Real-Time Calculus for Scheduling Hard Real-Time Systems

Lothar Thiele, Samarjit Chakraborty and Marting Naedele
ETH Zürich
Presented by Robert Staudinger
University of Salzburg

December 20, 2007

Motivation

Modular Performance Analysis and Design Space Exploration of Distributed Embedded Systems

System complexity is increasing, a few domains:

- Networks of sensors and actuators
- Building automation
- Car communication
- Environmental monitoring
- ...

Motivation

Fundamental problems:

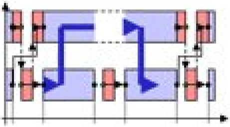
- Handling non-functional and resource constraints
- Design under multiple conflicting criteria
  - Performance vs power consumption
  - Data throughput vs error rate
  - ...
- Trade-off between average performance and predictability

Motivation

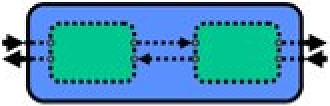
Modular Design Strategies

Predictable Wireless Embedded Systems

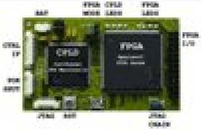
Giotto



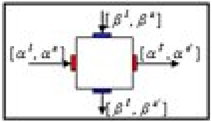
Interface Theories



HW/SW Task Machine



Real-Time Calculus



Motivation

Modular Design Strategies – Vision

Finite resources

- Buffer space
- Energy
- Communication
- Processing power
- Time
- ...

Requirements:

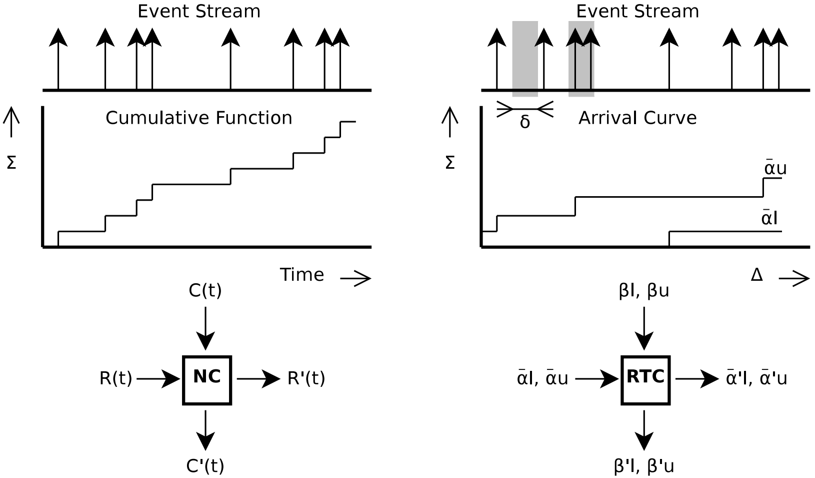
- Distributed systems
- Need for run-time adaptability
- Allow for guarantees

Real-Time Calculus

In a Nutshell

- Deterministic queueing theory
- Hard upper and lower bounds are *always* found
- No insight into average load of a system
- Extension of Network Calculus (R. L. Cruz, 1991)

Real-Time Calculus



Network Calculus **Real-Time Calculus**

Real-Time Calculus

Relationship

Network Calculus

- Time (absolute)
- Cumulative Function
- Event stream $R(t)$
- Resources $C(t)$

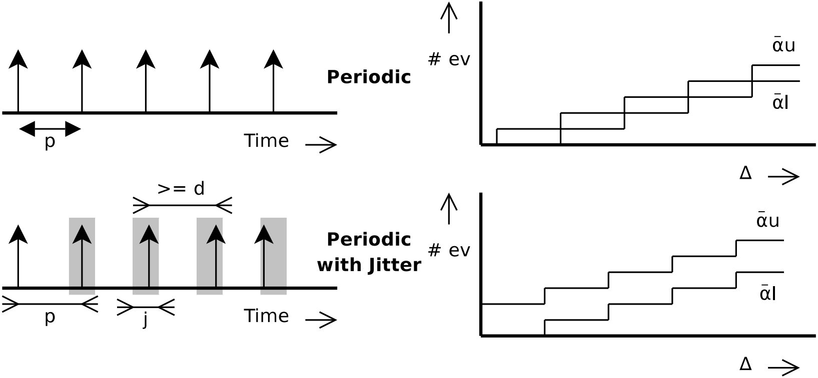
Real-Time Calculus

- Interval size Δ (relative time)
- Arrival Curve
- Event curve α_l, α_u
- Resource curve β_l, β_u

Real-Time Calculus

- Nodes are modeled as pure mathematical functions
- Computation resources: CPU cycles
- Communication resources: Transported number of bits

Real-Time Calculus



Real-Time Calculus

Proposition 1: Basic Model For a processing node characterised by the capacity function $C(t)$ and the incoming requests function $R(t)$ we have $C'(t) = C(t) - R(t)$ and

$$R'(t) = \min_{0 \leq u \leq t} \{R(u) + C(t) - C(u)\}$$

Real-Time Calculus

Bounds

Proposition 2: Request Curve For a given request function R, the minimum request curve α_r can be calculated as

$$\alpha_r = \max_{u \geq 0}\{R(\Delta + u) - R(u)\}$$

Proposition 3: Delivery Curve For a given capacity function C, the maximum delivery curve β can be calculated as

$$\beta = \min_{u \geq 0}\{C(\Delta + u) - C(u)\}$$

Real-Time Calculus

Processing of Bounds

Proposition 4: Remaining Delivery Curve Given request and capacity functions R and C, bounded by the request and delivery curves α_r and β respectively, C' according to Prop. 1 is then bounded by the delivery curve

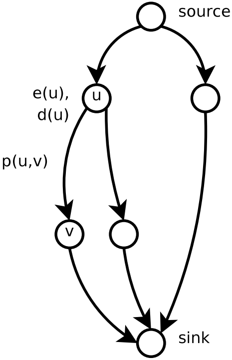
$$\beta'(\Delta) = \max_{0 \leq u \leq \Delta} \{\beta(u) - \alpha_r(u)\}$$

Proposition 5: Delivered Computation Bounds Given request and capacity functions R and C, bounded by the request and delivery curves α_r and β respectively, R' according to Prop. 1 is then bounded by the request curve

$$\alpha'(\Delta) = \max_{u \geq 0} \{\alpha_r(\Delta + u) - \beta(u)\}$$

Real-Time Calculus

Calculation of Bounds



Task Graph with Conditional Branches

c.f. S. Baruah, 1998

- Task graph: DAG with unique source and sync vertex
- Vertex u is subtask
- Associated with pair $(e(u), d(u))$
- Requires $e(u)$ time, finishes $d(u)$ after being triggered
- Directed edge (u, v) is control flow
- $p(u, v)$ is min time before v can be triggered

Real-Time Calculus

Calculation of Bounds

$L^{j+1} = (e(j+1), d(j+1))$
for each edge $(k, j+1)$ **do**
  for each tuple $(f, \Delta) \in L^k$ **do**
    $L^{j+1} \cup$
      $\{(f + e(j+1), \Delta + p(k, j+1)$
        $+ d(j+1) - d(k))\}$
$L(j+1) = L^{j+1} \cup L(j)$
Reduce sets $L(j+1)$ and L^{j+1}

- Incremental algorithm
- L^i set of tuples (f, Δ)
  sequences of executions
  i is last subtask
- $L(i)$ list of tuples
  subtasks up to, including i

Reduce removed tuples with less restrictive bounds

Real-Time Calculus

Theorem 1 Given a task graph of n vertices, $\alpha_d(\Delta)$ can be computed in $O(n^3)$ time if the execution times of the subtasks are equal, and in general it can be computed in pseudo-polynomial time.

Real-Time Calculus

Proposition 6: Schedulability Test The capacity offered by a task T_i can be described by the remaining delivery curve β' according to Prop. 4 where $\alpha_r = \sum_{j=1}^{i-1}$. Task Ti meets all its deadlines if

$$(\forall \Delta)(\beta'(\Delta) \geq \alpha_d^i(\Delta))$$

Conclusion

- Terse but concise paper
- Key contribution is algorithm in polynomial time

- Critique: request/arrival model and task graph model could be integrated better