1

Jorvik:

A Framework for **Flexible Real-Time** Systems

Guillem Bernat
Real-Time Systems Research Group
Department of Computer Science, University of York. 2002
http://www.cs.york.ac.uk/rts

Contents

3

1 Flexible Real-Time Systems

4

Hard real-time systems

- Real-time system: time is important
- All deadlines have to be guaranteed
- System is predictable => analysable for the worst case behaviour
- Analysis : WCET + Schedulability tests
- But:
  - System is rigid and pessimistic, and therefore system is overspecified
  - Not adequate model for the requirements of future real-time systems

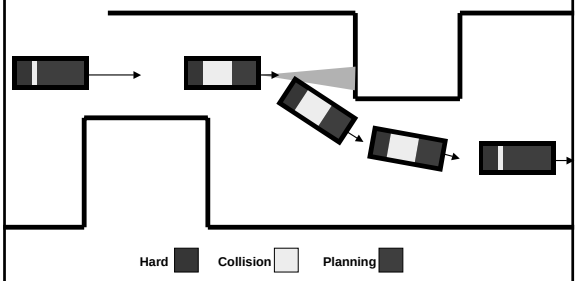
5

Flexible real-time systems

- Only a subset of the system is really hard:
  - firm, soft,
  - weakly-hard: i.e. meet *n* out of *m* consecutive invocations.
  - value/utility functions
- Some unpredictable or unbounded components
- Dynamic/changing environments. Modes of operation.
- Not enough time to finish all components.
- Adaptiveness: Trade-off between quality of the result and the time at which it is made available.
- Guarantee hard deadlines + maximise system utility (optimally?)

6

Example: mobile robot



7

Example 2: Robocup

- Robots
  - Small, limited bandwidth
- Central station
  - COMM: Broadcasts commands to team of robots
  - HI: High-level strategic reasoning
    - Strategy identification
    - Machine learning approaches
  - MI: Mid-level tactic planning
    - Path planning
    - Tactic manoeuvre
    - Vision subsystem
  - LO: Low-level reactive behaviour
  - RTDB: Real-time database management
- Most levels are highly computationally expensive
  - Cycle times, from 30ms to 5 secs
  - Ci is *useless*, it is to pessimistic

8

Robocup

9

10

Summary of requirements

- System decomposed in two broad subsystems A,B (that interact)
  - R1: Guarantee that all tasks of subsystem A *always* meets their deadlines
  - R2: Desirable that tasks of subsytem B meet their deadlines, but
- Load A + B exceeds computing power

- Q: What problems will be faced?
- Q: How to structure the system to satisfy these requirements?
- Q: How to evaluate how good the final solution is?
- ...

11

Meaning of flexible

- Resource adaptiveness (CPU/Network).

- Periods of transient overload.
  - Graceful degradation on the quality of the results,
  - *Guaranteed* minimum level of service, ...
  - and on time.

- Adjusting load by not running some invocations
  - Firm tasks skipped/aborted
  - Soft tasks finish late
  - Weakly-hard constraints. A minimum number of invocations guaranteed to finish on time

12

Issues

- Design
  - Design with flexible timing constraints in mind
  - Design using adaptive models of computation
  - Assigning Value/value functions
  - Modeling systems that may miss deadlines
- Architecture
  - Process model
    - functions to task decomposition
  - Scheduling algorithms
    - Hard / optimal usage of slack time
- Analysis techniques
  - Absolute guarantees for hard components
  - Achievable levels of service
  - Probabilistic models
- Implementation models
  - Low overheads

13

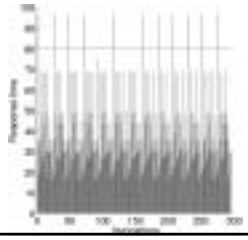
Dimensioning RT systems

- Dimensioning a system for the worst case is very pessimistic (expensive)
- Find the smallest/slowest/cheapest hardware that provides the desired service
- Tasks do not run “always” for their WCET

• Task with D= 80

• Processor A: Ri= 97

• But it may only miss it 11 times every 300 invocations (in the worst case)



14

Applications of flexible RTS

- Two objectives:
  - Maximise resource usage
  - Guaranteed level of service
    - Guaranteed result with minimum cost
- Scopes
  - Real-Time Artificial Intelligence
  - Embedded systems (big numbers)
    - Home appliances (i.e. DVD players, autonomous lawn mowers, autonomous vacuum cleaners,...),
    - X-by-wire,
  - Embedded Systems (limited resources)
    - Satellites,
    - *Pre-selected* hardware
  - Guaranteed minimum levels of QoS.
    - Video decoders for video on demand...
  - Flexible real-time networking. Video server.

15

2 Process Model

16

Process model

- System decomposed as a set of concurrent tasks/transactions
  - Periodic, non-periodic: repeating and isolated
  - Temporal constraints: hard, firm, soft, weakly-hard,
  - Period (T),
  - Deadline (D),
  - Execution time:
    - Worst-case execution time (C)
    - Execution profile
    - Probabilistic values of the above
  - Value (V)

17

On computation times

- Main assumption on most (all?) scheduling work:
  - Ci is known
  - Ci is accurate
  - Tasks run for Ci time units every time
- Significant points in time:
  - Average C
  - p-percentile C
  - Maximum measured C
  - *Maximum C (non computable)*
  - WCET
- Low correlation on WCET execution times
- Ineffective model for next generation RTS
  - Not suitable for data dependent code: AI, vision, ...

18

On computation times (2)

- WCET is "very" pessimistic
  - Implicit code structure
    - EX: long if branch almost never taken
  - Worst case data dependencies
    - EX: maximum data set size
  - Processor features
    - Determining exact impact is an NP-hard problem
    - Impact of CPU features considered "almost" separately
- Implications
  - *Huge* amounts of slack (gain time) may be actually available due to the pessimism of Ci only.
  - Research on WCET
    - Algebraic/symbolic WCET
    - Probabilistic WCET
    - Execution profiles

19

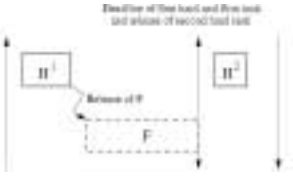
On Value

- Single or multi criteria
  - value is multidimensional <v1,v2,v3,...>
  - But generally captured in a single value (easier)
- Where does value come from?
  - Small ranges of values
  - comparable
    - which operations are allowed
    - precision?
- Traditional approach:
  - Minimise average response time of soft aperiodics (best effort?)
  - Maximise value in total.
- Jorvik approach. Time window:
  - Guaranteed minimum value in every window of time "w".
  - Provide the same metric over a window of time.

20

Adaptive components

- Imprecise computation model:
  - Anytime algorithms
  - Multiple versions
  - ...
- The HFH model
  - HF*H: general model
  - Weakly hard version



21

Other Models

- Generic transaction
  - Start of transaction is hard?
  - Hard, non-hard components
  - Non-hard should be also skippable
  - Fork-Join operators
  - Cumulative errors
- Multiprocessor
  - Scheduling 2 resources
    - CPUs
    - Network (s)
  - Global vs. local scheduling
- Optimal solutions are mostly NP-Hard

22

Summary

- Two objectives
  - Guaranteed minimum behaviour
    - Meet all hard deadlines
    - Guarantee minimum value
  - Use extra time "effectively"
- Structure solution in two levels
  - Level 1: guaranteed level
    - Predictable
    - Based on pessimistic assumptions
    - Pessimistic
  - Level 2: effective use of available resources
    - Due to gain time and slack time
    - Probabilistic / average behaviour
    - Non-optimal solutions

23

3orvik

24