Real-Time Systems
Achievements and Perspectives

Giorgio Buttazzo

Real-Time Systems Laboratory

Keynote at the 35th IEEE Real-Time Systems Symposium, Rome, 2014
Outline

1. A quick look into my past
2. Major achievements in the RT community
3. A look into the future
Back to 1987

- I started my PhD working on robotics
- Active perception and sensory-motor coordination

Diagram:
- PC 386
- RS232
- Unimation Controller
- PUMA robot
- Force sensor
- Camera
- Delay 60 ms
- Alter (T = 28 ms)
- Half frame every 20 ms
- A frame every 10 ms
Very soon, I realized a strong need for RT support:

- **Scheduling**: periodic & aperiodic tasks
- **Mixed criticality**: hard & soft deadlines
- **Data sharing**: async. comm. among periodic tasks
- **Analysis**: effects of delay and jitter on performance

**Crucial references**

The change: 1993

- I heard Jack Stankovic was looking around for sabbatical, so I invited him in Pisa.
- At that time there were not PDF files, so he brought a lot of RT papers to read ... very useful ... thanks Jack!

RTSS 1993
(Raleigh-Durham, NC, USA)
was my first RT conference
RT Conferences

- ICCPS
- SIES
- RTNS
- EMSOFT
- DATE
- ETFA
- RTCSA
- RTAS
- ECRITS
- RTSS

Year: 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 00, 02, 04, 06, 08, 10, 12, 14

Groups on RTS in Italy

- Trento (Palopoli, Abeni)
- Padova (Vardanega)
- Modena (Bertogna)
- Bologna (Caselli)
- Ancona (Dragoni)
- Milan (Leva)
- Pavia (Facchinetti)
- Pisa (ReTiS lab)
- Florence (Vicario)
- Siena (Giorgi)
- Rome (Spaccamela Bonifaci)
- Catania (Lo Bello)
Groups on RTS in Europe

**Austria**: Vienna

**Czech Rep**: Prague

**France**: Paris, Grenoble, Renne, Nantes, Nancy, Toulouse, Lille

**Germany**: Munich, Kaiserslautern, Dresden, Karlsruhe, Saarland

**Ireland**: Dublin, Cork

**Italy**: Pisa, Pavia, Catania, Siena, Florence, Bologna, Trento, Padova, Modena, Ancona, Rome

**Portugal**: Porto, Aveiro, Lisbon

**Spain**: Madrid, Cantabria, Valencia, Barcelona, Palma de Mallorca

**Sweden**: Stockholm, Lund, Vasteras, Uppsala, Halmstad, Linkoping

**Switzerland**: Zurich, Lausanne

**UK**: York
Solved problems

Platform

Constraints

Task model

EXECUTION MODEL
Execution Model

Task models

- Non recurring (single jobs)
- Recurring
  - Aperiodic
  - Sporadic
  - Periodic
  - Strictly periodic
  - Multiframe
  - DAGs
  - Digraf

Constraints

- Precedence
- Resources
- Self-suspensions
- Mode changes
- Fully Preemptive
- Fully Non preemptive
- Limited preemptive
  - Preemption thresholds
  - Floating regions
  - Deferred preemptions
  - Fixed preemption points

Task models

- Recurring
  - Aperiodic
  - Sporadic
  - Periodic
  - Strictly periodic
  - Multiframe
  - DAGs
  - Digraf
- Non recurring (single jobs)

Constraints

- Precedence
- Resources
- Self-suspensions
- Mode changes
- Fully Preemptive
- Fully Non preemptive
- Limited preemptive
  - Preemption thresholds
  - Floating regions
  - Deferred preemptions
  - Fixed preemption points
Platforms

Uniprocessor

- Fixed speed
- Variable speed (DVFS)
  - continuous
  - discrete
- Low power states (DMP)
- With cache

Multiprocessor

- Identical
- Uniform
- Heterogeneous
- Hybrid

Distributed

- Wired
  - crossbar
  - mesh
  - star
  - tree
  - bus
- Wireless
  - single-hop
  - multi-hop

Optimization criteria

- Feasibility
- Response time
- Maximum lateness
- Utilization bandwidth
- Energy consumption
- Temperature
- Number of processors
Results on RTS

Platform

MP-hetero
MP-uniform
MP-imprecise
DVS

Task model

Constraints

single job
periodic
per + server
skip
elastic
recurring
imprecise
mix critical

What is really being used?

Platform
- MP-hetero
- MP-unif
- MP-ident
- DVS+DMP
- DVS
- uni
- single job
- period
- per + server
- skip
- elastic
- recurring
- imprecise
- mix critical

Constraints
- dependent
- resource
- precedence
- res + prec.
- energy
- temperature

Task model

Some major achievements

model complexity (realism)

- Single job
- Periodic
- Aperiodic servers
- Resource sharing
- Resource Reservation
- Exact Analysis
- Recurring tasks
- Res.+ resources
- Variable-rate
- Mixed-criticality

Techniques:
- EDD
- EDF
- EDL
- EDF
- RTA
- PDC
- TBS
- CBS
- SRP
- PCP
- PIP
- SRP
- DVFS
- DPM
- PS
- DS
- SS
- Slack Stealer
- BROE
- SIRAP
- BWI
- Overrun
- Digraf
- DAGs

Key phrases:
- Resource sharing
- Exact Analysis
- Recurring tasks
- Res.+ resources
- Variable-rate
- Mixed-criticality

Years:
- 55
- 73
- 86
- 90
- 95
- 98
- 2000
- 04
- 07
- 11
- 14
Concrete impact

- **Rate Monotonic** is used in most industrial settings
- **Priority Inheritance** is in most RT kernels
- **Sporadic Server** is specified in POSIX
- **CBS** is implemented in LINUX
- **EDF** is now supported by a few kernels
  - **Erika Enterprise** (by Evidence): certified **OSEK** and adopted by Magneti Marelli in next generation ECUs
  - **Ada 2005** runtime support
  - **Linux** (**SCHED_DEADLINE** in mainline since June 2014)
- Lots of **RT tools** are now available for
  - WCET estimation, schedulability analysis, scheduling simulation, formal verification, etc.
From RTS to CPS

Are they just buzzwords?

In 2002 RTAS changed name
from Real-Time Technology and Applications Symposium
to Real-Time and Embedded Technology and Applications Symposium

In 2010 ICCPS appeared
International Conference on Cyber-Physical Systems

Definitions

Real-Time System
Computing system able to provide bounded response times to tasks with bounded execution, in all possible scenarios.

Embedded System
Computing system embedded into a larger device, dedicated to control its functions, manage the available resources, and simplify the interaction with the user.

Cyber-Physical System
System where software is tightly coupled with the physical characteristics of the plant to be controlled.
Cyber-Physical Systems

- Sensing
- Computation
- Control
- Communication

Design
Analysis
Programming

Software

System dynamics

Embedded Computer

Controlled System
One (wrong) view
My view

- TV remote control
- Light control system
- USB pen
- flight control
- engine control
- inverted pendulum
- robot control
- optimizing car shape
- energy / thermal management
- digital TV
The truth is that...

Complex systems require interdisciplinary knowledge

- System modeling
- Control design
- Source coding
- Constraints derivation
- Formal verification
- Task scheduling
- Resource management

- Physics
- RTOS
- Programming
- Real-time theory
- Control Theory
- Optimization
- Formal methods
An instructive example

**PROBLEM:** Find the sampling periods of the sensors that guarantee the feasibility of the goal.

**GOAL:** If an obstacle is detected, stop the train without hitting the obstacle.

Diagram:
- Human controls
- Dashboard Controls
- Distribution Unit
- BRAKES
- Sensors
- Condition checker
- Emergency stop
- Obstacle

Diagram notes:
- v: sensor visibility
- D: obstacle
Assumptions

- Tasks are scheduled by Rate Monotonic (implicit deadlines)
- Let $\tau_s(C_s, T_s)$ be the periodic task devoted to sampling
- Assume $\tau_s$ has the shortest period (highest priority).
- Let $U_{other}$ be the load of the other tasks
Minimum period

The minimum period can be computed imposing the system schedulability by the Liu & Layland test:

The system is schedulable if

\[
\frac{C_s}{T_s} + U_{other} \leq U_{lub}^{RM}
\]

that is

\[
T_s \geq \frac{C_s}{U_{lub}^{RM} - U_{other}}
\]

\[
T_s^{min} = \frac{C_s}{U_{lub}^{RM} - U_{other}}
\]
Maximum period

Worst-case reasoning

$\tau_s$  
$T_s$  
$\Delta$  
$T_b$  

obs T_s

speed

obstacle in the field  obstacle detected  brake pressed  train stopped
Safety condition

The space covered by the train in \((T_s + \Delta + T_b)\) should not exceed

\[
T_b = \frac{v}{\mu g}
\]

\[
X_b = \frac{v^2}{2\mu g}
\]

The diagram illustrates the processes:
- Obstacle in the field
- Obstacle detected
- Brake pressed
- Train stopped

The space covered by the train is calculated as

\[
D = \nu (T_s + \Delta + T_b)
\]

where
- \(D\) is the distance covered by the train
- \(\nu\) is the speed of the train
- \(T_s\) is the time to detect the obstacle
- \(\Delta\) is the time for the brake to be pressed
- \(T_b\) is the time it takes for the train to stop

\(X_b\) is the space covered by the train.
Safety condition

\[ v(T_s + \Delta) + X_b < D \]

\[ X_b = \frac{v^2}{2\mu g} \]

\[ v(T_s + \Delta) + \frac{v^2}{2\mu g} < D \]

\[ T_s < \left( \frac{D}{v} - \frac{v}{2\mu g} \right) - \Delta \]

\[ T_s^{\text{max}} \]
Safety condition

\[ T_{s}^{\max} = \frac{D}{v} - \frac{v}{2\mu g} - \Delta \]

**NOTE:** if \( v_{0} > v_{\max} \) no solution exists, no matter how fast the CPU is!
How can we increase impact?

Producing and publishing results is not enough...

- Show concrete benefits
- Convince industrial people (not easy)
- Simplify the use of results
- Teach people how to use the results
Showing the benefits

What do we gain using a specific result?

- performance
- reliability
- safety
- predictability
- efficiency
- faster design

Can you quantify in terms of money?

In mass productions every dollar you save can be worth a million $.

Convincing people

First of all convince yourself.

- Identify **critical scenarios** (may be rare, but dangerous)

- Build **working systems** around critical scenarios

- Show that your method is able to deal with critical situations, so it's worth paying extra overhead.
A working demo is not enough

"Look! It's working using a RT kernel"

"A spoon in the bottle keeps Champagne bubbly"

What it is missing is falsification
"Look! It's **NOT** working under Windows"

"But it **perfectly works** under Erika Enterprise"

Identify at least a critical situation in which this happens.
Simplify

If you want to have a big impact in the society, then seek for simplicity

- However, achieving simplicity is very difficult, because it implies
  - deep understanding
  - distilling ideas
  - beauty & elegance

- Hence, it takes time, effort, and skill
Read the literature

➢ To know the most recent results

➢ To avoid re-discovering the wheel

➢ Lots of results produced in the last 30 years
How about the future of RTS?

Will RTS become obsolete and reach a dead end?
How about the future of RTS?

New technologies will provide **new platforms**

New applications will require **new models**

Just keep your models **realistic** and **up-to-date**

**TIME** will always be a constraint
How about the future of RTS?

For the same reason

RTSS should keep its own identity and name

Including new topics is good and necessary, but following the wind and buzzwords is dangerous.

This year RTSS celebrates the 35th anniversary.
Happy Birthday and Long Life