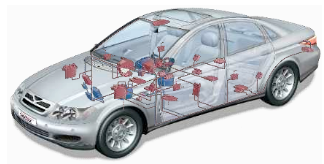




# Designing Distributed Real-Time Systems: a focus on holistic TT design


Luís Almeida

lda@det.ua.pt

Electronic Systems Lab-IEETA / DETI
Universidade de Aveiro
Aveiro, Portugal ieeta




# In this lecture...

- ✓ **Application requirements**
  - ✓ **Functional and non-functional**
- ✓ **System design**
  - ✓ **Entities and interactions**
- ✓ **The recurrent communicating tasks model**
  - ✓ **Tasks, messages, transactions**
- ✓ **Analysis and configuration methods**
  - ✓ **Holistic analysis**
- ✓ **The CAMBADA robots case study**

# DRTS requirements

- What do we want to do? (*functional req.*)
  - Feedback control
  - Environment monitoring
  - Multimedia communication
  - ...
  - **Set of actions and interactions**
- Performance req.s (*non-functional req.*)
  - Timing constraints
  - Fault tolerance
  - Concurrency control
  - ...
  - **Set of constraints concerning the actual implementation of the set of actions/interactions**

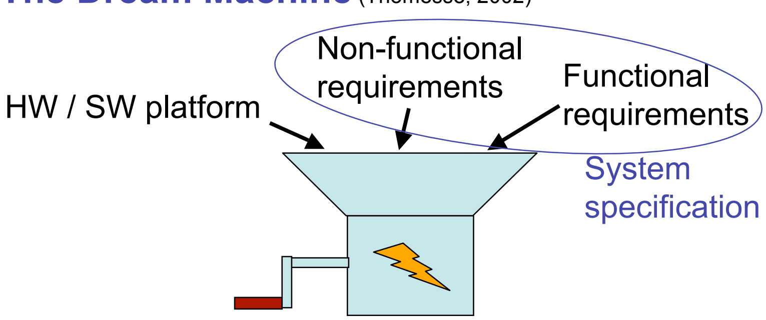
**System specification**





# Designing a DRTS

- **The Dream Machine** (Thomesse, 2002)




**The final system, ready to use !**

# Specifying a DRTS

- ✓ Can the Dream Machine make it up when the **specification is incomplete**?!!
- ✓ Can the Dream Machine cope with **inconsistent specifications**?

**Importance of formal modelling, formal specification, automated system design**

(Blair, 1998)

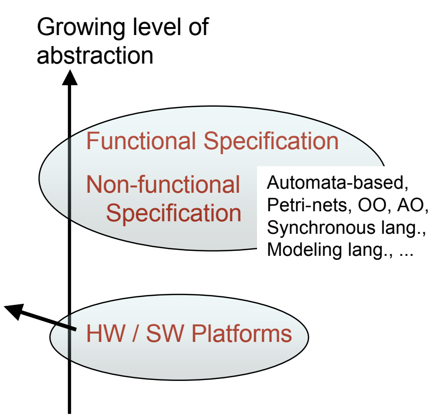
- ✓ **NO**!! We need to **specify exactly** what we want the system to do in **every anticipated operational scenario**.





# Designing a DRTS

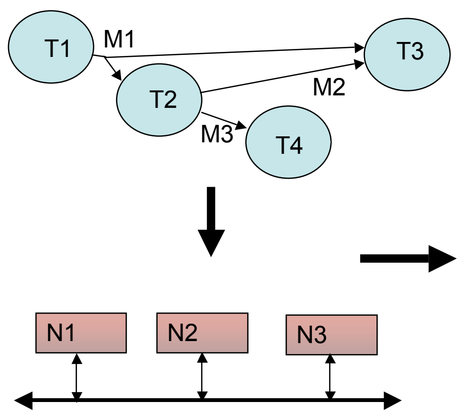
- ✓ Complex issue, goes through very different levels of abstraction



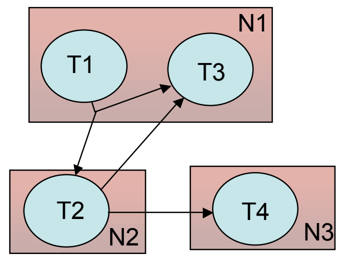

**Common abstraction at this level:**

Application specified as a **set of (recurrent) tasks** to be executed over **several processors** and exchanging **messages** over a **network**

# System model of a DRTS




**Set of (recurrent) tasks** to be executed over **several processors** and exchanging **messages** over a **network**

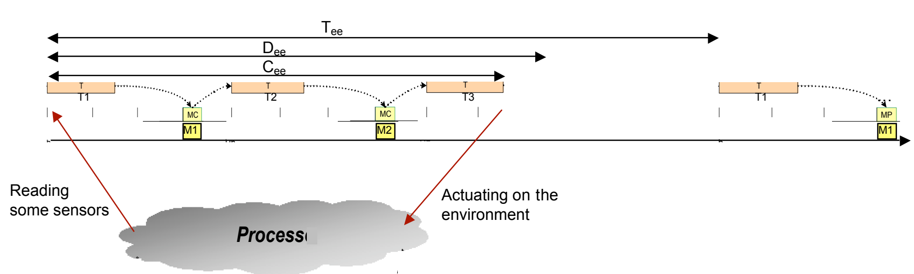






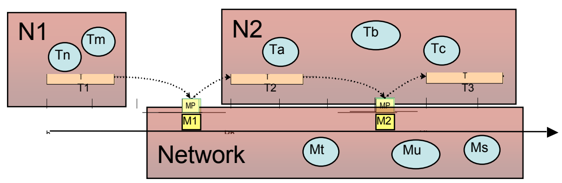

# System model of a DRTS

- Concept of **Transaction**
  - Represents the dataflow
  - Encompasses several tasks / messages
    - Possibly represented with a DAG
  - Has associated end-to-end properties and requirements



# Holistic analysis/design

- Given a set of **local timing attributes** (C,T,D,O,J,...) how can we verify whether **end-to-end requirements** are met?
  - This was named **Holistic Schedulability Analysis** (Tindell, 1994)
- But, more important, **how to come up with local timing** attributes (C,T,D,O,J,...) that allow meeting the end-to-end requirements?
  - This is very important when designing DRTS with common RTOS and network protocols.

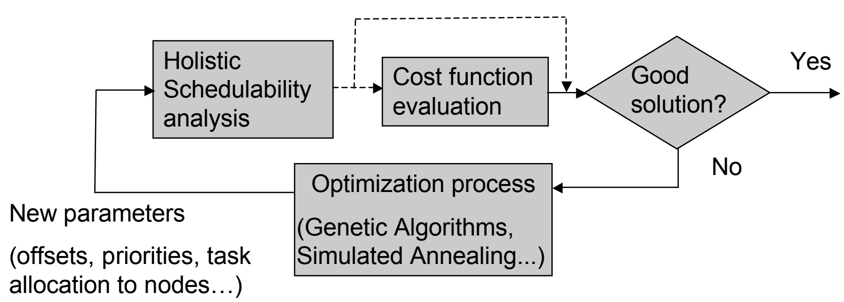







# Holistic analysis/design

- Once we have a holistic schedulability analysis tool, we can use an **optimization procedure** to come up with the *best* attributes (C,T,D,O,J,...) that allow meeting the end-to-end requirements.



# Holistic analysis/design

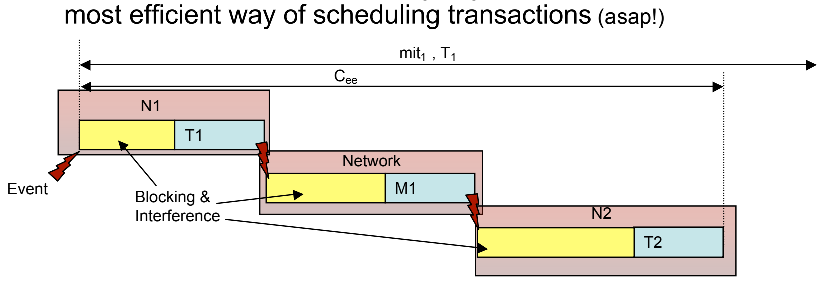
- When designing a DRTS we can follow two main approaches that lead to two different sets of analysis
  - **Event-triggered approach**
    - Transactions are initiated by events
    - All internal entities are triggered in sequence
    - Exact **activation instants are not known** at design time
  - **Time-triggered approach**
    - Transactions, as well as all internal entities are independent periodic processes that share the same period (or integer multiples of it, e.g. in multi-rate controllers).
    - Exact **activation instants are known** at design time





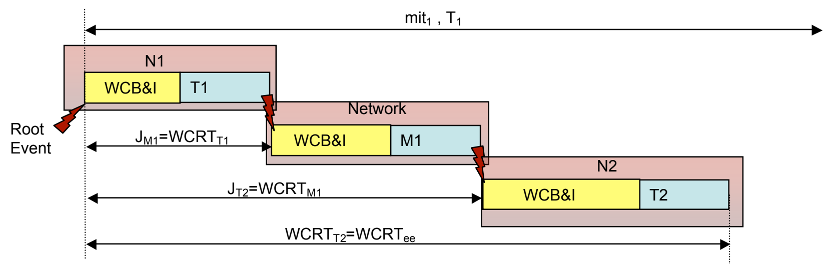
# Event-triggered approach

- A task **initiates a transaction** upon an **event** occurrence.
- From then on, all entities in the transaction are triggered by the termination of their predecessors – **event chain**.
  - Tight relationship among the actions in the transaction (changing actions has a direct impact at run-time).
  - Under certain assumptions, e.g., light load conditions, it is the most efficient way of scheduling transactions (asap!)



# Event-triggered approach

- Which conditions **maximize $C_{ee}$** ($WCRT_{ee}$)?
  - Consider the **worst-case Blocking & Interference** at each level
  - Account for a possible **release jitter** (J) caused by variations in the finish time (response-time) of the respective predecessors.

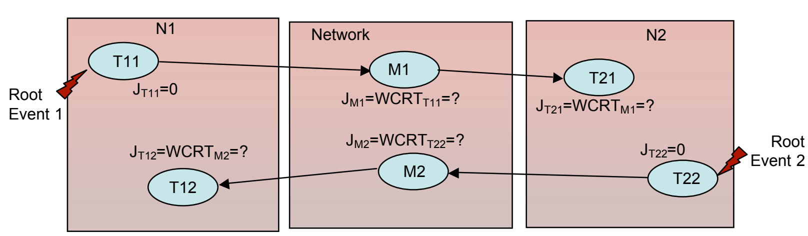





*Designing Distributed Real-Time Systems* © Luis Almeida

# Event-triggered approach

- Problem with the *release jitter*:
  - It establishes a **coupling** between the different active resources (nodes and network), **changing the WCB&I**
  - Solution, in FPS, iterate the whole analysis until convergence or deadline miss! (Tindell, 1994)



# Event-triggered approach

- Calculating the WCRT of tasks
  – Preemptive, independent, FPS, arbitrary deadlines

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

$$w_i(q) = (q+1)C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{J_j + w_i(q)}{T_j} \right\rceil C_j$$

- Calculating the WCRT of messages
  – CAN, Non-Preemptive, FPS, arbitrary deadlines

$$R_m = \max_{q=0..Q_m-1} \left(R_m(q)\right) \qquad R_m(q) = J_m + w_m(q) - qT_m + C_m$$

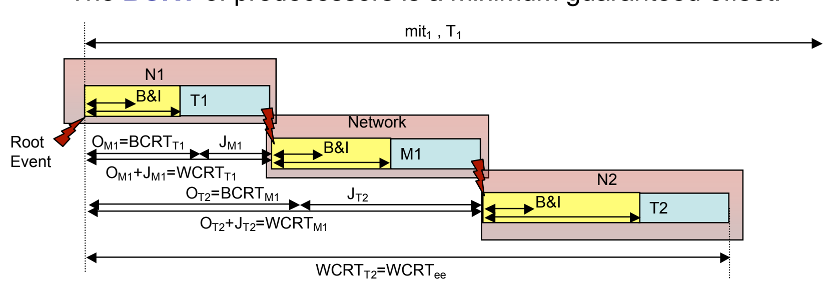
$$w_m^{n+1}(q) = B_m + qC_m + \sum_{\forall k \in hp(m)} \left\lceil \frac{w_m^n + J_k + \tau_{bit}}{T_k} \right\rceil C_k$$





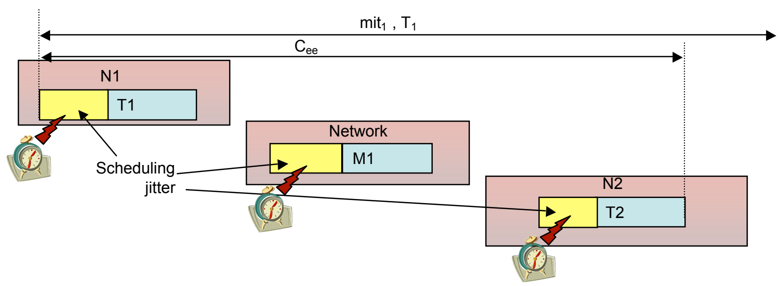
# Event-triggered approach

- Another approach consists on **using offsets** to capture the causal relationships within the event chain (Tindell, 1994a; Palencia & Harbour, 1998)
  – Offset analysis is less pessimistic than synchronous release analysis when there are offsets.
  – The **BCRT** of predecessors is a minimum guaranteed offset.



# Time-triggered approach

- There is a **global time** base (all nodes and network are synchronized).
- Transactions are **initiated at predefined instants** in time (so as all their internal entities).
  - The triggering events are all periodic and independent.
  - There can still be some scheduling jitter (certain exact periodic activation instants might be already taken by other entities)

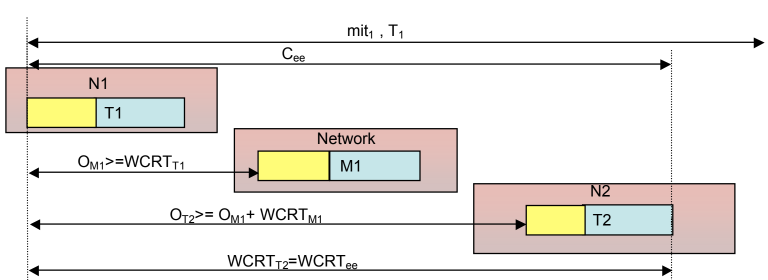







# Time-triggered approach

- Under certain conditions, e.g., harmonic periods, it is possible to find **offsets that eliminate scheduling jitter**
- **Offsets** can also be tuned for **optimizing** certain criteria
  - Reducing scheduling jitter (Coutinho, 2000)
  - Reducing end-to-end delay (Pop, 2003) (can be shorter than with ET approach when scheduling jitter is reduced and there is sufficient timing resolution)



# Time-triggered approach

- Since the **offsets** are **larger than WCRT** of predecessors, the scheduling at each active resource is **independent**.
  - No release jitter.
  - Whole analysis does not need to iterate.
- However, clear relationships can be identified among the local properties of the entities that make a transaction
  - **Data Streams** approach (Calha, 2005).

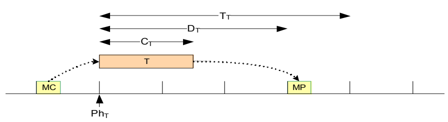
Task model

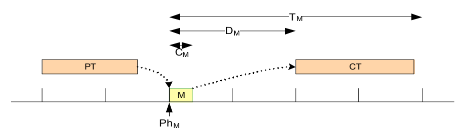
$S_T$ = { $ST_i$ : ($C_i, T_i, Ph_i, D_i, N_i, MP_i, MC_i$), $i$=1..NST }

$N$ – node of allocation

Message model

$S_M$ = { $SM_m$: ($C_m, T_m, Ph_m, D_m, PT_m, CTL_{mi}$), $m$=1..NSM , $i$=1..NCT }



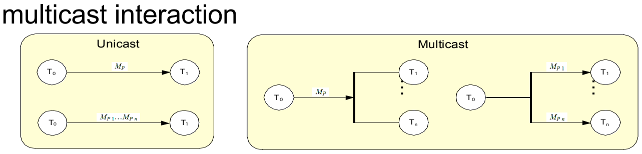





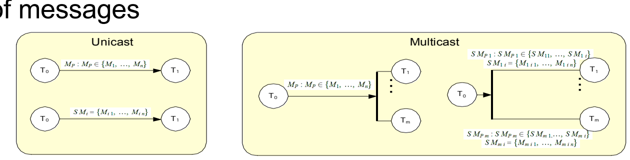

*Designing Distributed Real-Time Systems* © Luis Almeida

# Data Streams approach (by Mário Calha)

- Interaction between tasks
  - **Basic scenarios** represent the simplest forms of unicast and multicast interaction

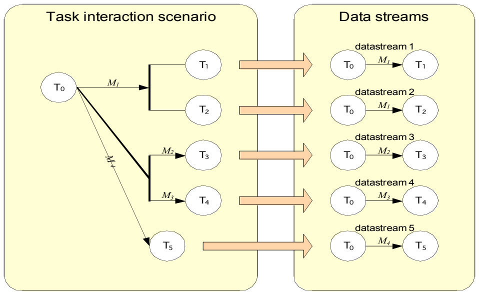



  - **Expansion scenarios** build upon the basic scenarios with sets of messages



# Data Streams approach

- Interactions are decomposed in *data streams*
  - Information flows concerning single logical entities






*Designing Distributed Real-Time Systems* © Luis Almeida

# Data Streams approach

- The core of the **Data Streams** approach is to start from:
  - The **specification of messages**, build a message schedule and propagate parameters to tasks (**net-centric approach**)

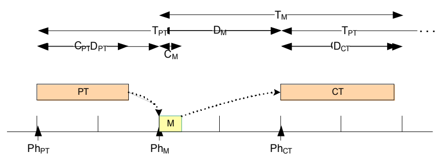
  or
  - The **specification of tasks**, build a task schedule in each node and propagate parameters to messages (**node-centric approach**)
  - Either way the approach delivers a set of relative phases for tasks and messages that determine the end-to-end delays

# Data Streams approach

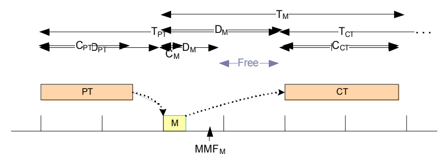
- **Net-centric** approaches

Calculated parameters
Required parameters

### Message Deadline




### Message Max Finishing time




| Approach | Method | Task and message scheduling | Periodic entities | Required parameters | Calculated parameters | Specific restrictions |
|---|---|---|---|---|---|---|
| Net-centric | Message Deadline (MD) | Independent | Tasks | C, MP, MC | T, D, Ph | |
| | | | Messages | C, T, D, PT, CTL | Ph | |
| | Message Maximum Finishing (MMF) | Dependent | Tasks | C, MP, MC | T, D, Ph | |
| | | | Messages | C, T, D, PT, CTL | Ph | |



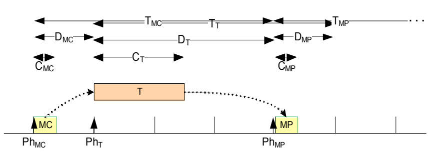


# Data Streams approach

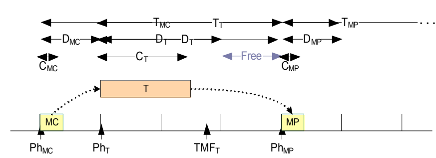
- **Node-centric** approaches

Calculated parameters
Required parameters

### Task Deadline

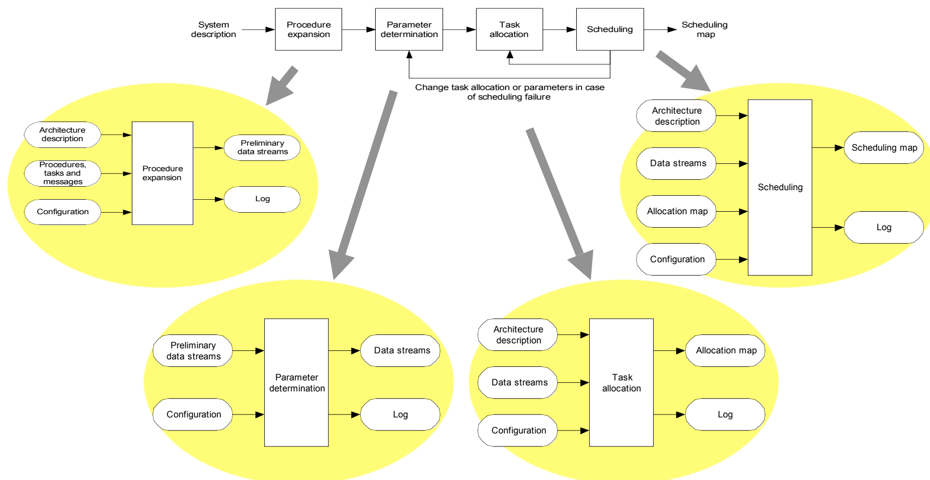



### Task Max Finishing time




| Approach | Method | Task and message scheduling | Periodic entities | Required parameters | Calculated parameters | Specific restrictions |
|---|---|---|---|---|---|---|
| Node-centric | Task Deadline (TD) | Independent | Tasks | C, T, D, MP, MC | Ph | |
| | | | Messages | C, PT, CTL | T, D, Ph | |
| | Task Maximum Finishing (TMF) | Dependent | Tasks | C, T, D, MP, MC | Ph | |
| | | | Messages | C, PT, CTL | T, D, Ph | |

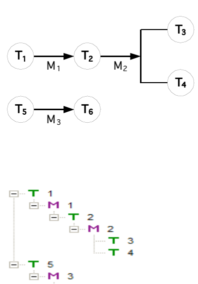
# Design flow using Data Streams

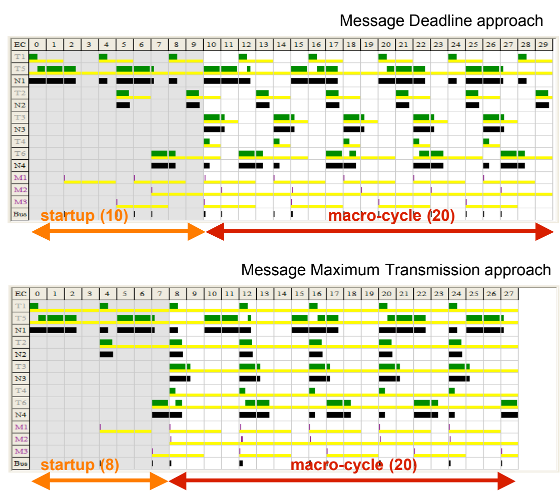







# SimHol, an analysis & config tool




*execution of a task*
*resource utilization*
*execution/transm. window*
*transmission of a message*

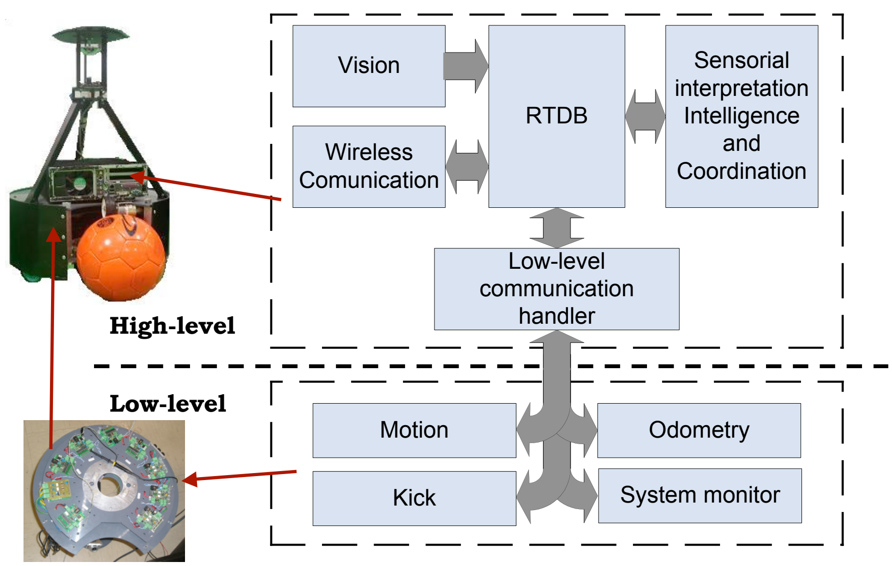
# Case study: a RoboCup MSL team

- MSL – Middle Size League
- **CAMBADA** – Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture
  - Requirements:
    - **Handle the ball in movement** (attackers)
      (ball speeds of 1m/s while passing)
    - **Intercept the ball** (goallee)
      (shots with ball speeds ~2m/s, sometimes >10m/s)
    - **Avoid obstacles** (other robots, goal posts, people!...)
      (robots speeds up to 2m/s)
    - **Keep a notion of localization**
    - **Interact with the other team members** (team strategy)





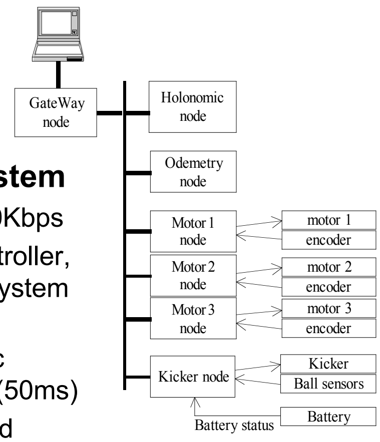
# Internal architecture of each robot






14

# Internal distributed sensing and actuation system




## Distributed computer 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), enconders reading(10ms)

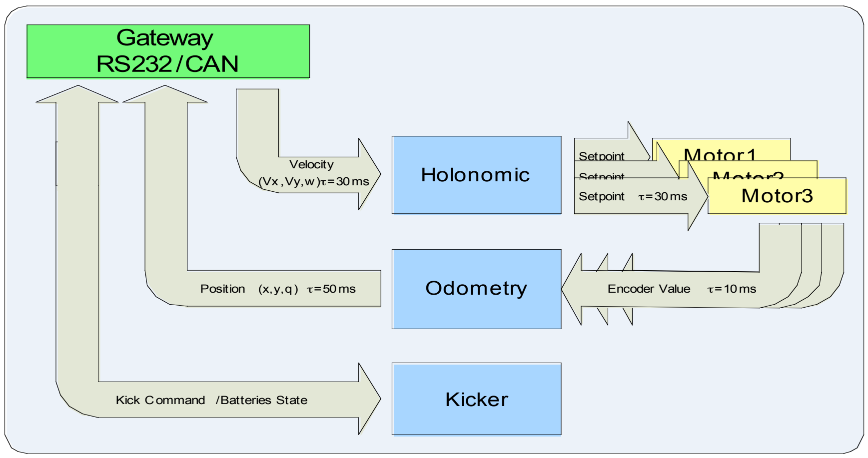




# Low level communication requirements

| 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 |

# Information Flows

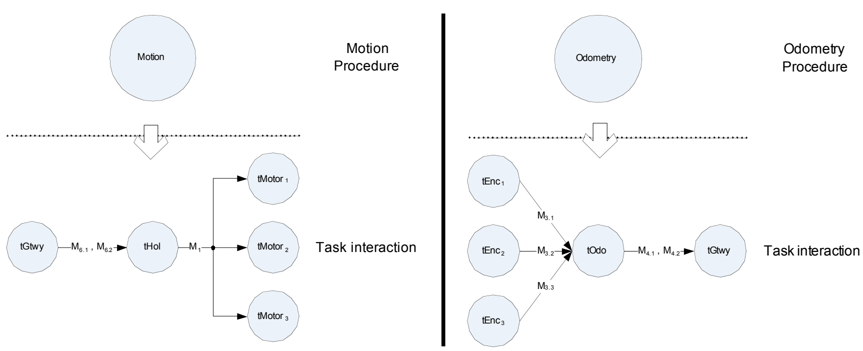






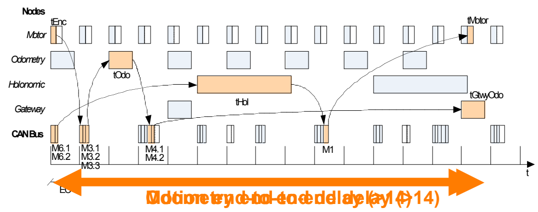

# Using Data Streams

- Real-time procedures identified: **motion** and **odometry**.

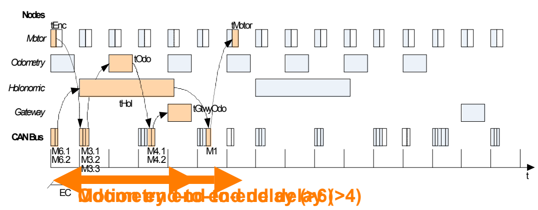


# Using Data Streams

- **Net-centric** approach
  - Message deadline approach:




  - Message maximum transmission approach:

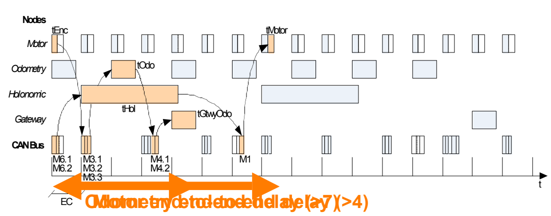




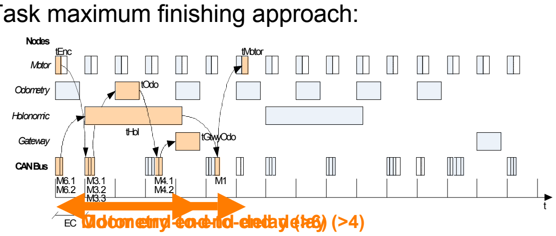

*Designing Distributed Real-Time Systems* © Luis Almeida

# Using Data Streams

- **Node-centric** approach
  - Task deadline approach:



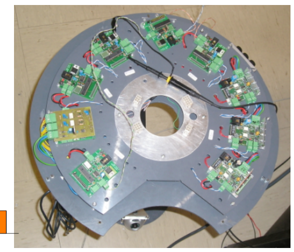

  - Task maximum finishing approach:

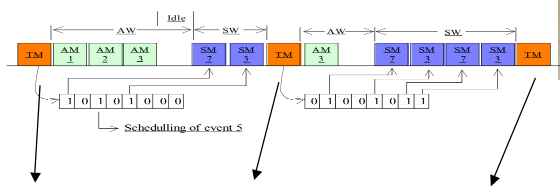


# Internal distributed sensing and actuation system

## Two implementations

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





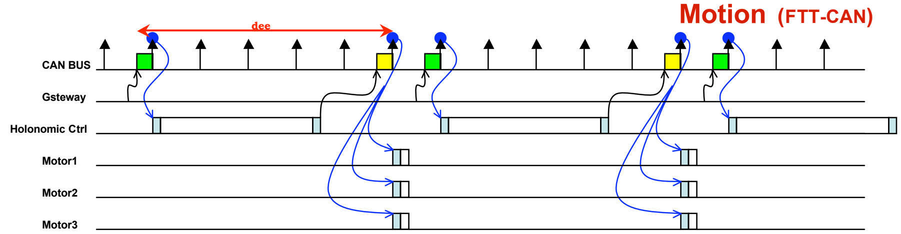

**Trigger message** sent regularly by the Master every Elementary Cycle:

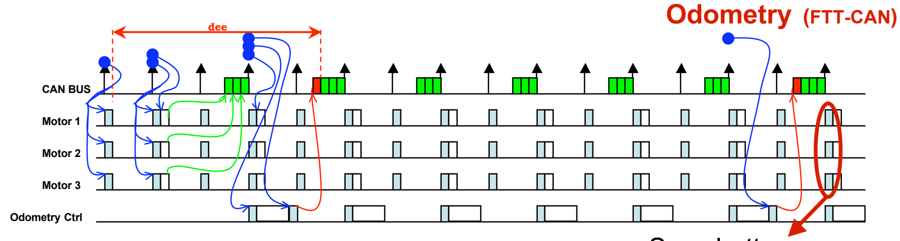
Triggers synchronous **messages and tasks**





# Cambada – Information flow with FTT




Sync better than **130 µs**

# 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 (as opposed to a drift up to 10ms with CAN)

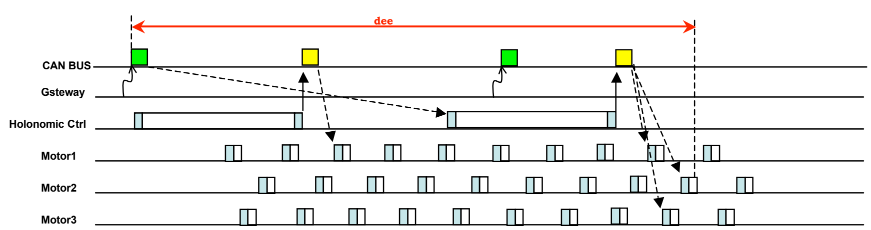
| Measure | Without FTT (ms) | With FTT (ms) |
|---|---|---|
| Setpoints from Gateway to actuation on motors | 38.8 to 64.4 | 26.7 to 27.7 |
| Encoders acquisition to Gateway reception of actual position | 12 to 21 | 21.6 to 21.7 |





# Results

- **Event-triggered approaches** (such as using simple CAN) **do not cope** well with **multi-rate** applications
  - Even without events coming from the Gtw, the Holonomic controller had to continue execution at 30ms rate and the motor controllers had to continue execution at 5ms
  - Drifts lead to bad phasing that may cause extra delay of 35ms!



# Rewinding

- We have addressed the problem of **specifying and designing a DRTS** using tasks and messages
- Particularly we considered **holistic scheduling** analysis methods
- Two major approaches can be followed in designing a DRTS, **ET and TT**
- Analysis for **ET** must account for the **coupling between tasks across nodes**
  - This can be modeled as release jitter or static and dynamic offsets
- Analysis for **TT** considers **independent periodic entities** scheduled with **static offsets**.
- We saw a **case study** where a TT aproach was used, namely **Data Streams**.





# References

Thomesse (2002). J.-P. Thomesse, “Open Issues in Fieldbus-based Systems”. IFAC World Congress, Barcelona, Spain, July 2002.

Blair (1998). G.S. Blair, L. Blair, H. Bowman, A. G. Chetwynd, "Formal Specification of Distributed Multimedia Systems", London: UCL Press, 1998.

Tindell (1994). K. Tindell, J. Clark, “Holistic Schedulability Analysis for Distributed Hard Real-Time Systems”. Microprocessors and Microprogramming, 40, Elsevier, 1994.

Tindell (1994a). K. Tindell. “Adding Time-Offsets to Schedulability Analysis”. Technical Report YCS221. Department of Computer Science, University of York. 1994.

Palencia & Harbour (1998). Palencia, J. C., and Harbour, M. G. “Schedulability Analysis for Tasks with Static and Dynamic Offsets”. *IEEE Real-time Systems Symposium.* 1998.

Coutinho (2000). F. Coutinho, J.A. Fonseca, J. Barreiros, E. Costa. "Jitter Minimisation with Genetic Algorithms“. Proceedings WFCS'2000 - 3rd IEEE International Workshop on Factory Communication Systems, Porto, Portugal, 5-8 September 2000

Pop (2003). T. Pop, P. Eles, Z. Peng. “Schedulability Analysis for Distributed Heterogeneous Time/event Triggered Real-Time Systems”. 15th Euromicro Conf. on Real-Time Systems (ECRTS 2003). Porto, Portugal. July 2003.

Calha (2005). Calha, M.J., J.A. Fonseca, “Data Streams – an Analysis of the Interactions Between Real-Time Tasks”, Proceedings of the 10th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2005), Catania, Italy, Sept. 2005.