Real-Time Workload Models with Efficient Analysis
Advanced Course, 3 Lectures, September 2014

Martin Stigge

Uppsala University, Sweden
Fahrplan

1 DRT Tasks in the Model Hierarchy
   - Liu and Layland and Sporadic Tasks
   - Frames and Branching
   - The Digraph Real-Time (DRT) Task Model
   - Adaptive Variable-Rate (AVR) Tasks

2 Feasibility Analysis of DRT
   - Feasibility Theorem
   - Demand Pairs
   - Test Termination
   - Evaluation

3 Static Priority Schedulability Analysis of DRT
   - Response-Time Analysis
   - Request Functions
   - Refinement Algorithm
   - Evaluation
Fahrplan

1. DRT Tasks in the Model Hierarchy
   - Liu and Layland and Sporadic Tasks
   - Frames and Branching
   - The Digraph Real-Time (DRT) Task Model
   - Adaptive Variable-Rate (AVR) Tasks

2. Feasibility Analysis of DRT
   - Feasibility Theorem
   - Demand Pairs
   - Test Termination
   - Evaluation

3. Static Priority Schedulability Analysis of DRT
   - Response-Time Analysis
   - Request Functions
   - Refinement Algorithm
   - Evaluation
Problem Overview

Workload Model

Task A

Task B

Task C

Scheduler Model

EDF/Static Prio/...

Feasible? Schedulable? Response times?

Our Setting:
• DRT tasks
• Static Priorities
• Precise Test
The Liu and Layland (L&L) Task Model
(Liu and Layland, 1973)

- Tasks are *periodic*
  - Job WCET $e$
  - Period $p$ (implicit deadline)

- Advantages: Well-known model; *efficient* schedulability tests
- Disadvantage: Very *limited expressiveness*
A Hierarchy of Models

Schedulability Analysis

difficult

efficient

Expressiveness

high

low

Liu & Layland

\((e, d = p)\)
The Sporadic Task Model
(Mok, 1983)

- Deadlines ≠ periods; releases \textit{sporadic}
- Each tasks defined by:
  - Job WCET $e$
  - Relative deadline $d$
  - Minimum inter-release delay $p$

$$(e, d, p) \sim$$
The General Multiframe (GMF) Task Model

- Behavior is not always periodic

  ![Diagram showing frames 0 to 3 with execution times and deadlines]

- Task is split into *frames*, each with own
  - Execution time $e^{(i)}$
  - Inter-release separation $p^{(i)}$
  - Deadline $d^{(i)}$ for the job
The General Multiframe (GMF) Task Model

- Behavior is not always periodic

- Task is split into *frames*, each with own
  - Execution time $e^{(j)}$
  - Inter-release separation $p^{(j)}$
  - Deadline $d^{(j)}$ for the job
The General Multiframe (GMF) Task Model (cont.)
(Baruah et al., 1999)

- Tasks *cycle* through job types
  - Vector for WCET \((e^{(1)}, \ldots, e^{(n)})\)
  - Vector for deadlines \((d^{(1)}, \ldots, d^{(n)})\)
  - Vector for minimum inter-release delays \((p^{(1)}, \ldots, p^{(n)})\)
A Hierarchy of Models

- **generalized multiframe (GMF)**: \((e_i, d_i, p_i)\)
- **multiframe**: \((e_i, d = p)\)
- **sporadic**: \((e, d, p)\)
- **Liu & Layland**: \((e, d = p)\)

**Schedulability Analysis**
- **difficult**
- **efficient**

**Expressiveness**
- **high**
- **low**
The Non-Cyclic GMF Task Model
(Moyo et al., 2010)

- Frame order unknown \( a \text{ priori} \)
- Syntax similar to GMF:
  - Vector for WCET \((e^{(1)}, \ldots, e^{(n)})\)
  - Vector for deadlines \((d^{(1)}, \ldots, d^{(n)})\)
  - Vector for minimum inter-release delays \((p^{(1)}, \ldots, p^{(n)})\)
The Recurring Branching (RB) Task Model
(Baruah, 1998)

- Introduces branching structures
- Tree for tasks
  - Vertices $J$: jobs to be released (with WCET and deadline)
  - Edges $(J_i, J_j)$: minimum inter-release delays $p(J_i, J_j)$
  - General period parameter $P$

\[ P = 57 \]
The Recurring Branching (RB) Task Model
(Baruah, 1998)

- Introduces \textit{branching} structures
- \textit{Tree} for tasks
  - Vertices $J$: jobs to be released (with WCET and deadline)
  - Edges $(J_i, J_j)$: minimum inter-release delays $p(J_i, J_j)$
  - General period parameter $P$

\[
\begin{align*}
J_1 & \rightarrow J_2 \\
J_2 & \rightarrow J_3 \\
J_3 & \rightarrow J_5 \\
J_2 & \rightarrow J_4 \\
J_4 & \rightarrow J_6 \\
J_5 & \\
\end{align*}
\]

Period $P = 57$
The Recurring Real-Time (RRT) Task Model
(Baruah, 1998)

- Compact Branching representation
- *Directed acyclic graph* (DAG) for tasks
  - Vertices $J$: jobs to be released (with WCET and deadline)
  - Edges $(J_i, J_j)$: minimum inter-release delays $p(J_i, J_j)$
  - General period parameter $P$

![Diagram of DAG with jobs $J_1$ to $J_5$ and arrows indicating dependencies. The period $P = 57$.]
A Hierarchy of Models

- **sporadic** \((e, d, p)\)
- **multiframe** \((e_i, d = p)\)
- **generalized multiframe (GMF)** \((e_i, d_i, p_i)\)
- **recurring branching (RB)** \((\text{tree}, p)\)
- **recurring RT (RRT)** \((\text{DAG}, p)\)
- **non-cyclic RRT** \((\text{DAG}, p_i)\)
- **non-cyclic GMF** \((\text{order arbitrary})\)
- **Liu & Layland** \((e, d = p)\)

**Schedulability Analysis**

- **difficult**
- **efficient**

**Expressiveness**

- **high**
- **low**
Restrictions of RRT

- Tasks are still *recurrent*
  - Always revisit source $J_1$
  - *No cycles allowed!*

Consequences:

- *No local loops*

- Not *compositional* (for modes etc.)
Restrictions of RRT

• Tasks are still *recurrent*
  • Always revisit source $J_1$
  • *No cycles allowed!*

• Consequences:
  • *No local loops*

• Not *compositional* (for modes etc.)
Restrictions of RRT

- Tasks are still *recurrent*
  - Always revisit source $J_1$
  - *No cycles allowed!*

- Consequences:
  - No *local loops*

- Not *compositional* (for modes etc.)
The Digraph Real-Time (DRT) Task Model
(S. et al., 2011)

- Generalizes periodic, sporadic, GMF, RRT, ...
- Directed graph for each task
  - Vertices $v$: jobs to be released (with WCET and deadline)
  - Edges $(u, v)$: minimum inter-release delays $p(u, v)$
DRT: Semantics

Path $\pi = (v_4)$
Path $\pi = (v_4, v_2)$
Path $\pi = (v_4, v_2, v_3)$
Path $\pi = (v_4)$
DRT: Semantics

Path $\pi = (v_4, v_2)$
Path $\pi = (v_4, v_2, v_3)$
A Hierarchy of Models

- **Schedulability Analysis**
  - **difficult**
  - **efficient**

- **Expressiveness**
  - **low**
  - **high**

1. **Liu & Layland**
   - $(e, d = p)$

2. **Sporadic**
   - $(e, d, p)$

3. **Multiframe**
   - $(e_i, d = p)$

4. **Generalized Multiframe (GMF)**
   - $(e_i, d_i, p_i)$

5. **Recurring Branching (RB)**
   - $(\text{tree, } p)$

6. **Recurring RT (RRT)**
   - $(\text{DAG, } p)$

7. **Non-cyclic RRT**
   - $(\text{DAG, } p_i)$

8. **Non-cyclic GMF**
   - $(\text{order arbitrary})$

9. **Digraph (DRT)**
   - $(\text{arbitrary graph})$

The diagram illustrates the hierarchy and expressiveness of workload models, with Liu & Layland at the top and sporadic at the bottom, indicating a spectrum from efficient to difficult analysis.
A Hierarchy of Models

- **Schedulability Analysis**
  - **efficient**
  - **difficult**
  - Strongly (co)NP-hard
  - Pseudo-Polynomial

- **Expressiveness**
  - low
  - high

- **Liu & Layland**
  - recurring RT (RRT) (DAG, \( p \))
  - recurring branching (RB) (tree, \( p \))
  - generalized multiframe (GMF) \((e_i, d_i, p_i)\)
  - multiframe \((e_i, d = p)\)
  - sporadic \((e, d, p)\)
  - non-cyclic GMF (order arbitrary)
  - non-cyclic RRT (DAG, \( p_i \))
  - Digraph (DRT) (arbitrary graph)

- **Efficiency & Expressiveness trade-off**
Extended DRT (EDRT)

- Extends DRT with *global delay constraints*
- Directed graph for each task
  - Vertices $v$: jobs to be released (with WCET and deadline)
  - Edges $(u, v)$: minimum inter-release delays $p(u, v)$
  - $k$ global constraints $(u, v, \gamma)$

![Directed graph diagram]

**Theorem (S. et al., 2011)**

For $k$-EDRT task systems with bounded utilization, feasibility is

1. **decidable in pseudo-polynomial time** if $k$ is constant, and
2. **strongly coNP-hard** in general.
Extended DRT (EDRT)

- Extends DRT with *global delay constraints*
- Directed graph for each task
  - Vertices $v$: jobs to be released (with WCET and deadline)
  - Edges $(u, v)$: minimum inter-release delays $p(u, v)$
  - $k$ global constraints $(u, v, \gamma)$

![Graph Diagram]

**Theorem (S. et al., 2011)**

For $k$-EDRT task systems with bounded utilization, feasibility is

1. **decidable in pseudo-polynomial time** if $k$ is constant, and
2. **strongly coNP-hard** in general.
A Hierarchy of Models

Schedulability Analysis

- **efficient**
  - Liu & Layland
    - sporadic
    - (e, d = p)
  - generalized multiframe (GMF)
    - (e_i, d_i, p_i)
    - recurring branching (RB)
      - (tree, p)
      - recurring RT (RRT)
        - (DAG, p)
  - multiframe
    - (e_i, d = p)
  - non-cyclic GMF
  - digraph (DRT)
  - non-cyclic RRT
    - (DAG, p_i)
  - k-EDRT
    - (k constraints)
  - Extended DRT (EDRT)
    - (constraints)

Expressiveness

- **difficult**
  - Strongly (co)NP-hard
  - Pseudo-Polynomial
- **low**
DRT Examples

\[ (e, d) \]

Sporadic Task

\[ \text{Sporadic Task with 2 modes} \]

\[ v_1 \quad v_2 \quad v_3 \quad v_4 \quad v_5 \]

Branching Task

\[ \text{GMF Task} \]
DRT Examples

\( \langle e, d \rangle \)

Sporadic Task

\( \langle e, p \rangle \)

Sporadic Task (implicit deadline)
DRT Examples

Sporadic Task

\langle e, d \rangle

\langle e, p \rangle

Sporadic Task (implicit deadline)

Sporadic Task with 2 modes

\langle e_1, d_1 \rangle

\langle e_2, p_2 \rangle
DRT Examples

Sporadic Task

\[ \langle e, d \rangle \]

Sporadic Task (implicit deadline)

\[ \langle e, p \rangle \]

Sporadic Task with 2 modes

\[ \langle e_1, d_1 \rangle \]

GMF Task

\[ \langle e_2, p_2 \rangle \]
DRT Examples

Sporadic Task

\[ (e, d) \]

\[ (e, p) \]

Sporadic Task (implicit deadline)

\[ (e_1, d_1) \]

\[ (e_2, p_2) \]

Sporadic Task with 2 modes

GMF Task

Branching Task
Adaptive Variable-Rate (AVR) Tasks

- *Rate-Adaptive Tasks* (Buttazzo et al., DATE 2014)
- *Variable Rate-dependent Behaviour (VRB)* (Davis et al., RTAS 2014)
- *Adaptive Variable-Rate (AVR) Tasks* (Biondi et al., ECRTS 2014)
- ...
AVR Tasks: Execution Modes

(Biondi, ECRTS 2014)
AVR Business

(This morning in Pisa)
Fahrplan

1 DRT Tasks in the Model Hierarchy
   - Liu and Layland and Sporadic Tasks
   - Frames and Branching
   - The Digraph Real-Time (DRT) Task Model
   - Adaptive Variable-Rate (AVR) Tasks

2 Feasibility Analysis of DRT
   - Feasibility Theorem
   - Demand Pairs
   - Test Termination
   - Evaluation

3 Static Priority Schedulability Analysis of DRT
   - Response-Time Analysis
   - Request Functions
   - Refinement Algorithm
   - Evaluation
Fahrplan

1 DRT Tasks in the Model Hierarchy
   - Liu and Layland and Sporadic Tasks
   - Frames and Branching
   - The Digraph Real-Time (DRT) Task Model
   - Adaptive Variable-Rate (AVR) Tasks

2 Feasibility Analysis of DRT
   - Feasibility Theorem
   - Demand Pairs
   - Test Termination
   - Evaluation

3 Static Priority Schedulability Analysis of DRT
   - Response-Time Analysis
   - Request Functions
   - Refinement Algorithm
   - Evaluation
Theorem

For a task set \( \tau \), the following three properties are equivalent.

1. Task set \( \tau \) is feasible.
2. Task set \( \tau \) is EDF schedulable.
3. The following condition holds: \( \forall t \geq 0 : \sum_{T \in \tau} dbf_T(t) \leq t \)

What is \( dbf_T(t) \)? How to compute it?
Feasibility Theorem

Theorem

For a task set $\tau$, the following three properties are equivalent.

1. Task set $\tau$ is feasible.
2. Task set $\tau$ is EDF schedulable.
3. The following condition holds: $\forall t \geq 0 : \sum_{T \in \tau} \text{dbf}_T(t) \leq t$

What is $\text{dbf}_T(t)$? How to compute it?
**Demand-Bound Functions**

\[ \text{dbf}_T(t): \text{Maximal demand in any window of size } t \]

Demand: \( 5 + 1 + 3 = 9 \)
Feasibility Test

Theorem (Feasibility Theorem)

For a task set $\tau$, the following three properties are equivalent.

1. Task set $\tau$ is feasible.
2. Task set $\tau$ is EDF schedulable.
3. The following condition holds: $\forall t \geq 0 : \sum_{T \in \tau} dbf_T(t) \leq t$

\[ \sum_{T \in \tau} dbf_T(t) \]

\[ \rightarrow t \]
Feasibility Test

Theorem (Feasibility Theorem)

For a task set $\tau$, the following three properties are equivalent.

1. Task set $\tau$ is feasible.
2. Task set $\tau$ is EDF schedulable.
3. The following condition holds: $\forall t \geq 0 : \sum_{T \in \tau} dbf_T(t) \leq t$

How to calculate $dbf_T(t)$?

How to check existence of violating $t$?
Feasibility Test

Theorem (Feasibility Theorem)

For a task set $\tau$, the following three properties are equivalent.

1. Task set $\tau$ is feasible.
2. Task set $\tau$ is EDF schedulable.
3. The following condition holds: $\forall t \geq 0 : \sum_{T \in \tau} \text{dbf}_T(t) \leq t$

How to calculate $\text{dbf}_T(t)$?

How to check existence of violating $t$?
Feasibility Test

Theorem (Feasibility Theorem)

For a task set $\tau$, the following three properties are equivalent.

1. Task set $\tau$ is feasible.
2. Task set $\tau$ is EDF schedulable.
3. The following condition holds: $\forall t \geq 0 : \sum_{T \in \tau} dbf_T(t) \leq t$

How to calculate $dbf_T(t)$?

How to check existence of violating $t$?
dbf for Sporadic Tasks

Sporadic task $T = (e, d, p)$

$$\sum \text{dbf}_T(t)$$

![Graph showing dbf for Sporadic Tasks]

$T = (e, d, p)$

$e, d, p$
Sporadic task $T = (e, d, p)$

$$\sum \text{dbf}_T(t) = \max \left\{ 0, \left\lfloor \frac{t - d}{p} + 1 \right\rfloor \cdot e \right\}$$
dbf for DRT: From $G(T)$ to $dbf_T$

Path Abstraction: Demand Pair

$\langle 2, 5 \rangle \\ 10 \\ 11$

$\langle 1, 8 \rangle \\ 20 \\ 15$

$\langle 1, 5 \rangle \\ 20 \\ 10$

$\langle 3, 8 \rangle \\ 20 \\ 20$

$dbf_T(t)$

Martin Stigge

Workload Models + Analysis
dbf for DRT: From $G(T)$ to $\text{dbf}_T$

Path Abstraction: Demand Pair

\[ \text{dbf}_T(t) \]

\[ \langle 9, 43 \rangle \]

Martin Stigge
Workload Models + Analysis

29
dbf for DRT: From $G(T)$ to $\text{dbf}_T$

Path Abstraction: Demand Pair

$\text{dbf}_T(t)$
dbf for DRT: From $G(T)$ to $dbf_T$

Path Abstraction: Demand Pair

$\langle 2, 5 \rangle$
$\langle 1, 5 \rangle$
$\langle 1, 8 \rangle$
$\langle 3, 8 \rangle$
$\langle 5, 10 \rangle$
$\langle 9, 43 \rangle$
dbf for DRT: From $G(T)$ to $\text{dbf}_T$

Path Abstraction: Demand Pair

$\langle 2, 5 \rangle$ $v_1$ $\langle 1, 8 \rangle$ $v_2$ $\langle 3, 8 \rangle$ $v_3$

$\langle 1, 5 \rangle$ $v_5$ $\langle 5, 10 \rangle$ $v_4$

$\langle 9, 43 \rangle$ $\text{dbf}_T(t)$
Demand Pairs

Formally:
- Given path $\pi = (\pi_0, \ldots, \pi_l)$
- **Execution demand**: $e(\pi) := \sum_{i=0}^{l} e(\pi_i)$
- **Deadline**: $d(\pi) := \sum_{i=0}^{l-1} p(\pi_i, \pi_{i+1}) + d(\pi_l)$
- $\langle e(\pi), d(\pi) \rangle$ is a demand pair for $\pi$

$$\text{dbf}_T(t) = \max \{ e \mid \langle e, d \rangle \text{ demand pair with } d \leq t \}$$

How to compute all demand pairs?
- Enumerate paths: Too expensive! (Exponential..)
- Better: Iteration using abstraction
**Demand Pairs**

Formally:

- **Given path** \( \pi = (\pi_0, \ldots, \pi_l) \)
- **Execution demand:** \( e(\pi) := \sum_{i=0}^{l} e(\pi_i) \)
- **Deadline:** \( d(\pi) := \sum_{i=0}^{l-1} p(\pi_i, \pi_{i+1}) + d(\pi_l) \)
- \( \langle e(\pi), d(\pi) \rangle \) is a **demand pair** for \( \pi \)

\[
\text{dbf}_T(t) = \max \{ e \mid \langle e, d \rangle \text{ demand pair with } d \leq t \}
\]

How to compute all demand pairs?

- **Enumerate paths**: Too expensive! (Exponential..)
- **Better**: Iteration using **abstraction**
Demand Triples

- Idea: Start with 0-paths (one vertex), extend stepwise
- We need: Abstraction which
  1. allows to extend paths,
  2. contains demand pair information,
  3. without visiting/storing all paths
- Idea: *Demand triples*
  - Execution demand $e(\pi)$
  - Deadline $d(\pi)$
  - Last vertex $\pi_l$
- Demand triple $\langle e(\pi), d(\pi), \pi_l \rangle$ is another abstraction!
Demand Triples

• Idea: Start with 0-paths (one vertex), extend stepwise
• We need: Abstraction which
  1. allows to extend paths,
  2. contains demand pair information,
  3. without visiting/storing all paths
• Idea: *Demand triples*
  • Execution demand $e(\pi)$
  • Deadline $d(\pi)$
  • Last vertex $\pi_l$
• Demand triple $\langle e(\pi), d(\pi), \pi_l \rangle$ is another abstraction!

![Diagram of a graph with vertices and edges labeled with values and demand triple notations.](image-url)
Demand Triples

- Idea: Start with 0-paths (one vertex), extend stepwise
- We need: Abstraction which
  1. allows to *extend* paths,
  2. contains demand pair information,
  3. without visiting/storing all paths
- Idea: *Demand triples*
  - Execution demand $e(\pi)$
  - Deadline $d(\pi)$
  - Last vertex $\pi_l$
- Demand triple $\langle e(\pi), d(\pi), \pi_l \rangle$ is another abstraction!
Demand Triples

- Idea: Start with 0-paths (one vertex), extend stepwise
- We need: Abstraction which
  1. allows to extend paths,
  2. contains demand pair information,
  3. without visiting/storing all paths
- Idea: Demand triples
  - Execution demand $e(\pi)$
  - Deadline $d(\pi)$
  - Last vertex $\pi_l$
- Demand triple $\langle e(\pi), d(\pi), \pi_l \rangle$ is another abstraction!

```
10 20 20 15
(2,5) <---- (1,8) <---- (1,5) ----< (3,8)
   11

10 20 20
(1,5) ----< (1,8) ----< (3,8)
   15

20
(5,10)
   10

Path ($v_4$) 
$\leadsto$ $\langle 5, 10, v_4 \rangle$

Path ($v_4, v_2$) 
$\leadsto$ $\langle 6, 28, v_2 \rangle$

Path ($v_4, v_2, v_3$) 
$\leadsto$ $\langle 9, 43, v_3 \rangle$
```
Iterative Procedure

- Create all demand triples up to some $D$:
  1. Start with all 0-paths, i.e., $\langle e(v), d(v), v \rangle$ for all vertices $v$
  2. Pick some stored demand triple $\langle e, d, u \rangle$
  3. Create new demand triple:
     - Choose successor vertex $v$ of $u$
     - $e' = e + e(v)$
     - $d' = d - d(u) + p(u, v) + d(v)$
     - $\langle e', d', v \rangle$ is new demand triple!
  4. Store $\langle e', d', v \rangle$ if
     - not stored yet, and
     - $d' \leq D$
  5. Repeat from 2 until no change

- Efficient procedure!
  - Note: Actual paths never stored
  - Optimizations: Discard non-critical triples along the way

- Exercise: What’s $\text{dbf}_{\tau_i}(26)$ for graph on previous slide?
Example

\[
dbf_{\tau}(t)
\]

Optimization: Discard dominated triples \((e, d, v) \succeq (e', d', v') \iff e \geq e' \land d \leq d'\)

Martin Stigge

Workload Models + Analysis
Example

Optimization: Discard dominated triples \((e, d, v) \succeq (e', d', v')\) ⇔ \(e \geq e' \land d \leq d'\)

\[\begin{align*}
\langle v_1, 20 \rangle & \rightarrow \langle v_2, 11 \rangle \\
\langle v_2, 11 \rangle & \rightarrow \langle v_3, 15 \rangle \\
\langle v_3, 15 \rangle & \rightarrow \langle v_4, 20 \rangle \\
\langle v_4, 20 \rangle & \rightarrow \langle v_5, 10 \rangle \\
\langle v_5, 10 \rangle & \rightarrow \langle v_1, 20 \rangle
\end{align*}\]
Optimization: Discard dominated triples \((e, d, v) \succeq (e', d', v') \iff e \geq e' \land d \leq d'

\[ \text{dbf}_T(t) \]

\[ \begin{align*}
\langle 1, 5 \rangle & \rightarrow \langle 2, 5 \rangle \\
\langle 3, 8 \rangle & \rightarrow \langle 1, 8 \rangle \\
\langle 5, 10 \rangle & \rightarrow \langle 1, 5 \rangle \\
\end{align*} \]
Example

\[ \text{dbf}_T(t) \]

\[ \langle 6, 20, v_4 \rangle \quad \langle 6, 30, v_4 \rangle \]

Optimization: Discard dominated triples \((e, d, v) \succeq (e', d', v') \iff e \geq e' \land d \leq d'\).
Example

Optimization: Discard *dominated* triples

\((e, d, v) \succ (e', d', v) \iff e \geq e' \land d \leq d'\)
Example

Optimization: Discard *dominated* triples

\((e, d, v) \succ (e', d', v) \iff e \geq e' \land d \leq d'\)
Example

Optimization: Discard *dominated* triples

\[(e, d, v) \succeq (e', d', v) \iff e \geq e' \land d \leq d'\]
Example

Optimization: Discard *dominated* triples

\[(e, d, v) \succ (e', d', v) \iff e \geq e' \land d \leq d'\]
Example

Optimization: Discard *dominated* triples

\[(e, d, v) \succeq (e', d', v) \iff e \geq e' \land d \leq d'\]
Example

Optimization: Discard *dominated* triples

\[(e, d, v) \succeq (e', d', v) \iff e \geq e' \land d \leq d'\]
Example

Optimization: Discard *dominated* triples

\[(e, d, v) \succ (e', d', v) \iff e \geq e' \land d \leq d'\]
Example

Optimization: Discard \textit{dominated} triples

\[(e, d, v) \succ (e', d', v) \iff e \geq e' \land d \leq d'\]
Feasibility Test Revisited

Theorem (Feasibility Theorem)

For a task set $\tau$, the following three properties are equivalent.

1. Task set $\tau$ is feasible.
2. Task set $\tau$ is EDF schedulable.
3. The following condition holds: $\forall t \geq 0 : \sum_{T \in \tau} \text{dbf}_T(t) \leq t$

How to calculate $\text{dbf}_T(t)$?
How to check existence of violating $t$?
Feasibility Test Revisited

Theorem (Feasibility Theorem)

For a task set $\tau$, the following three properties are equivalent.

1. Task set $\tau$ is feasible.
2. Task set $\tau$ is EDF schedulable.
3. The following condition holds: $\forall t \geq 0 : \sum_{T \in \tau} dbf_T(t) \leq t$

How to calculate $dbf_T(t)$?

How to check existence of violating $t$?
Calculating the Bound

\[ \sum \text{dbf}_T(t) \]

- Linear bound for \( \text{dbf}(t) \)
- Slope: Less than 1
- Intersection with \( t \) gives bound \( D \)
- Check only up to \( D \)

\( \text{dbf}(t) \leq t \cdot U(\tau) + e^{\text{sum}} \)

“Most dense” cycle
Calculating the Bound

\[ \sum \text{dbf}_T(t) \]

- Linear bound for \( \text{dbf}(t) \)
  - Slope: Less than 1
- Intersection with \( t \) gives bound \( D \)
- Check only up to \( D \)

\[ \text{dbf}(t) \leq t \cdot U(\tau) + e^{\text{sum}} \]
Calculating the Bound

\[ \sum \text{dbf}_T(t) \]

- Linear bound for \( \text{dbf}(t) \)
  - Slope: Less than 1
  - Intersection with \( t \) gives bound \( D \)
  - Check only up to \( D \)

\[ \text{dbf}(t) \leq t \cdot U(\tau) + e^{sum} \]
**Theorem (S. et al., 2011)**

*For DRT task systems $\tau$ with a utilization bounded by any $c < 1$, feasibility can be decided in pseudo-polynomial time.*

Pseudo-polynomial time $=$ Tractable/efficient
Evaluation: Runtime vs. Utilization

Setting:
- Randomly generated task sets
- 1-30 tasks, 5-10 vertices per task, branching degree 1-3, ...
Fahrplan

1 DRT Tasks in the Model Hierarchy
   - Liu and Layland and Sporadic Tasks
   - Frames and Branching
   - The Digraph Real-Time (DRT) Task Model
   - Adaptive Variable-Rate (AVR) Tasks

2 Feasibility Analysis of DRT
   - Feasibility Theorem
   - Demand Pairs
   - Test Termination
   - Evaluation

3 Static Priority Schedulability Analysis of DRT
   - Response-Time Analysis
   - Request Functions
   - Refinement Algorithm
   - Evaluation
Fahrplan

1. DRT Tasks in the Model Hierarchy
   - Liu and Layland and Sporadic Tasks
   - Frames and Branching
   - The Digraph Real-Time (DRT) Task Model
   - Adaptive Variable-Rate (AVR) Tasks

2. Feasibility Analysis of DRT
   - Feasibility Theorem
   - Demand Pairs
   - Test Termination
   - Evaluation

3. Static Priority Schedulability Analysis of DRT
   - Response-Time Analysis
   - Request Functions
   - Refinement Algorithm
   - Evaluation
Complexity of Schedulability Tests

- Pseudo-polynomial schedulability tests possible?

<table>
<thead>
<tr>
<th></th>
<th>EDF</th>
<th>SP</th>
</tr>
</thead>
<tbody>
<tr>
<td>L&amp;L</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>GMF</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>DRT</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>EDRT</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Flawed!

Replaced by:
Theorem (S. et al., 2012)

For GMF task systems, the schedulability problem for static priority schedulers is strongly coNP-hard.
Complexity of Schedulability Tests

- Pseudo-polynomial schedulability tests possible?

<table>
<thead>
<tr>
<th></th>
<th>EDF</th>
<th>SP</th>
</tr>
</thead>
<tbody>
<tr>
<td>L&amp;L</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GMF</td>
<td>Yes</td>
<td>Yes*</td>
</tr>
<tr>
<td>DRT</td>
<td>Yes</td>
<td>?</td>
</tr>
<tr>
<td>EDRT</td>
<td>No</td>
<td>?</td>
</tr>
</tbody>
</table>

- $^*$ = Takada & Sakamura, 1997

For GMF task systems, the schedulability problem for static priority schedulers is strongly coNP-hard.
Complexity of Schedulability Tests

- Pseudo-polynomial schedulability tests possible?

<table>
<thead>
<tr>
<th></th>
<th>EDF</th>
<th>SP</th>
</tr>
</thead>
<tbody>
<tr>
<td>L&amp;L</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GMF</td>
<td>Yes</td>
<td>Yes* No!</td>
</tr>
<tr>
<td>DRT</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>EDRT</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

* = Takada & Sakamura, 1997

- Flawed!

- Replaced by:

**Theorem (S. et al., 2012)**

For GMF task systems, the schedulability problem for static priority schedulers is strongly coNP-hard.
Hardness Result: Proof Sketch

3-PARTITION instance \( I \):

\[
\begin{align*}
\text{m bins} & \quad \text{3m items} \\
\end{align*}
\]

Possible to (exactly) fit all items? (strongly NP-hard)

Reduction to GMF schedulability:

Thus: \( \tau(I) \) unsched. \( \iff \) \( I \in \text{3-PARTITION} \)
Hardness Result: Proof Sketch

**3-PARTITION instance I:**

Possible to (exactly) fit all items? (strongly NP-hard)

**Reduction to GMF schedulability:**

Thus: $\tau(I)$ unsched. $\iff I \in 3$-PARTITION
Model Hierarchy Revisited

EDF

L&L

sporadic

coNP-hard

EDRT

k-EDRT

DRT

RRT

RB

GMF

ncGMF

MC

EDF

SP

L&L

sporadic


EDRT

k-EDRT

DRT

RRT

RB

GMF

ncGMF


coNP-hard

Martin Stigge

Workload Models + Analysis
Response-Time Analysis (RTA)

Use RTA for SP Schedulability Analysis.

![Response time diagram]

Standard RTA for static priorities + periodic/sporadic tasks:

\[ R_j = C_j + \sum_{i \in hp(j)} \left\lceil \frac{R_j}{T_i} \right\rceil C_i \]
Problem: Path Combinations

Response time

Response time

Combinatorial Explosion!
Problem: Path Combinations

Combinatorial Explosion!
Intuition: No Task-Local Worst Cases

Which path is worse: \((v_1, v_2)\) or \((v_2, v_1)\)?
Intuition: No Task-Local Worst Cases

Task $T_{high}$:

Which path is worse: $(v_1, v_2)$ or $(v_2, v_1)$?

It depends! $T_1: \langle 2, 7 \rangle$ versus $T_2: \langle 4, 10 \rangle$
Intuition: No Task-Local Worst Cases

Task $T_{\text{high}}$:

Which path is worse: $(v_1, v_2)$ or $(v_2, v_1)$?

It depends! $T_1: (2, 7)$ versus $T_2: (4, 10)$

$v_1$ $v_2$

$T_{\text{high}}$: 

$T_1$: 

$T_{\text{high}}$: 

$T_1$: 

Martin Stigge
Workload Models + Analysis
44
Intuition: No Task-Local Worst Cases

Task $T_{\text{high}}$: $\langle v_1, v_2 \rangle$ versus $\langle v_2, v_1 \rangle$

Which path is worse: $(v_1, v_2)$ or $(v_2, v_1)$?

It depends! $T_1: \langle 2, 7 \rangle$ versus $T_2: \langle 4, 10 \rangle$
Request Functions

\[
rf(t) := \max \{ e(\pi') \mid \pi' \text{ is prefix of } \pi \text{ and } p(\pi') < t \}
\]
Request Functions (cont.)

Useful for deriving response time:

\[
R_{SP}(v, \bar{r}f) = \min \left\{ t \geq 0 \mid e(v) + \sum_{T'>T} rf(T')(t) \leq t \right\}
\]

\[
R_{SP}(v) = \max_{\bar{r}f \in RF(\tau)} R_{SP}(v, \bar{r}f)
\]
Request Functions (cont.)

Useful for deriving response time:

\[ R_{SP}(v, \bar{r}) = \min \left\{ t \geq 0 \mid e(v) + \sum_{T' > T} rf(T')(t) \leq t \right\} \]

\[ R_{SP}(v) = \max_{\bar{r} \in RF(\tau)} R_{SP}(v, \bar{r}) \]

Combinatorial Explosion?!
Abstract Request Functions

\[ rf(t) \]

\[ rf(v_4, v_2, v_3) \]

\[ rf(v_5, v_2, v_3) \]
Abstract Request Functions

\[ rf(t) \]

- \( rf(v_4, v_2, v_3) \)
- \( rf(v_5, v_4, v_2) \)
Abstract Request Functions

\[
\begin{align*}
&v_1 \langle 2, 5 \rangle \\
&v_2 \langle 3, 8 \rangle \\
&v_3 \langle 1, 8 \rangle \\
&v_4 \langle 5, 10 \rangle \\
&v_5 \langle 1, 5 \rangle
\end{align*}
\]

\[rf(t)\]

\[arf\]

\[rf(v_4, v_2, v_3)\]

\[rf(v_5, v_4, v_2)\]
Define an *abstraction tree* per task:

- Leaves are concrete *rf*
- Each node: maximum function of child nodes
- Root is maximum of all *rf*
Define an *abstraction tree* per task:

- Leaves are concrete *rf*
- Each node: maximum function of child nodes
- Root is maximum of *all rf*
Define an *abstraction tree* per task:

- Leaves are concrete $rf$
- Each node: maximum function of child nodes
- Root is maximum of all $rf$
Define an *abstraction tree* per task:

- Leaves are concrete *rf*
- Each node: maximum function of child nodes
- Root is maximum of *all* *rf*
Define an abstraction tree per task:

- Leaves are concrete rf
- Each node: maximum function of child nodes
- Root is maximum of all rf
Define an *abstraction tree* per task:

- Leaves are concrete *rf*
- Each node: maximum function of child nodes
- Root is maximum of *all rf*

Allows stepwise refinement!
Refinement Algorithm

Tuple: \( \bar{rf} = (rf(T_1), rf(T_2), rf(T_3)) \)
Refinement Algorithm

Tuple: \( \bar{r}f = (r(T_1), r(T_2), r(T_3)) \)

Response time: \( R_{SP}(v, \bar{r}f) = 23 \)

Using: \( R_{SP}(v, \bar{r}f) = \min \left\{ t \geq 0 \mid e(v) + \sum_{T' > T} r(T')(t) \leq t \right\} \)
Refinement Algorithm

\[
\text{Tuple:} \\
\begin{align*}
\bar{r}_f^1 &= (r_f(T_1), r_f(T_2), r_f(T_3)) \\
\bar{r}_f^2 &= (r_f(T_1), r_f(T_2), r_f(T_3)) \\
\bar{r}_f^3 &= (r_f(T_1), r_f(T_2), r_f(T_3)) \\
\end{align*}
\]

Store

\[
(23, \bar{r}_f^1)
\]

\[
\begin{align*}
\text{Initialiation:} & \quad \bullet \quad \text{Most abstract functions} \\
\text{Each iteration:} & \quad \bullet \quad \text{Replace functions along abstraction trees} \\
\text{Termination:} & \quad \bullet \quad \text{All functions are concrete} \\
& \quad \text{OR:} \quad \bullet \quad \text{Estimate is already safe}
\end{align*}
\]

Using:

\[
R_{SP}(v, \bar{r}_f) = \min \left\{ t \geq 0 \mid e(v) + \sum_{T' \in T_f} T'_f(t) \leq t \right\}
\]
Refinement Algorithm

Step:

\[ \bar{rf}_1 = (rf(T_1), rf(T_2), rf(T_3)) \]

\[ \downarrow \]

\[ \bar{rf}_2 = (rf'(T_1), rf(T_2), rf(T_3)) \]

\[ \bar{rf}_3 = (rf''(T_1), rf(T_2), rf(T_3)) \]

In \( T_1 \):

\[ rf \]

\[ rf' \]

\[ rf'' \]

Store

(23, \( \bar{rf}_1 \))
Refinement Algorithm

Step:

\[ \bar{rf}_1 = (rf(T_1), rf(T_2), rf(T_3)) \]

\[ \downarrow \]

\[ \bar{rf}_2 = (rf'(T_1), rf(T_2), rf(T_3)) \rightarrow 18 \]

\[ \bar{rf}_3 = (rf''(T_1), rf(T_2), rf(T_3)) \rightarrow 21 \]

In \( T_1 \):

\[ rf' \quad \rightarrow \quad rf \quad \rightarrow \quad rf'' \]

Store

\[ (23, \bar{rf}_1) \]
Refinement Algorithm

Step:

\[ \bar{rf}_1 = (rf(T_1), rf(T_2), rf(T_3)) \]
\[ \downarrow \]
\[ \bar{rf}_2 = (rf'(T_1), rf(T_2), rf(T_3)) \rightarrow 18 \]
\[ \bar{rf}_3 = (rf''(T_1), rf(T_2), rf(T_3)) \rightarrow 21 \]

In \( T_1 \):

\[ rf \]
\[ rf' \]
\[ rf'' \]

Store

(23, \( \bar{rf}_1 \))
(21, \( \bar{rf}_2 \))
(18, \( \bar{rf}_3 \))
Refinement Algorithm

\[ \bar{rf} = (rf(T_1), rf(T_2), rf(T_3)) \]

\[ \downarrow \]

\[ \bar{rf}_2 = (rf'(T_1), rf(T_2), rf(T_3)) \]

\[ \downarrow \]

\[ \bar{rf}_4 = (rf(T_1), rf'(T_2), rf(T_3)) \]

\[ \downarrow \]

\[ \bar{rf}_5 = (rf(T_1), rf''(T_2), rf(T_3)) \]

\[ \rightarrow \]

\[ \rightarrow \]

Initialization:
• Most abstract functions

Each iteration:
• Replace functions along abstraction trees

Termination:
• All functions are concrete
  OR:
  • Estimate is already safe

Using:
\[ R_{SP}(v, \bar{rf}) = \min \{ t \geq 0 | e(v) + \sum_{T' \in T_{rf}} T'(t) \leq t \} \]
Refinement Algorithm

Step:

\[ \bar{rf}_2 = \langle rf(T_1), rf(T_2), rf(T_3) \rangle \]

Store

- \((21, \bar{rf}_2)\)
- \((18, \bar{rf}_3)\)
Refinement Algorithm

Step:

\[
\bar{rf}_2 = (rf(T_1), rf(T_2), rf(T_3))
\]

\[
\downarrow
\]

\[
\bar{rf}_4 = (rf(T_1), rf'(T_2), rf(T_3))
\]

\[
\bar{rf}_5 = (rf(T_1), rf''(T_2), rf(T_3))
\]

In \(T_2\):

\(rf\)

\(rf'\)

\(rf''\)

Store

(21, \(\bar{rf}_2\))

(18, \(\bar{rf}_3\))
Refinement Algorithm

Step:

\[
\bar{rf}_2 = (rf(T_1), rf(T_2), rf(T_3))
\]

\[
\downarrow
\]

\[
\bar{rf}_4 = (rf(T_1), rf'(T_2), rf(T_3)) \rightarrow 20
\]

\[
\bar{rf}_5 = (rf(T_1), rf''(T_2), rf(T_3)) \rightarrow 17
\]

In \( T_2 \):

\[
\begin{array}{c}
rf' \\
rf \\
r''
\end{array}
\]

Store

\[
(21, \bar{rf}_2)
\]

\[
(18, \bar{rf}_3)
\]
Refinement Algorithm

Step:

\( \bar{rf}_2 = (rf(T_1), rf(T_2), rf(T_3)) \)

\( \downarrow \)

\( \bar{rf}_4 = (rf(T_1), rf'(T_2), rf(T_3)) \rightarrow 20 \)

\( \bar{rf}_5 = (rf(T_1), rf''(T_2), rf(T_3)) \rightarrow 17 \)

In \( T_2: \)

\( rf' \)

\( rf'' \)

Store

(21, \( \bar{rf}_2 \))

(20, \( \bar{rf}_4 \))

(18, \( \bar{rf}_3 \))

(17, \( \bar{rf}_5 \))
Refinement Algorithm

\[
\bar{r}_f = (r_f(T_1), r_f(T_2), r_f(T_3))
\]

\[
R_{SP}(v, \bar{r}_f) = 23
\]

\[
\bar{r}_f_1 = (r_f(T_1), r_f(T_2), r_f(T_3))
\]

\[
\bar{r}_f_2 = (r_f(T_1), r_f(T_2), r_f(T_3))
\]

\[
\bar{r}_f_3 = (r_f(T_1), r_f(T_2), r_f(T_3))
\]

\[
\bar{r}_f_4 = (r_f(T_1), r_f(T_2), r_f(T_3))
\]

\[
\bar{r}_f_5 = (r_f(T_1), r_f(T_2), r_f(T_3))
\]

\[
\bar{r}_f_6 = (r_f(T_1), r_f(T_2), r_f(T_3))
\]

\[
\bar{r}_f_7 = (r_f(T_1), r_f(T_2), r_f(T_3))
\]

...
Refinement Algorithm

Initialization:
- Most abstract functions

Each iteration:
- Replace functions along *abstraction trees*

Termination:
- All functions are *concrete*
  - OR:
  - Estimate is already safe

Store

(20, $\bar{rf}_4$)
(18, $\bar{rf}_3$)
(17, $\bar{rf}_5$)
...
Refinement Algorithm

Initialization:
- Most abstract functions

Each iteration:
- Replace functions along abstraction trees

Termination:
- All functions are concrete
- Estimate is already safe

Store

(20, $\bar{r}f_4$)
(18, $\bar{r}f_3$)
(17, $\bar{r}f_5$)

...
1-30 tasks with 5-10 vertices each, branching degree 1-3
Evaluation: Precision Improvement

Type A: lower parameter variance
Type B: higher parameter variance
Generality

- Exact solution for NP-hard problem
  - *Efficient* method
  - Iterative refinement
- General!
  - Apply to *any* combinatorial problem
  - ... with monotonic abstraction lattice
Fahrplan

1 DRT Tasks in the Model Hierarchy
   - Liu and Layland and Sporadic Tasks
   - Frames and Branching
   - The Digraph Real-Time (DRT) Task Model
   - Adaptive Variable-Rate (AVR) Tasks

2 Feasibility Analysis of DRT
   - Feasibility Theorem
   - Demand Pairs
   - Test Termination
   - Evaluation

3 Static Priority Schedulability Analysis of DRT
   - Response-Time Analysis
   - Request Functions
   - Refinement Algorithm
   - Evaluation
Summary

- RRT
- RB
- GMF
- MF
- L&L
- ncRRT
- ncGMF
- sporadic
- coNP-hard
Summary

coNP-hard


EDRT

k-EDRT

DRT

RRT

ncRRT

RB

ncGMF

GMF

MF

sporadic

L&L

EDF
Summary

- **coNP-hard**
- **p.p.**

EDF

- L&L
- sporadic
- MF
- GMF
- RB
- RRT
- DRT
- k-EDRT
- EDRT

SP

- L&L
- sporadic
- MF
- GMF
- RB
- RRT
- DRT
- k-EDRT
- EDRT

RRT

- ncRRT
- ncGMF
- GMF
- MF
- sporadic

EDF

- ncRRT
- ncGMF
- GMF
- MF
- sporadic

SP

- ncRRT
- ncGMF
- GMF
- MF
- sporadic
Summary

CoNP-hard


EDRT

k-EDRT

DRT

ncRRT

RB

ncGMF

GMF

MF

L&L

sporadic

EDF

SP

CoNP-hard


EDRT

k-EDRT

DRT

ncRRT

RB

ncGMF

GMF

MF

L&L

sporadic

EDF

SP
Thanks!
Backup Slides Coming Up . . .
Path Abstractions: SP + EDF
Path Abstractions: Static Priorities

\[ rf_{\pi}(t) := \max \left\{ e(\pi') \mid \pi' \text{ is prefix of } \pi \text{ and } p(\pi') < t \right\} \]
Path Abstractions: EDF

\[ \tau_2 \nu \pi(t, t') \coloneqq \max \left\{ e(\pi') \mid \pi' \text{ is prefix of } \pi, p(\pi') < t \text{ and } d(\pi') \leq t' \right\} \]
$\text{Path Abstractions: EDF}$

$\text{wf}_\pi(t, t') := \max\{e(\pi') \mid \pi' \text{ is prefix of } \pi,\ p(\pi') < t \text{ and } d(\pi') \leq t'\}.$