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**

Designing a DRTS

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

HW / SW platform

Non-functional requirements

Functional requirements

System specification

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?

✓ 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

Growing level 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

• 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, \textbf{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.

New parameters
(offsets, priorities, task allocation to nodes…)

- Holistic Schedulability analysis
- Cost function evaluation
- Optimization process (Genetic Algorithms, Simulated Annealing...)
- Good solution?
  - Yes
  - No
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.

![Event-triggered approach diagram]

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 diagram]
Event-triggered approach

- Calculating the WCRT of tasks
  - Preemptive, independent, FPS, arbitrary deadlines
    \[
    \eta_j = \max_{q=0,1,2,...} \left( J_j + w_j(q) - qT_j \right)
    \]
    \[
    w_j(q) = (q+1)C_i + \sum_{\forall j \in \beta(q)} \left( \frac{J_j + w_j(q)}{T_j} \right) C_j
    \]

- Calculating the WCRT of messages
  - CAN, Non-Preemptive, FPS, arbitrary deadlines
    \[
    R_m = \max_{q=0,1,2,...} \left( R_m(q) \right)
    \]
    \[
    R_m(q) = J_{n_x} + w_m(q) - qT_m + C_n
    \]
    \[
    w_m^{n_x}(q) = B_m + qC_m + \sum_{\tau \in \mathcal{P}(m)} \left( \frac{w_m^{n_x} + J_x}{T_x} \right) C_x
    \]

---

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)

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).

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

\[
S_M = \{ SM_m : (C_m, T_m, Ph_m, D_m, PT_m, CTL_m) \}, m=1..NSM, i=1..NCT
\]

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

![Diagram showing task interaction and data streams]

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**

  **Message Deadline**

  ![Diagram of Message Deadline]

  **Message Max Finishing time**

  ![Diagram of Message Max Finishing time]

<table>
<thead>
<tr>
<th>Approach</th>
<th>Method</th>
<th>Task and message scheduling</th>
<th>Periodic entities</th>
<th>Required parameters</th>
<th>Calculated parameters</th>
<th>Specific restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net-centric</td>
<td>Message Deadline (MD)</td>
<td>Independent</td>
<td>Tasks C, ME, MR</td>
<td>T, D, Ph</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Message Minimum Finishing (MMF)</td>
<td>Dependent</td>
<td>Tasks C, ME, MR</td>
<td>T, D, Ph</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

November 28, 2006  
University of Pennsylvania, Philadelphia, USA  
23

---

Data Streams approach

- **Node-centric approaches**

  **Task Deadline**

  ![Diagram of Task Deadline]

  **Task Max Finishing time**

  ![Diagram of Task Max Finishing time]

<table>
<thead>
<tr>
<th>Approach</th>
<th>Method</th>
<th>Task and message scheduling</th>
<th>Periodic entities</th>
<th>Required parameters</th>
<th>Calculated parameters</th>
<th>Specific restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node-centric</td>
<td>Task Deadline (TD)</td>
<td>Independent</td>
<td>Tasks C, ME, MR</td>
<td>T, D, Ph</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Task Maximum Finishing (TMF)</td>
<td>Dependent</td>
<td>Tasks C, ME, MR</td>
<td>T, D, Ph</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

November 28, 2006  
University of Pennsylvania, Philadelphia, USA  
24
Design flow using Data Streams

SimHol, an analysis & config tool
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** (goalie) (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**

- **Vision**
- **Wireless Communication**
- **RTDB**
- **Sensorial interpretation Intelligence and Coordination**
- **Low-level communication handler**
- **Motion**
- **Kick**
- **Odometry**
- **System monitor**
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), encoders reading (10ms)

Low level communication requirements

<table>
<thead>
<tr>
<th>ID</th>
<th>Source</th>
<th>Target</th>
<th>Type</th>
<th>Period/Period (ms)</th>
<th>Size (B)</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>Holonomic ctrl</td>
<td>Motor node [1-3]</td>
<td>Periodic</td>
<td>30</td>
<td>6</td>
<td>Aggregate motor speeds setpoints</td>
</tr>
<tr>
<td>M2</td>
<td>Kicker</td>
<td>Gateway</td>
<td>Periodic</td>
<td>1000</td>
<td>2</td>
<td>Battery status</td>
</tr>
<tr>
<td>M3.1- M3.3</td>
<td>Motor node [1-3]</td>
<td>Odometry node</td>
<td>Periodic</td>
<td>5 to 20</td>
<td>3+3</td>
<td>Wheels encoder values</td>
</tr>
<tr>
<td>M4.1- M4.2</td>
<td>Odometry node</td>
<td>Gateway</td>
<td>Periodic</td>
<td>50</td>
<td>7+4</td>
<td>Robot Position+orientation</td>
</tr>
<tr>
<td>M5.1-M5.2</td>
<td>Gateway</td>
<td>Odometry node</td>
<td>Sporadic</td>
<td>500</td>
<td>7+4</td>
<td>Set/Reset robot position+orientation</td>
</tr>
<tr>
<td>M6.1- M6.2</td>
<td>Gateway</td>
<td>Holonomic ctrl</td>
<td>Periodic</td>
<td>30</td>
<td>7+4</td>
<td>Velocity vector (linear+angular)</td>
</tr>
<tr>
<td>M7</td>
<td>Gateway</td>
<td>Kicker</td>
<td>Sporadic</td>
<td>1000</td>
<td>1</td>
<td>Kicker actuation</td>
</tr>
<tr>
<td>M8-M12</td>
<td>Every node</td>
<td>Gateway</td>
<td>Sporadic</td>
<td>1000</td>
<td>5^2</td>
<td>Node hard reset</td>
</tr>
</tbody>
</table>
Information Flows

Using Data Streams

- Real-time procedures identified: motion and odometry.
Using Data Streams

- **Net-centric** approach
  - Message deadline approach:
  
  ![Net-centric Message Deadline Diagram]

  ![Net-centric Message Deadline Diagram](image)

  - Message maximum transmission approach:
  
  ![Net-centric Message Max Transmission Diagram]

  ![Net-centric Message Max Transmission Diagram](image)

- **Node-centric** approach
  - Task deadline approach:
  
  ![Node-centric Task Deadline Diagram]

  ![Node-centric Task Deadline Diagram](image)

  - Task maximum finishing approach:
  
  ![Node-centric Task Max Finishing Diagram]

  ![Node-centric Task Max Finishing Diagram](image)
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

Motion (FTT-CAN)

Odometry (FTT-CAN)

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)

<table>
<thead>
<tr>
<th>Measure</th>
<th>Without FTT (ms)</th>
<th>With FTT (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setpoints from Gateway to actuation on motors</td>
<td>38.8 to 64.4</td>
<td>26.7 to 27.7</td>
</tr>
<tr>
<td>Encoders acquisition to Gateway reception of actual position</td>
<td>12 to 21</td>
<td>21.6 to 21.7</td>
</tr>
</tbody>
</table>

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 approach was used, namely Data Streams.

References