technische universität
dortmund

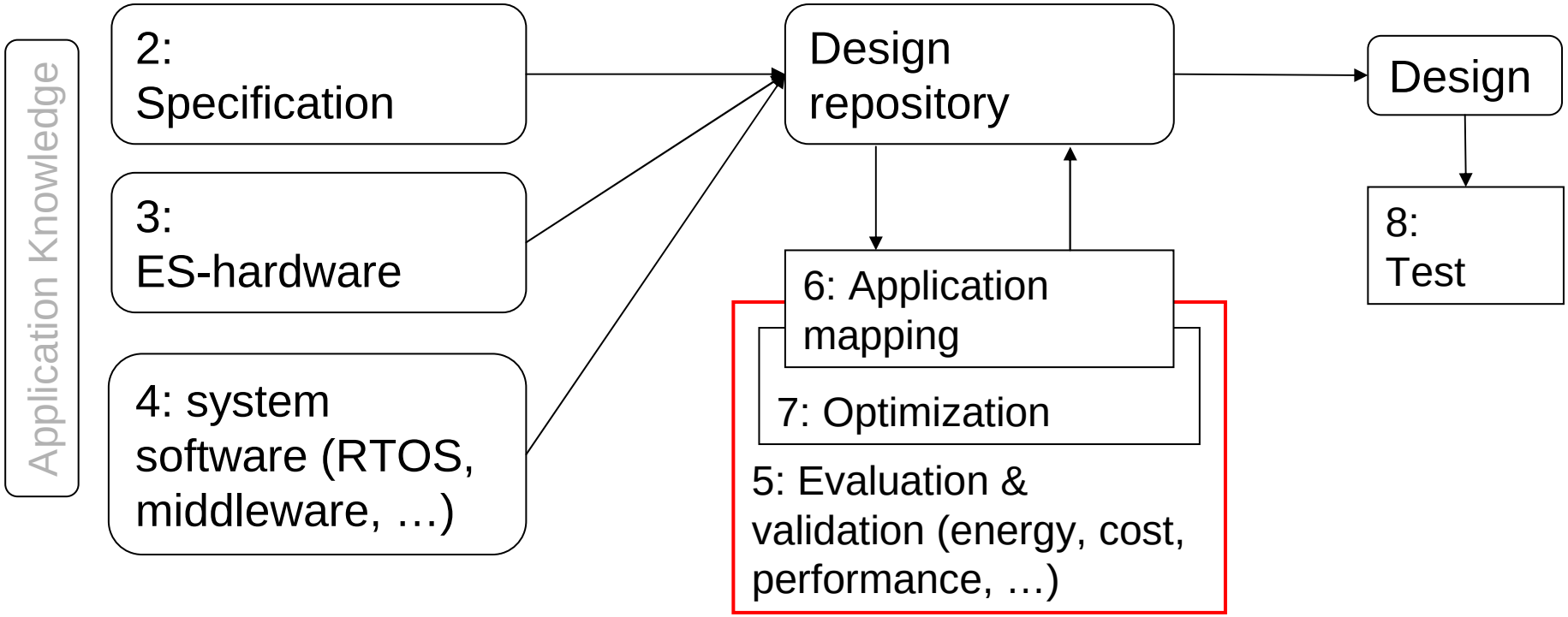
fakultät für informatik
informatik 12

Evaluation and Validation

Peter Marwedel
TU Dortmund, Informatik 12
Germany

2010年 12 月 05 日

Structure of this course



Numbers denote sequence of chapters

Validation and Evaluation

Definition: *Validation* is the process of checking whether or not a certain (possibly partial) design is appropriate for its purpose, meets all constraints and will perform as expected (yes/no decision).

Definition: Validation with mathematical rigor is called *(formal) verification*.

Definition: *Evaluation* is the process of computing quantitative information of some key characteristics of a certain (possibly partial) design.

How to evaluate designs according to multiple criteria?

In practice, many different criteria are relevant for evaluating designs:

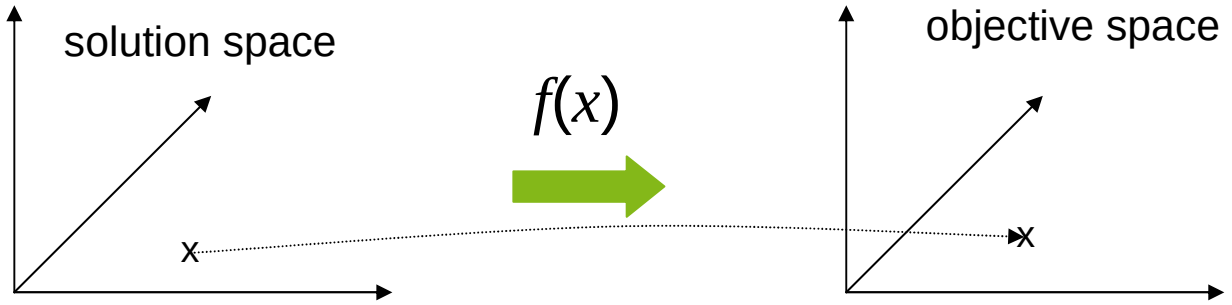
Different design objectives/criteria are relevant:

- Average performance
- Worst case performance
- Energy/power consumption
- Thermal behavior
- Reliability
- Electromagnetic compatibility
- Numeric precision, Testability, Cost, weight, robustness, usability, extendibility, security, safety, environmental ..

How to compare designs? Some designs “better” than others.

Definitions

- Let X: m-dimensional **solution space** for the design problem. Example: dimensions correspond to # of processors, size of memories, type and width of busses etc.
- Let F: n-dimensional **objective space** for the design problem. Example: dimensions correspond to speed, cost, power consumption, size, weight, reliability, …
- Let $f(x)=(f_1(x),\ldots,f_n(x))$ where $x \in X$ be an **objective function.** We assume that we are using $f(x)$ for evaluating designs.



Pareto points

- We assume that, for each objective, a total order < and the corresponding order ≤ are defined.
- **Definition**:
  Vector $u=(u_1,\ldots,u_n)\in F$ **dominates** vector $v=(v_1,\ldots,v_n)\in F$
  $\Leftrightarrow$
  u is “better” than v with respect to one objective and not worse than v with respect to all other objectives:

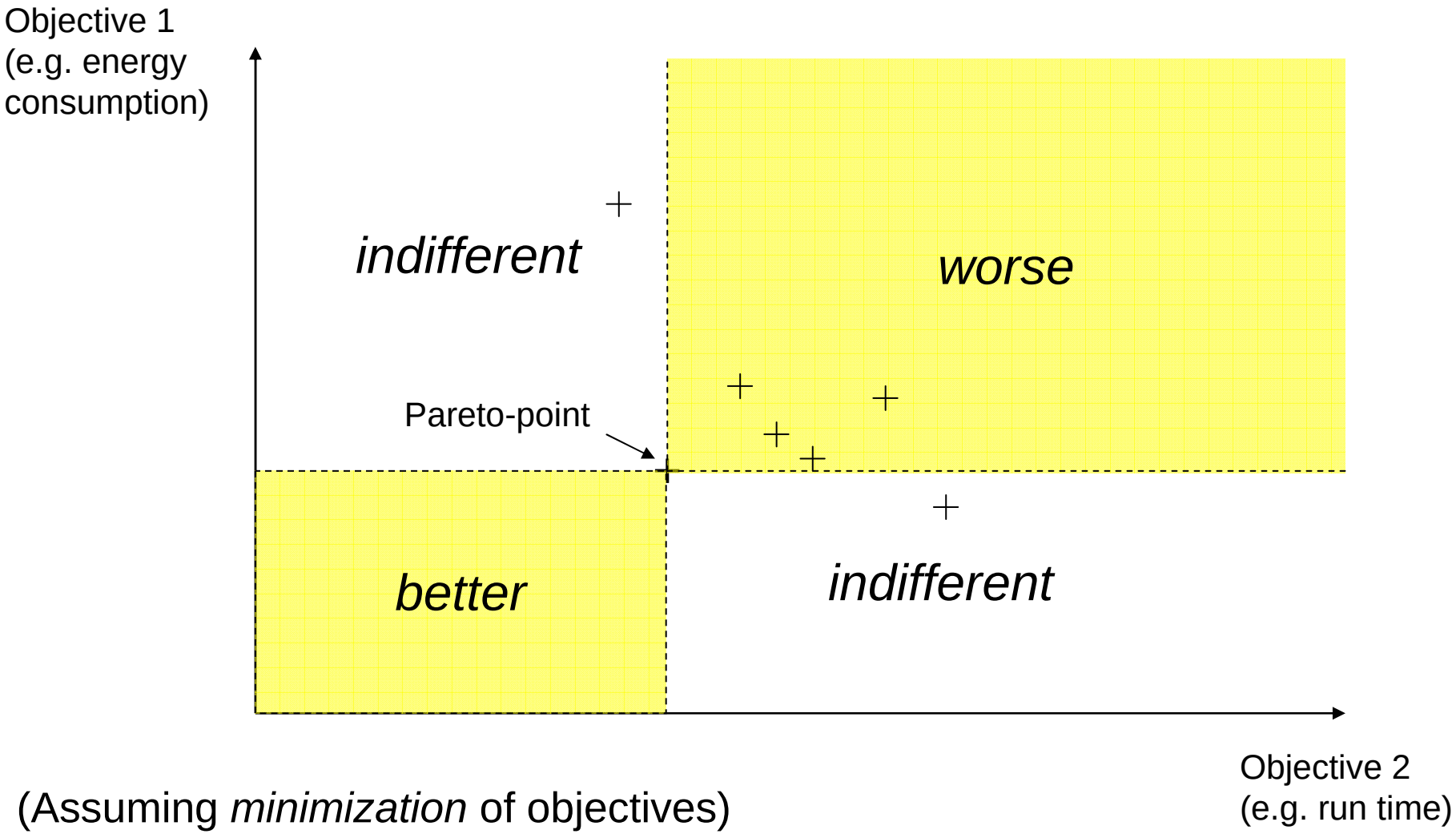
$$\forall i \in \{1,\ldots,n\} : u_i \leq v_i \wedge$$
$$\exists i \in \{1,\ldots,n\} : u_i < v_i$$

- **Definition:**
  Vector $u\in F$ is **indifferent** with respect to vector $v\in F$
  $\Leftrightarrow$ neither u dominates v nor v dominates u

Pareto points

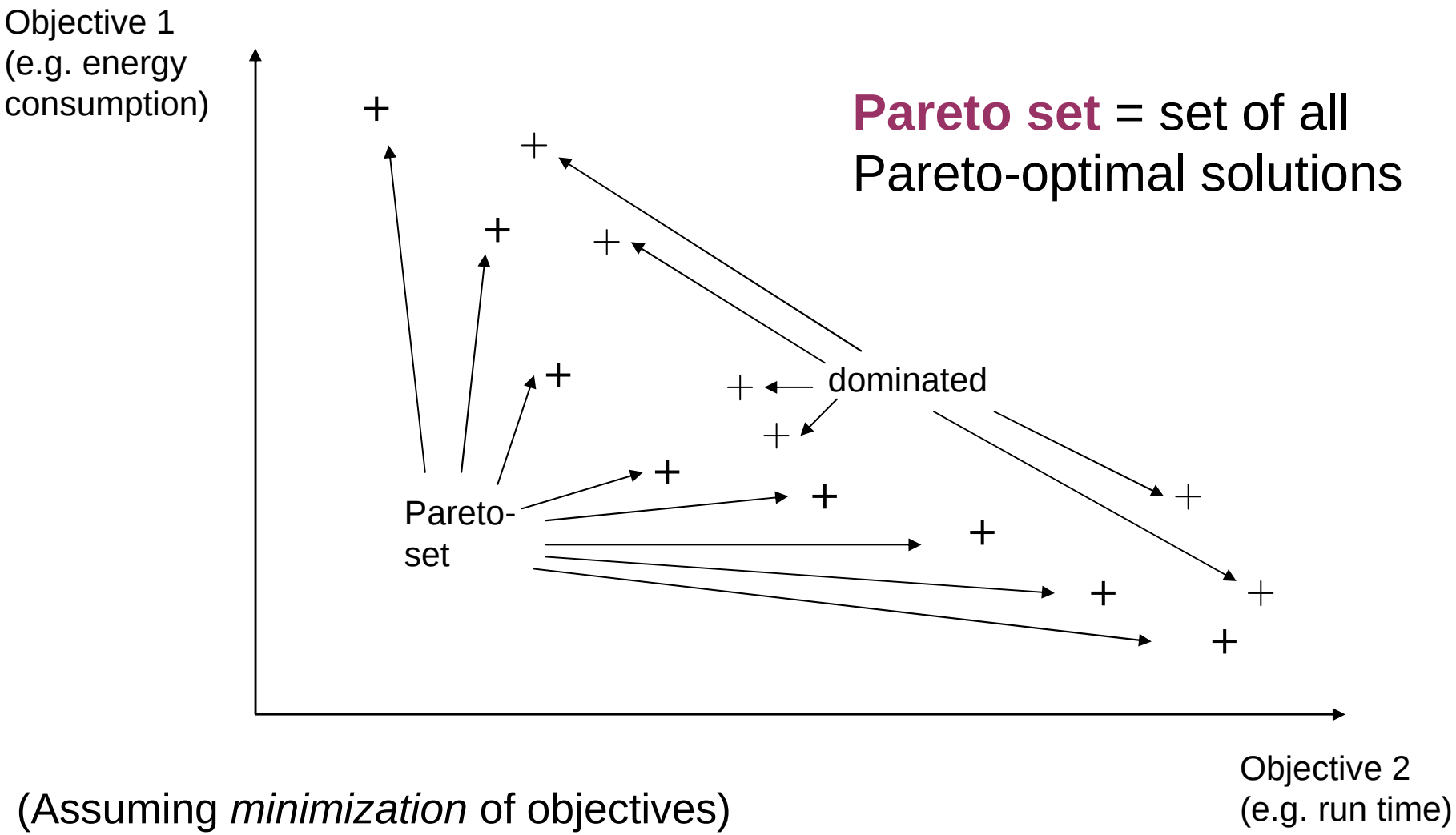
- A solution $x \in X$ is called **Pareto-optimal** with respect to X
  $\Leftrightarrow$
  there is no solution $y \in X$ such that $u=f(x)$ is dominated by $v=f(y)$
- **Definition**: Let $S \subseteq F$ be a subset of solutions.
  v is called a **non-dominated solution** with respect to S
  $\Leftrightarrow$ v is not dominated by any element $\in S$.
- v is called **Pareto-optimal**
  $\Leftrightarrow$ v is non-dominated with respect to all solutions F.

Pareto Point



(Assuming *minimization* of objectives)

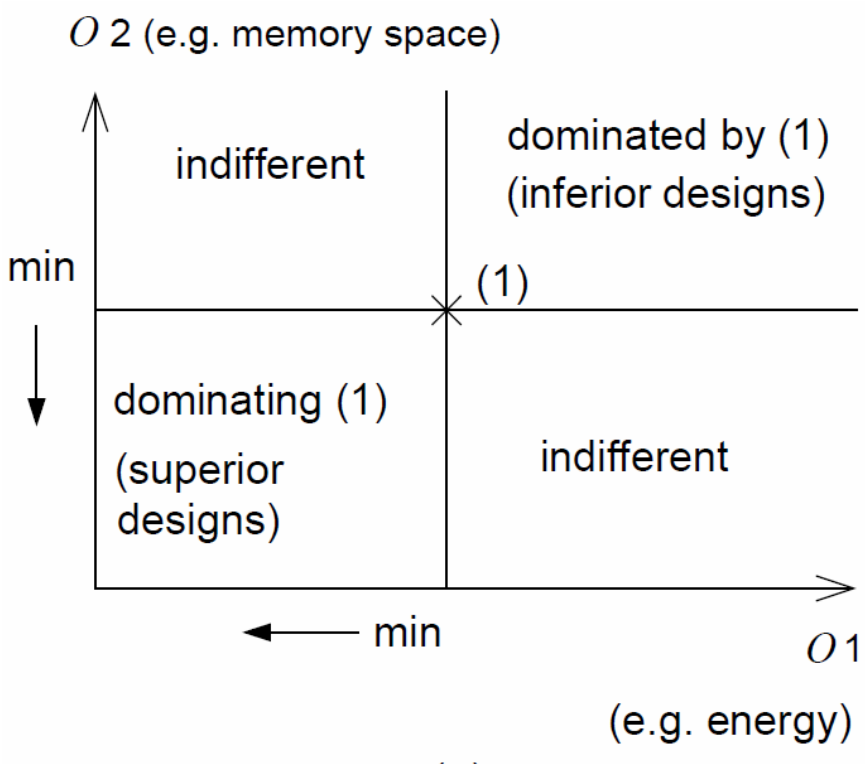
Pareto Set



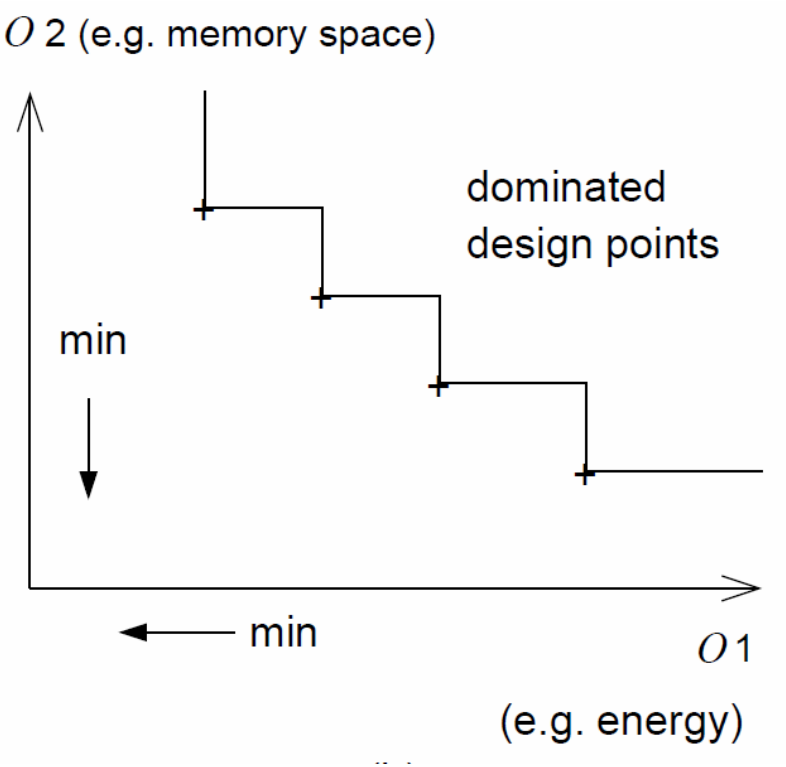
(Assuming *minimization* of objectives)

One more time …

Pareto point



Pareto front



Design space evaluation

Design space evaluation (DSE) based on Pareto-points is the process of finding and returning a set of Pareto-optimal designs to the user, enabling the user to select the most appropriate design.

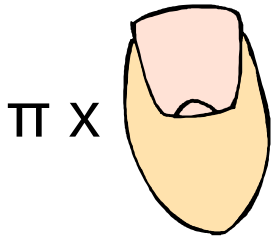
Evaluation of designs according to multiple objectives

Different design objectives/criteria are relevant:

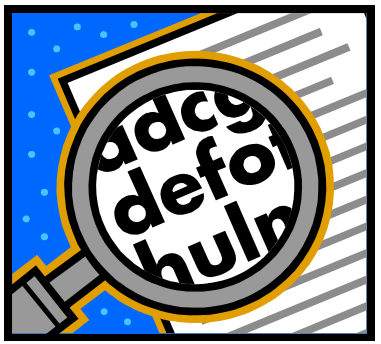
- Average performance
- Worst case performance
- Energy/power consumption
- Thermal behavior
- Reliability
- Electromagnetic compatibility
- Numeric precision
- Testability
- Cost
- Weight, robustness, usability, extendibility, security, safety, environmental friendliness

Performance evaluation

- **Estimated performance values:**
  Difficult to generate sufficiently precise estimates;
  Balance between run-time and precision

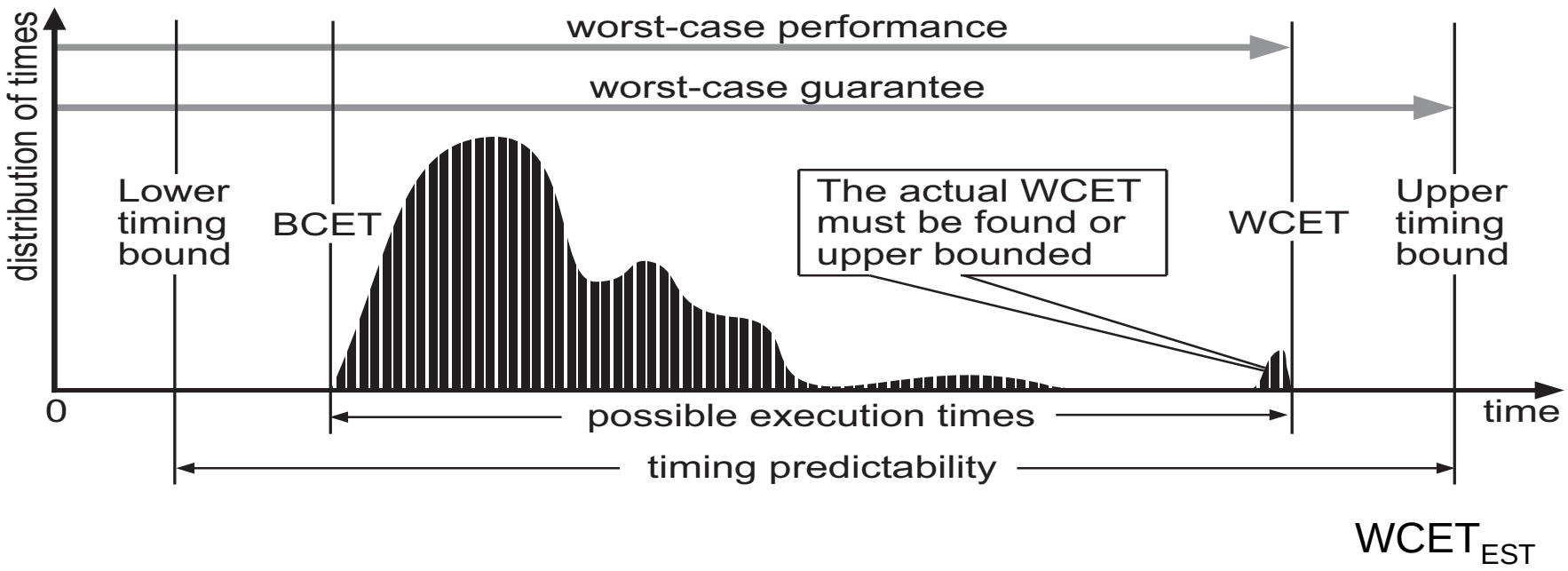


- **Accurate performance values:**
  As precise as the input data is.



We need to compute **average** and **worst case** execution times

Worst/best case execution times



Requirements on WCET estimates:

- *Safeness:* WCET ≤ $WCET_{EST}$!
- *Tightness:* $WCET_{EST}$ – WCET → minimal

Worst case execution times (2)

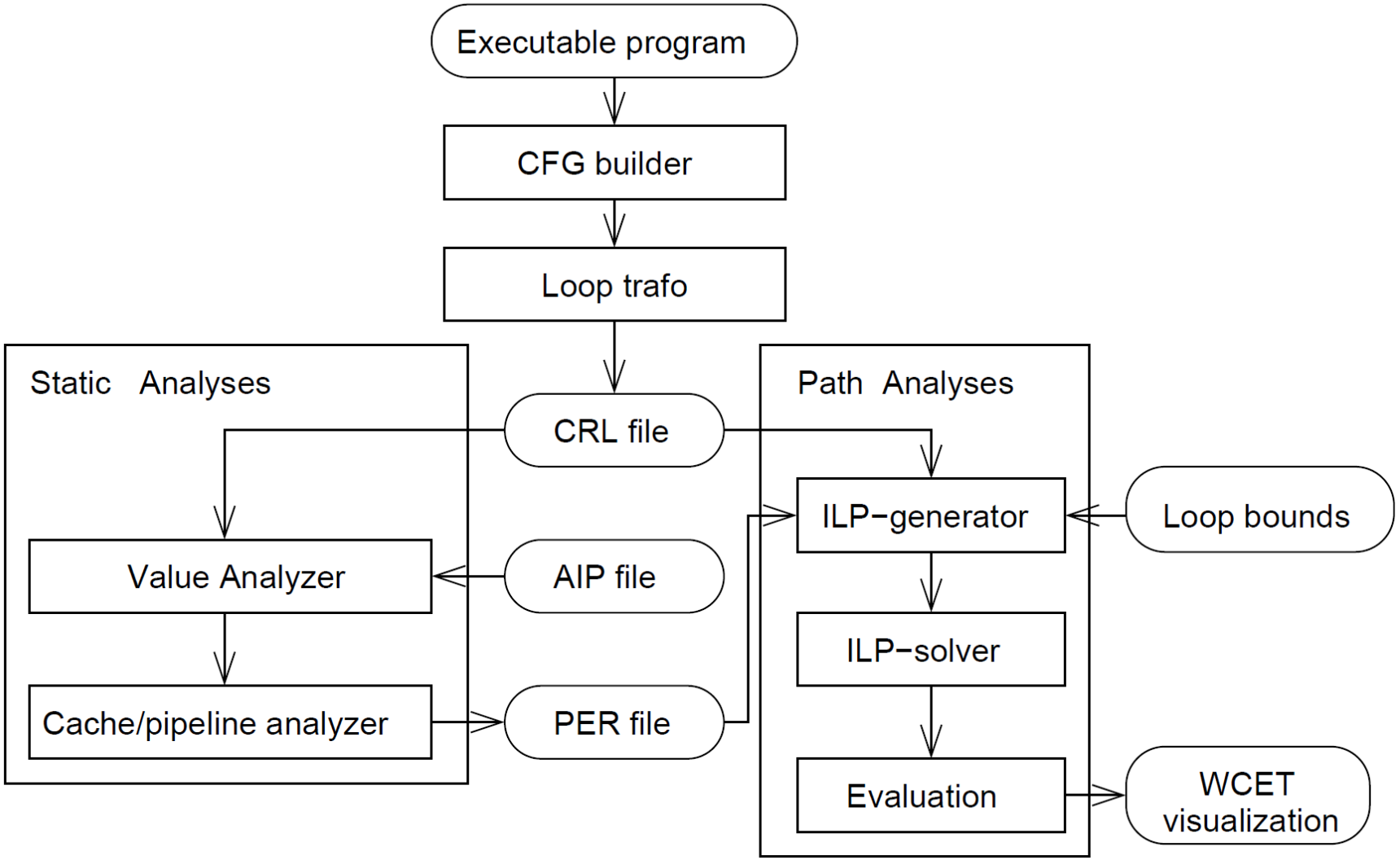
Complexity:

- in the general case: undecidable if a bound exists.
- for restricted programs: simple for “old“ architectures, very complex for new architectures with pipelines, caches, interrupts, virtual memory, etc.

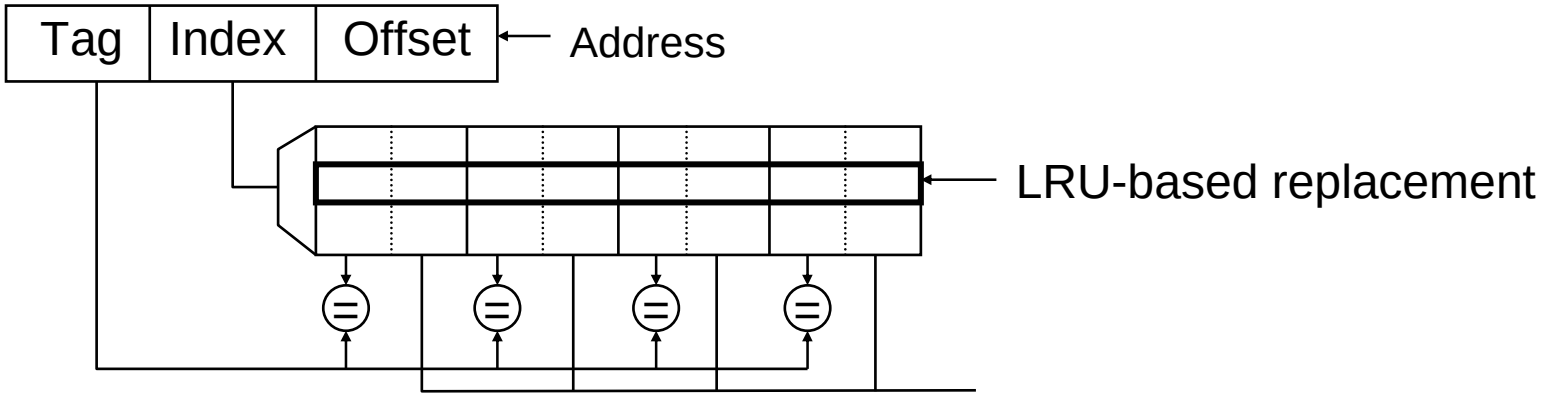
Approaches:

- for hardware: requires detailed timing behavior
- for software: requires availability of machine programs; complex analysis (see, e.g., www.absint.de)

WCET estimation: AiT (AbsInt)

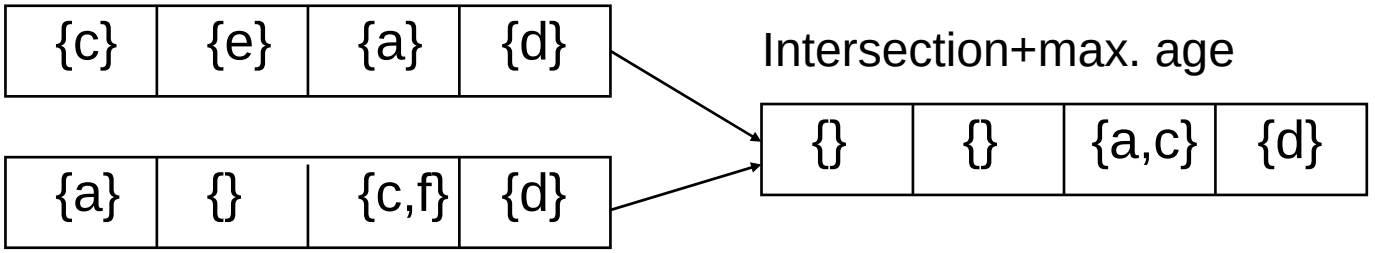


WCET estimation for caches

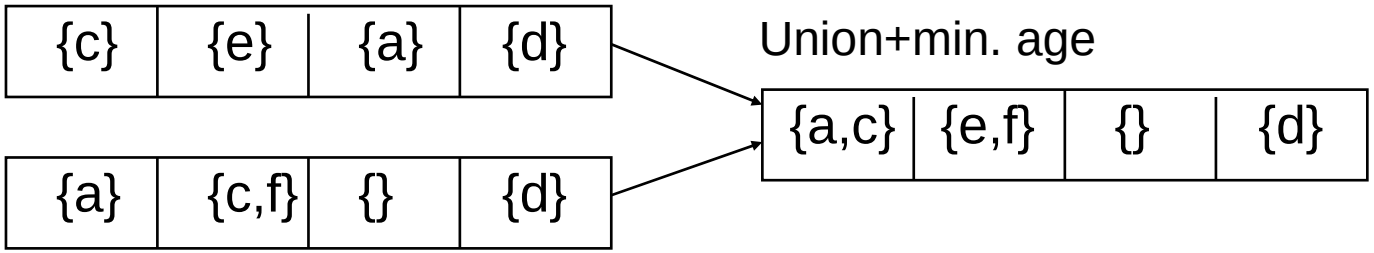


Behavior at program joins

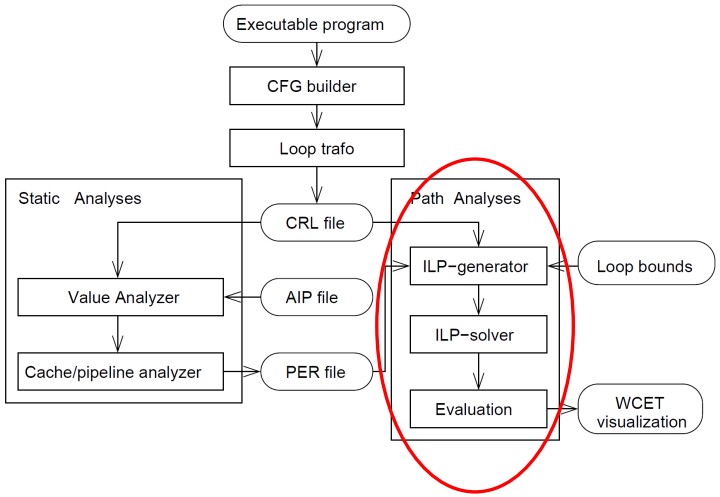
Worst case



Best case



ILP model



- Objective function reflects execution time as as function of the execution time of blocks. To be **maximized**.
- Constraints reflect dependencies between blocks.
- Avoids explicit consideration of all paths

☞ Called **implicit path enumeration** technique.

Example (1)

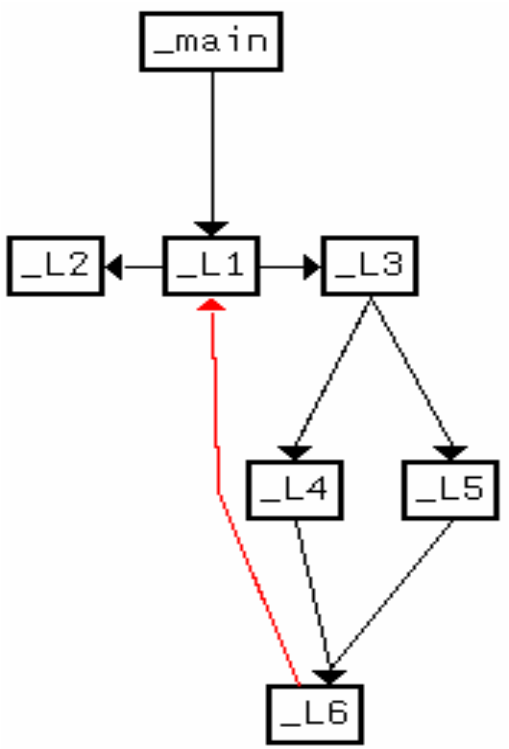
Program

```
int main()
{
  int i, j = 0;

  _Pragma( "loopbound min
                  100 max 100" );
  for ( i = 0; i < 100; i++ ) {
    if ( i < 50 )
      j += i;
    else
      j += ( i * 13 ) % 42;
  }

  return j;
}
```

CFG



WCETs of BB
(aiT 4 TriCore)

_main: 21 cycles
_L1: 27
_L3: 2
_L4: 2
_L5: 20
_L6: 13
_L2: 20

Example (2)

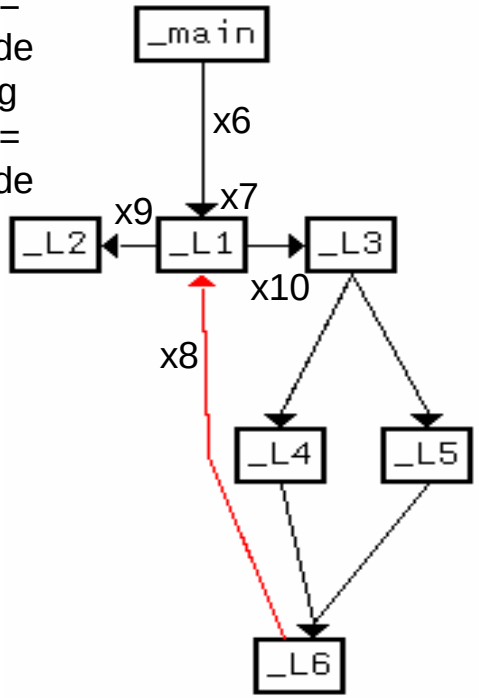
- Virtual start node
- Virtual end node
- Virtual end node per function

Variables:

- 1 variable per node
- 1 variable per edge

Constraints: „Kirchhoff“ equations per node

- Sum of incoming edge variables = flux through node
- Sum of outgoing edge variables = flux through node

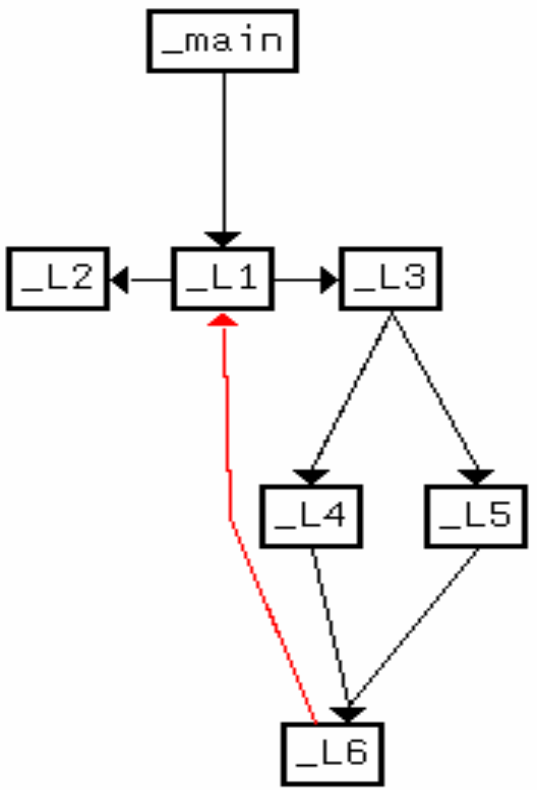


_main: 21 cycles
_L1: 27
_L3: 2
_L4: 2
_L5: 20
_L6: 13
_L2: 20

ILP

```
/* Objective function = WCET to be maximized*/
21 x2 + 27 x7 + 2 x11 + 2 x14 + 20 x16 + 13 x18 + 20 x19;
/* CFG Start Constraint */
x0 - x4 = 0;
/* CFG Exit Constraint */
x1 - x5 = 0;
/* Constraint for flow entering function main */
x2 - x4 = 0;
/* Constraint for flow leaving exit node of main */
x3 - x5 = 0;
/* Constraint for flow entering exit node of main */
x3 - x20 = 0;
/* Constraint for flow entering main = flow leaving main */
x2 - x3 = 0;
/* Constraint for flow leaving CFG node _main */
x2 - x6 = 0;
/* Constraint for flow entering CFG node _L1 */
x7 - x8 - x6 = 0;
/* Constraint for flow leaving CFG node _L1 */
x7 - x9 - x10 = 0;
/* Constraint for lower loop bound of _L1 */
x7 - 101 x9 >= 0;
/* Constraint for upper loop bound of _L1 */
x7 - 101 x9 <= 0; ….
```

Example (3)



Value of objective function: 6268

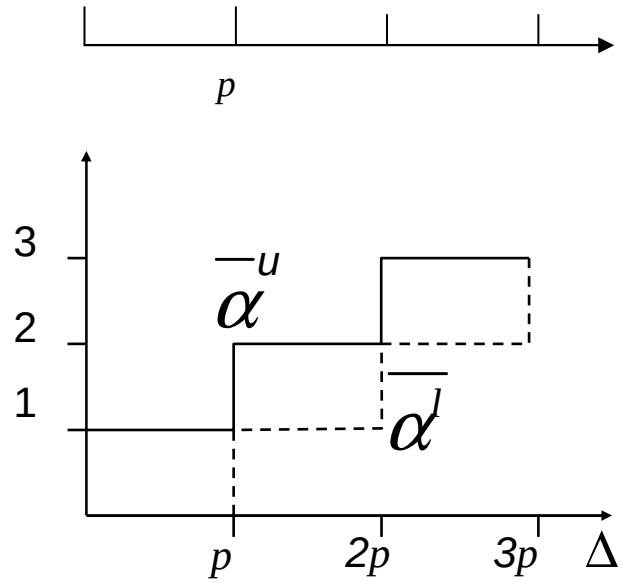
Actual values of the variables:

x2	1
x7	101
x11	100
x14	0
x16	100
x18	100
x19	1
x0	1
x4	1
x1	1
x5	1
x3	1
x20	1
x6	1
x8	100
x9	1
x10	100
x12	100
x13	0
x15	0
x17	100

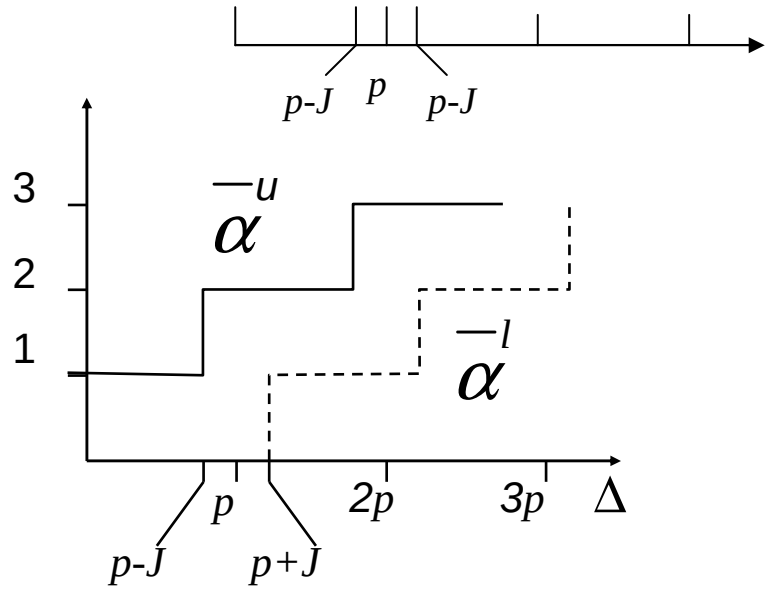
Real-time calculus (RTC)/ Modular performance analysis (MPA)

Thiele et al. (ETHZ): Extended **network calculus:**
Arrival curves describe the maximum and minimum number of events arriving in some time interval Δ.
Examples:

periodic event stream



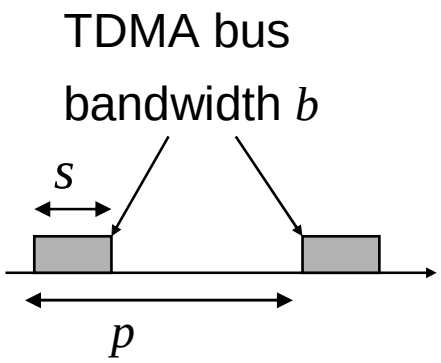
periodic event stream with jitter

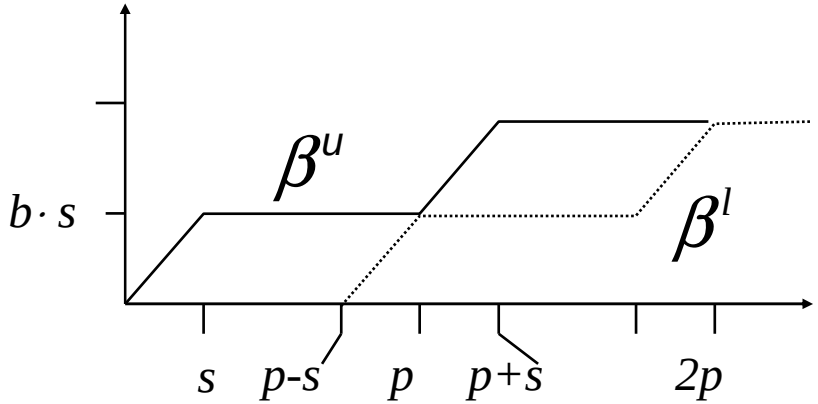


RTC/MPA: Service curves

Service curves β^u resp. β^ℓ describe the maximum and minimum service capacity available in some time interval Δ

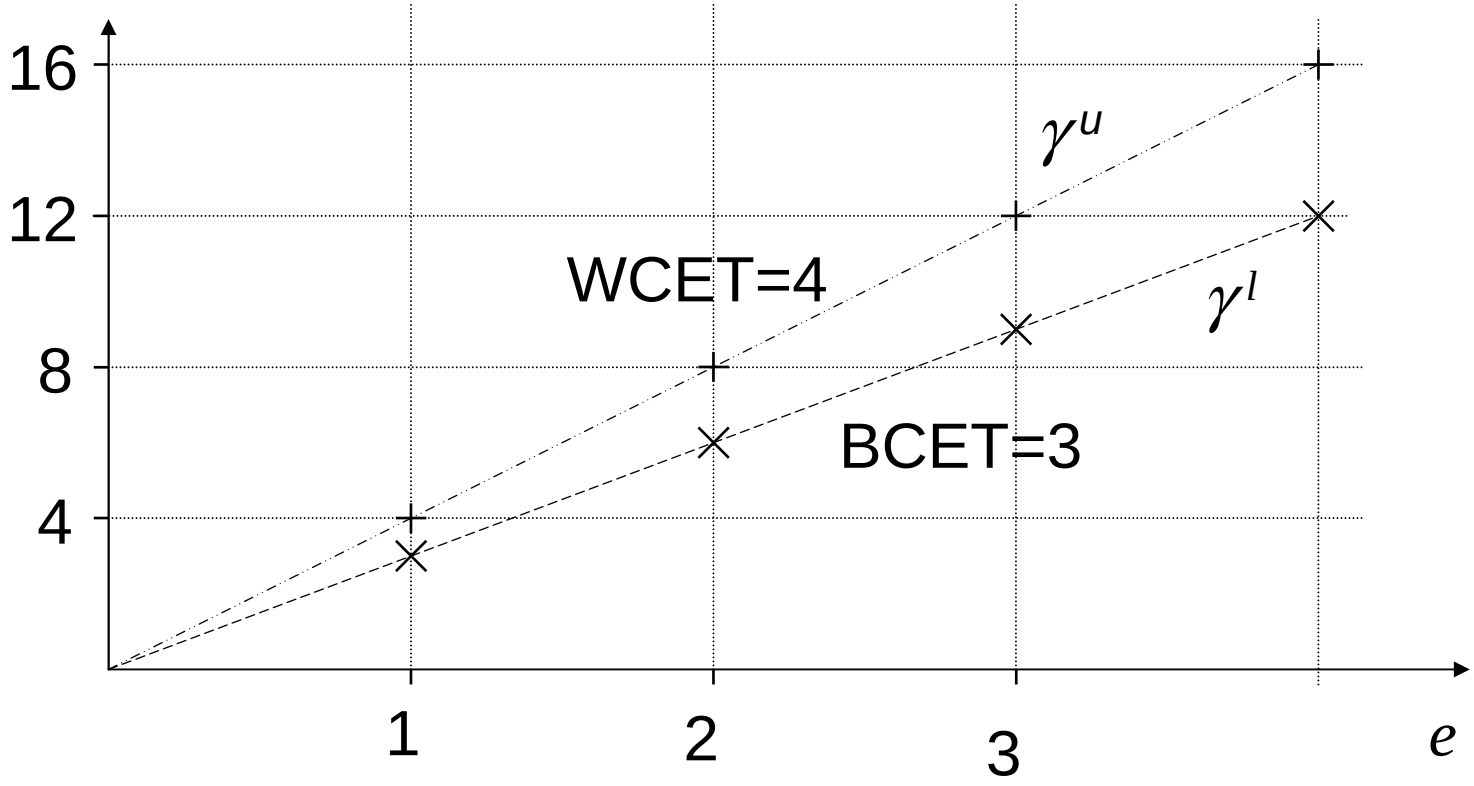
Example:





RTC/MPA: Workload characterization

γ^u resp. γ^ℓ describe the maximum and minimum service capacity required as a function of the number e of events. Example:



(Defined only for an integer number of events)

RTC/MPA: Workload required for incoming stream

Incoming workload

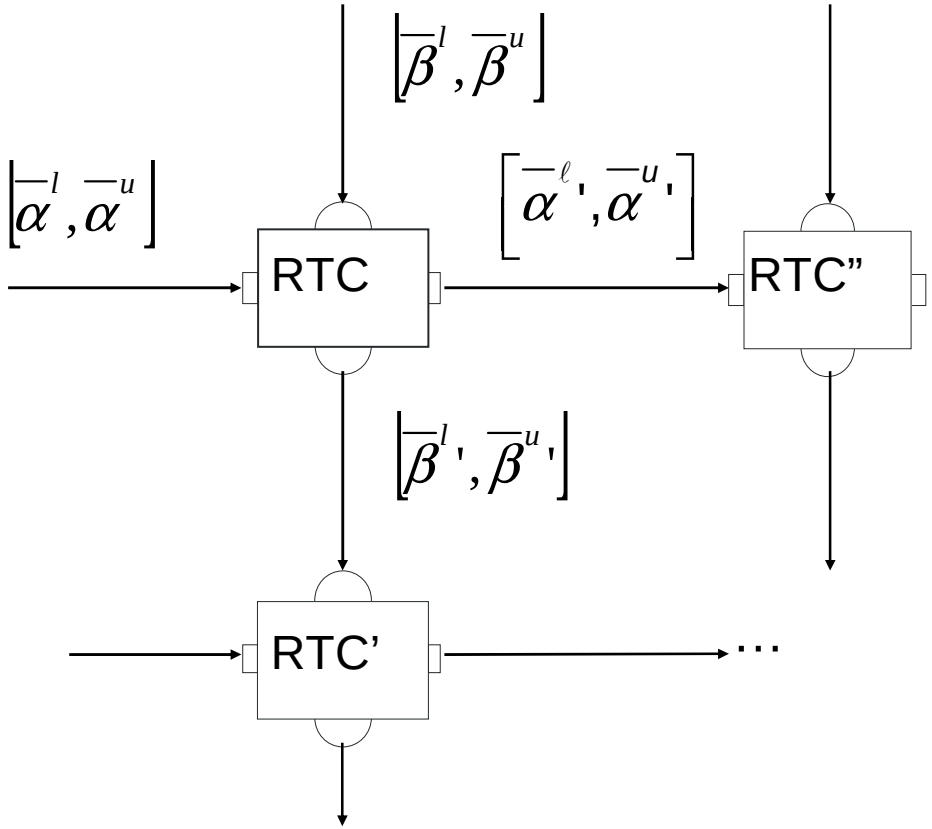
$$\alpha^u(\Delta) = \gamma^u\left(\overline{\alpha^u}(\Delta)\right) \qquad \alpha^l(\Delta) = \gamma^l\left(\overline{\alpha^l}(\Delta)\right)$$

Upper and lower bounds on the number of events

$$\overline{\beta}^u(\Delta) = \left(\gamma^l\right)^{-1}\left(\beta^u(\Delta)\right) \qquad \overline{\beta}^l(\Delta) = \left(\gamma^u\right)^{-1}\left(\beta^l(\Delta)\right)$$

RTC/MPA: System of real time components

Incoming event streams and available capacity are transformed by real-time components:



Theoretical results allow the computation of properties of outgoing streams ☞

RTC/MPA: Transformation of arrival and service curves

Resulting arrival curves:

$$\overline{\alpha}^{u\,\prime} = \min\left(\left[\left(\overline{\alpha}^{u} \underline{\otimes} \overline{\beta}^{u} \right) \overline{\oplus} \overline{\beta}^{l} \right], \overline{\beta}^{u} \right)$$

$$\overline{\alpha}^{l\,\prime} = \min\left(\left[\left(\overline{\alpha}^{l} \overline{\oplus} \overline{\beta}^{u} \right) \underline{\otimes} \overline{\beta}^{l} \right], \overline{\beta}^{l} \right)$$

Remaining service curves:

$$\overline{\beta}^{u\,\prime} = \left(\overline{\beta}^{u} - \overline{\alpha}^{l} \right) \underline{\oplus} 0$$

$$\overline{\beta}^{l\,\prime} = \left(\overline{\beta}^{l} - \overline{\alpha}^{u} \right) \overline{\otimes} 0$$

Where:

$$(f \underline{\otimes} g)(t) = \inf_{0 \le u \le t} \{ f(t-u) + g(u) \} \qquad (f \overline{\otimes} g)(t) = \sup_{0 \le u \le t} \{ f(t-u) + g(u) \}$$

$$(f \underline{\oplus} g)(t) = \inf_{u \ge 0} \{ f(t+u) - g(u) \} \qquad (f \overline{\oplus} g)(t) = \sup_{u \ge 0} \{ f(t+u) - g(u) \}$$

RTC/MPA: Remarks

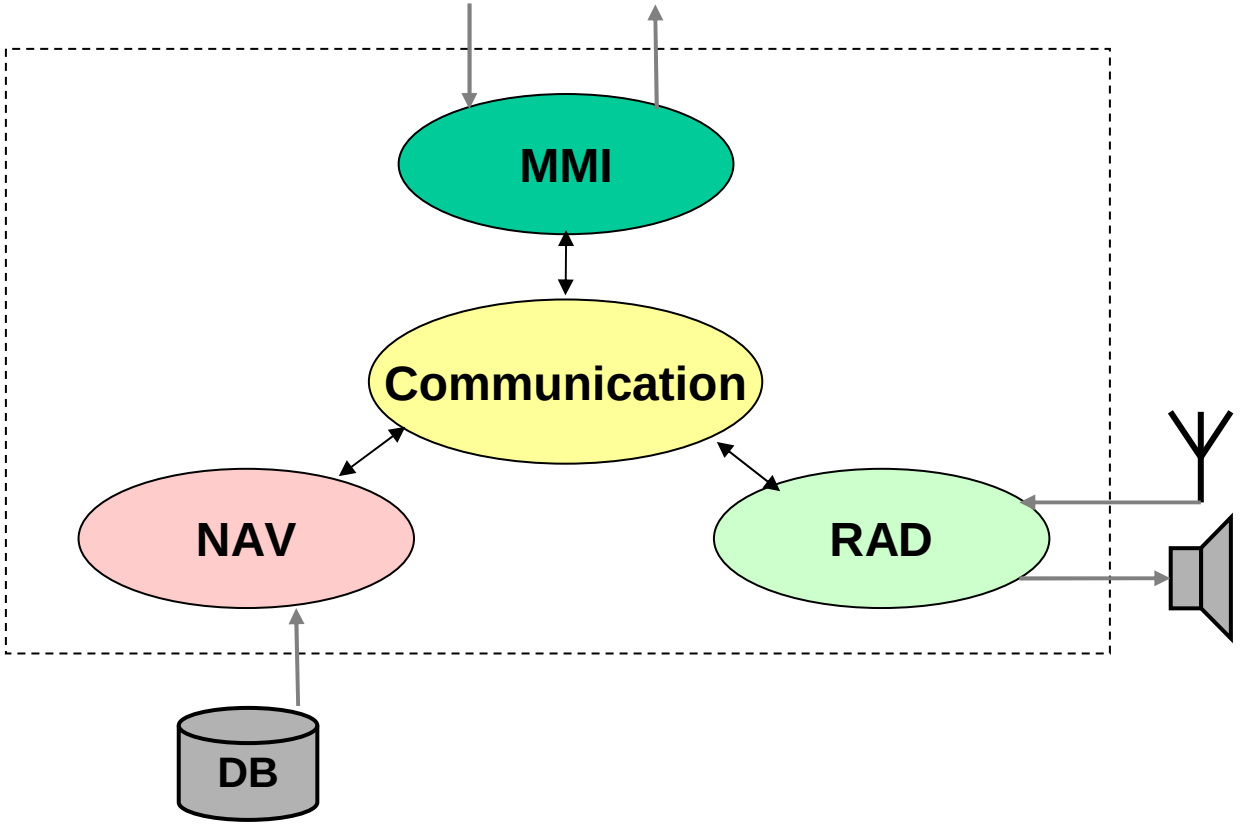
- Details of the proofs can be found in relevant references
- Results also include bounds on buffer sizes and on maximum latency.
- Theory has been extended into various directions, e.g. for computing remaining battery capacities

Application: In-Car Navigation System

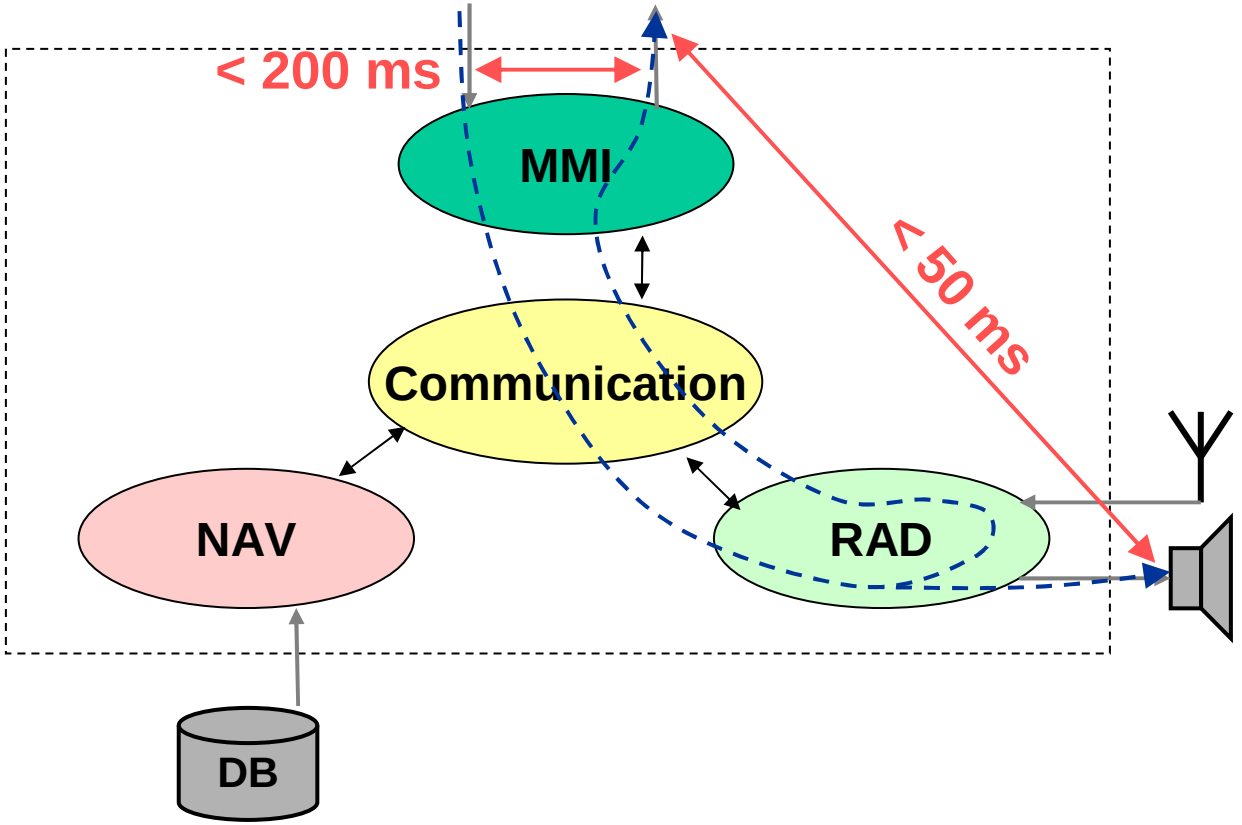
Car radio with navigation system
User interface needs to be responsive
Traffic messages (TMC) must be processed in a timely way
Several applications may execute concurrently

© Thiele, ETHZ

System Overview

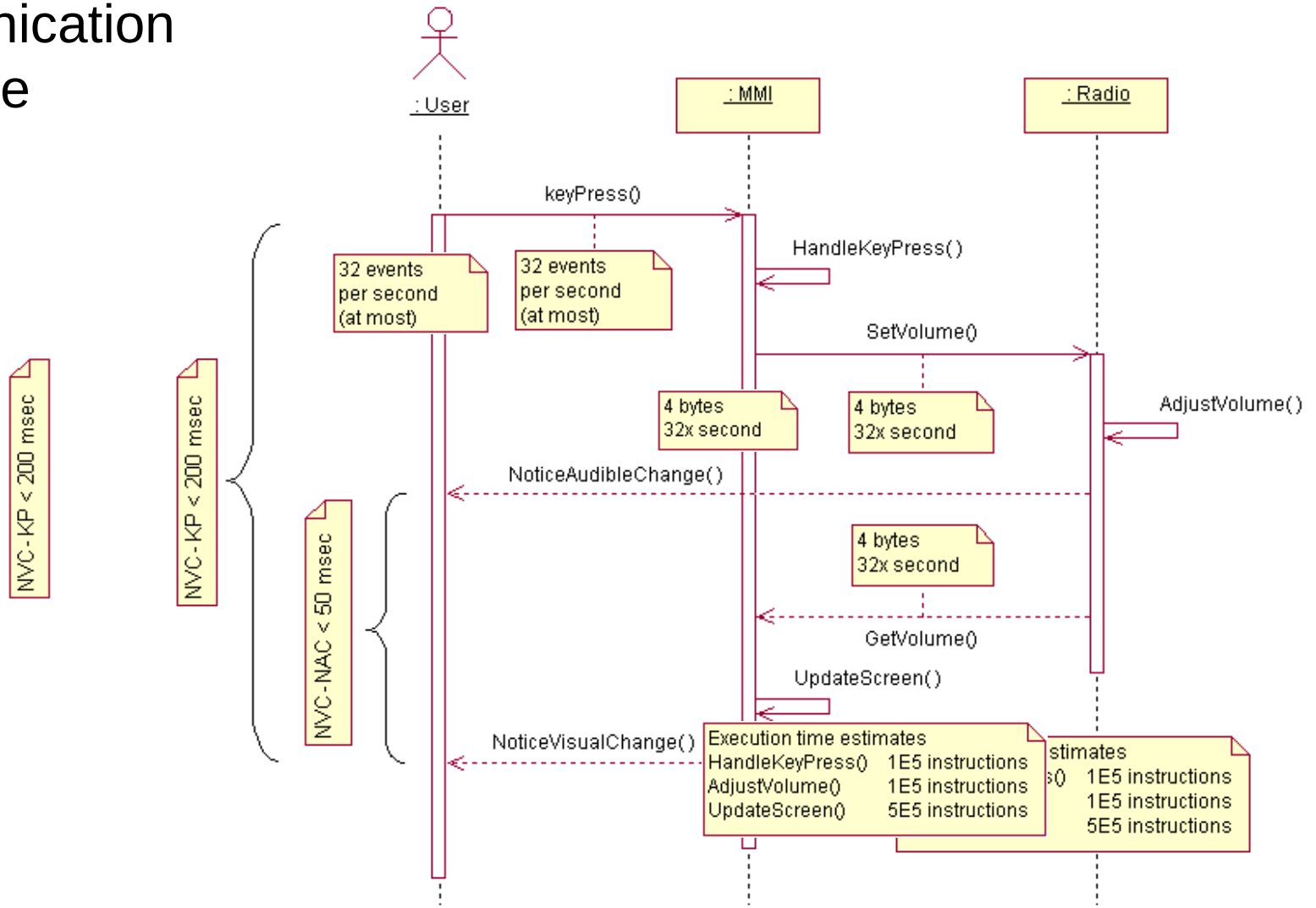


Use case 1: Change Audio Volume

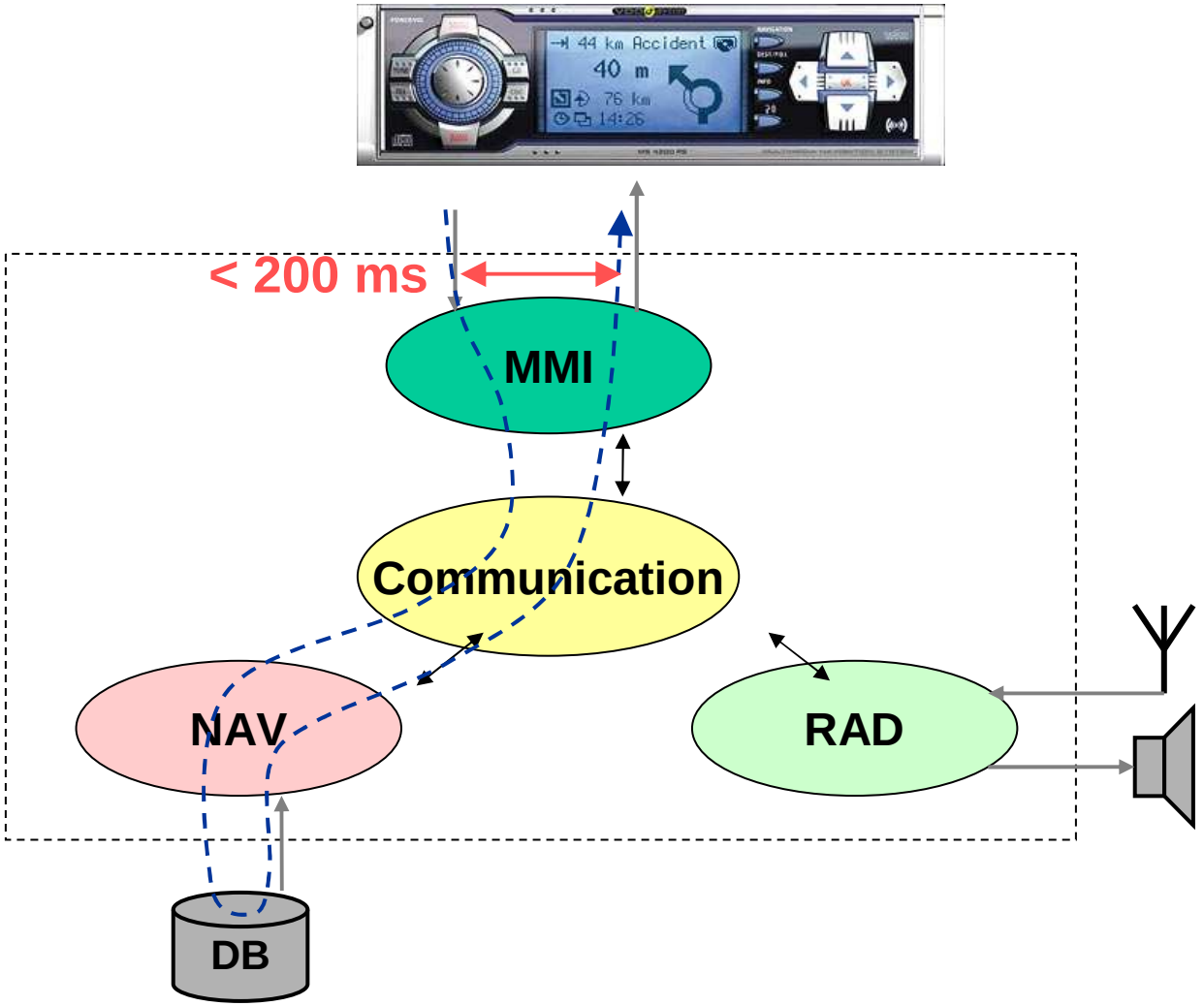


Use case 1: Change Audio Volume

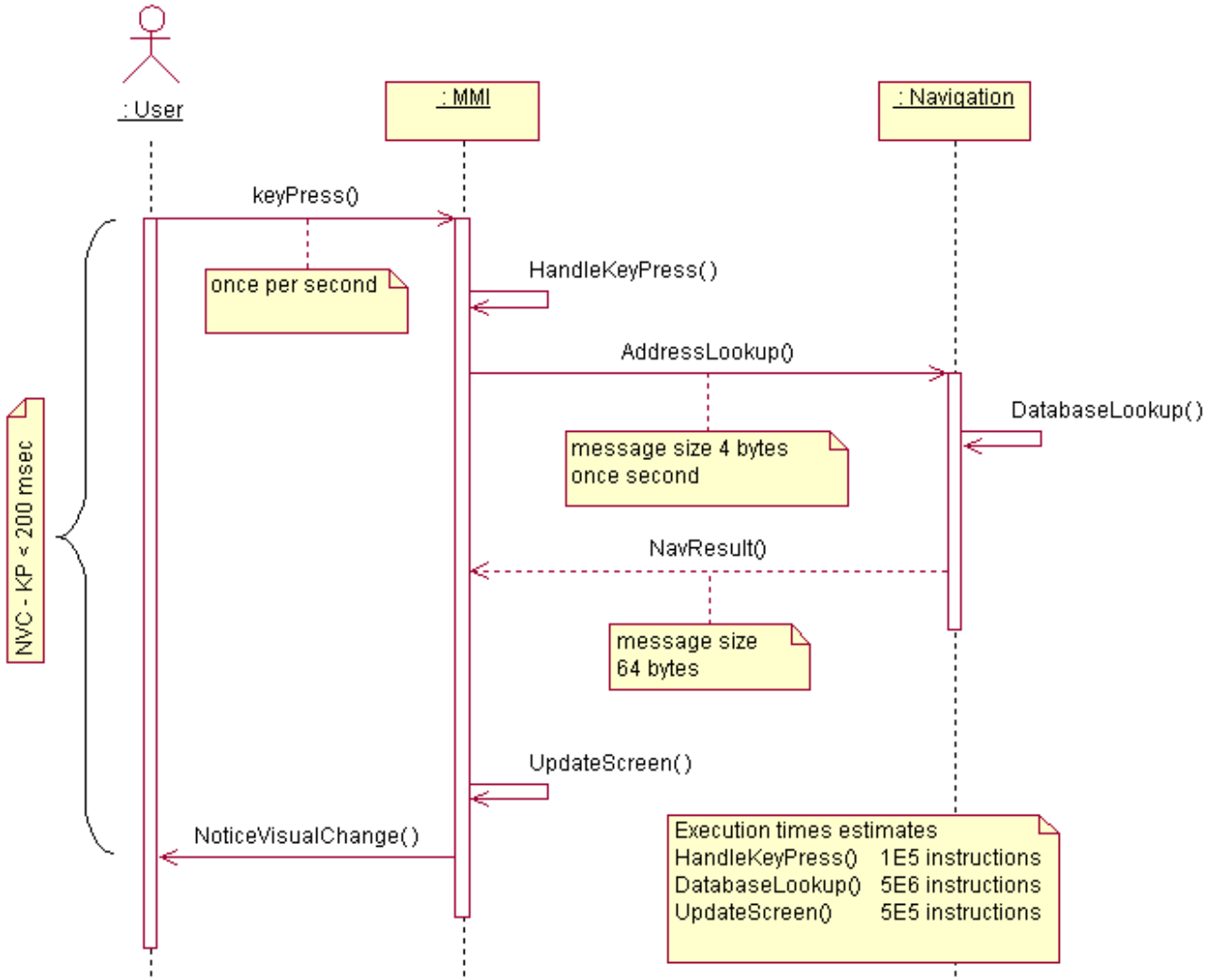
Communication
Resource
Demand



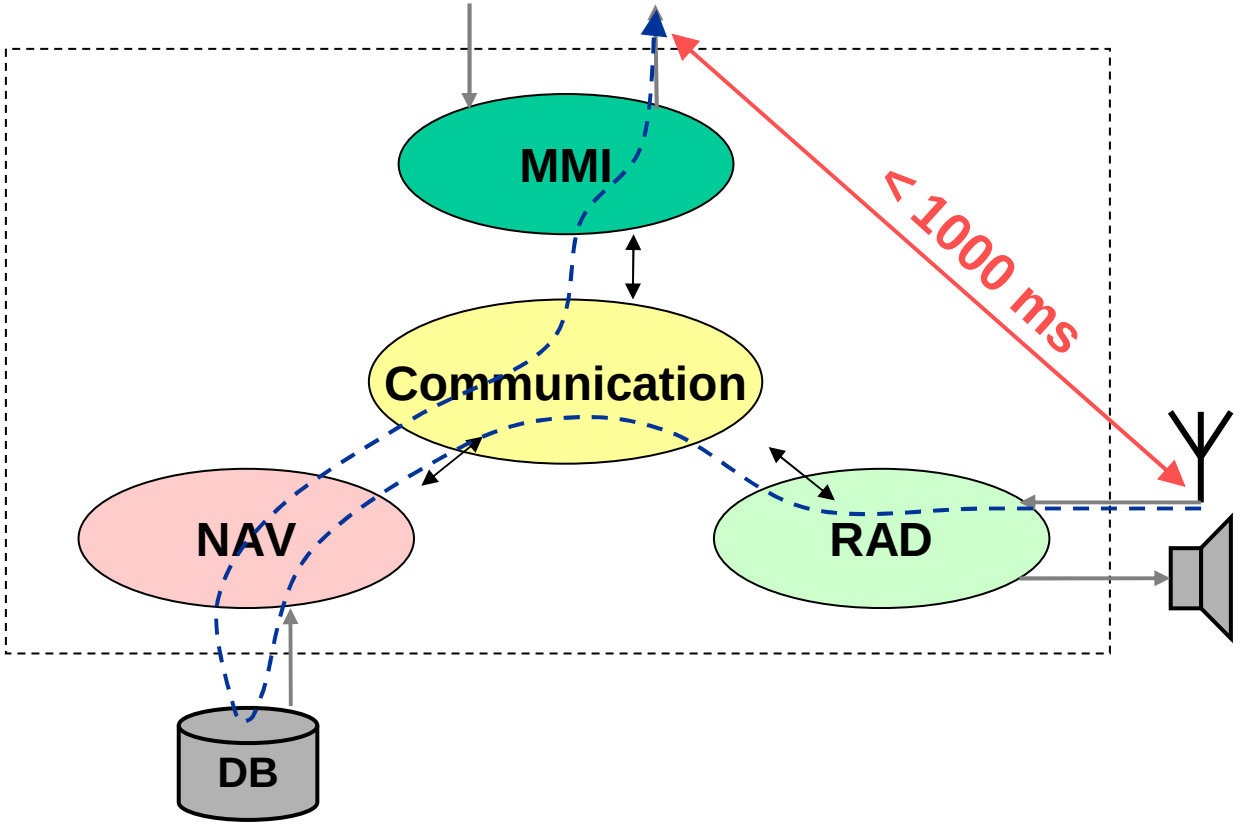
Use case 2: Lookup Destination Address



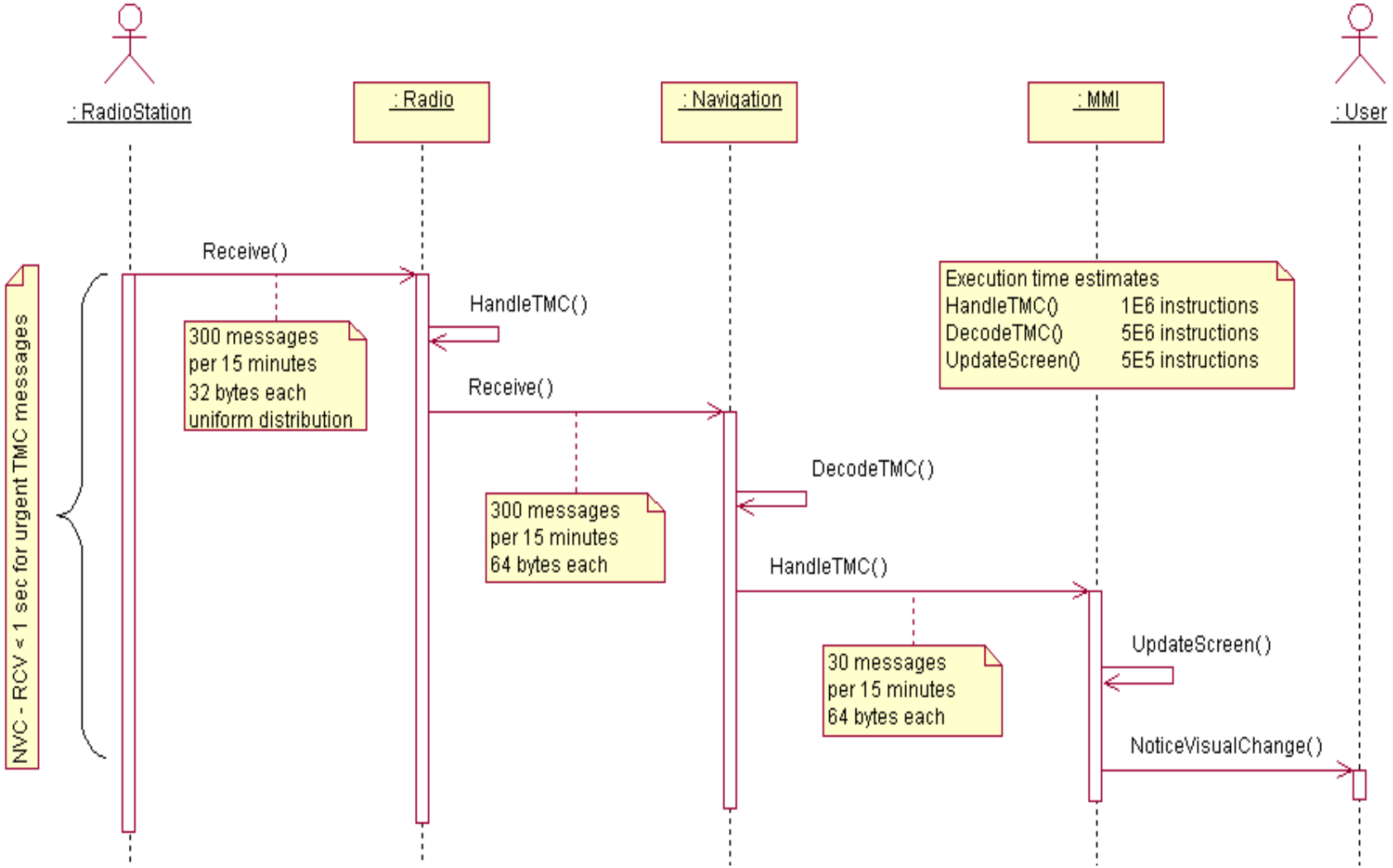
Use case 2: Lookup Destination Address



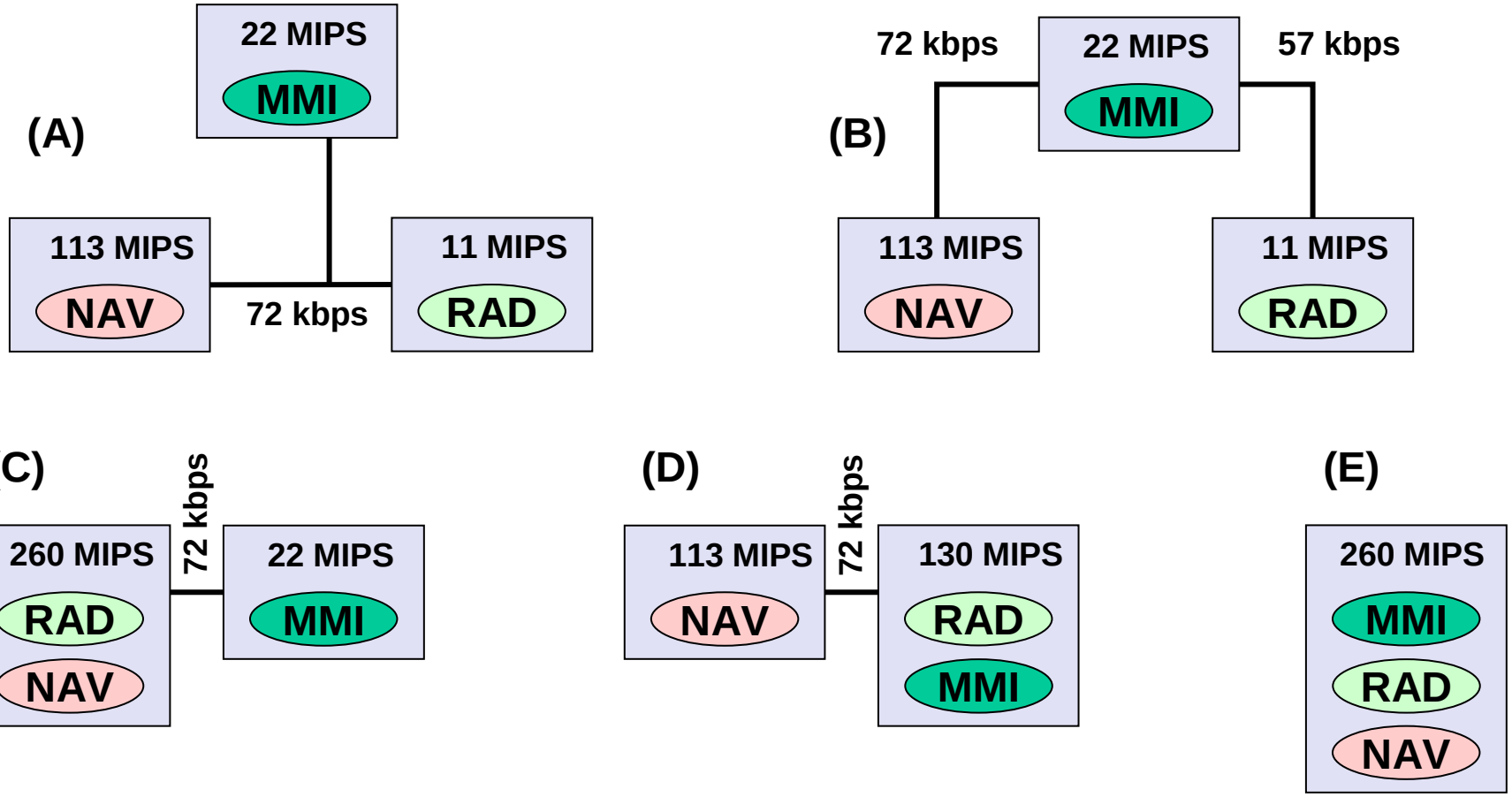
Use case 3: Receive TMC Messages



Use case 3: Receive TMC Messages



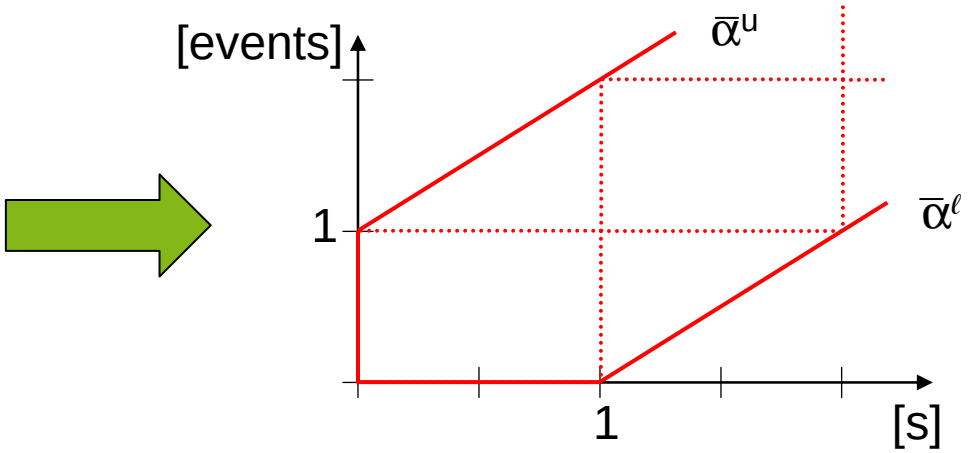
Proposed Architecture Alternatives



Step 1: Environment (Event Steams)

Event Stream Model

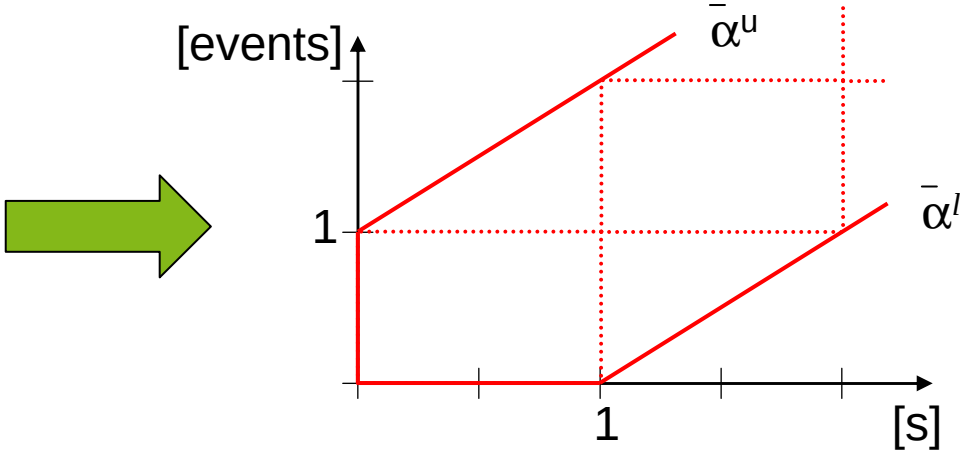
e.g. Address Lookup
(1 event / sec)



Step 2: Architectural Elements

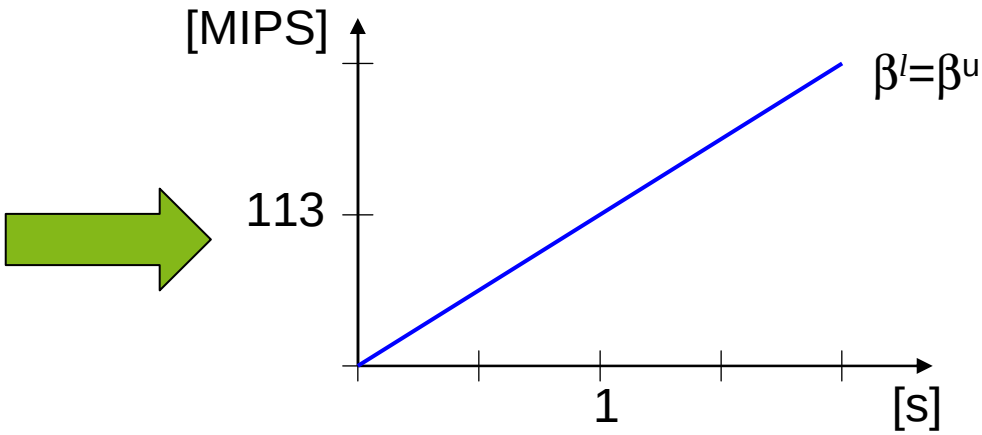
Event Stream Model

e.g. Address Lookup
(1 event / sec)



Resource Model

e.g. unloaded RISC CPU
(113 MIPS)

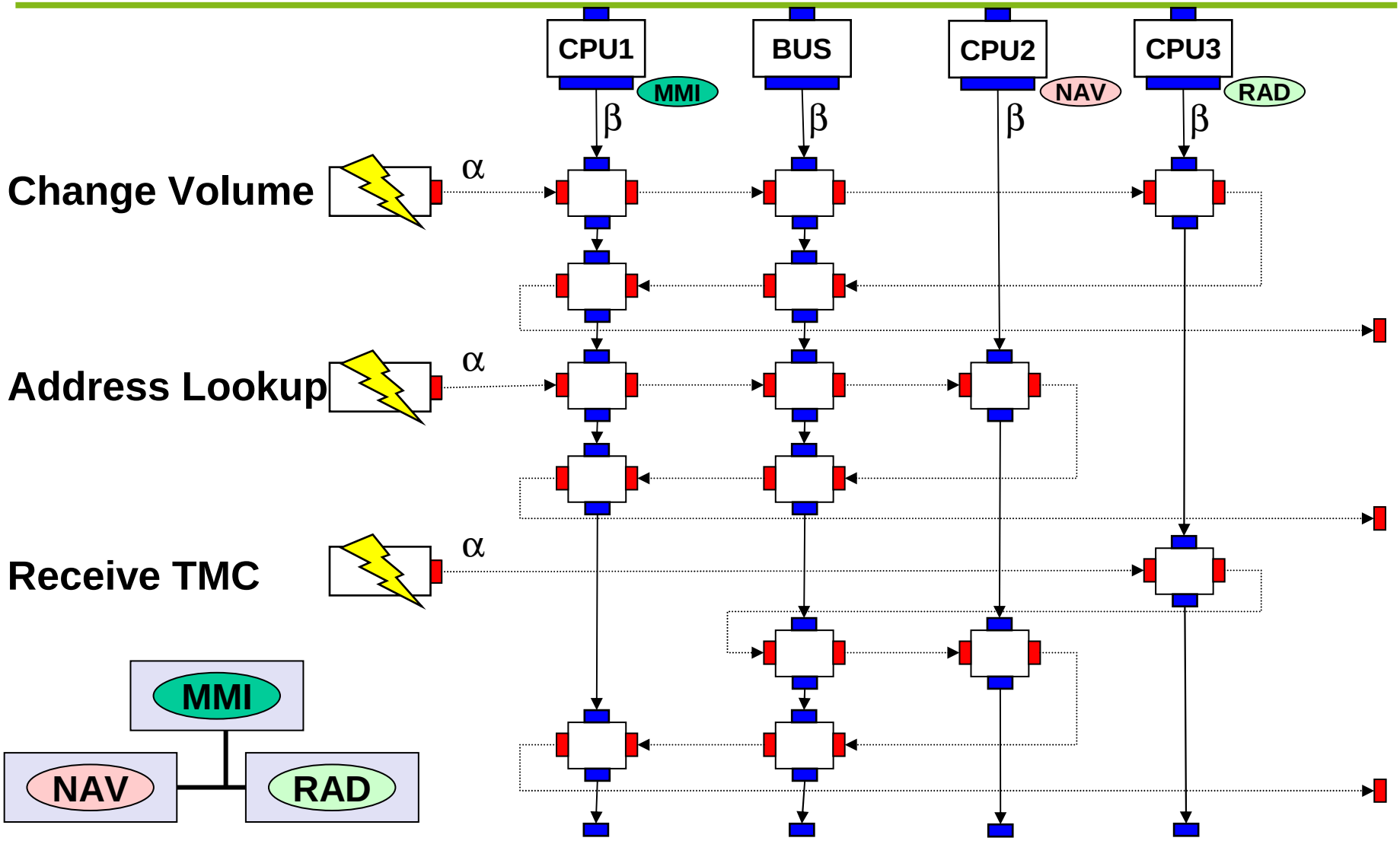


Step 3: Mapping / Scheduling

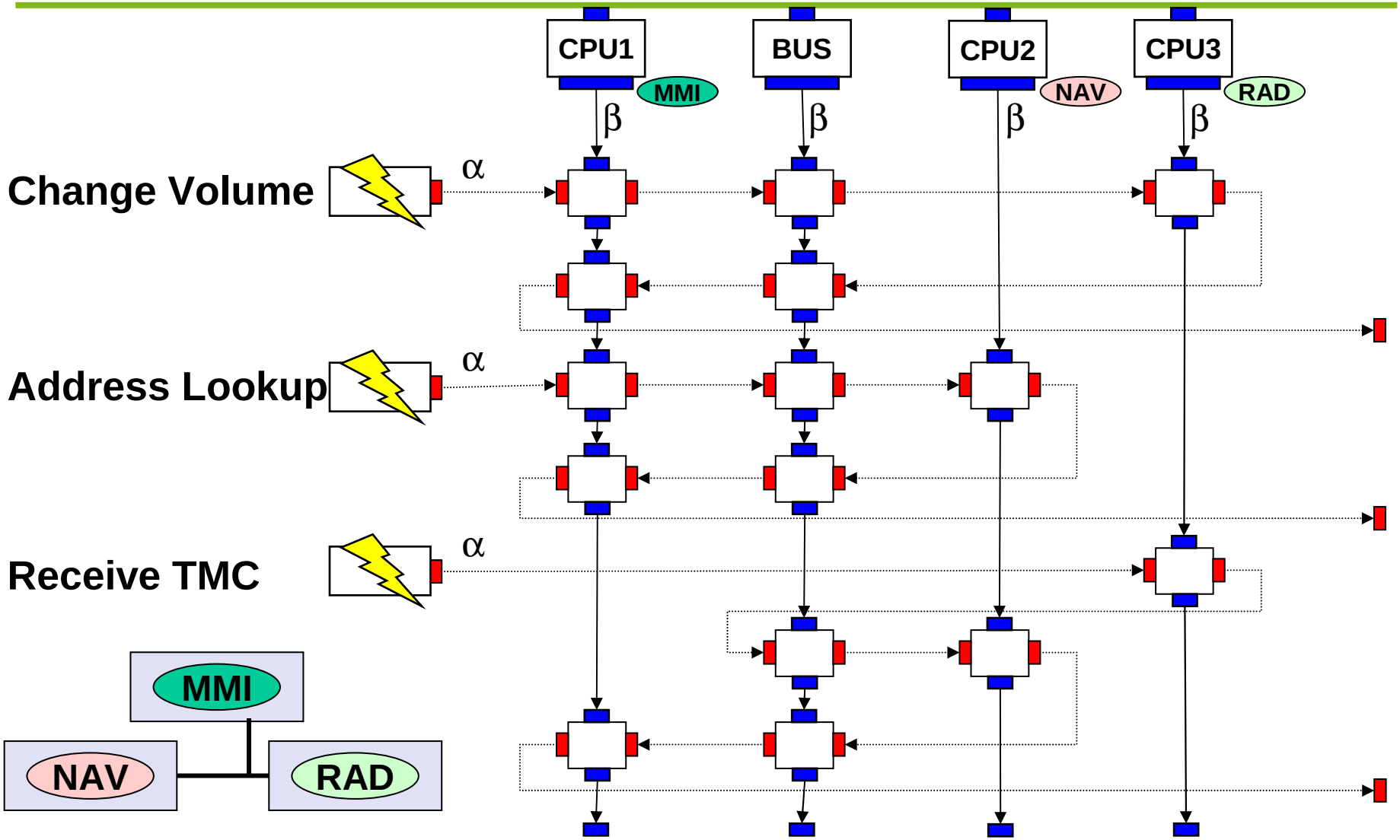
Rate Monotonic Scheduling
(Pre-emptive fixed priority scheduling):

- Priority 1: Change Volume (p=1/32 s)
- Priority 2: Address Lookup (p=1 s)
- Priority 3: Receive TMC (p=6 s)

Step 4: Performance Model



Step 5: Analysis

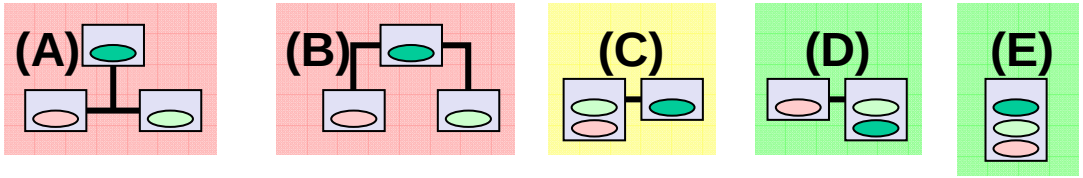


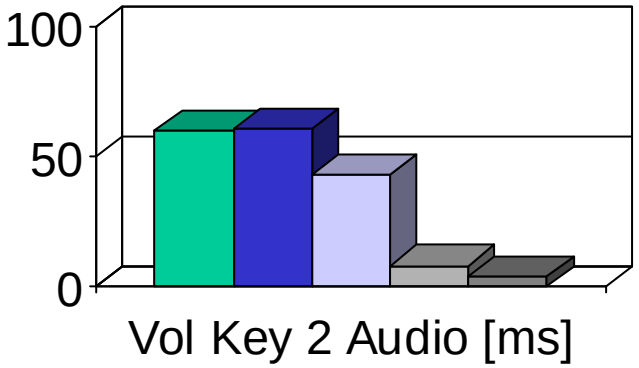
Analysis – Design Question 1

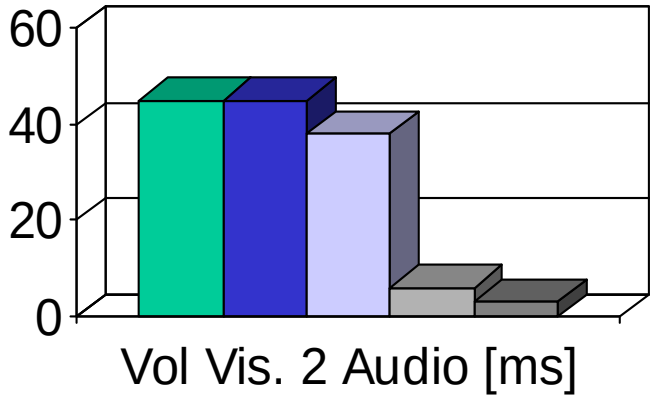
How do the proposed system architectures compare in respect to end-to-end delays?

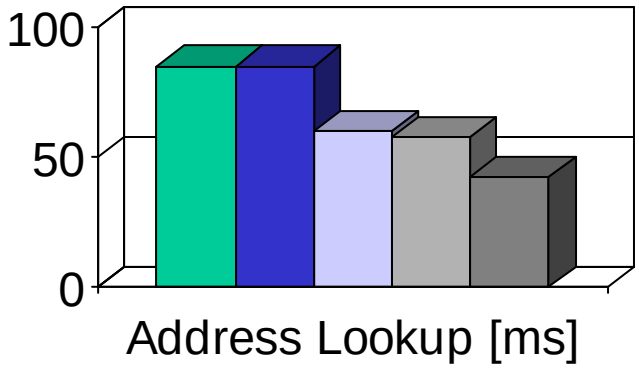
Analysis – Design Question 1

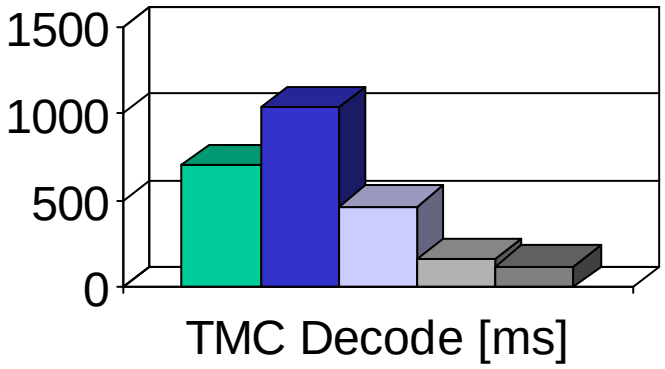
End-to-end delays:







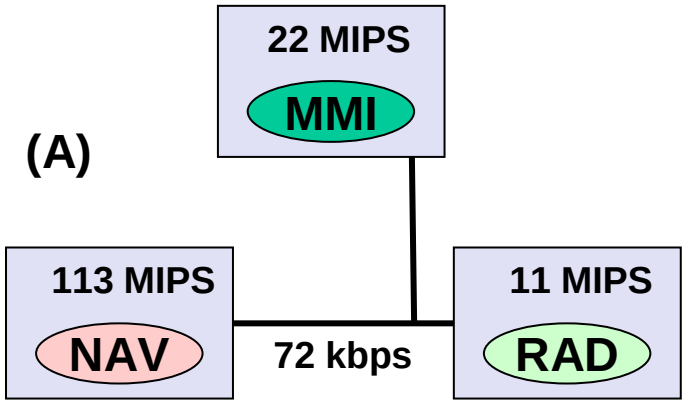




Analysis – Design Question 2

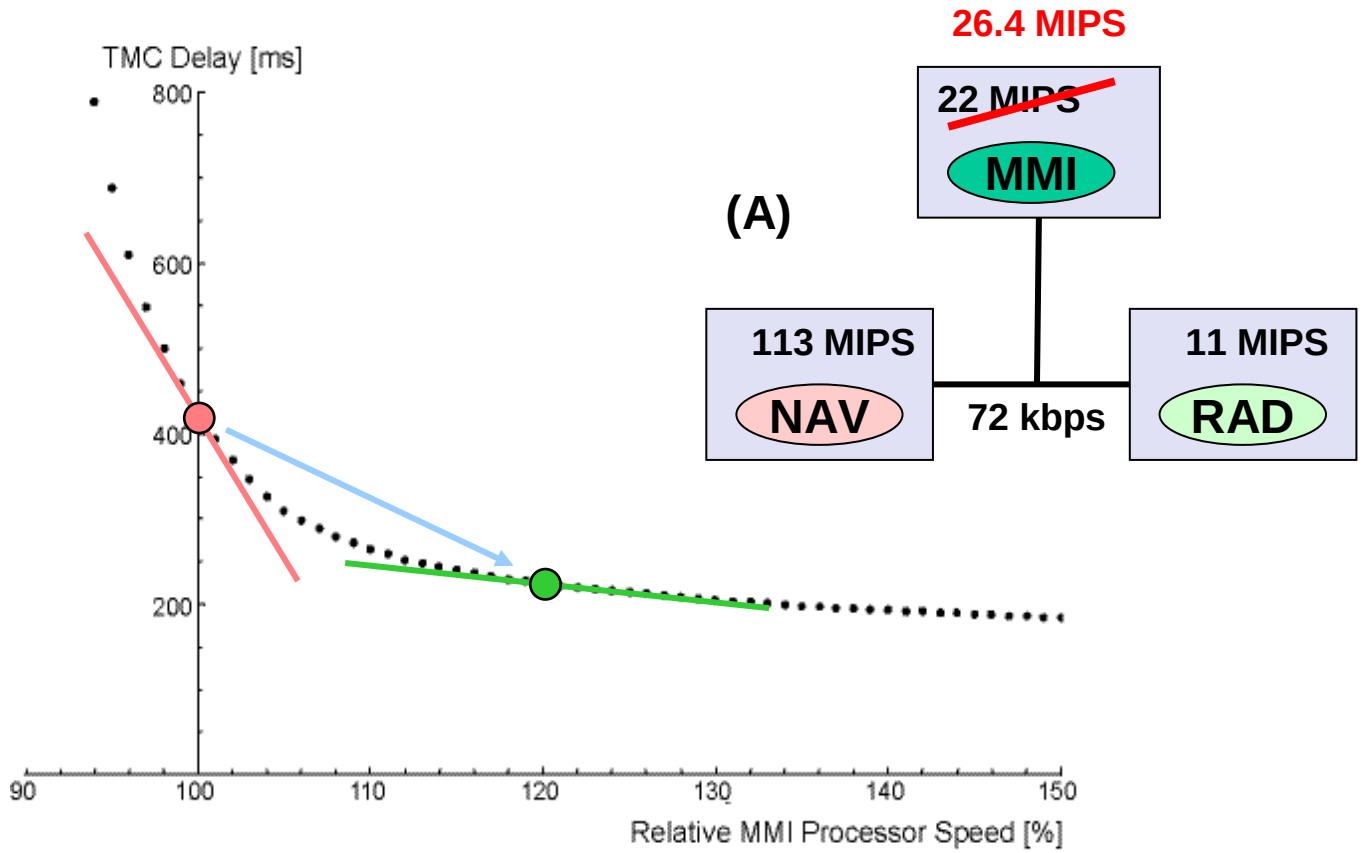
How robust is architecture A?

Where is the bottleneck of this architecture?



Analysis – Design Question 2

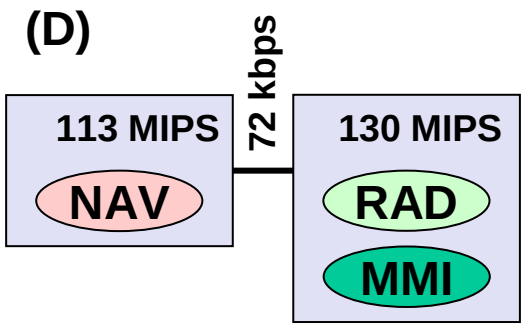
TMC delay vs. MMI processor speed:



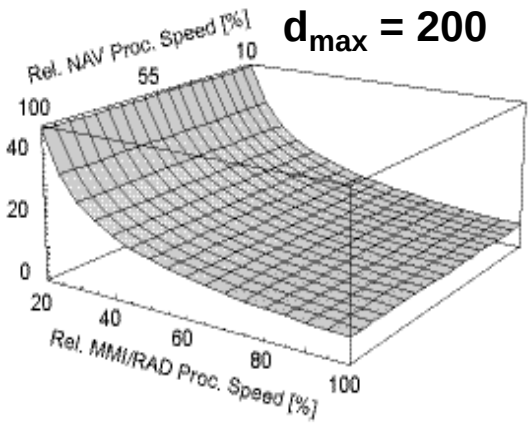
Analysis – Design Question 3

Architecture D is chosen for further investigation.

How should the processors be dimensioned?

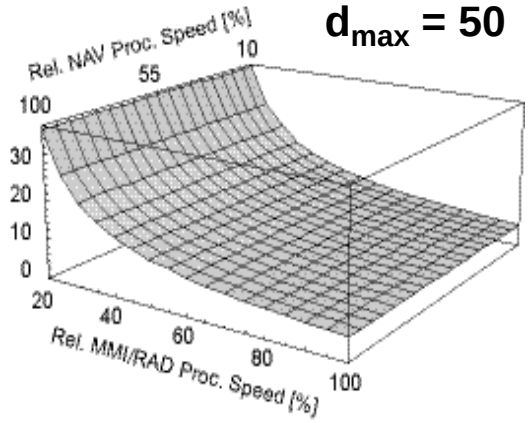


Analysis – Design Question 3



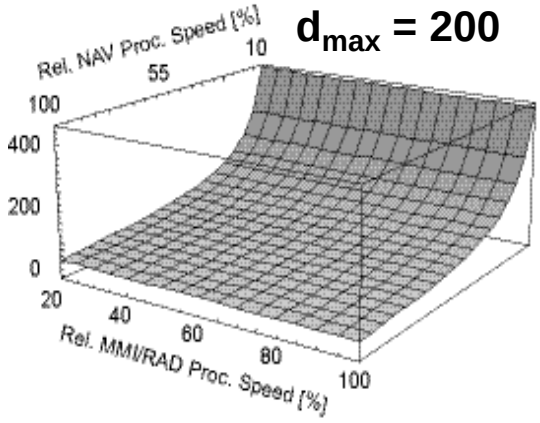
d_{max} = 200

Vol K2V Delay [ms]



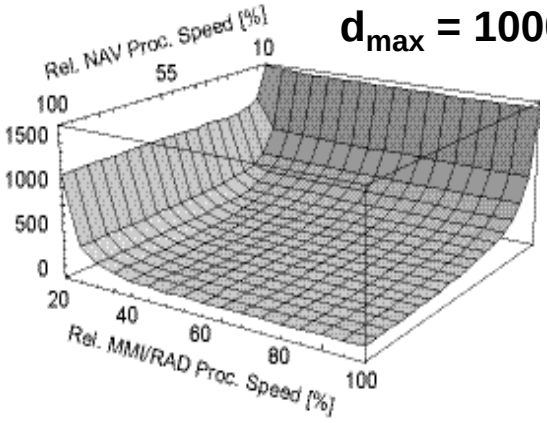
d_{max} = 50

Vol A2V Delay [ms]



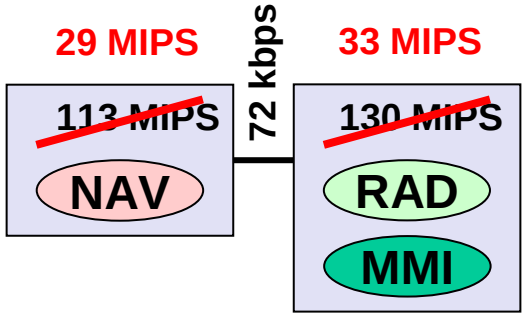
d_{max} = 200

Addr Delay [ms]



d_{max} = 1000

TMC Delay [ms]



Conclusions

- Easy to construct models (~ half day)
- Evaluation speed is fast and linear to model complexity (~ 1s per evaluation)
- Needs little information to construct early models (Fits early design cycle very well)
- Even though involved mathematics is very complex, the method is easy to use (Language of engineers)

Acknowledgement and References

The presentation contains contributions by

- Samarjit Chakraborty (NUS)
- Simon Künzli, Ernesto Wandeler, Alexander Maxiaguine (ETHZ)
- Andreas Herkersdorf, Patricia Sagmeister (IBM)
- Jonas Greutert (Netmodule)

Many publications are available from
http://www.tik.ee.ethz.ch/~thiele

SYMTA

SYMTA (Ernst, TU Braunschweig) works with standard streams (periodic, periodic with jitter streams) and focuses on computing end-to-end guarantees in multiprocessor based systems

See www.symtavision.com

Summary

Evaluation and Validation

- In general, multiple objectives
- Pareto optimality
- Design space evaluation (DSE)
- WCET estimation
- Real-time calculus

Summary

- Performance analysis
  - Trade-off between speed and accuracy
  - Computation of worst case execution times
    - Cache/pipeline analysis
    - ILP model for computing WCET of application from WCET of blocks
  - Thiele's real-time calculus (RTC)/MPA
    - Using bounds on the number of events in input streams
    - Using bounds on available processing capability
    - Derives bounds on the number of events in output streams
    - Derives bound on remaining processing capability, buffer sizes, …
    - Examples demonstrate design procedure based on RTC