Coordination of tasks on a Real-Time OS

Guillermina Cledou & José Proença & Bernhard H.C. Sputh & Eric Verhulst

Coordination 2019 - Copenhagen, 20 June

Slides: coord19.proenca.org
Real-Time OS

Coordination

VirtuosoNext™
(the real-time OS)

Visual Designer
(the IDE)

Gluing tasks
Real-Time OS

VirtuosoNext™
(the real-time OS)

Visual Designer
(the IDE)

Coordination

Gluing tasks

Our contribution:
How to improve this part
Real-Time OS

Coordination

Gluing tasks

Our contribution:
How to improve this part

Virtual
Visual

DaVinci project

Distributed architectures:
Variability and interaction in CPS

KURT modular vehicles

CISTER
Research Center in Real-Time & Embedded Computing Systems

Altreonic

DaVinci project

KURT modular vehicles

CISTER
Research Center in Real-Time & Embedded Computing Systems
Outline

Programming Real-time systems with **VirtuosoNext**™

Understanding interactions between tasks

Building interaction protocols

Online prototype to analyse protocols

http://arcatools.org/#virtuoso
Programming a RTOS
The classical way

Task 1
Initialise (some work)
Every 10s (more work)

Task 2
Initialise (some work)
Every 10s (more work)

Actuator task
Initialise (some work)
Every 5s (Consume data from Task 1 & 2)

Scheduler
Deadline? Computation time? Priority?
Schedule...

Shared data!
Programming a RTOS
The VirtuosoNext™ way
Programming a RTOS
The *VirtuosoNext™* way

Task 1

Initialise
...
while(true)
  test(SemaB)
  (some work)
  put(Actuate)
  signal(SemaA)
Programming a RTOS

The **VirtuosoNext™** way

Initialise
...
while(true)
  test(SemaB)
  (some work)
  put(Actuate)
  signal(SemaA)

Descheduling point

**HUBs** → Services by the OS

Slides: coord19.proenca.org
**Hubs in VirtuosoNext™**

- Executed by a dedicated **Kernel Task**
- Decide who can be scheduled
- May have state
Hubs in VirtuosoNext™

Executed by a dedicated **Kernel Task**

Decide who can be scheduled

May have state

Semaphore
Port
Event
DataEvent
Resource
Fifo
Blackboard

Slides: coord19.proenca.org
Running Visual Designer
Timing executions

[Image of a timing diagram showing events and semaphore operations.]
Hubs Semantics

Semaphore

- **signal** – signals the semaphore, incrementing an internal counter $c$. Succeeds if $c < \text{MAX}$.
- **test** – checks if $c > 0$, in which case succeeds, and decrements $c$.

Port

- **put** – signals some data entering the port
- **get** – signals some data leaving the port

*Both must synchronize to succeed.*
Hubs Semantics

**update** – sets an event and buffers some data, overriding any previous data. Always succeeds.

**read** – reads the data. Succeeds if the event is set.

**clear** – clears the buffer and the event.

**enqueue** – buffers some data in the queue. Succeeds if the queue is not full.

**dequeue** – reads the next data. Succeeds if the queue is not empty.
Challenge

How to add new Hubs?
Challenge

How to

- **generalise** Hubs
- **build** complex (and useful) Hubs
- **analyse** Hubs (trust Hubs)

How to **add** new Hubs?
Hubs ++

Automata semantics

Semaphore

entry point

update: \[ u := x \leftarrow e \mid u; u \mid u\mid u \]

guard

state

\[ \langle c < \text{MAX} \rangle \]
\[ c \leftarrow c + 1 \]

\[ \langle c > 0 \rangle \]
\[ c \leftarrow c - 1 \]
Hubs ++

Automata semantics

with data + synchronisation

Semaphore

signal

\[ c < \text{MAX} \]

\[ c \leftarrow c + 1 \]

idle

test

\[ c > 0 \]

\[ c \leftarrow c - 1 \]

Port

idle

\( \text{put} | \text{get} \)

\( \text{get} \leftarrow \text{put} \)

many entry points
Hubs ++

Automata semantics
with data + synchronisation

with composition
Hubs ++

Automata semantics with data + synchronisation

with composition

- online tool -

http://arcatools.org/#virtuoso
Example 1

Tasks 1 & 2 alternate between signal, test, and put.
Example 2

Tasks 1 & 2 can start and eventually put a value
Examples 3 & 4

Actuator alternates between tasks

Hub imposes alternation
Insights gained

**Performance (1000 rounds)**
Example 1 - **40ms**
Run complex example (2)
- as a **user task** - **60ms**
- as simpler **native hub** - **30ms**
Simplest (example 3) - **20ms**

**Coordination burden**
often moved to tasks

**Nr. Context Switches**
Ex. 1 - **17** / Ex. 2 - **13** / Ex. 3&4 - **9**

**The devil is in the details**

**Blackbox tasks:**
harder to reason
Wrap up

Automata semantics

Semaphore

\[ \mathcal{X} = \{ c : \mathbb{N} \} \]

\begin{align*}
\text{signal} & \quad \langle c < \text{MAXINT} \rangle \\
\text{test} & \quad \langle c > 1 \rangle \\
\end{align*}

\[ c \leftarrow c + 1 \]

\[ c \leftarrow c - 1 \]
Wrap up

Automata semantics

\[ \mathcal{X} = \{ c : \mathbb{N} \} \]

- signal: \( \langle c < \text{MAXINT} \rangle \)
  - \( c \leftarrow c + 1 \)

- test: \( \langle c > 1 \rangle \)
  - \( c \leftarrow c - 1 \)

Complex hubs by composition

Slides: coord19.proenca.org
Wrap up

Automata semantics

```
Semaphore
\( X = \{ c : \mathbb{N} \} \)

\( \begin{align*}
\text{signal} & \quad \langle c < \text{MAXINT} \rangle \\
\text{idle} & \quad c \leftarrow c + 1 \\
\text{test} & \quad \langle c > 1 \rangle \\
& \quad c \leftarrow c - 1
\end{align*} \)
```

Ongoing work:
- DSL for tasks
  (Funct. React. Prog + hubs)
- Commun. across nodes

Thank you!