technische universität
dortmund

fakultät für informatik
informatik 12

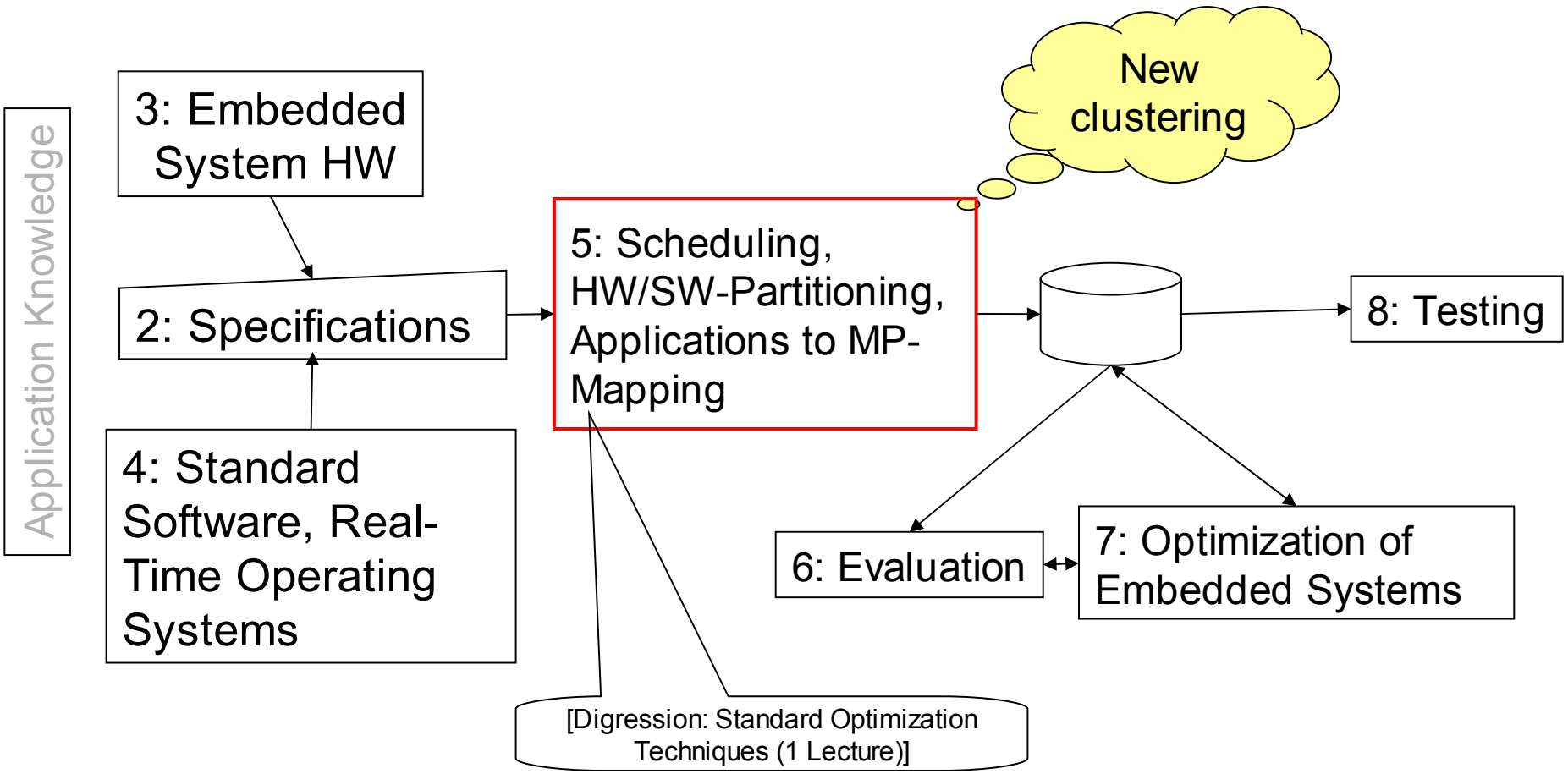
Evaluation and Validation

Peter Marwedel
TU Dortmund, Informatik 12
Germany

2008/12/28

Graphics: © Alexandra Nolte, Gesine Marwedel, 2003

Structure of this course



Evaluation and Validation

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

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*.

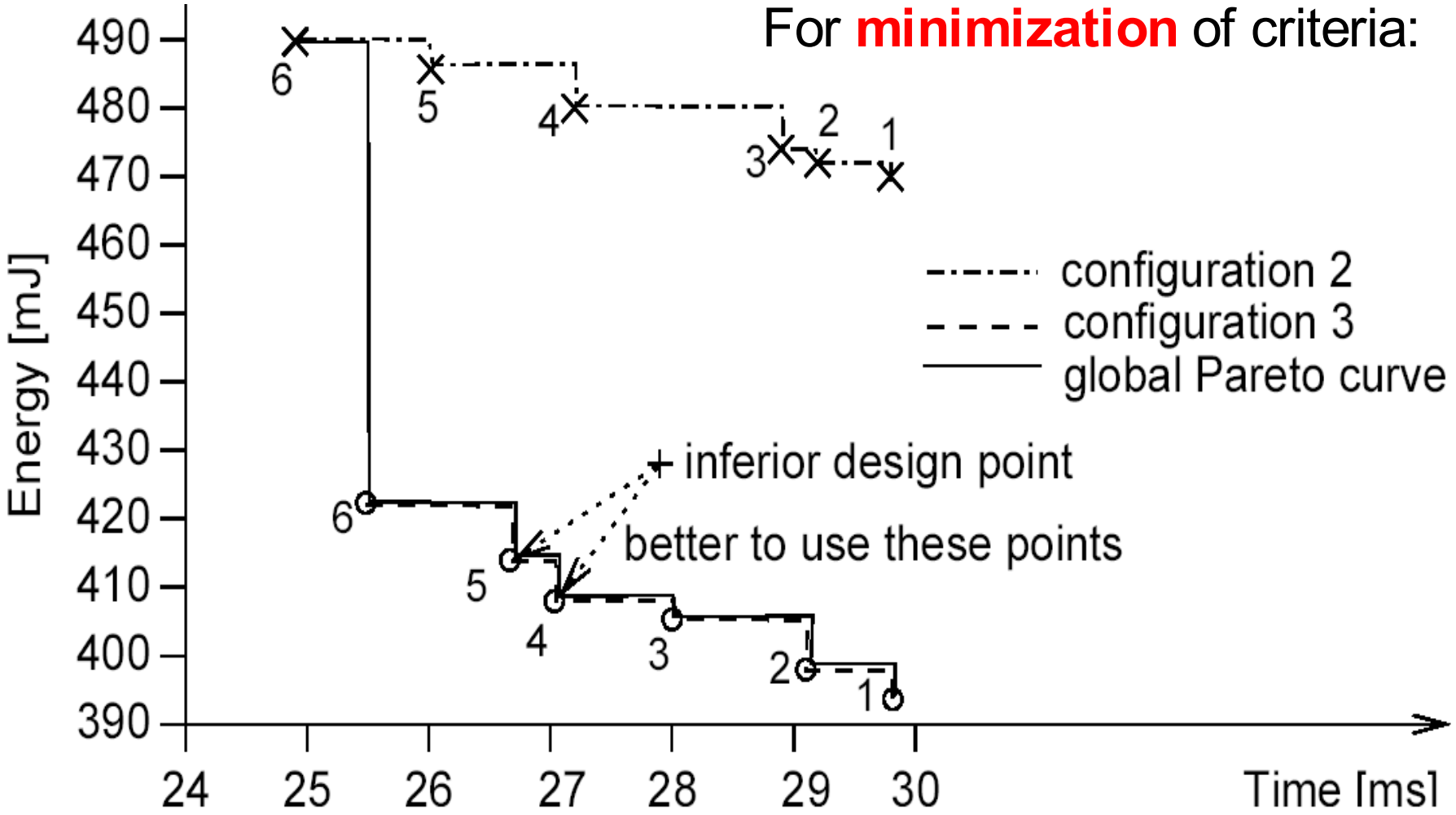
How to evaluate designs according to multiple criteria?

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

- (average) speed
- worst case speed
- power consumption
- cost
- size
- weight
- radiation hardness
- environmental friendliness ….

How to compare different designs?
(Some designs are “better” than others)

Pareto curves



Pareto points

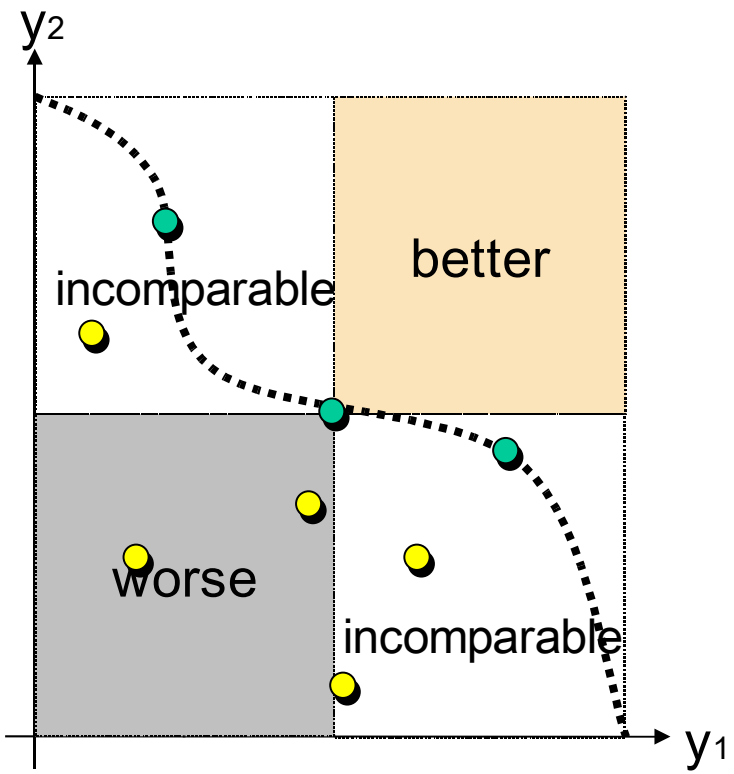
Definition: A (design) point J_i is **dominated** by point J_k, if J_k is equal or better than J_i in each criterion ($J_i \leq J_k$).

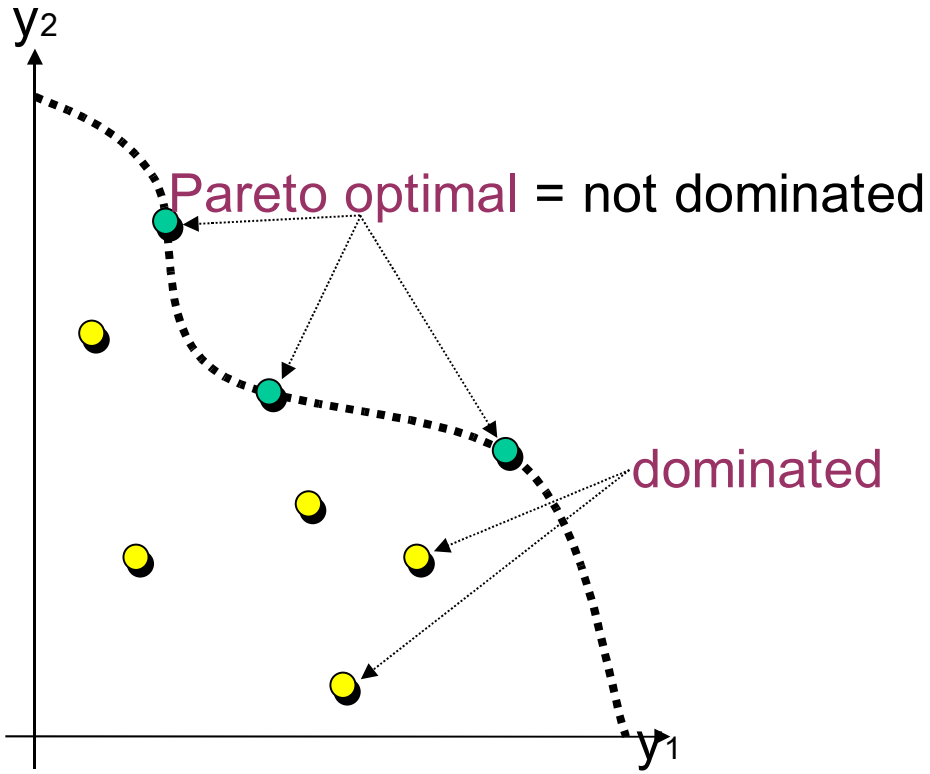
Definition: A (design) point is **Pareto-optimal** or a **Pareto point**, if it is not dominated by any other point.

Multiobjective Optimization

For **maximization** of criteria:

Maximize $(y_1, y_2, \ldots, y_k) = f(x_1, x_2, \ldots, x_n)$





Pareto set = set of all Pareto-optimal solutions

Simulations

- Simulations try to imitate the behavior of the real system on a (typically digital) computer.
- Simulation of the functional behavior requires executable models.
- Simulations can be performed at various levels.
- Some non-functional properties (e.g. temperatures, EMC) can also be simulated.
- Simulations can be used to **evaluate** and to **validate** a design

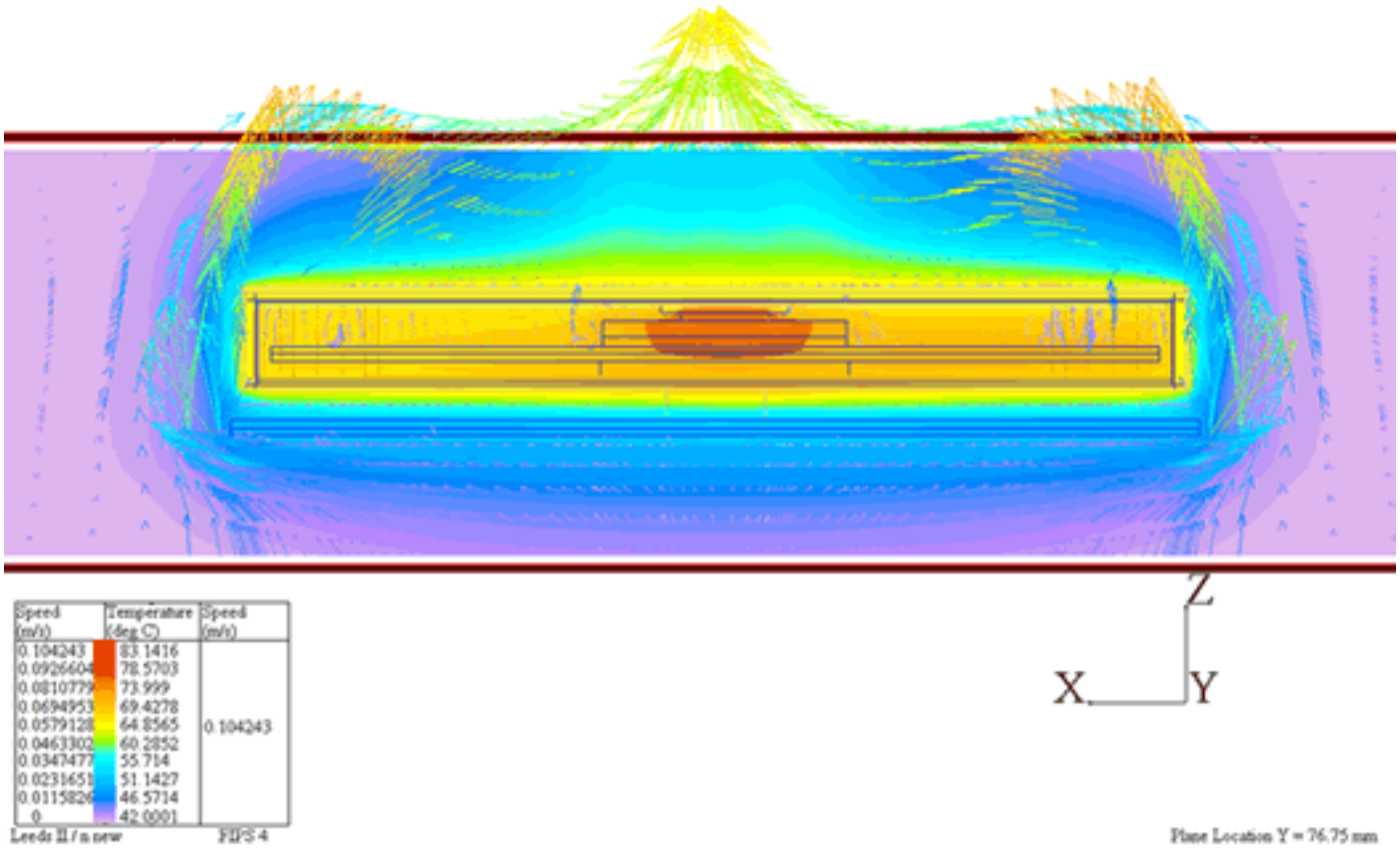
Validating functional behavior by simulation

Various levels of abstractions used for simulations:

- High-level of abstraction: fast, but sometimes not accurate
- Lower level of abstraction: slow and typically accurate
- Choosing a level is always a compromise

Non-functional behavior: Examples of thermal simulations (1)

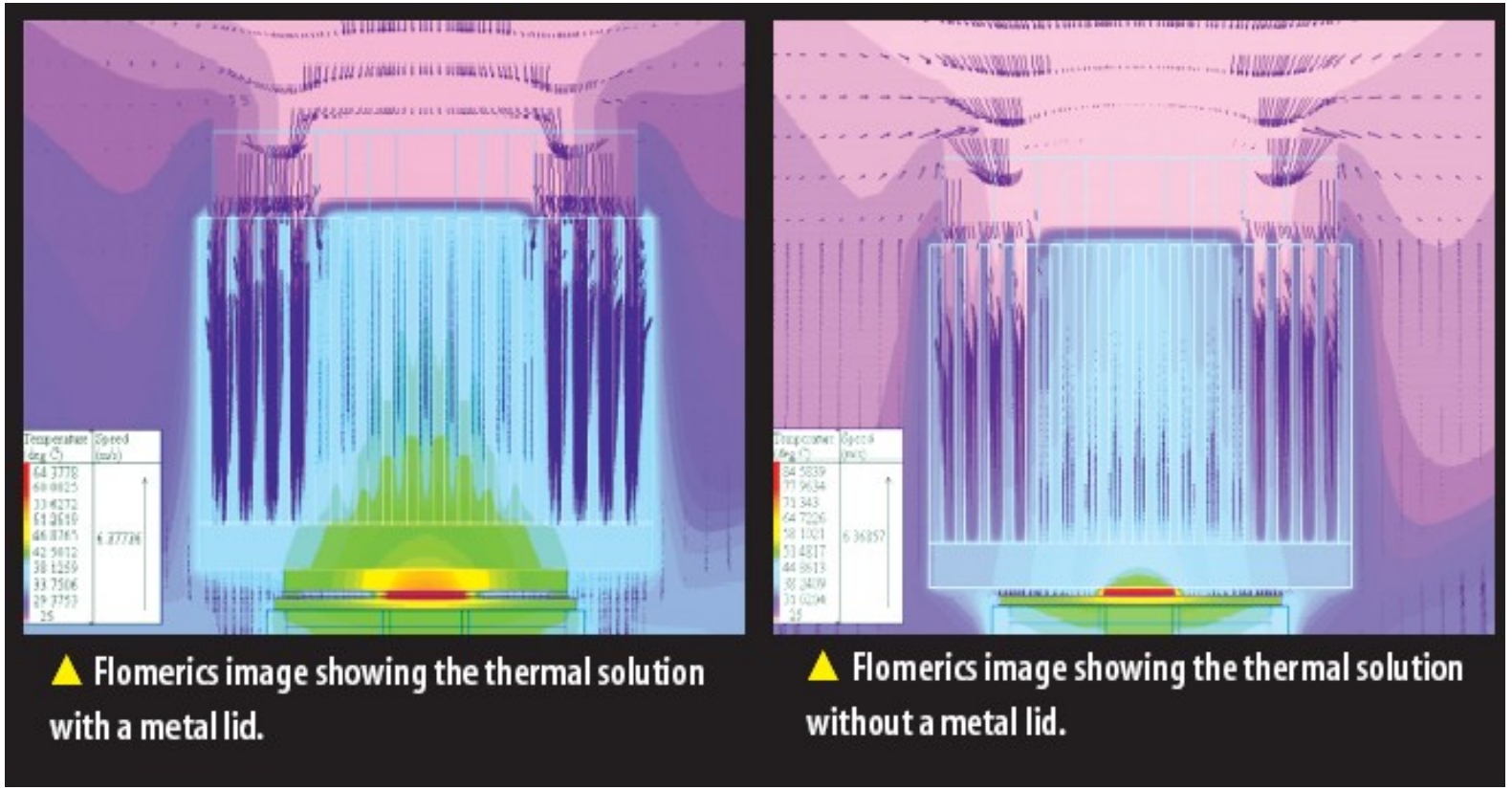
Encapsulated cryptographic coprocessor:



Source: http://www.coolingzone.com/Guest/News/NL_JUN_2001/Campi/Jun_Campi_2001.html

Examples of thermal simulations (2)

Microprocessor



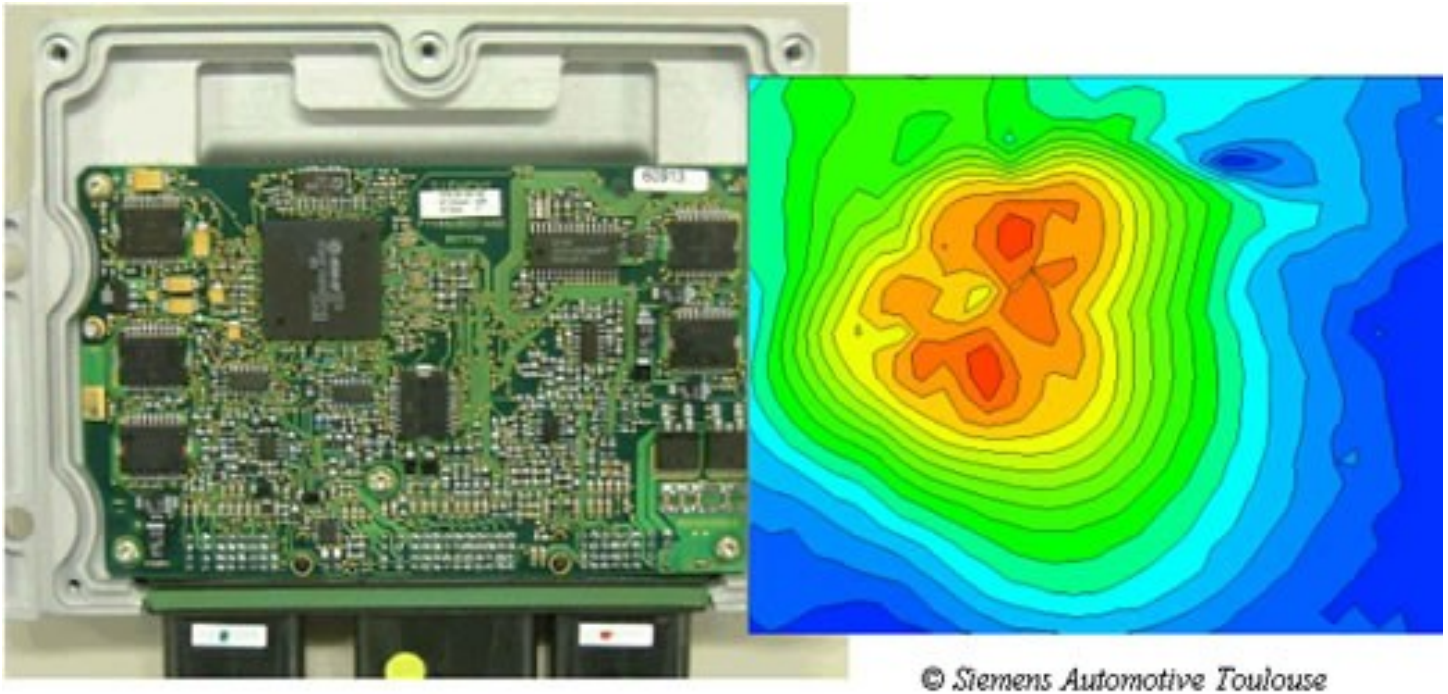
▲ Flomerics image showing the thermal solution with a metal lid.

▲ Flomerics image showing the thermal solution without a metal lid.

Source: http://www.flotherm.com/applications/app141/hot_chip.pdf

EMC simulation

Example: car engine controller



© Siemens Automotive Toulouse

Red: high emission
Validation of EMC properties often done at the end of the design phase.

Source: http://intrage.insa-tlse.fr/~etienne/emccourse/what_for.html

Simulations
Limitations

- Typically slower than the actual design.
  ☞ **Violations of timing constraints** likely if simulator is connected to the actual environment

- Simulations in the real environment may be **dangerous**

- There may be huge amounts of data and it may be impossible to simulate enough data in the available time.

- Most actual systems are too complex to allow simulating all possible cases (inputs).
  Simulations can help finding errors in designs, but they cannot guarantee the absence of errors.

Rapid prototyping/Emulation

- Prototype: Embedded system that can be generated quickly and behaves very similar to the final product.
- May be larger, more power consuming and have other properties that can be accepted in the validation phase
- Can be built, for example, using FPGAs.

Example: Quickturn Cobalt System (1997), ~0.5M$ for 500kgate entry level system

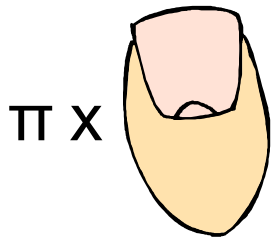
Source & ©: http://www.eedesign. com/editorial/1997/toolsandtech9703.html

Example of a more recent commercial emulator

[www.verisity.com/images/products/xtremep{1|3}.gif]

Performance evaluation

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



- **Accurate cost and performance values:**
  Can be done with normal tools (such as compilers).
  As precise as the input data is.

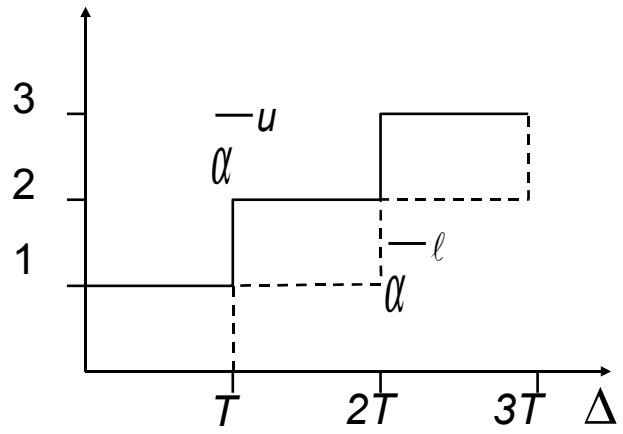
We need to compute average and worst case execution times

Thiele's real-time calculus
- Arrival curves -

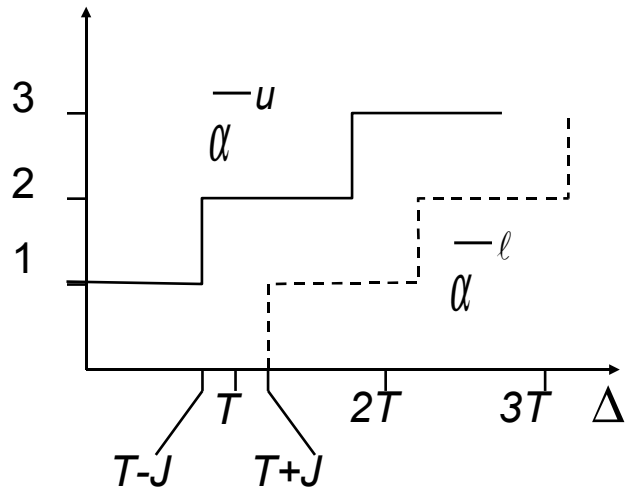
Arrival curves describe the maximum and minimum number of events arriving in some time interval Δ

Examples

periodic event stream



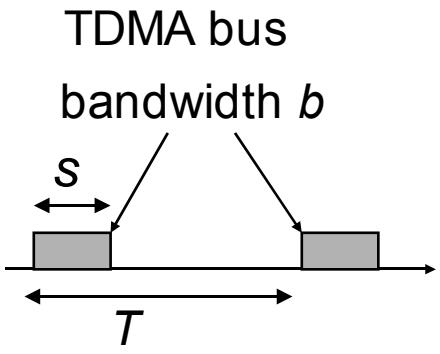
periodic event stream with jitter

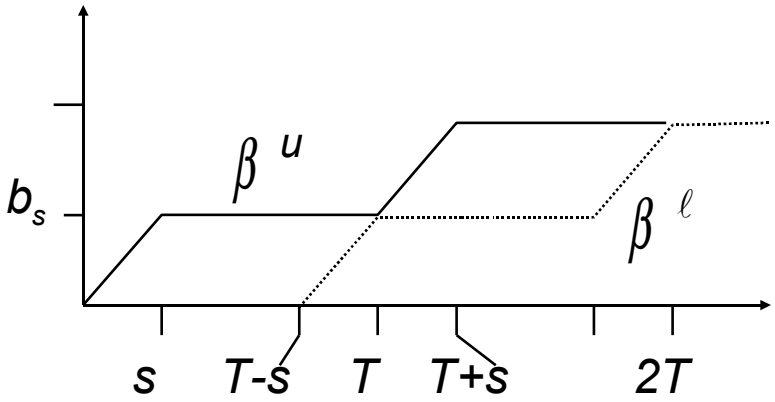


Thiele's real-time calculus
- Service curves -

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

Example:

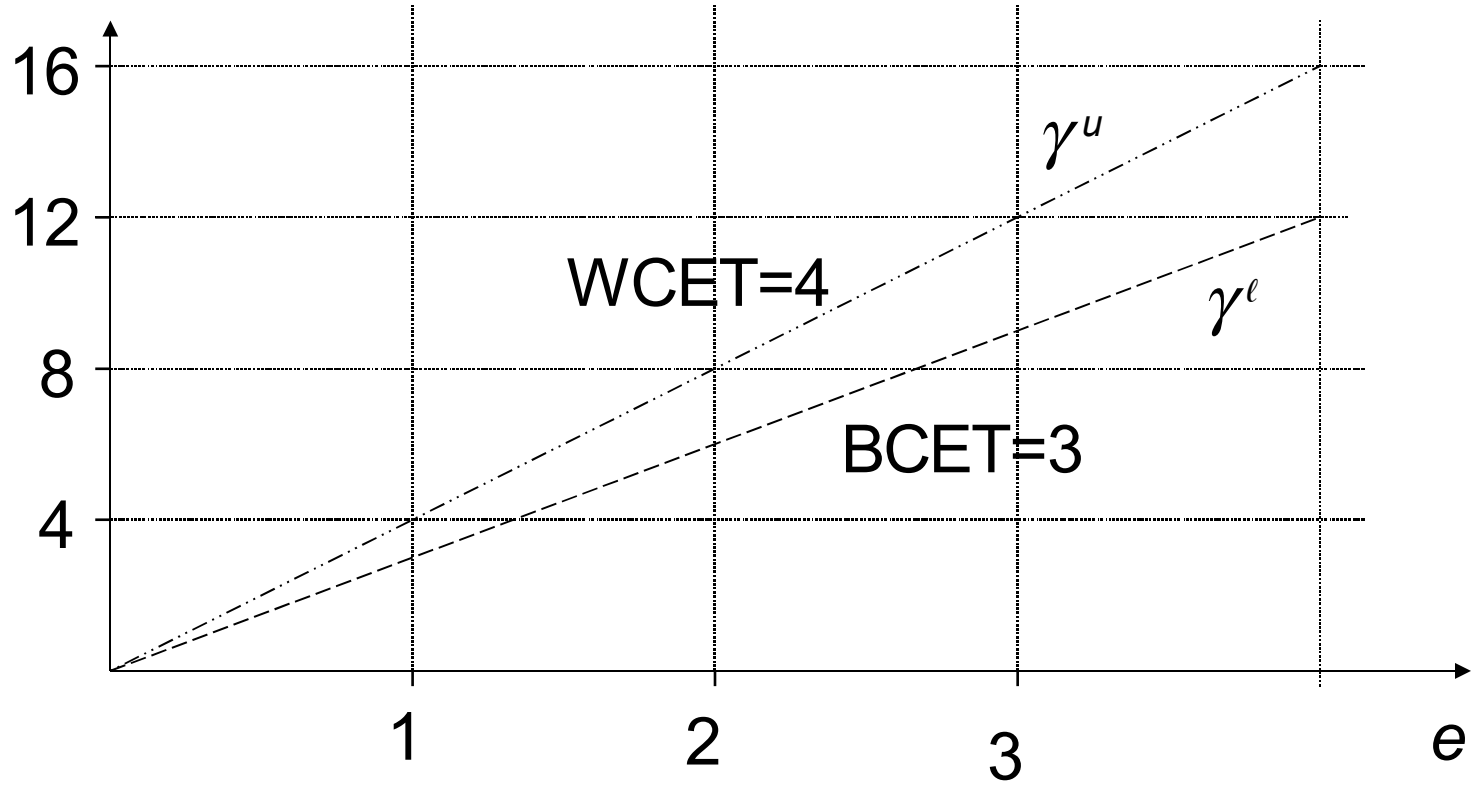




Thiele's real-time calculus
- Workload characterization -

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

Example



Workload required for incoming stream

Incoming workload

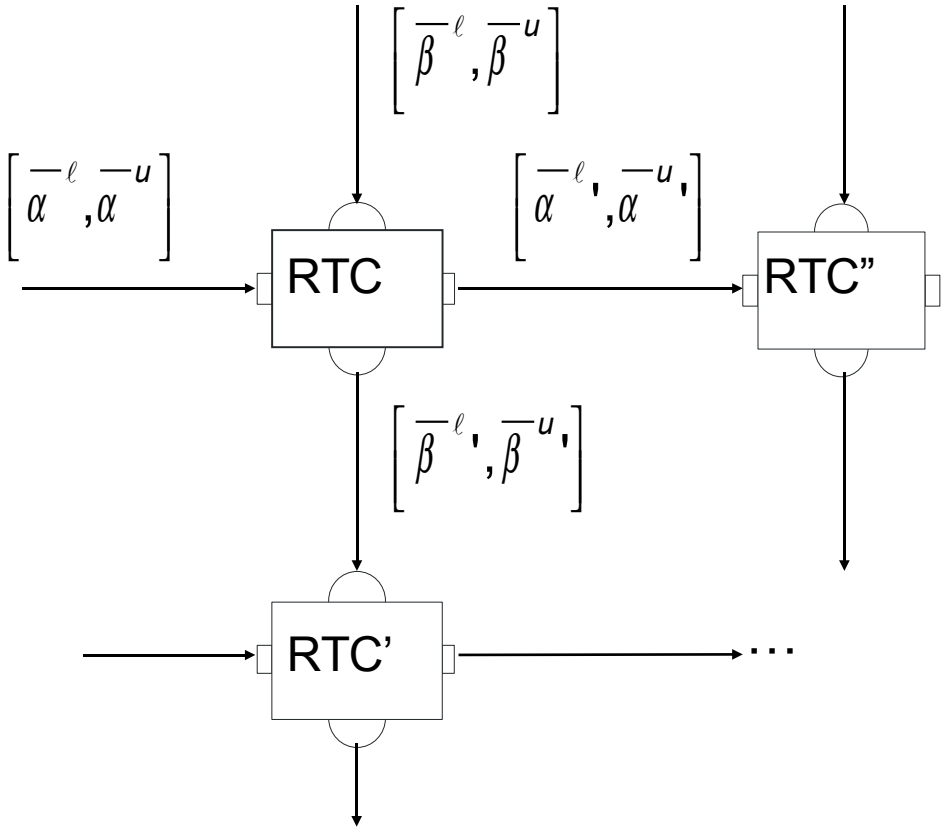
$$\alpha^{u}(\Delta) = \gamma^{u}\left(\overline{\alpha^{u}}(\Delta)\right) \qquad \alpha^{\ell}(\Delta) = \gamma^{\ell}\left(\overline{\alpha^{\ell}}(\Delta)\right)$$

Upper and lower bounds on the number of events

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

Thiele's real-time calculus
- 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 ☞

Thiele's real-time calculus
- Transformation of arrival and service curves -

Resulting arrival curves:

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

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

Remaining service curves:

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

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

Where:

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

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

Thiele's real-time calculus
Remarks

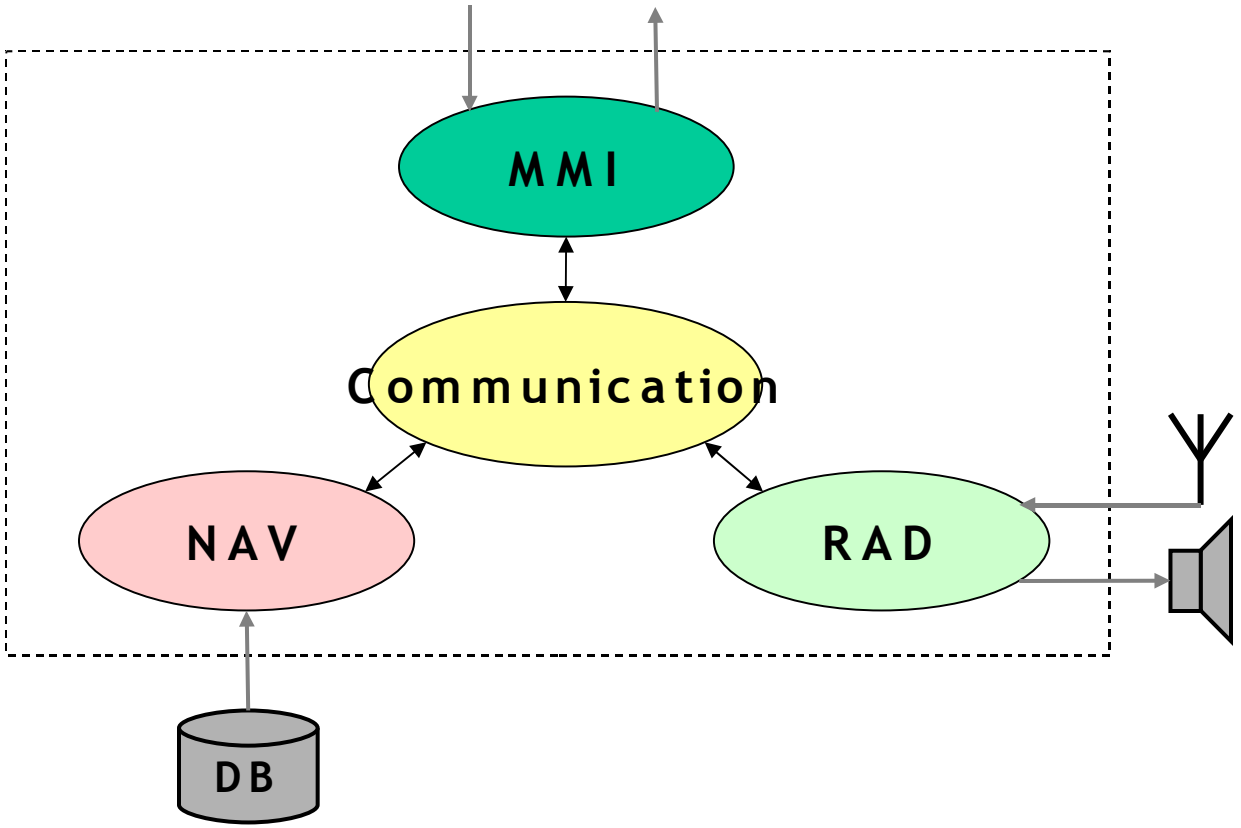
- Details of the proofs can be found in relevant references
- Results also include bounds on buffer sizes and on maximum latency.
- Theory has been extended into various directions, e.g. for computing remaining battery capacities

Application: In-Car Navigation System

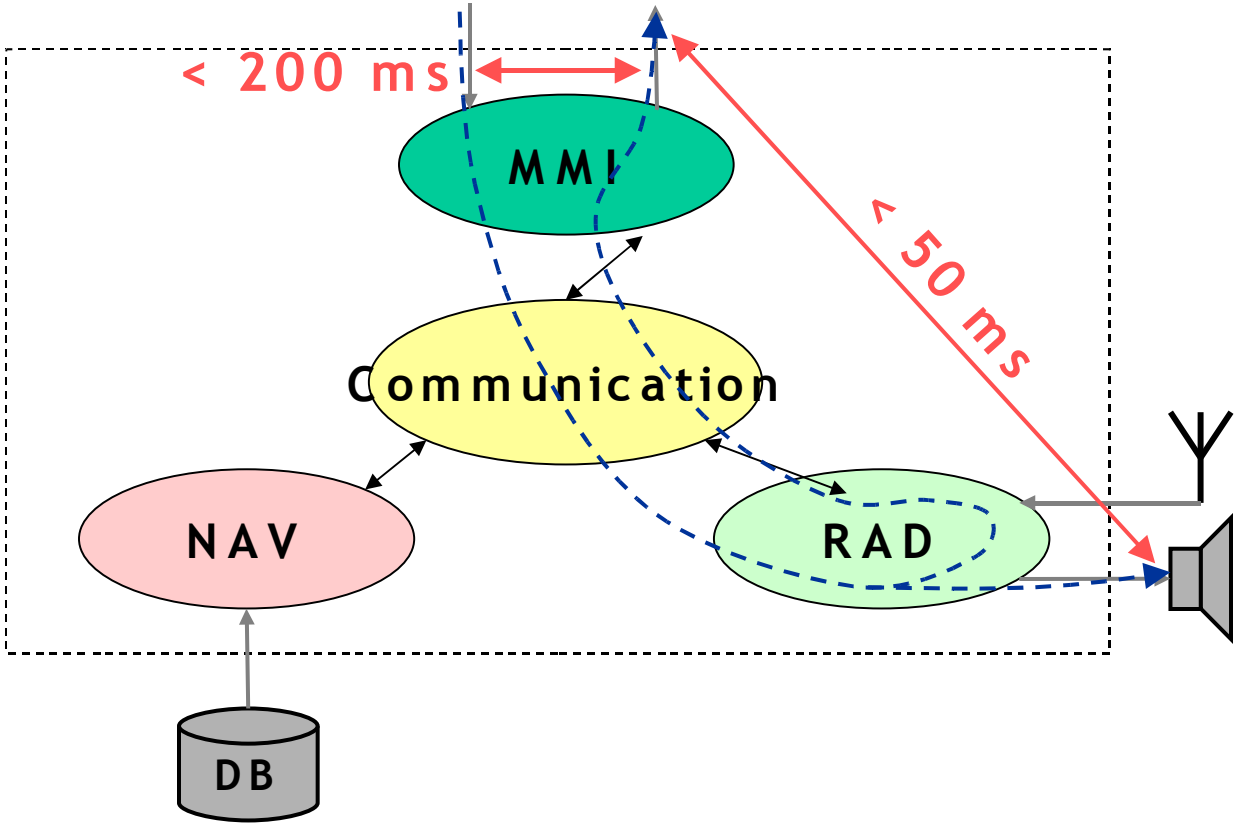
Car radio with navigation system
User interface needs to be responsive
Traffic messages (TMC) must be processed in a timely way
Several applications may execute concurrently

© Thiele, ETHZ

System Overview



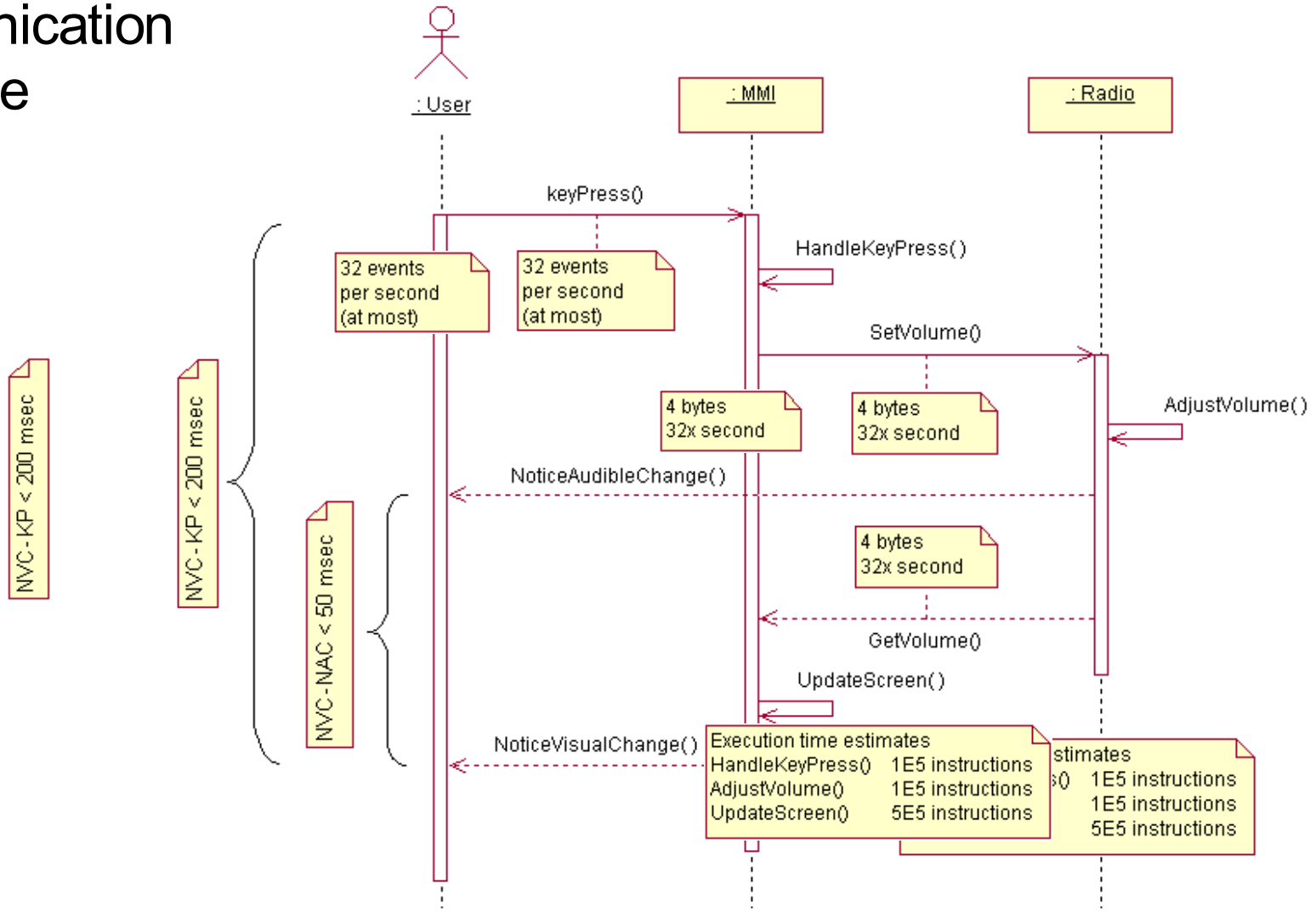
Use case 1: Change Audio Volume



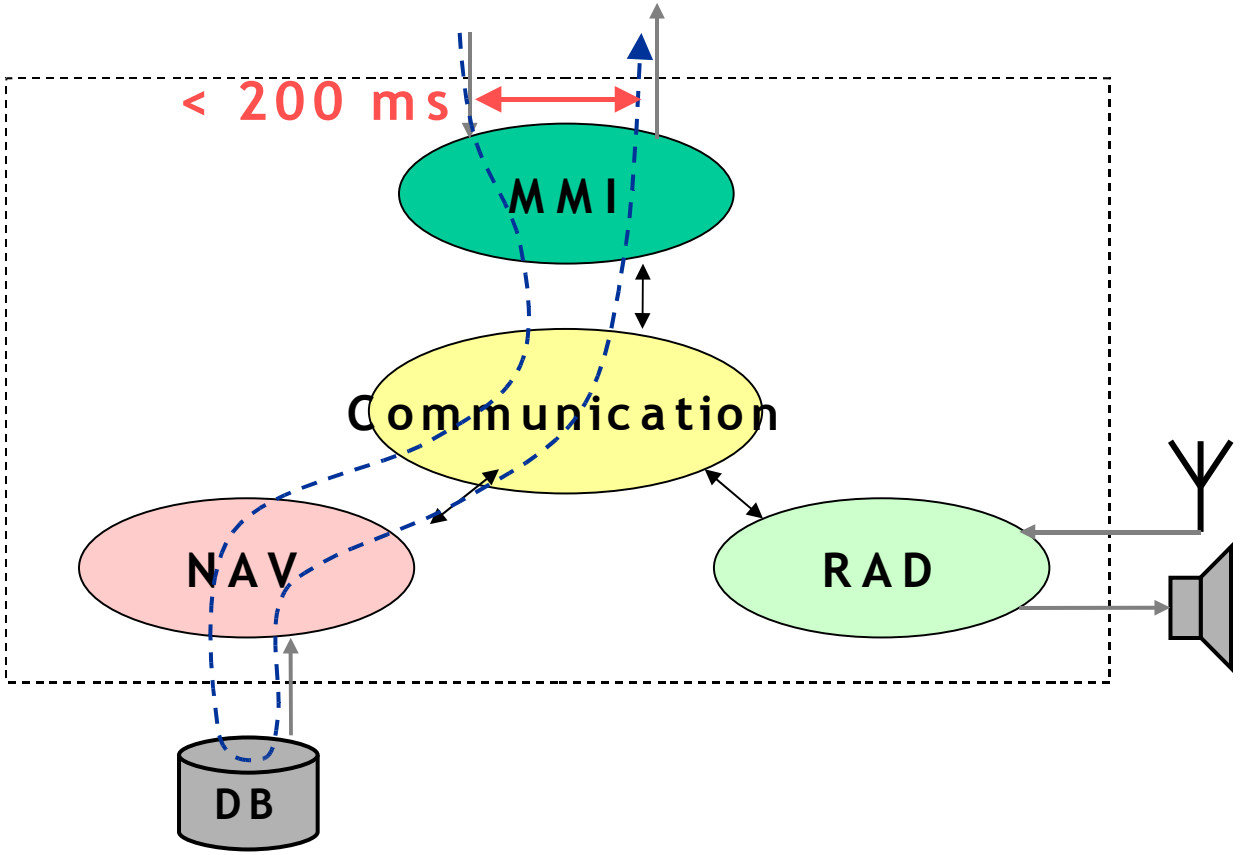
© Thiele, ETHZ

Use case 1: Change Audio Volume

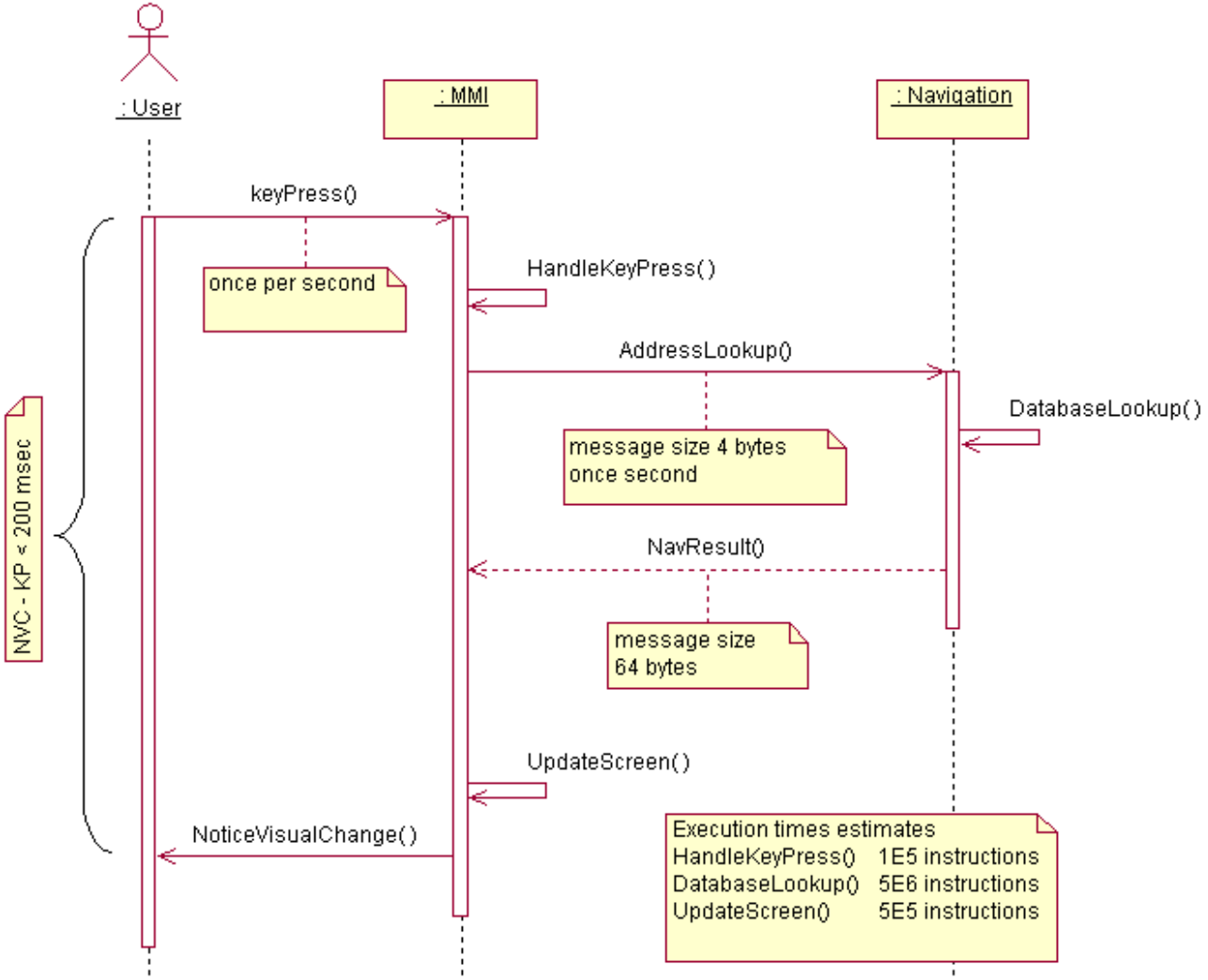
Communication Resource Demand



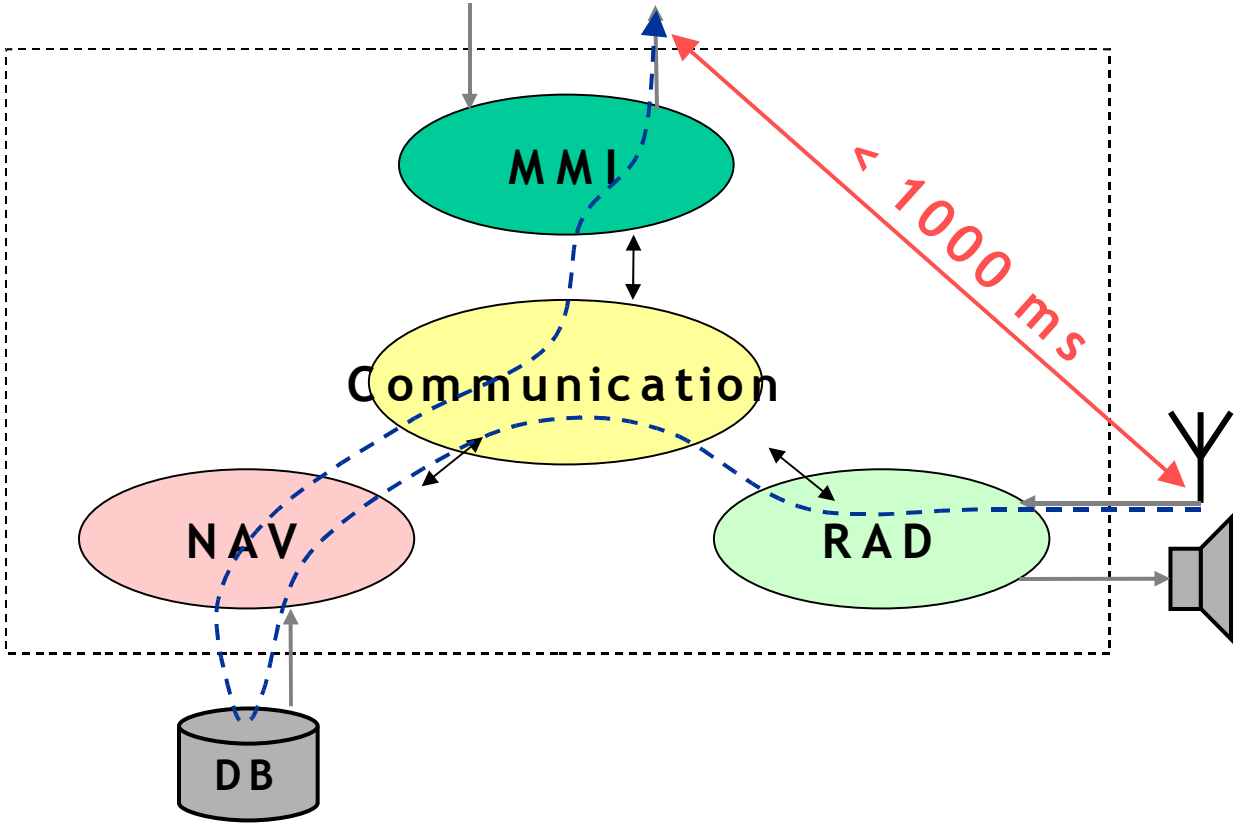
Use case 2: Lookup Destination Address



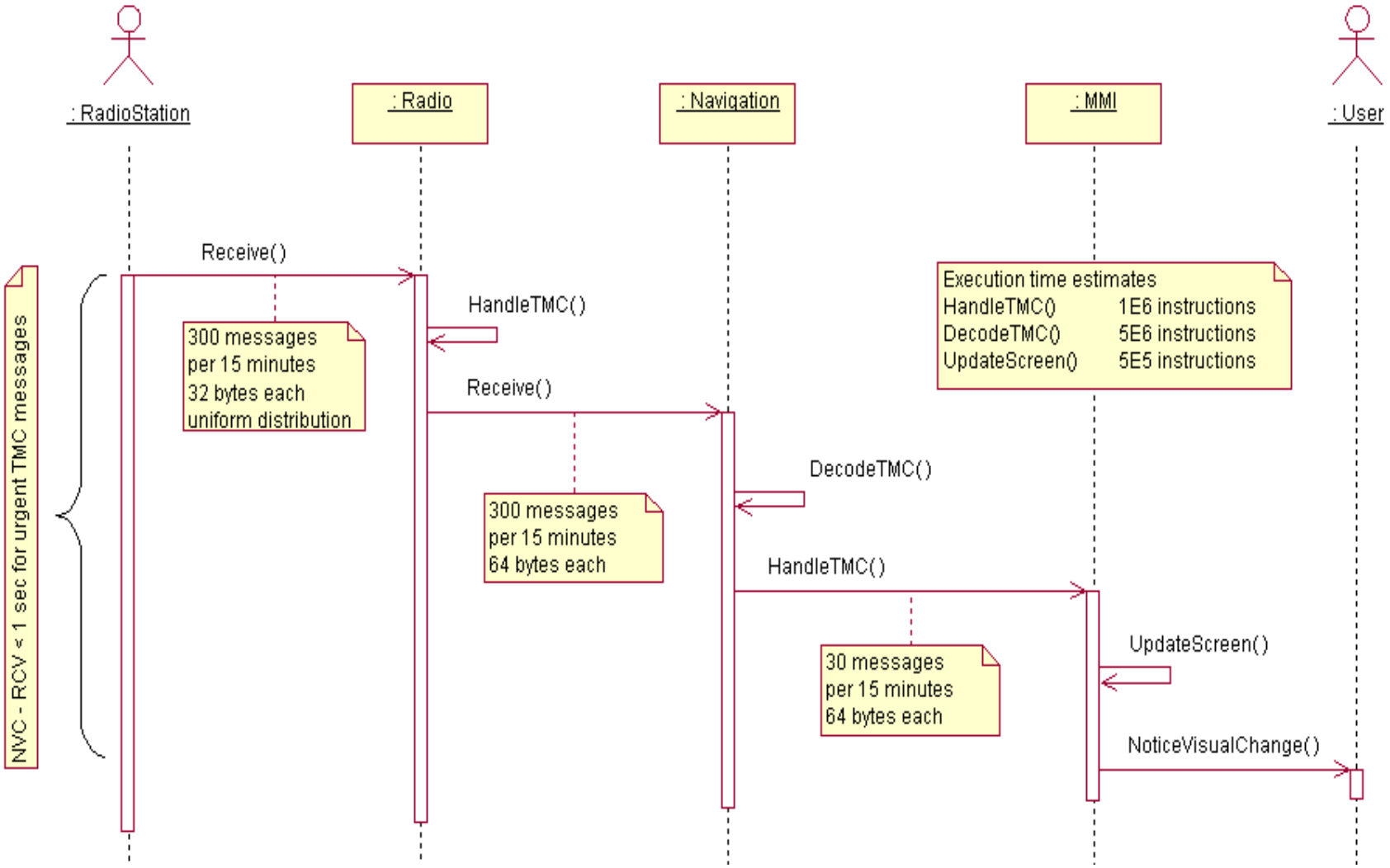
Use case 2: Lookup Destination Address



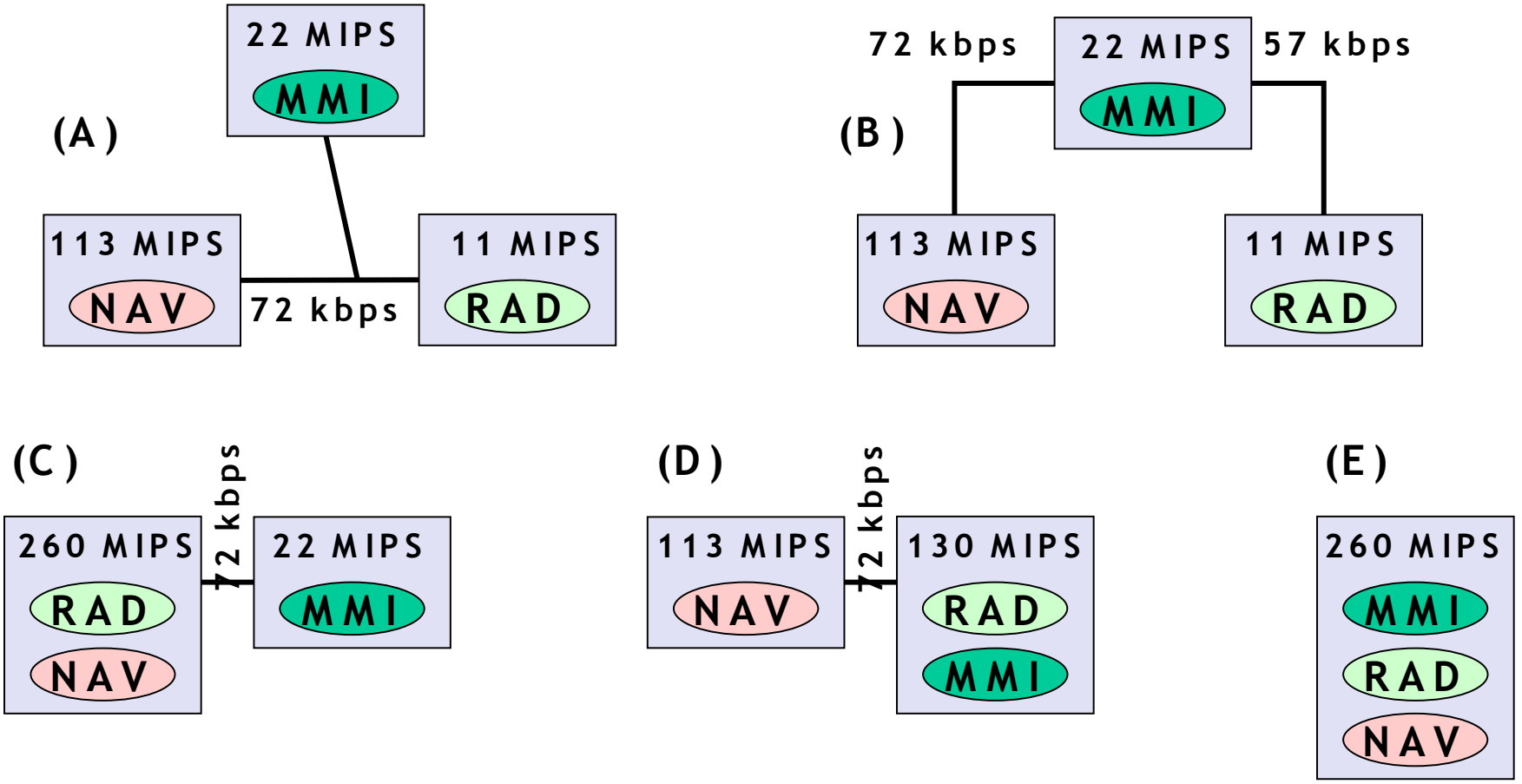
Use case 3: Receive TMC Messages



Use case 3: Receive TMC Messages



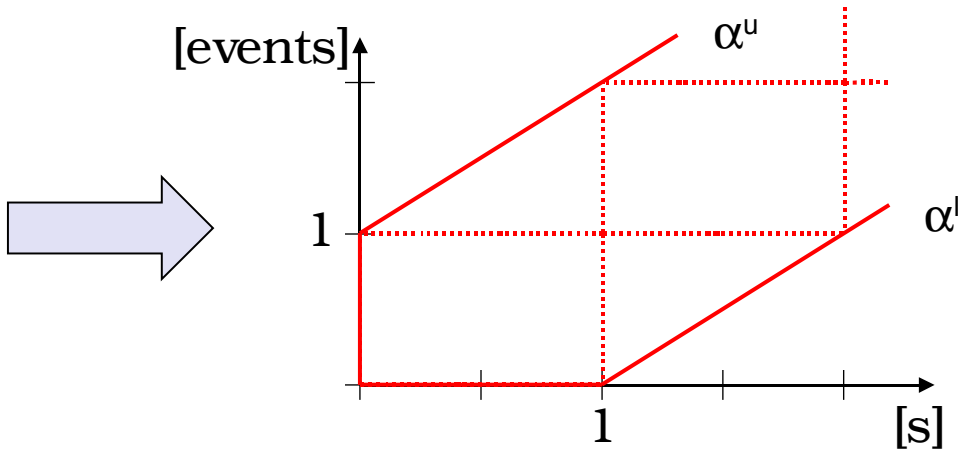
Proposed Architecture Alternatives



Step 1: Environment (Event Steams)

Event Stream Model

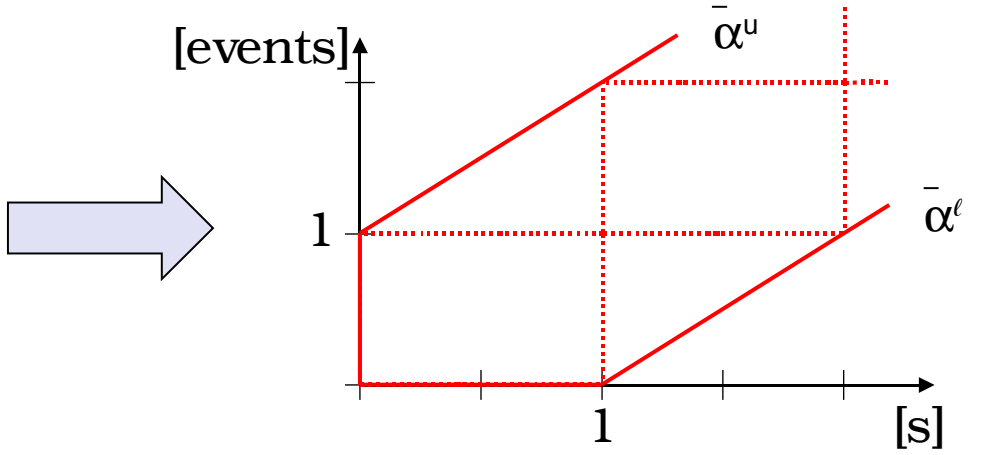
e.g. Address Lookup
(1 event / sec)



Step 2: Architectural Elements

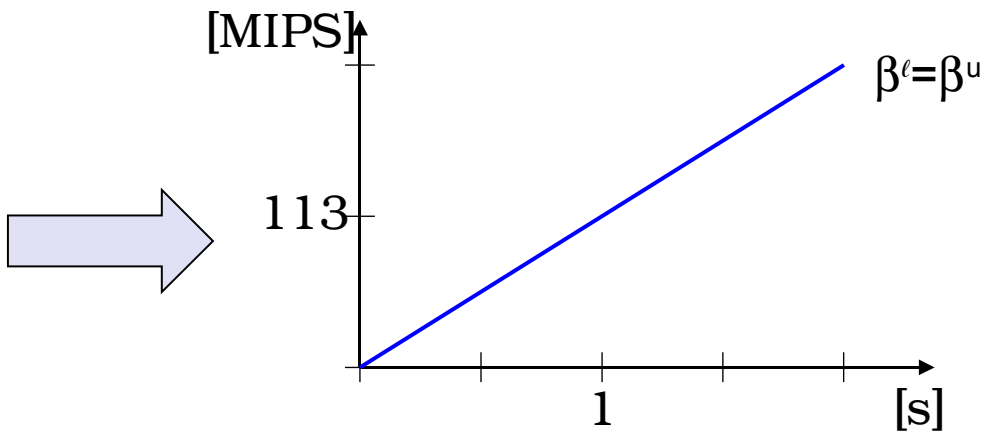
Event Stream Model

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



Resource Model

e.g. unloaded RISC CPU
(113 MIPS)

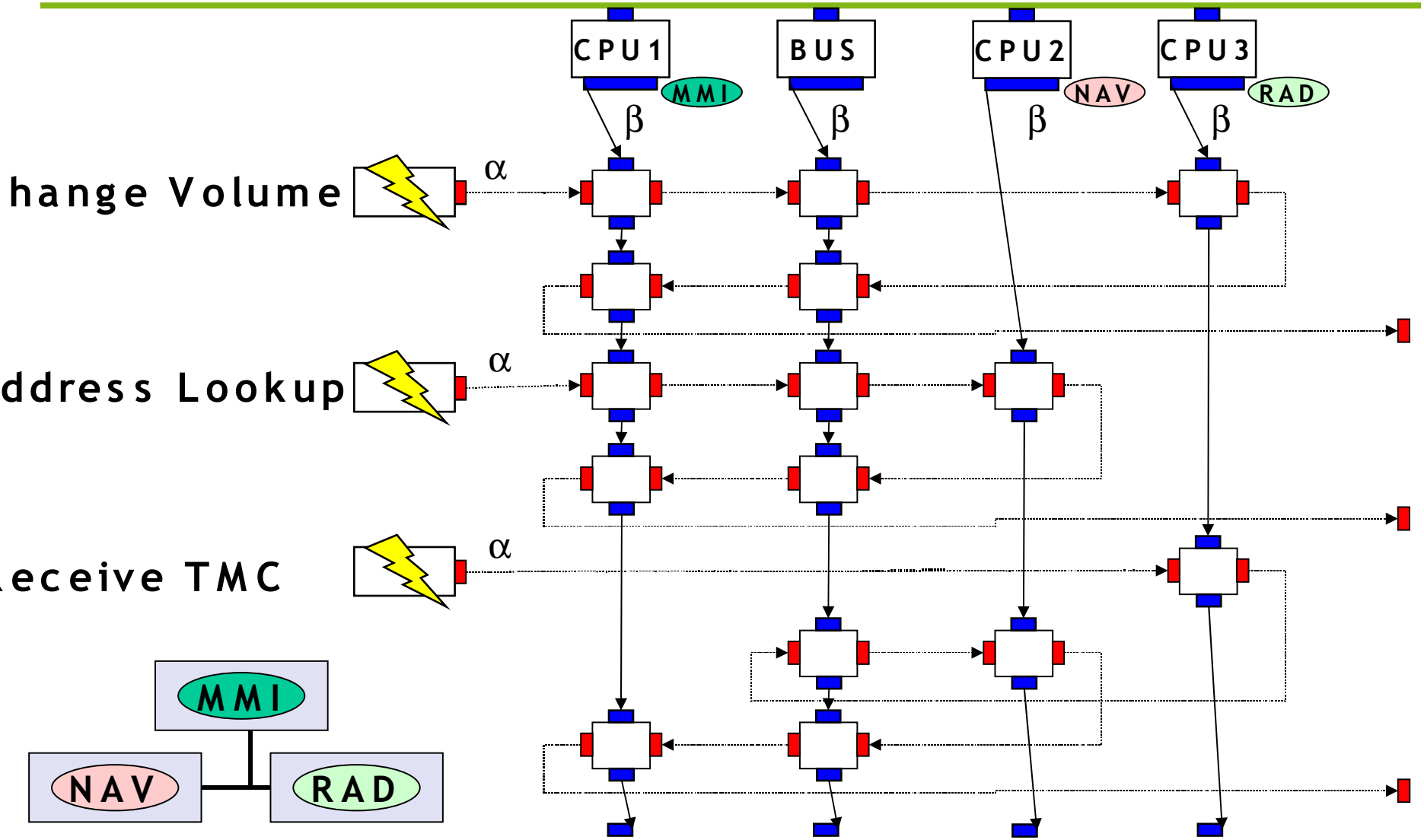


Step 3: Mapping / Scheduling

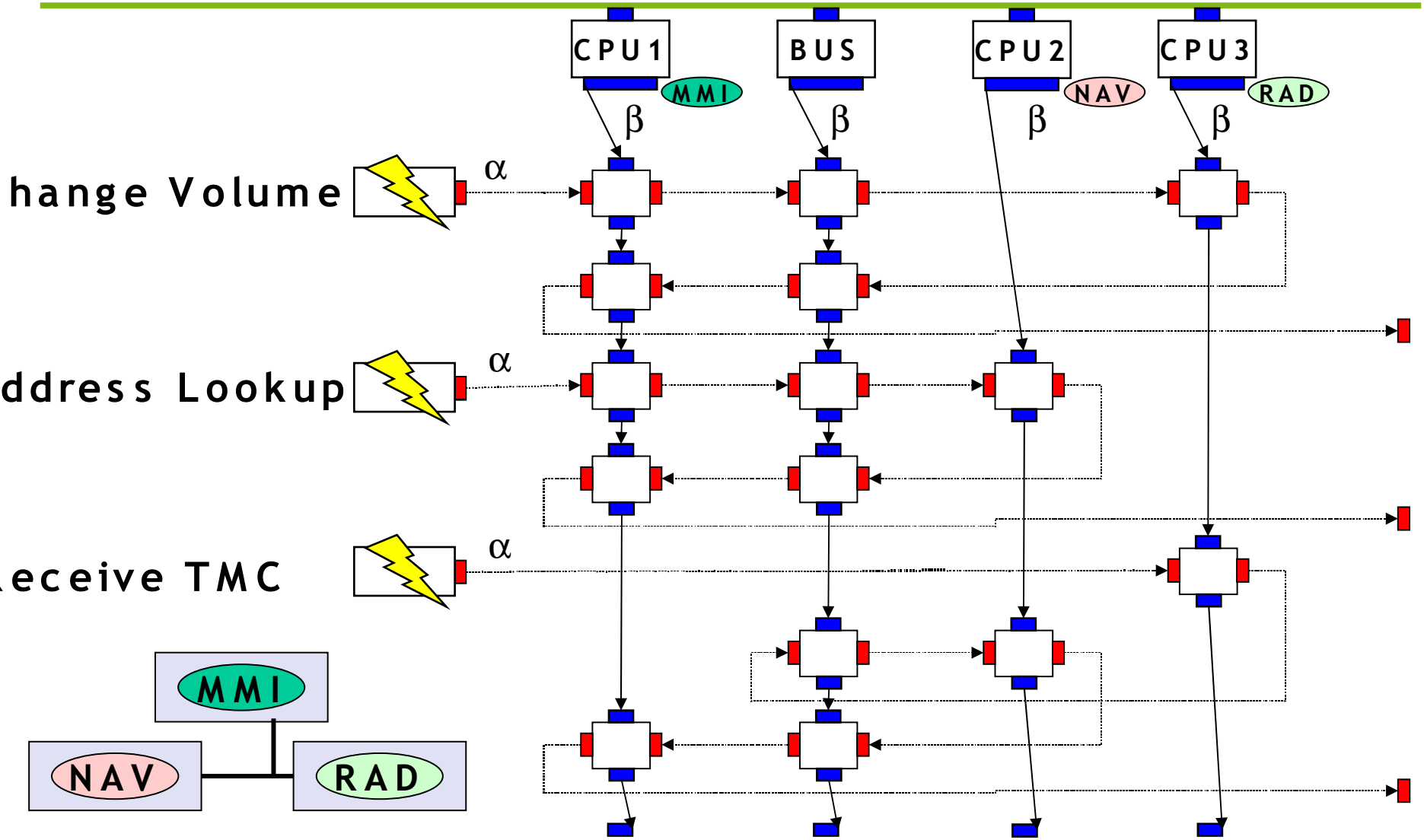
Rate Monotonic Scheduling
(Pre-emptive fixed priority scheduling):

- 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

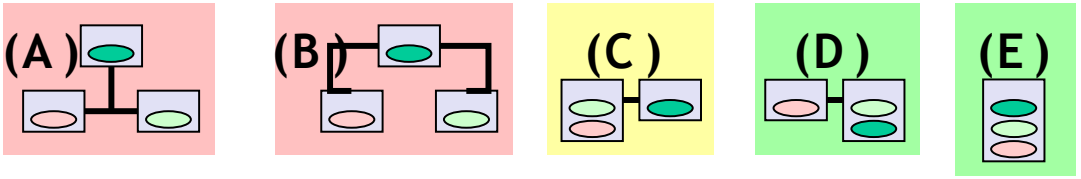


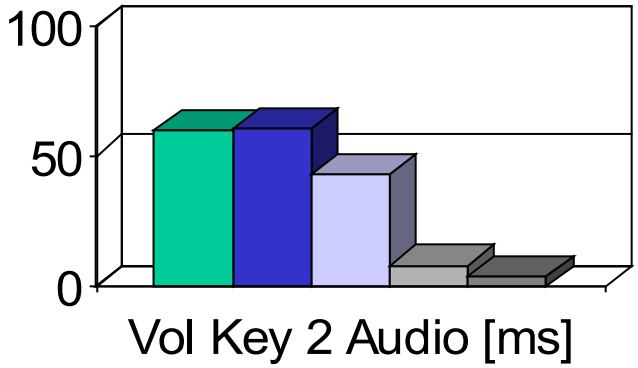
Analysis – Design Question 1

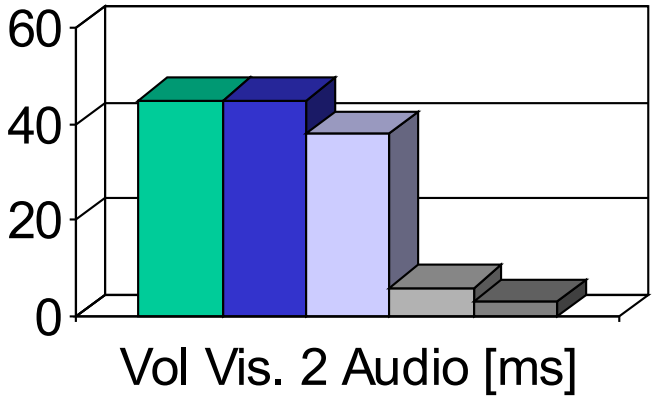
How do the proposed system architectures compare in respect to end-to-end delays?

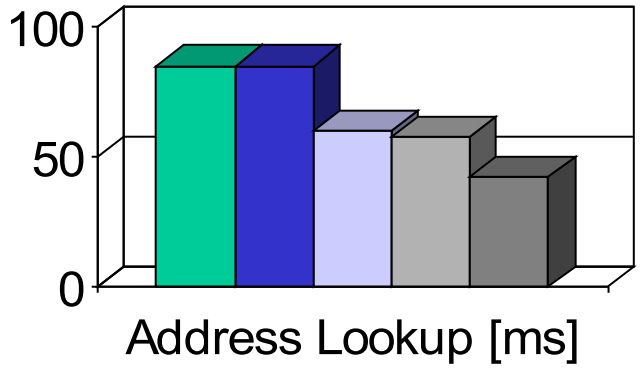
Analysis – Design Question 1

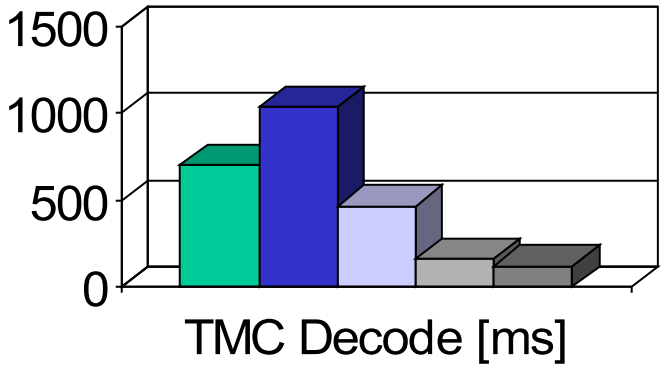
End-to-end delays:







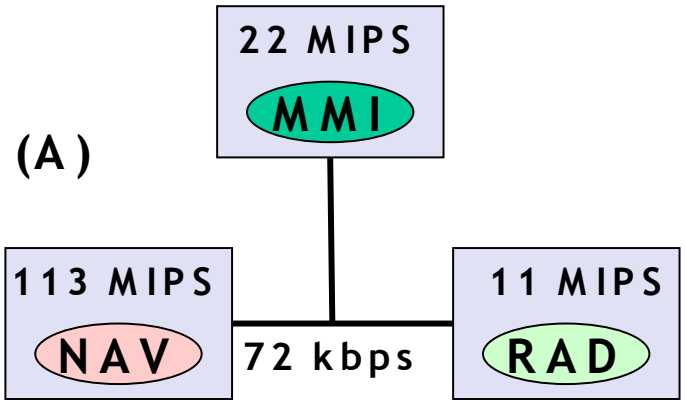




Analysis – Design Question 2

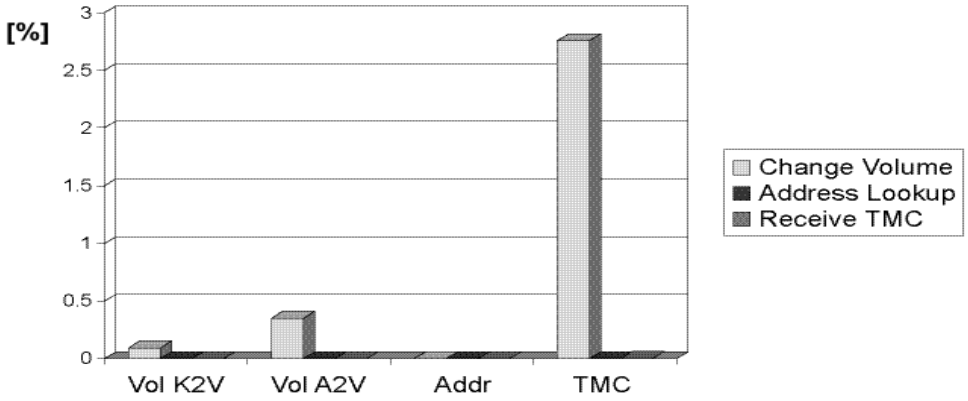
How robust is architecture A?

Where is the bottleneck of this architecture?

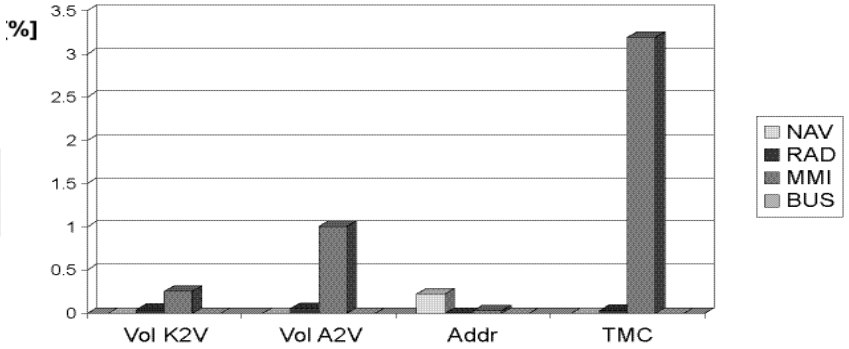


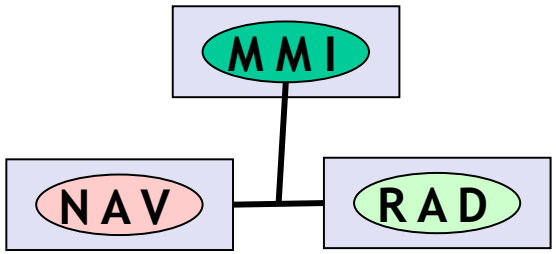
Analysis – Design Question 2

Sensitivity to input rate:



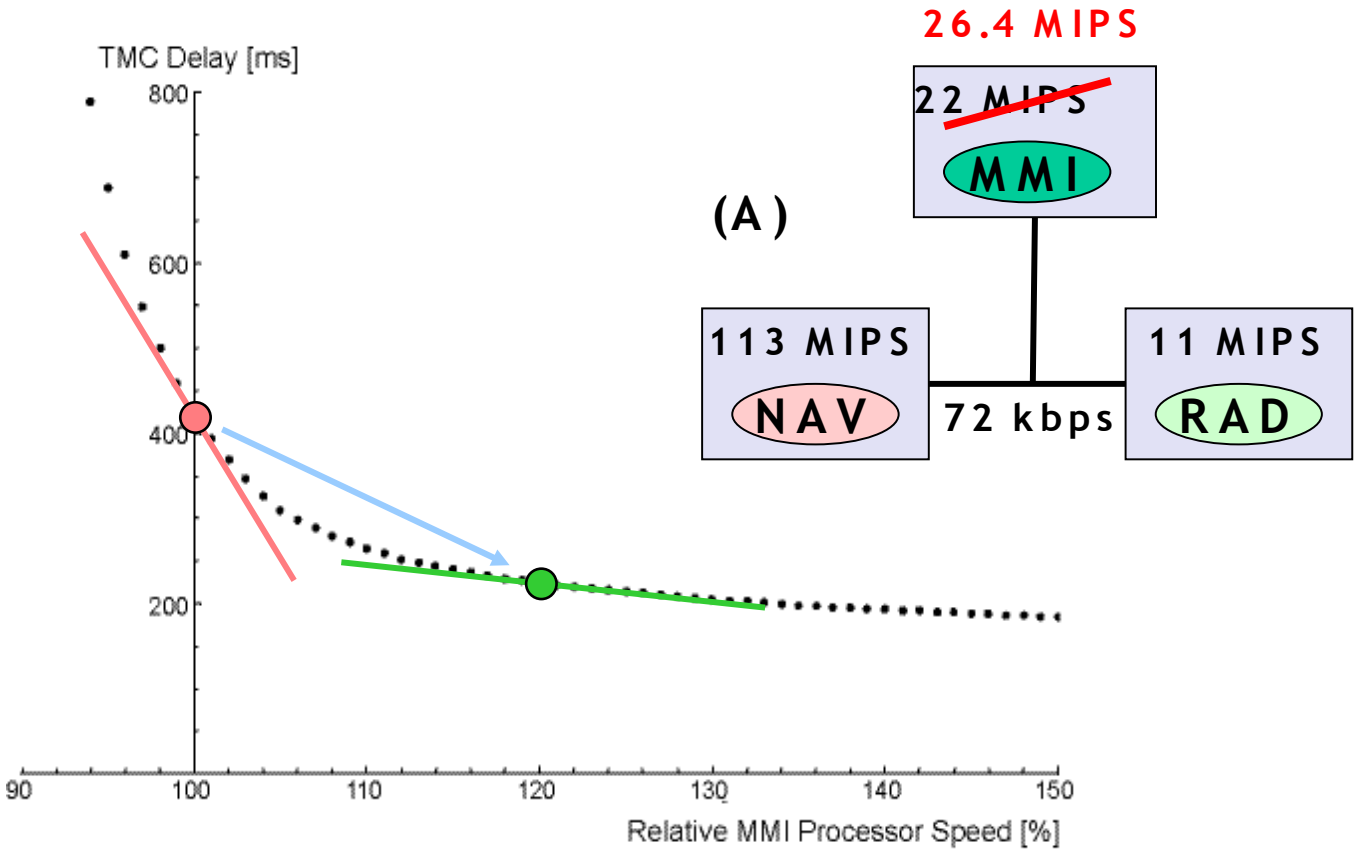
Sensitivity to resource capacity:





Analysis – Design Question 2

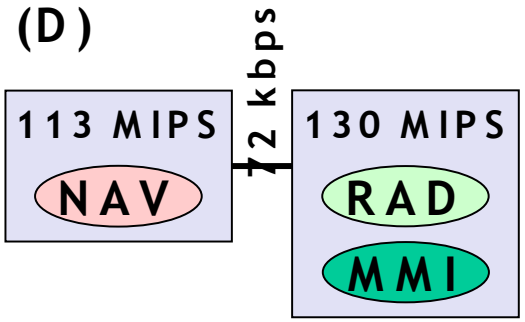
TMC delay vs. MMI processor speed:



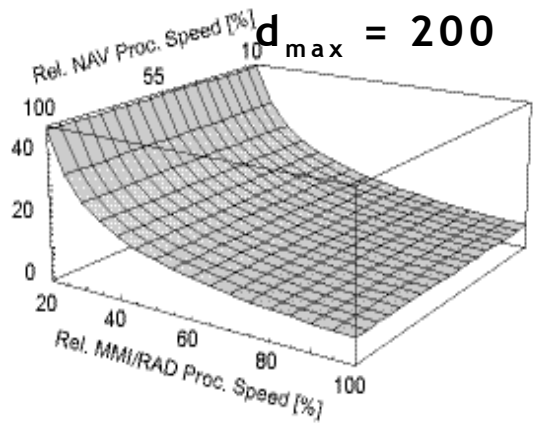
Analysis – Design Question 3

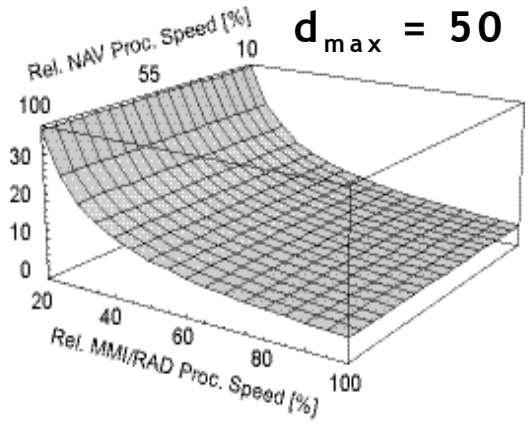
Architecture D is chosen for further investigation.

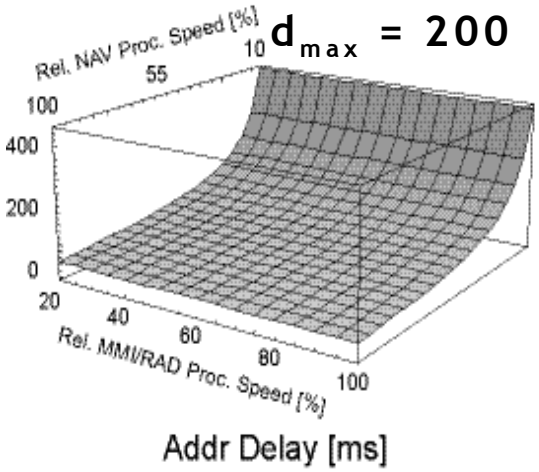
How should the processors be dimensioned?

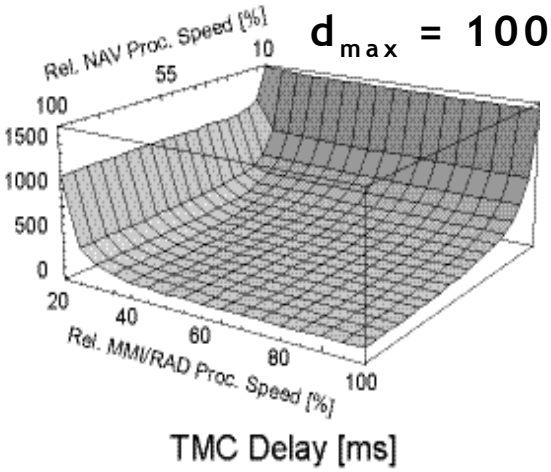


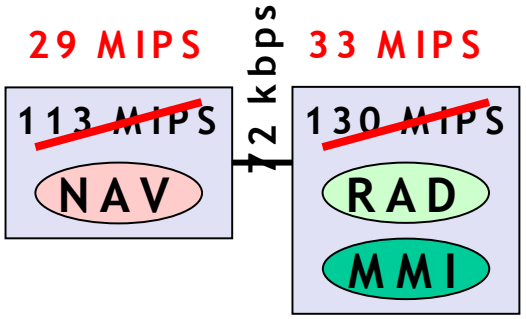
Analysis – Design Question 3











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

Validation and evaluation of designs

- Simulation-based techniques
  - Trade-off between speed and accuracy (☞ end of chapter 2)
  - Problematic for large input sets, dangerous environments, …
- Formal performance analysis
  - Thiele's real-time calculus (RTC)
    - 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