System Architecture Evaluation Using Modular Performance Analysis - A Case Study

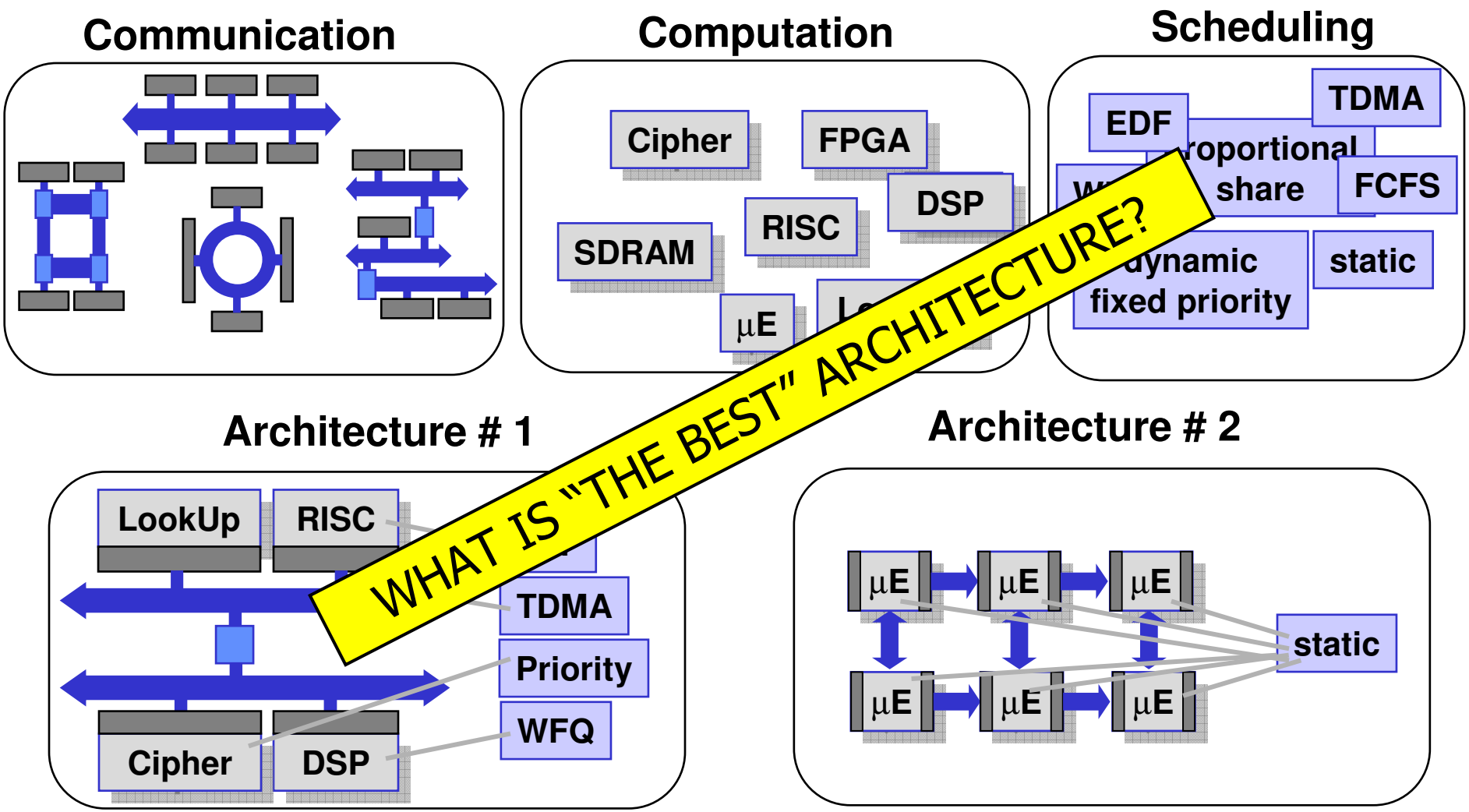
Marcel Verhoef (Chess Information Technology, NL)
Ernesto Wandeler, Lothar Thiele (ETH Zürich, CH)
Paul Lieverse (Siemens VDO Automotive, NL)

LaQuSo (Laboratory for Quality Software) symposium,
24 november 2004, Eindhoven, The Netherlands

Contents of this talk

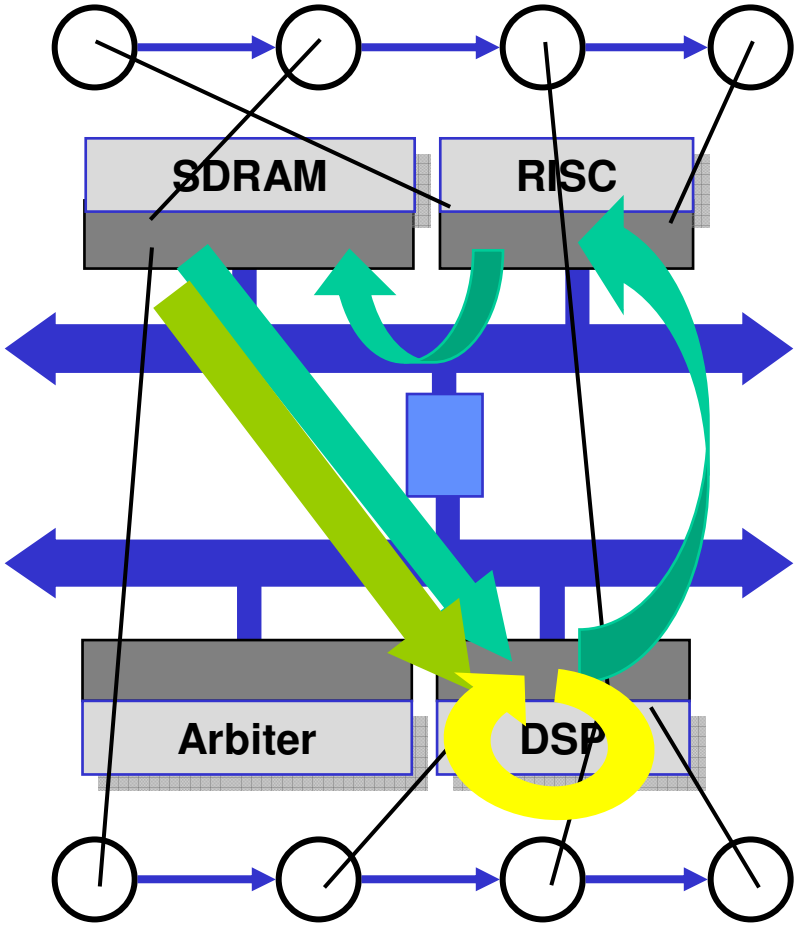
- System Architecture Evaluation
- Modular Performance Analysis
- Case Study – In Car Radio Navigation System
- Conclusions & Future Work

System Architecture Evaluation (1)



System Architecture Evaluation (2)

- Distributed processing on different resources
- Interactions



System Architecture Evaluation (3)

- Simulation Based Methods
  - Detailed, explicit, models are required to generate sufficiently useful results
  - Building models is in general costly, as is their evaluation and interpretation
  - No hard guarantee to find the best- or worst case
- Queuing Networks
  - Probabilistic approach causes state space explosion if level of model detail is increased
  - Exponential distributions used in Markov chain analysis does not accurately reflect real-life systems properties

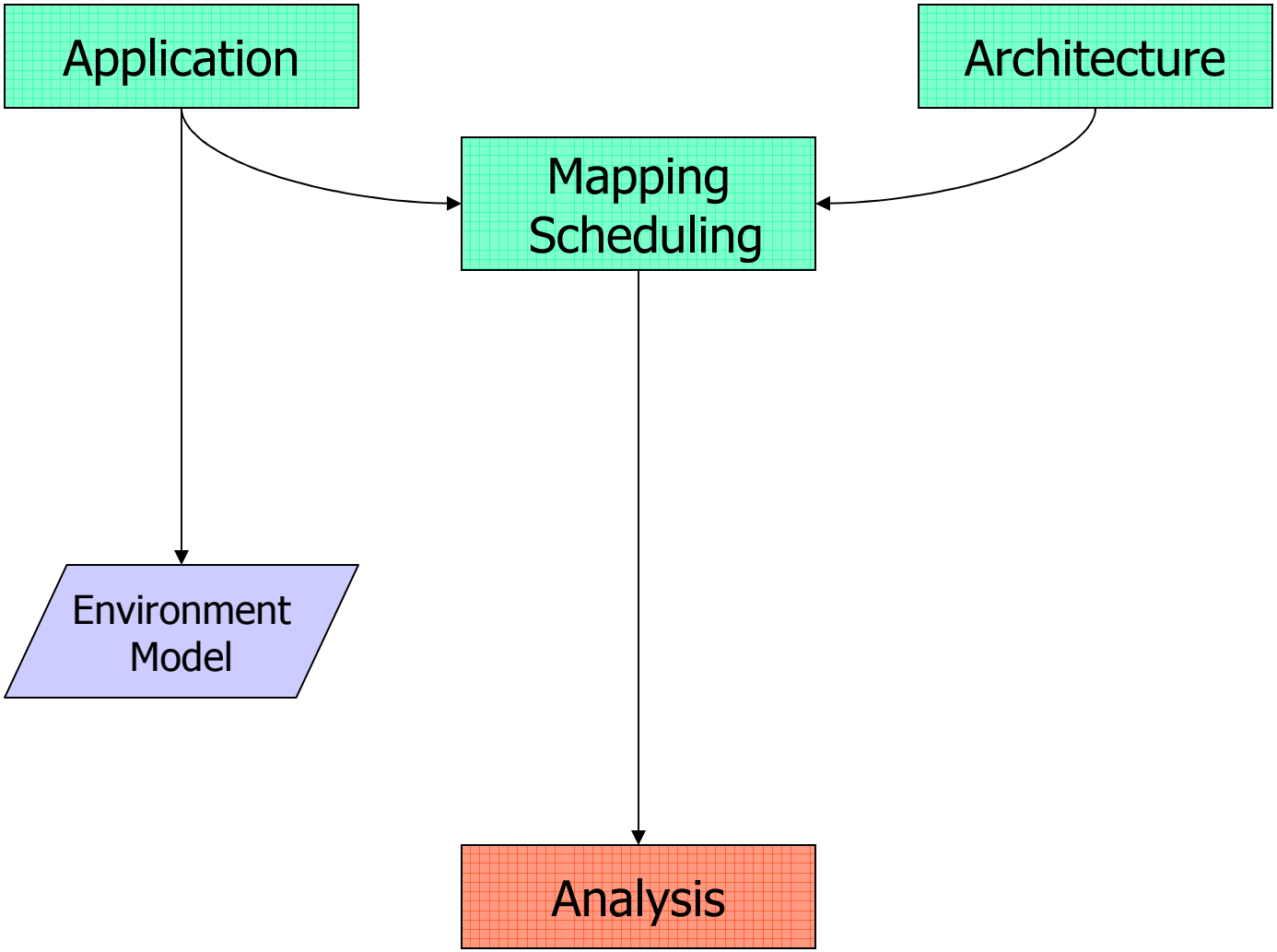
Modular Performance Analysis (1)

- Based on Real-Time Calculus [Thiele-2000/2004]
- Real-Time Calculus extends the well-known Network Calculus [LNCS2050-2001] and early work by [Cruz-1991]
  - RTC deals with both *computation* and *communication* in a single mathematical framework
  - Implements standard event models: *periodic, periodic with jitter, periodic with bursts and sporadic*
- Network Calculus uses max-plus algebra to compute the results [Bacelli-1992]

Modular Performance Analysis (2)

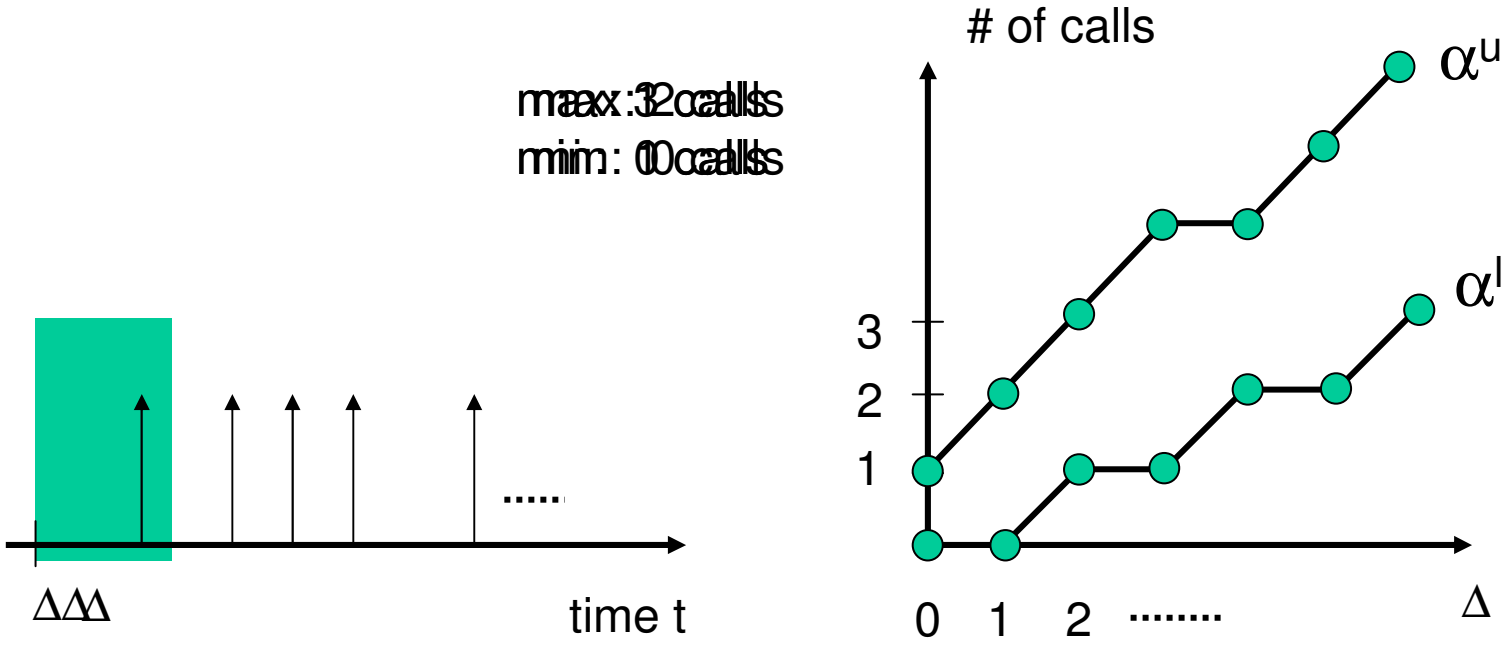
- Characterizes a system by describing
  - How many resources are needed to fulfill a function?
  - How often are functions needed?
  - When are resources available?
- Compositional, heterogeneous
  - Compose networks of MPA elements
  - Decompose MPA elements into MPA element networks
- Deterministic, analytic, algorithm (no simulation)
- Hard upper- and lower bounds are always found

Modular Performance Analysis (3)



MPA – Environment model (3.1)

We use so-called **arrival curves** to describe abstract event streams

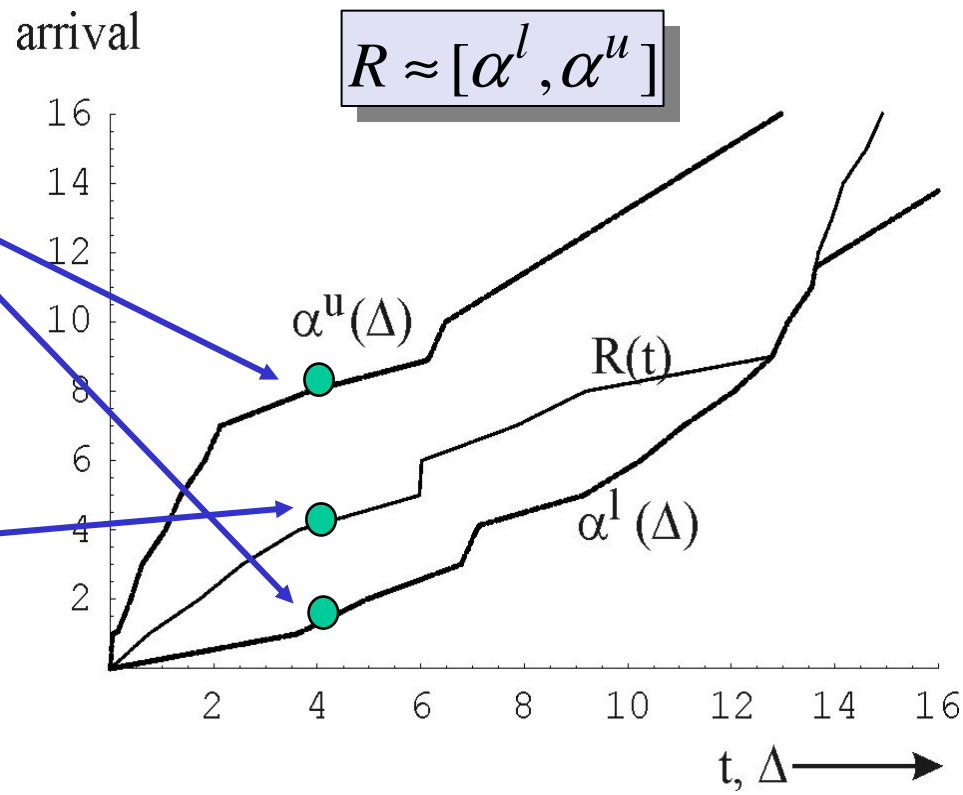


MPA – Environment model (3.2)

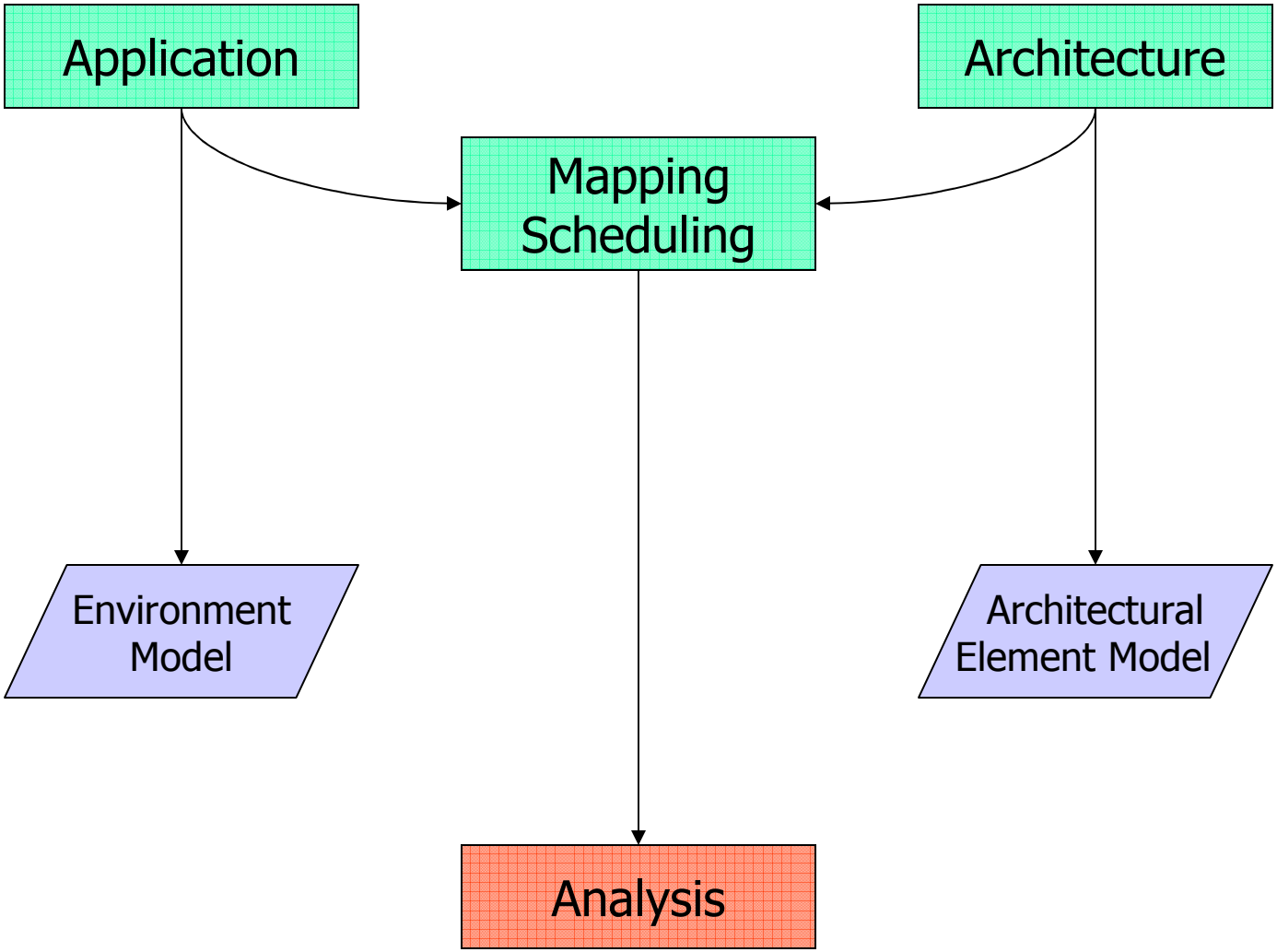
Upper and lower arrival curves

maximum/minimum number of calls in *any interval* of length 4

number of calls in time interval [0,4]



Modular Performance Analysis (4)

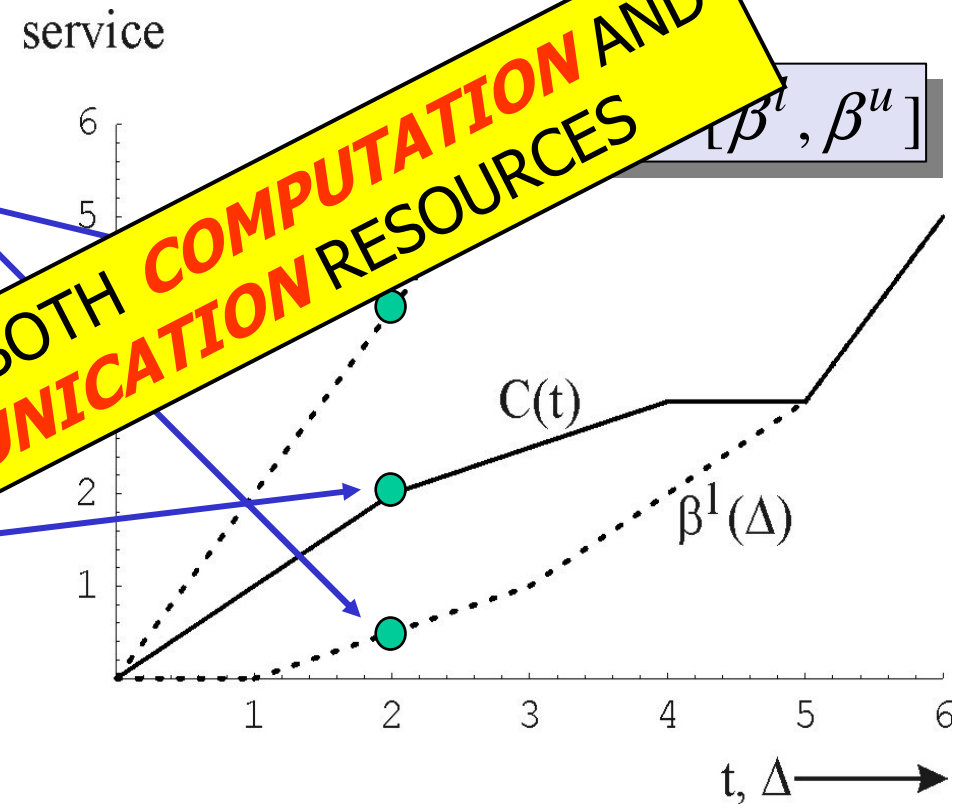


MPA – Environment model (4.1)

- Upper and lower service curves

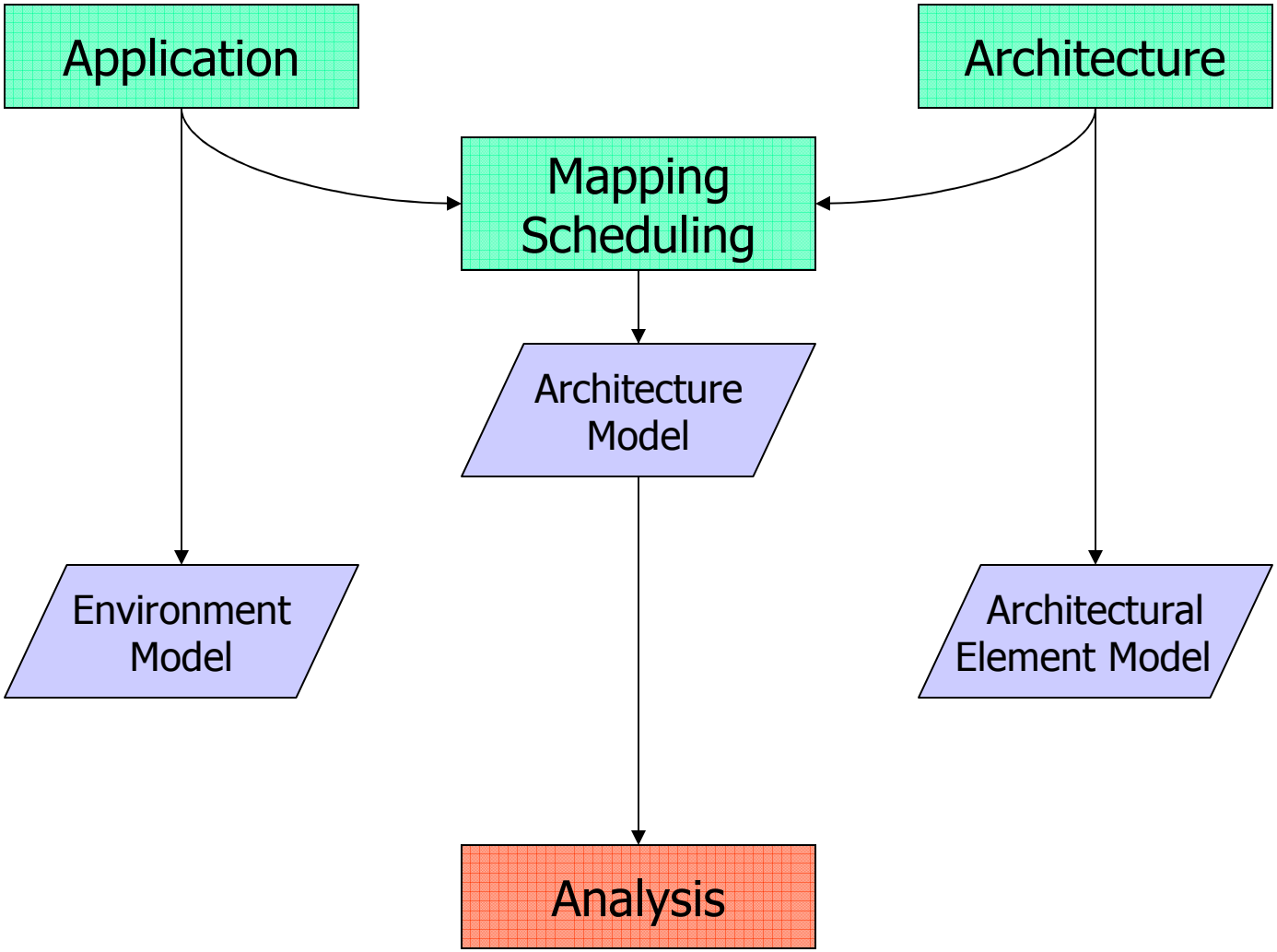
maximum/minimum computing power or communication capacity in *any interval* of length 2

in time [0,2]



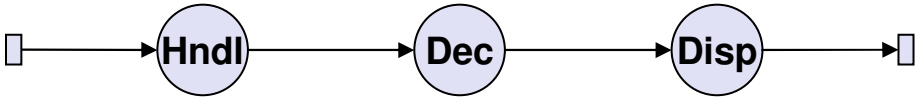
WORKS FOR BOTH **COMPUTATION** AND **COMMUNICATION** RESOURCES

Modular Performance Analysis (5)



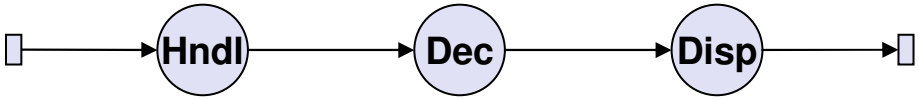
MPA – System Architecture Model (5.1)

Application



MPA – System Architecture Model (5.2)

Application



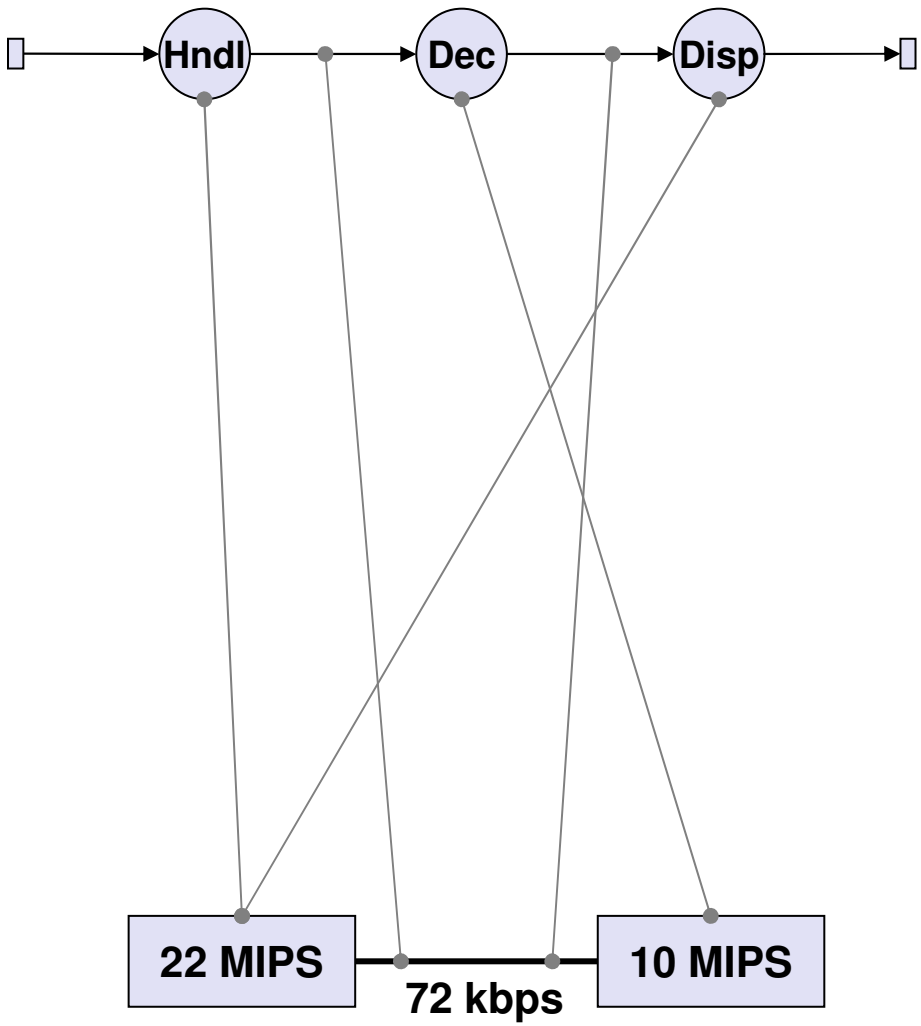
HW Architecture

MPA – System Architecture Model (5.3)

Application

Mapping

HW Architecture

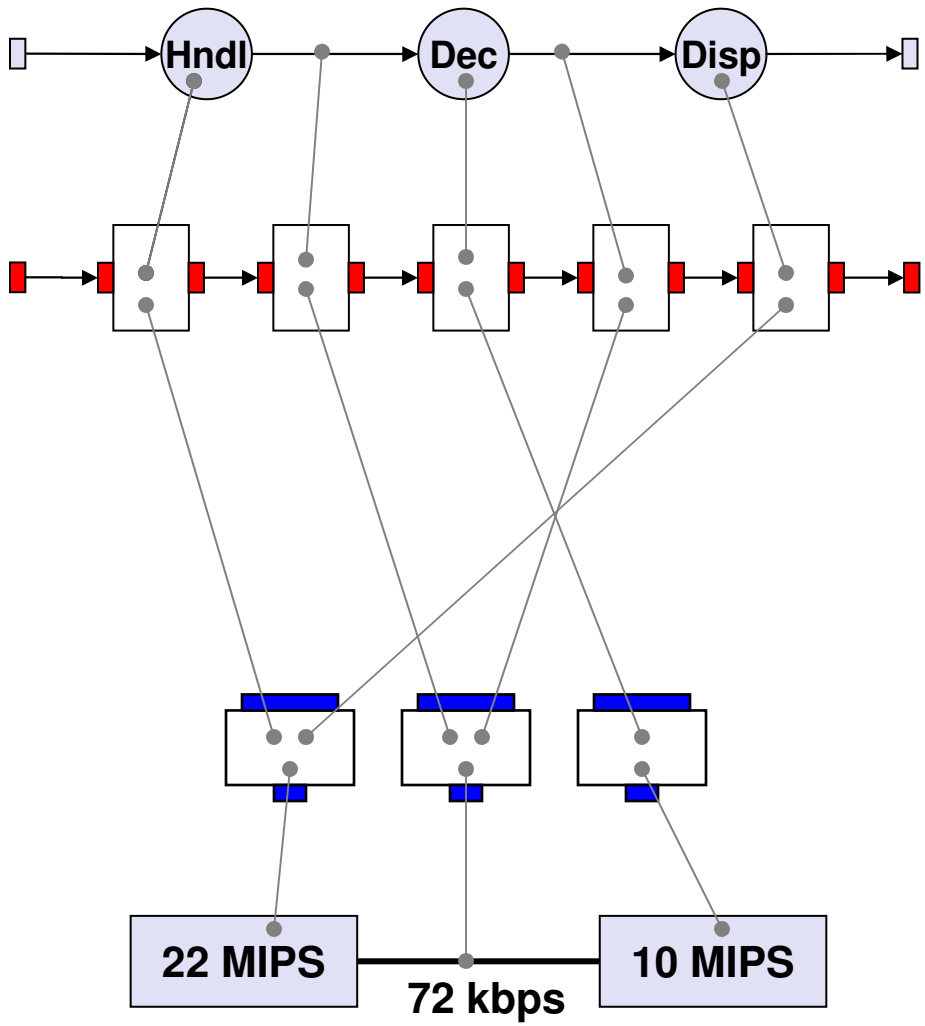


MPA – System Architecture Model (5.4)

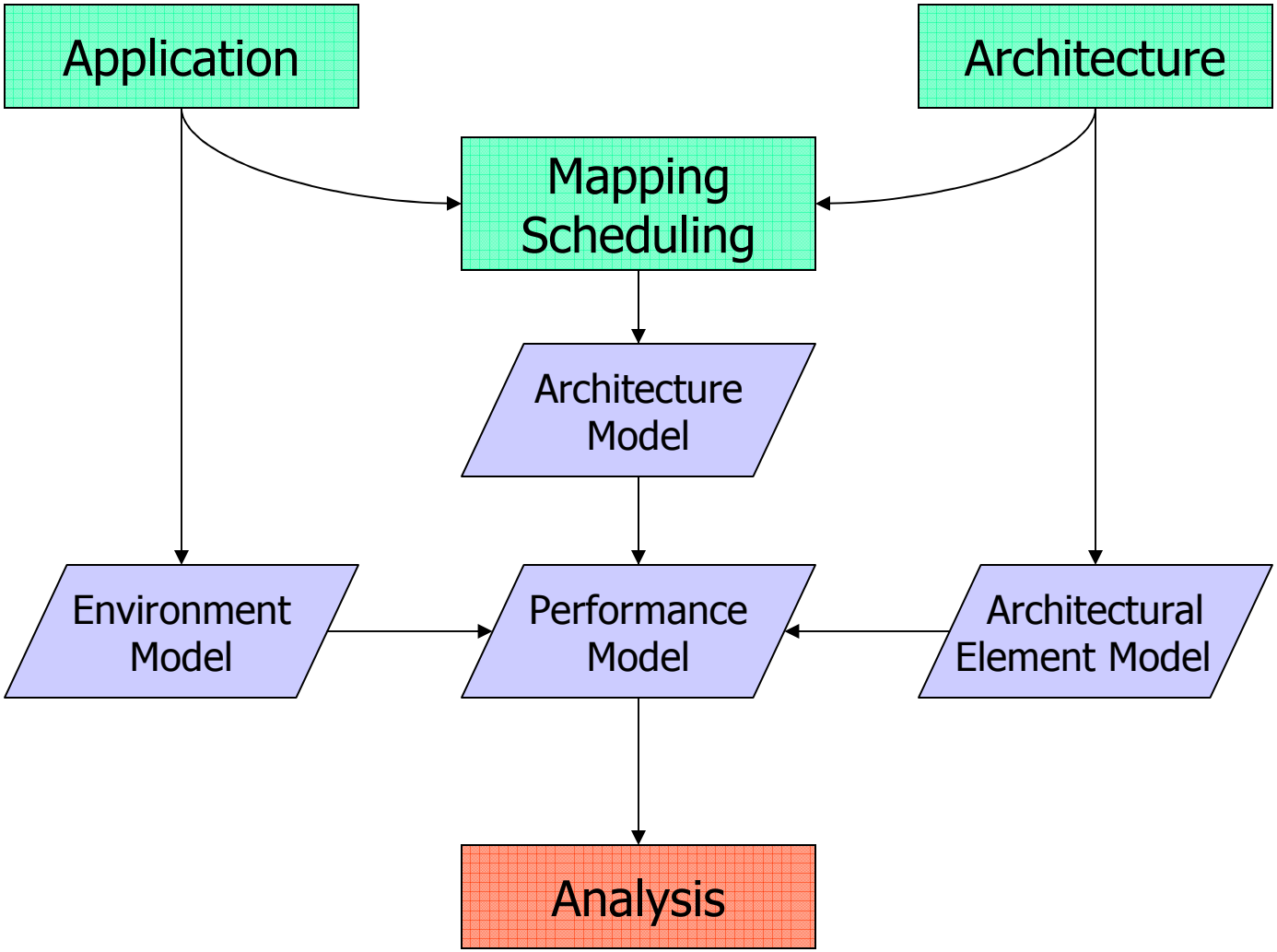
Application

Mapping

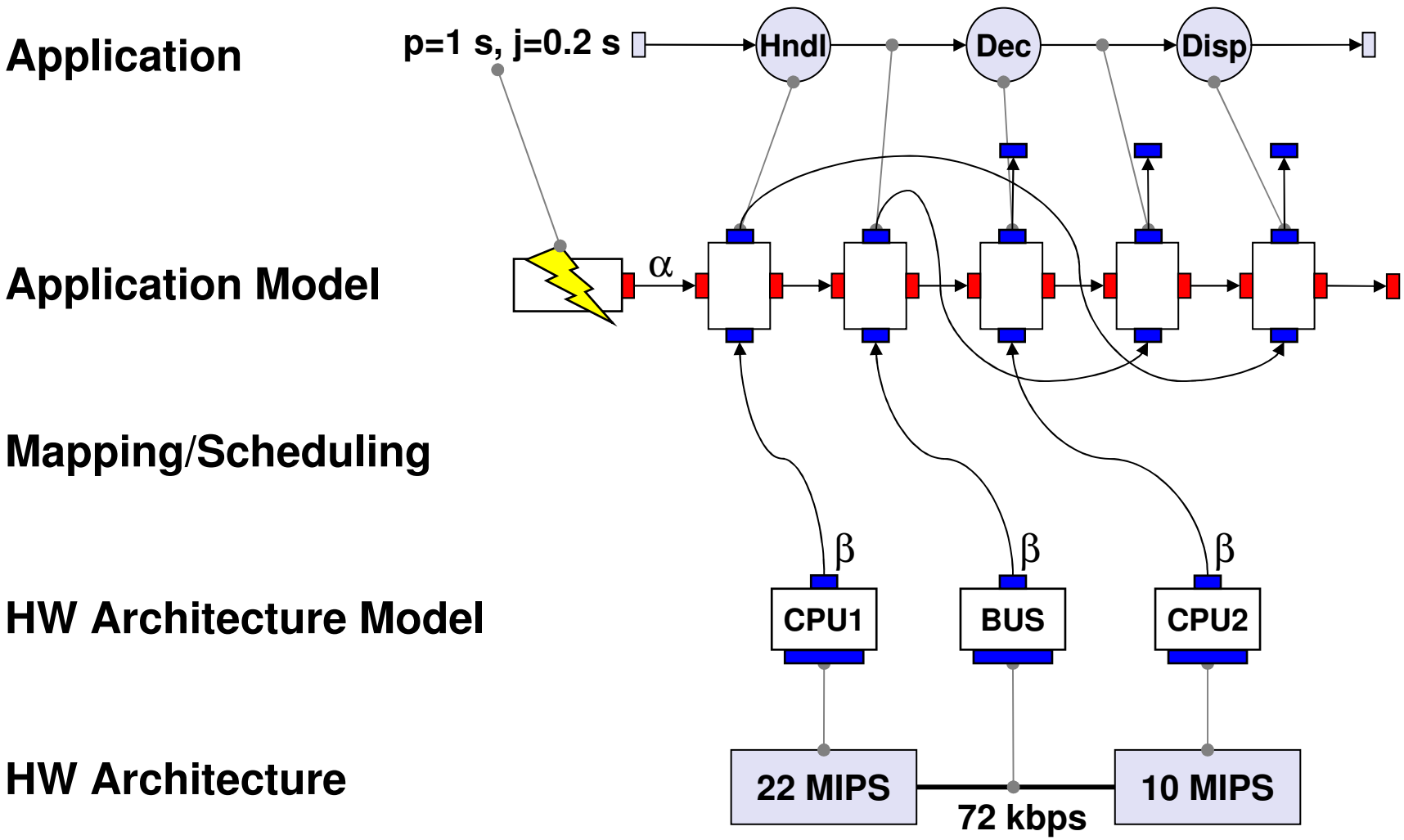
HW Architecture



Modular Performance Analysis (6)



MPA – Performance Model (6.1)

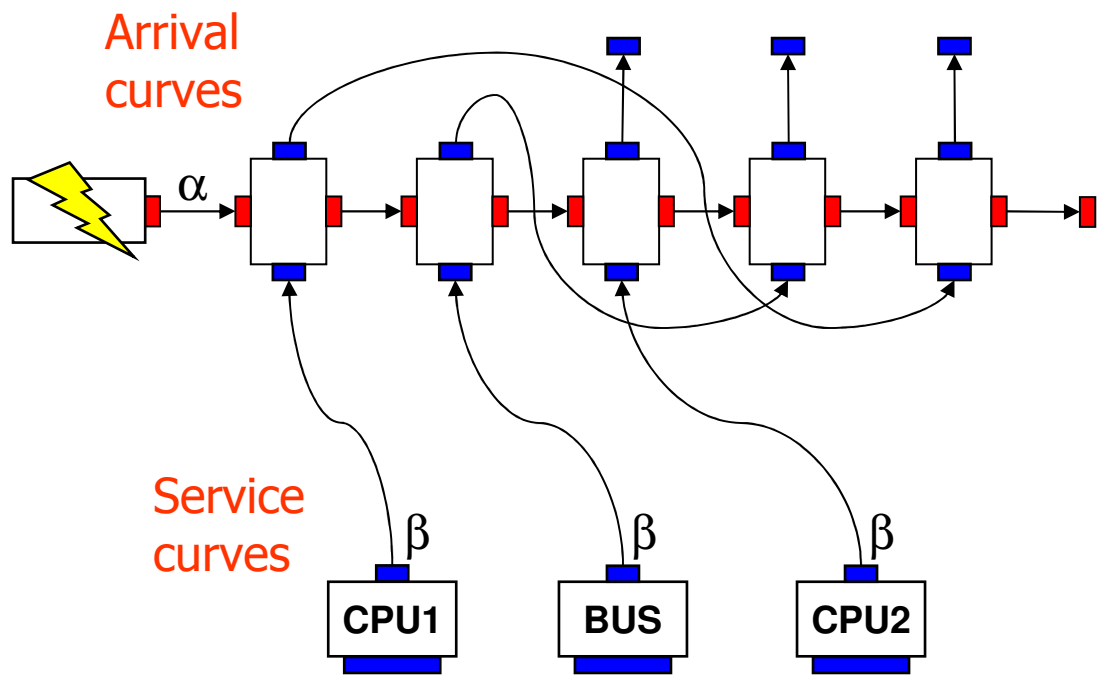


MPA – Performance Model (6.2)

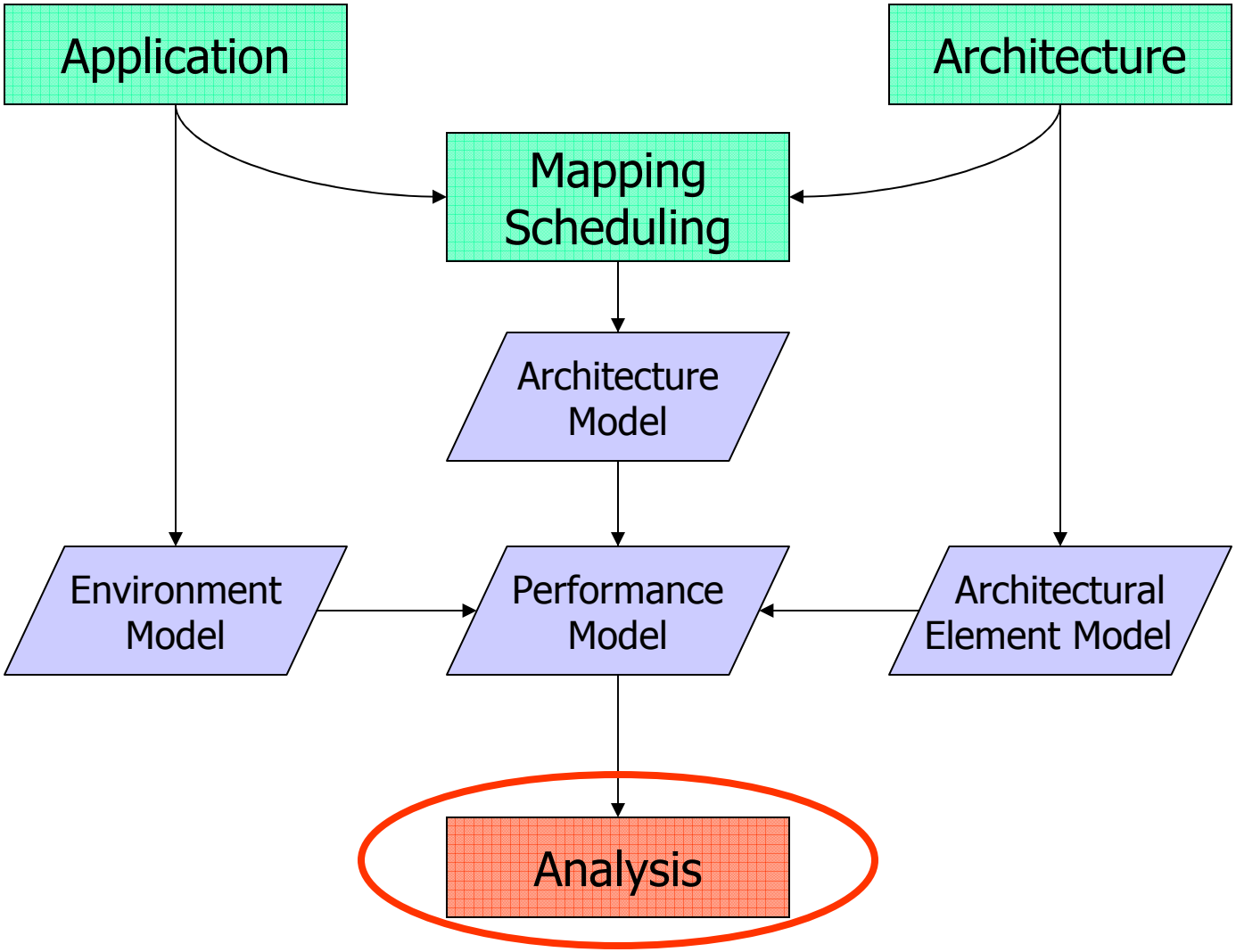
Application Model

Mapping/Scheduling

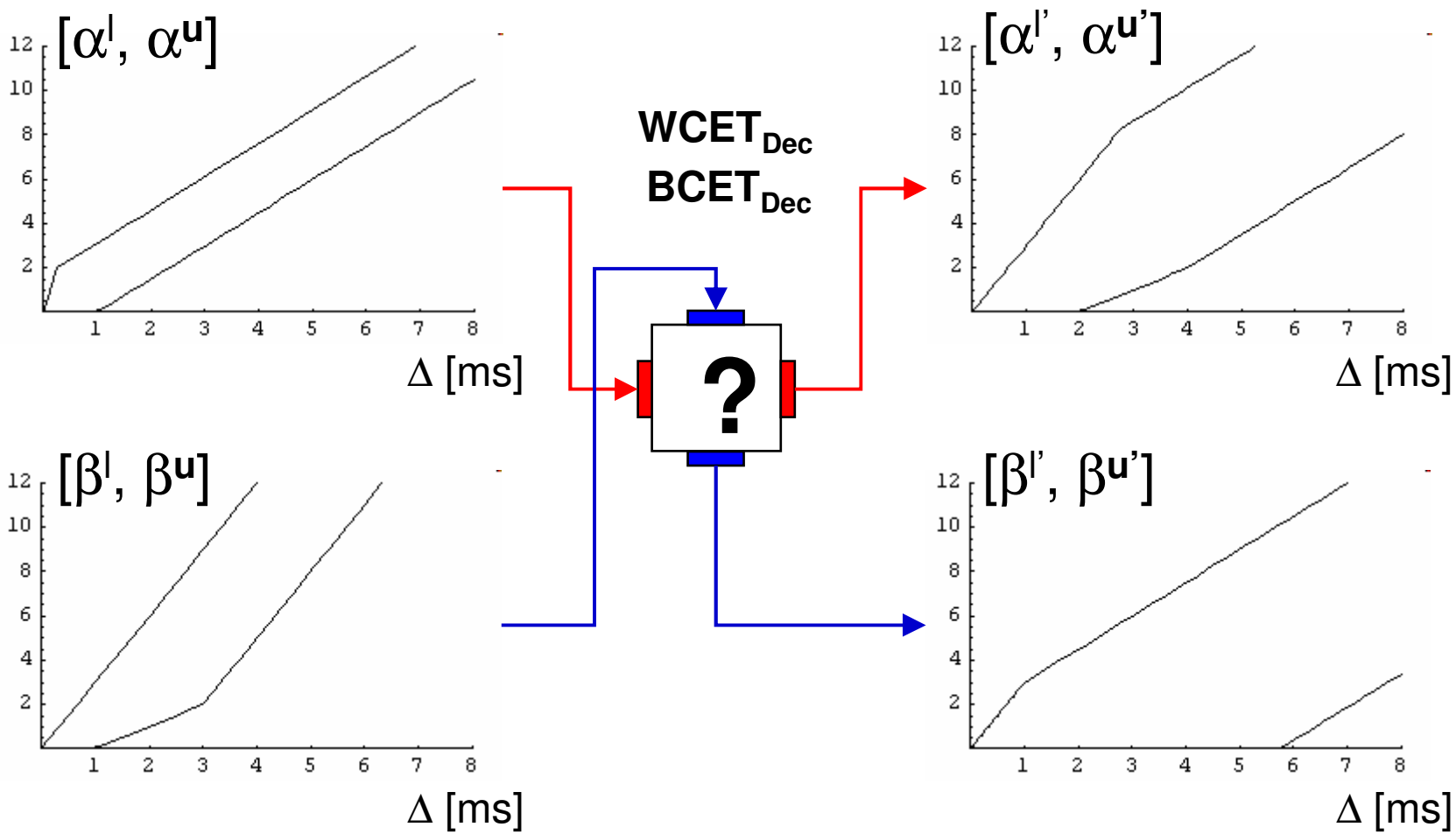
HW Architecture Model



Modular Performance Analysis (7)



MPA – Analysis (7.1)

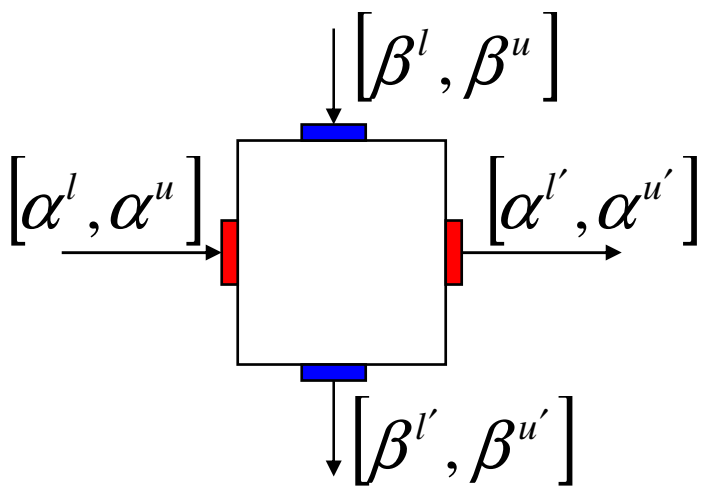


MPA – Analysis (7.2)

$$v(\Delta) \wedge w(\Delta) = \min\{v(\Delta), w(\Delta)\}$$

$$v(\Delta) \underline{\oplus} w(\Delta) = \inf_{0 \le \lambda \le \Delta} \{v(\lambda) + w(\Delta - \lambda)\} \qquad v(\Delta) \underline{\otimes} w(\Delta) = \inf_{0 \le \lambda} \{v(\Delta + \lambda) - w(\lambda)\}$$

$$v(\Delta) \overline{\oplus} w(\Delta) = \sup_{0 \le \lambda \le \Delta} \{v(\lambda) + w(\Delta - \lambda)\} \qquad v(\Delta) \overline{\otimes} w(\Delta) = \sup_{0 \le \lambda} \{v(\Delta + \lambda) - w(\lambda)\}$$



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

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

$$\beta^{u'} = \left(\beta^u - \alpha^l\right) \underline{\otimes} 0$$

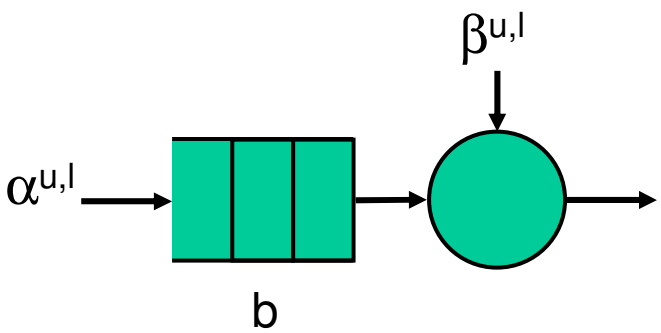
$$\beta^{l'} = \left(\beta^l - \alpha^u\right) \overline{\oplus} 0$$

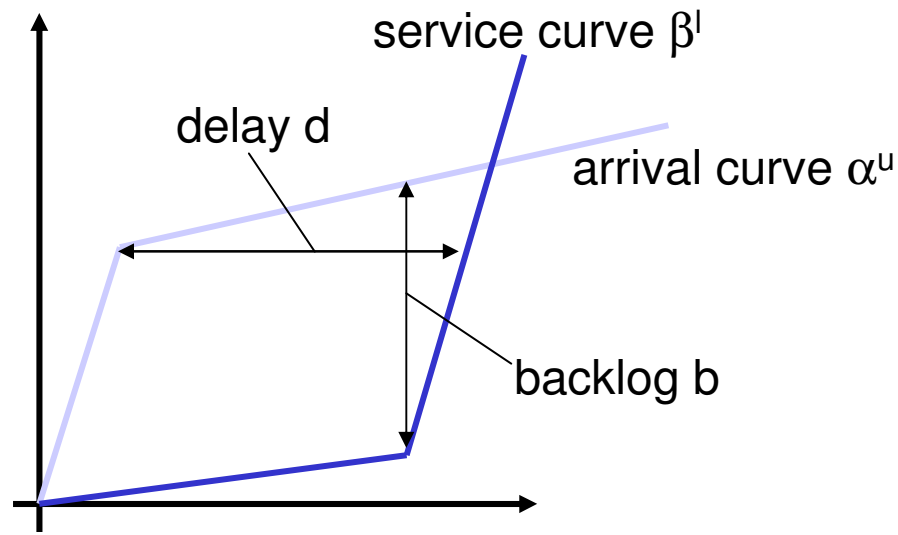
MPA – Analysis (7.3)

Delay and Memory

$$d(t) = \inf\{\tau \geq 0 : R(t) \leq R'(t+\tau)\} \leq \sup_{u \geq 0} \left\{ \inf\{\tau \geq 0 : \alpha^u(u) \leq \beta^l(u+\tau)\} \right\}$$

$$b(t) = R(t) - R'(t) \leq \sup_{u \geq 0} \{\alpha^u(u) - \beta^l(u)\}$$

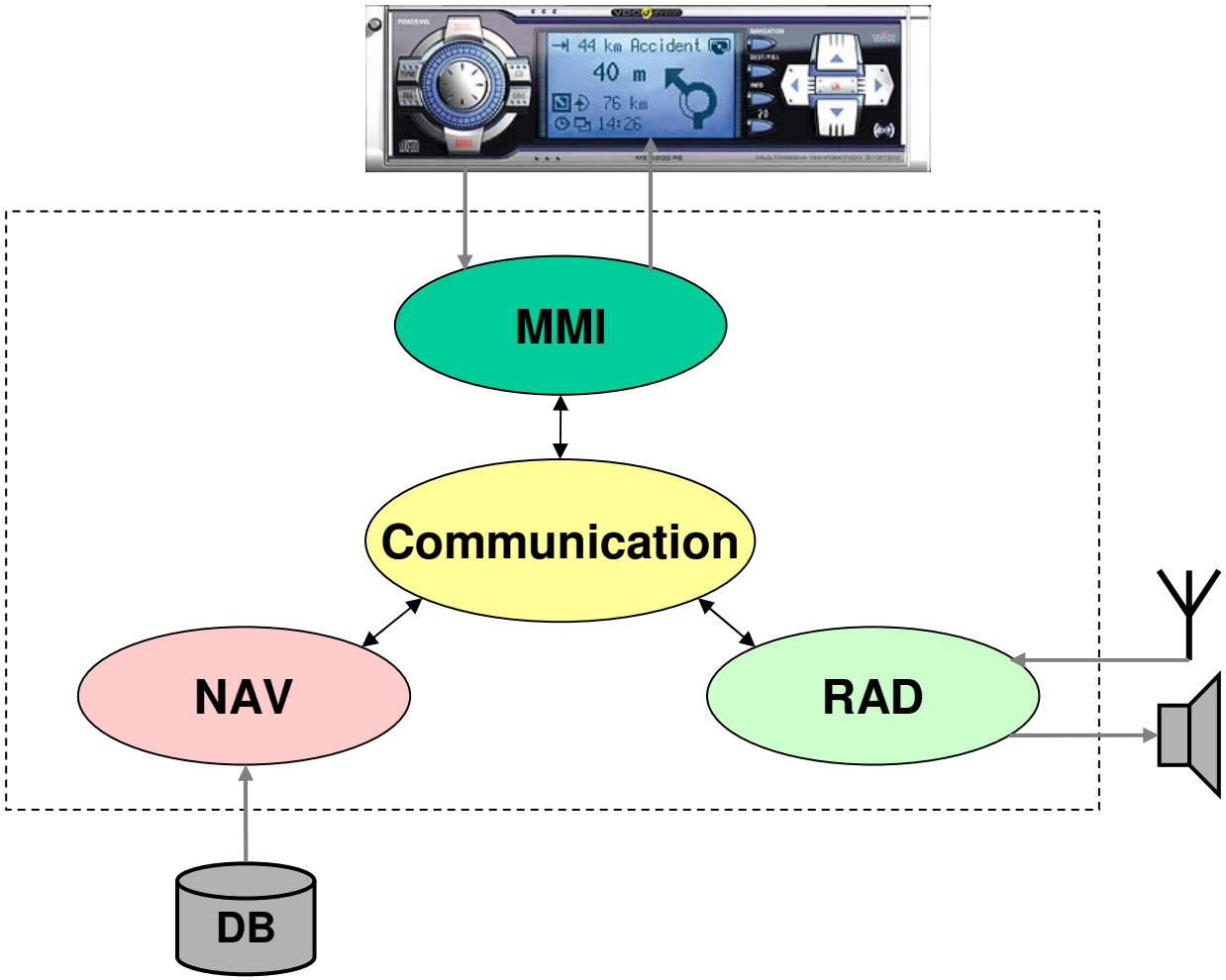




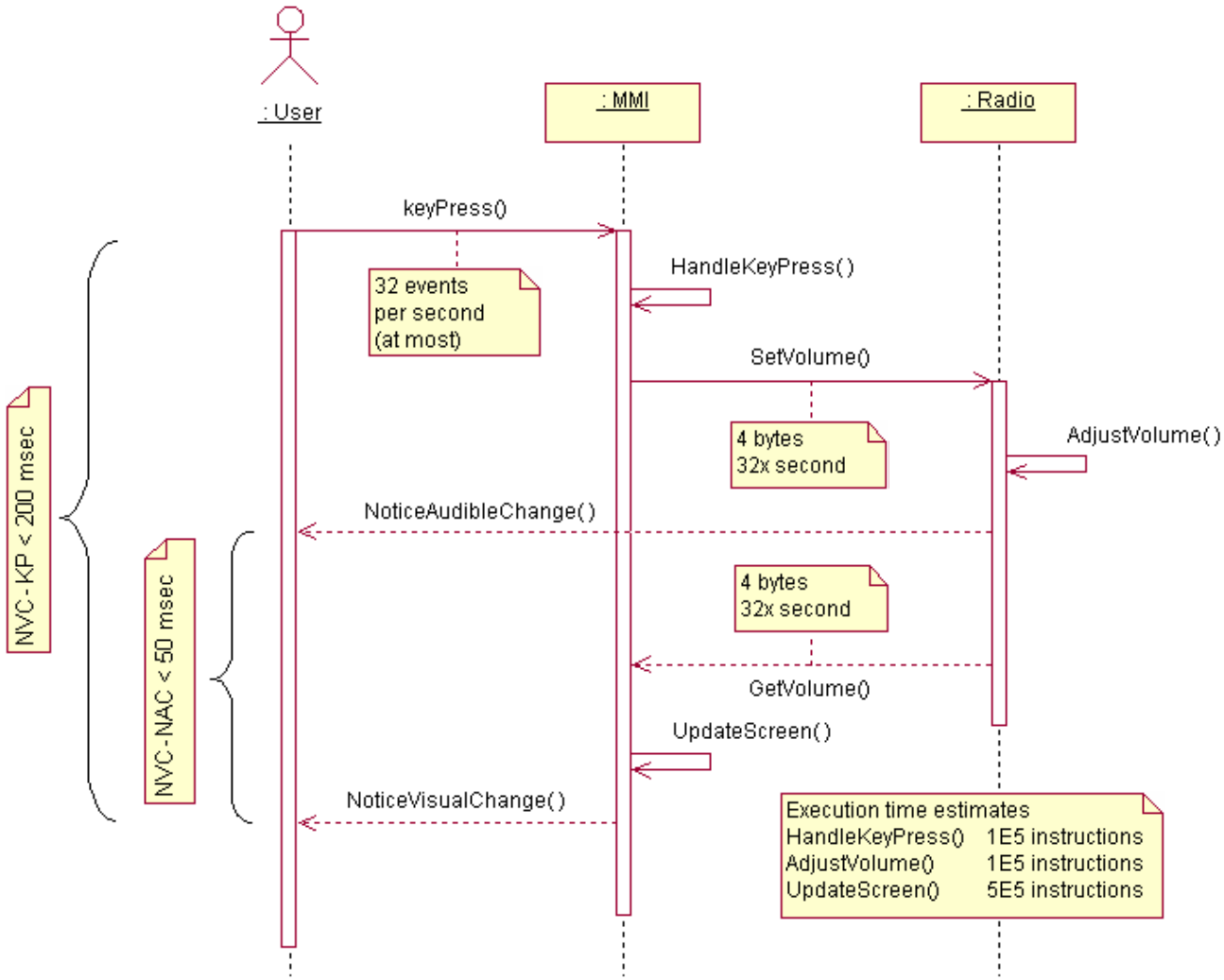
Case Study: In-Car Navigation System

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

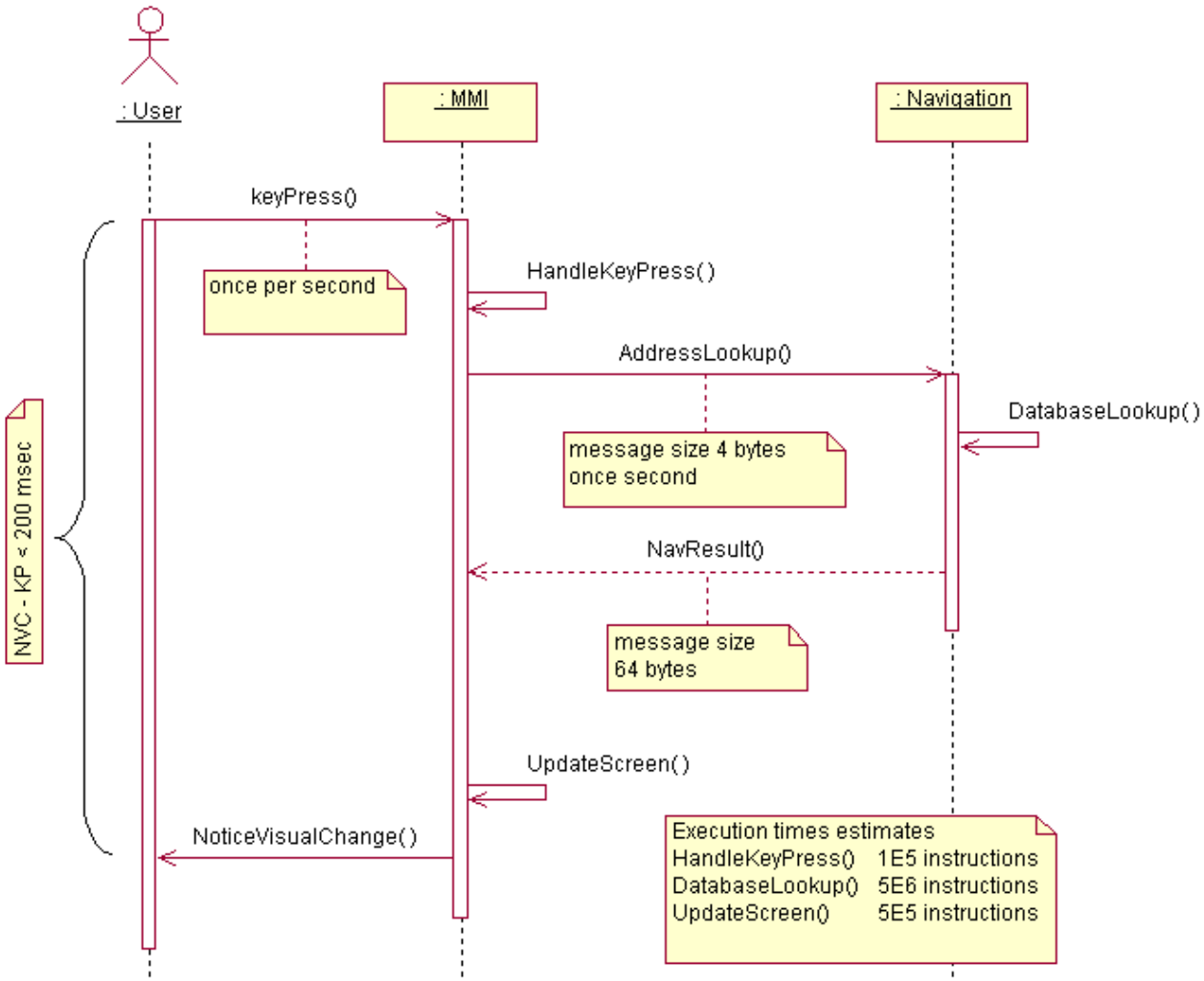
System Overview



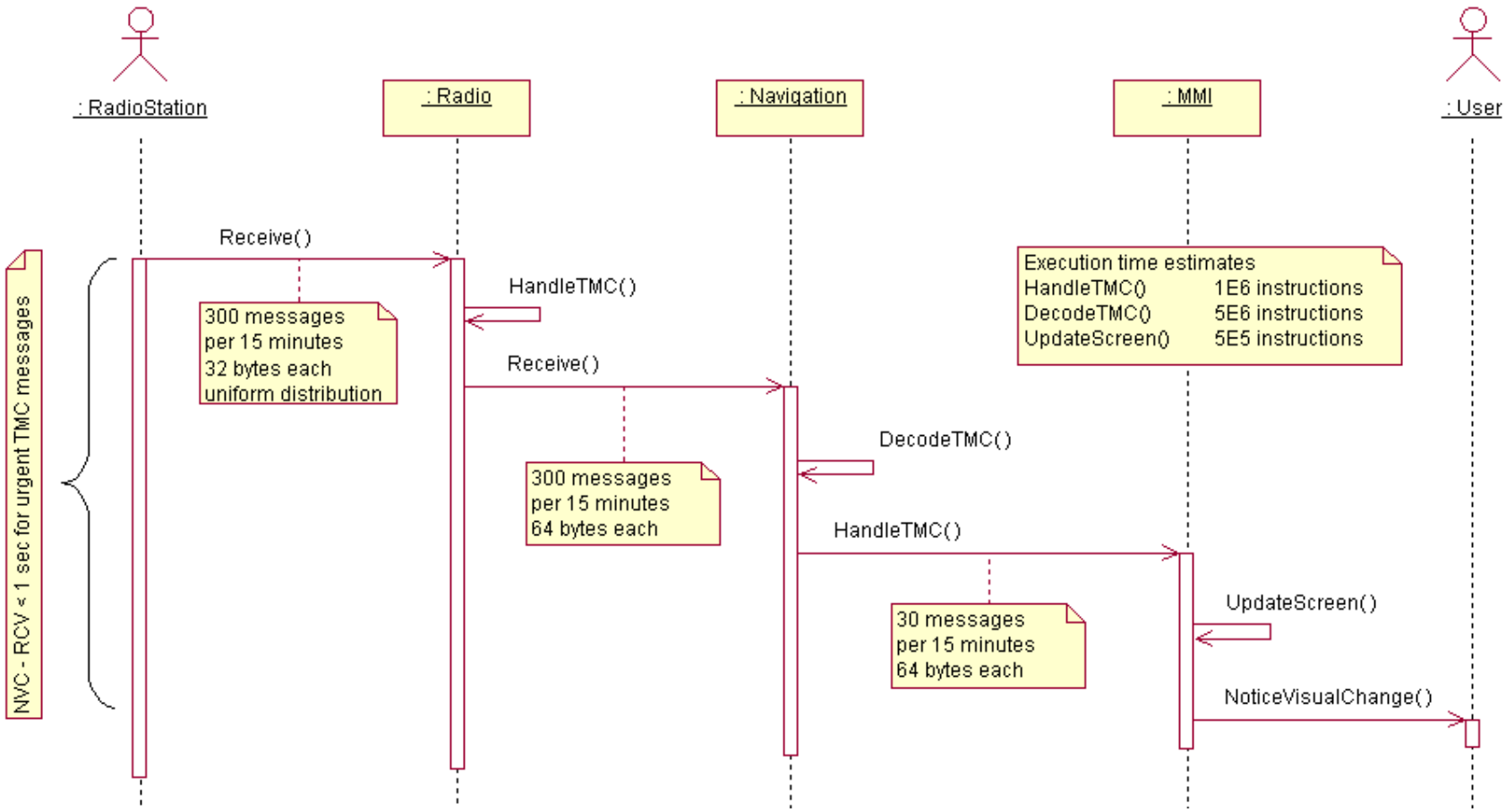
Application 1: Change Audio Volume



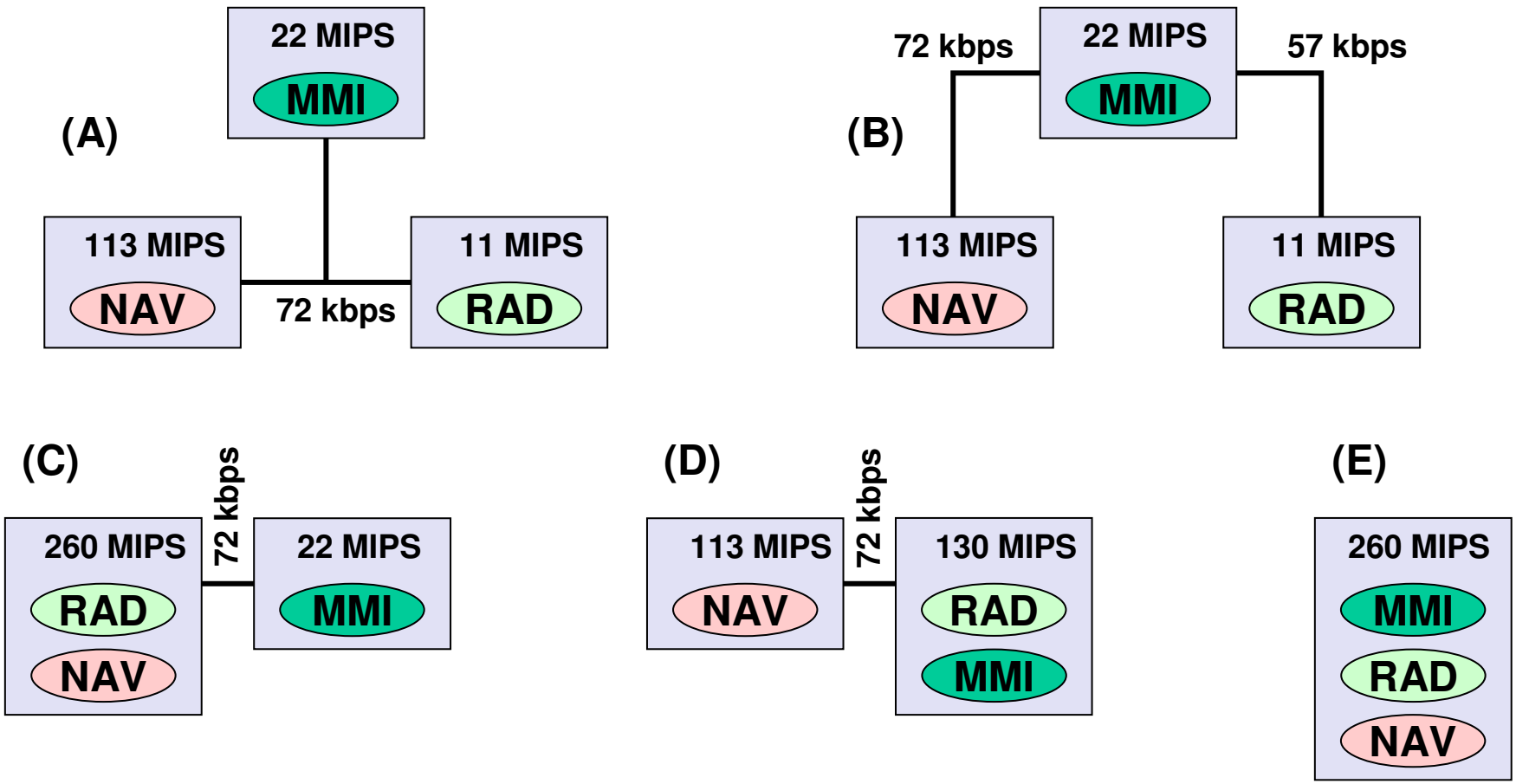
Application 2: Lookup Destination Address



Application 3: Receive TMC Messages



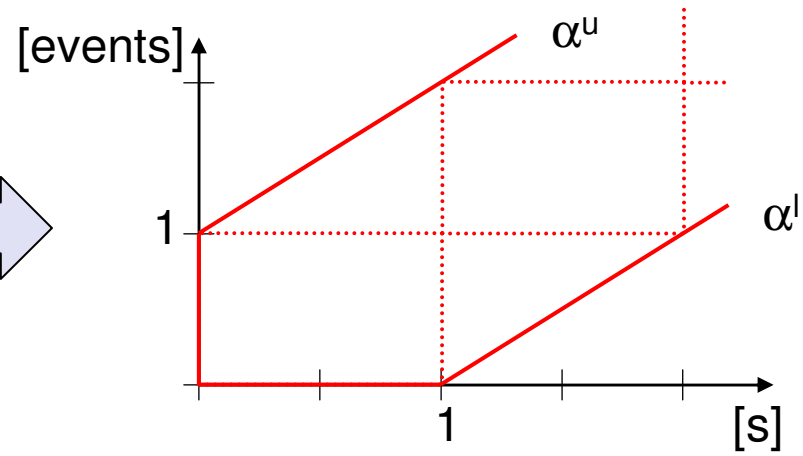
Proposed Architecture Alternatives



Step 1: Environment (Event Steams)

Event Stream Model

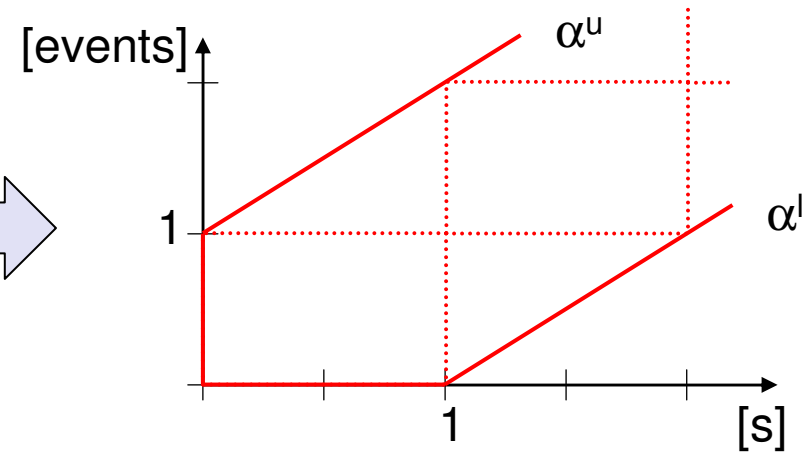
e.g. Address Lookup
(1 event / sec)



Step 2: Architectural Elements (Resources)

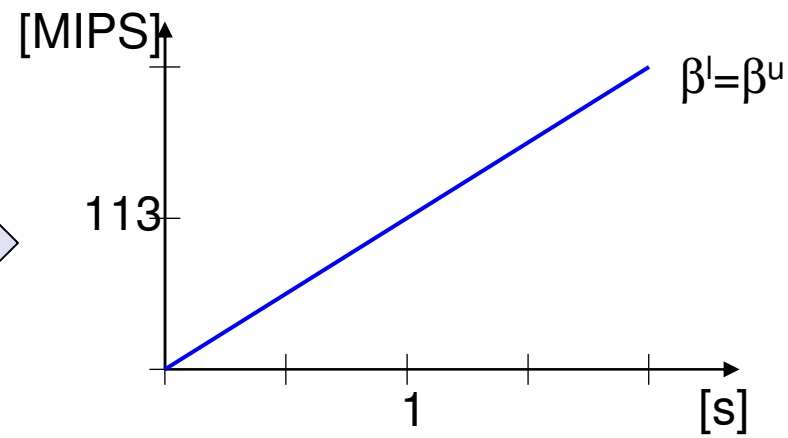
Event Stream Model

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



Resource Model

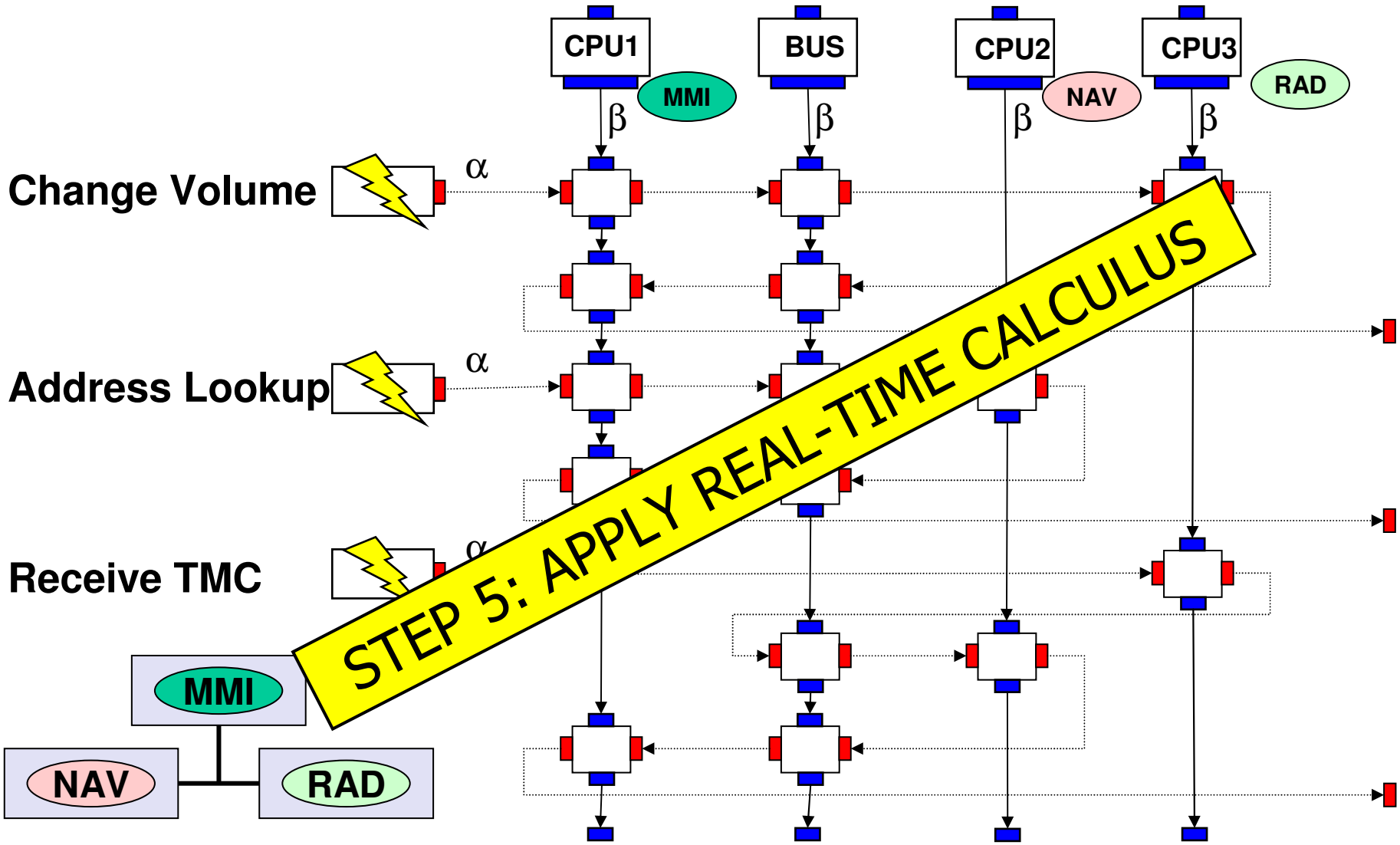
e.g. unloaded RISC CPU
(113 MIPS)



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)
- Consider scenario combinations
  - Change Volume / Receive TMC
  - Address Lookup / Receive TMC

Step 4: Scheduling Network

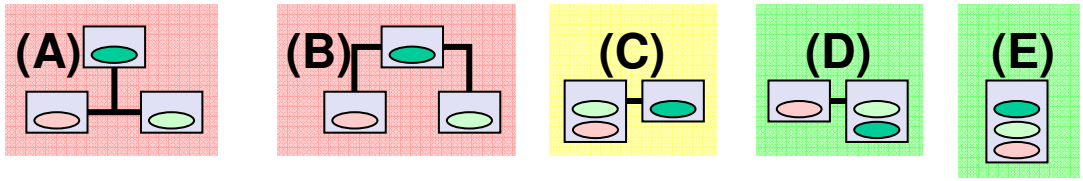


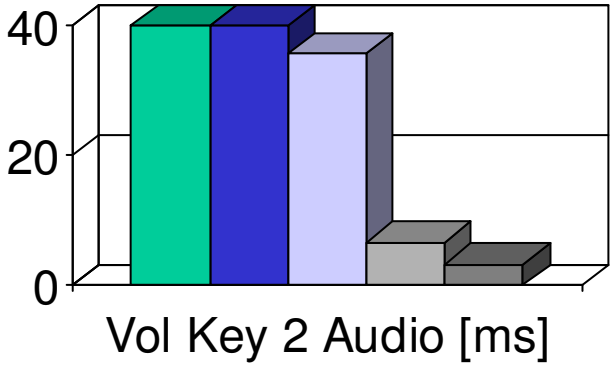
Analysis – Design Question 1

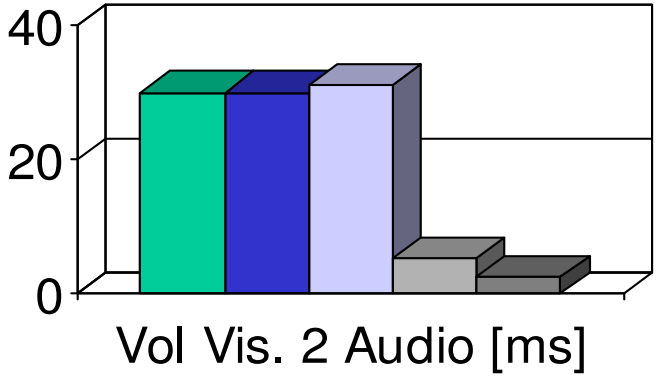
How do the proposed system architectures compare in respect to end-to-end delays?

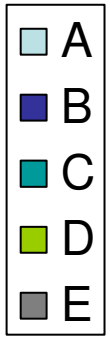
Analysis – Design Question 1

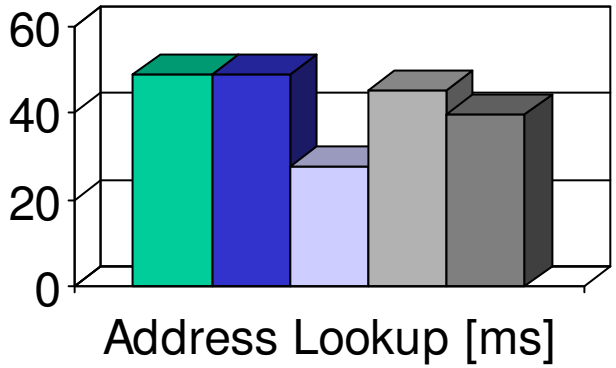
End-to-end delays:

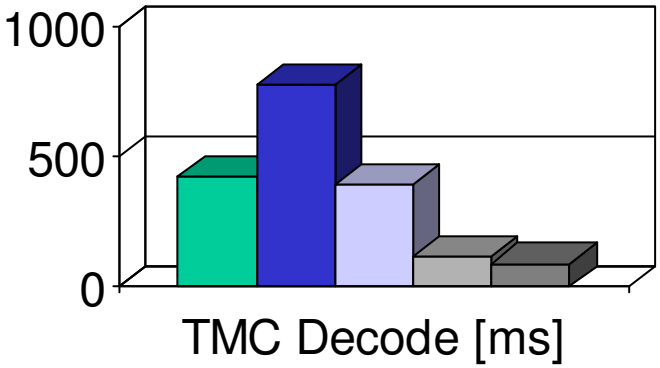








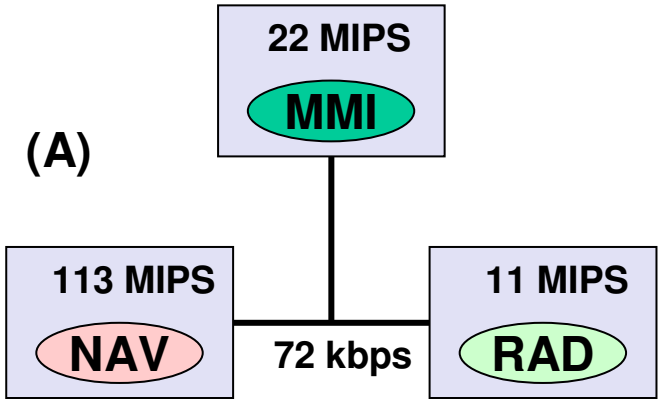




Analysis – Design Question 2

How robust is architecture A?

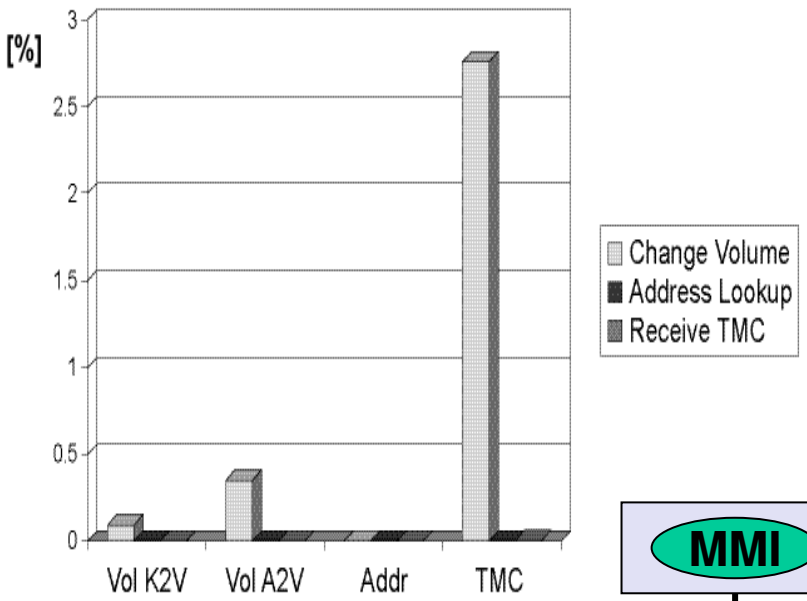
Where is the bottleneck of this architecture?

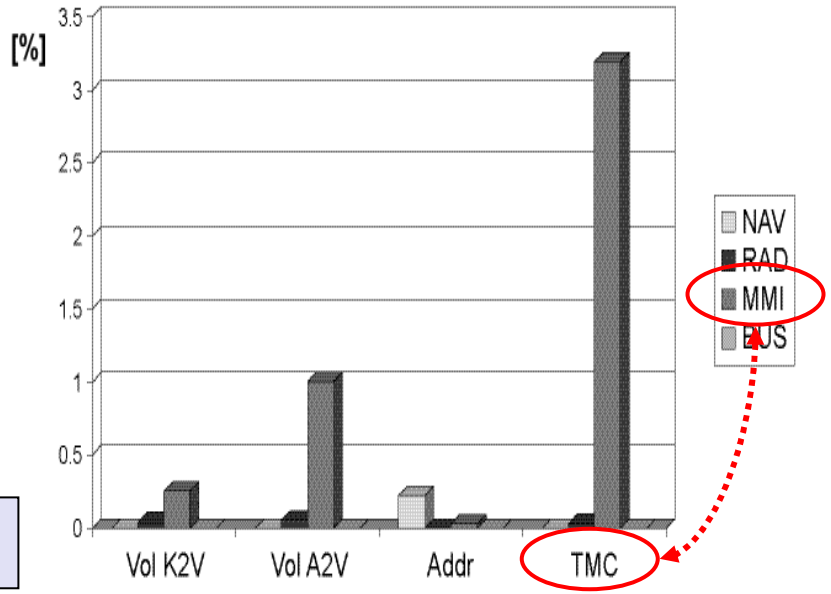


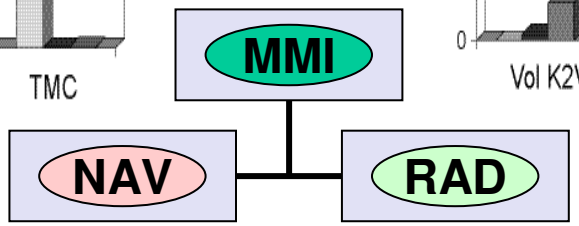
Analysis – Design Question 2

Sensitivity to input rate:

Sensitivity to resource capacity:



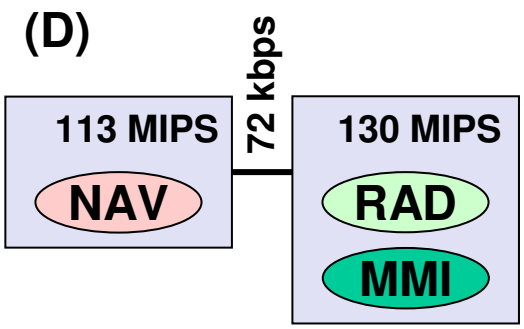




Analysis – Design Question 3

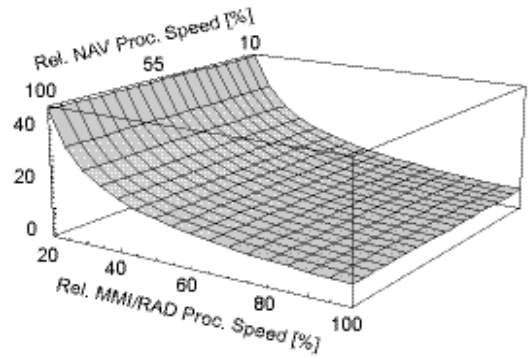
Architecture D is chosen for further investigation.

How should the processors be dimensioned?



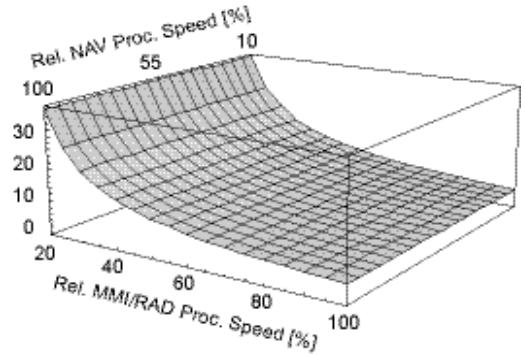
Analysis – Design Question 3

d_{max} = 200



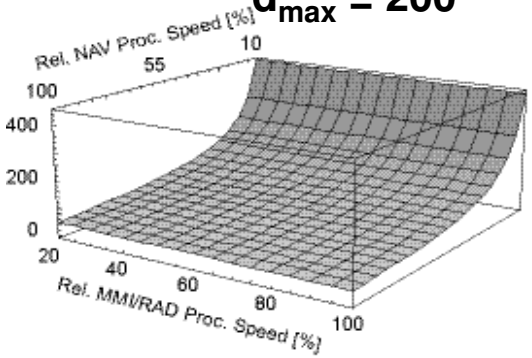
Vol K2V Delay [ms]

d_{max} = 50



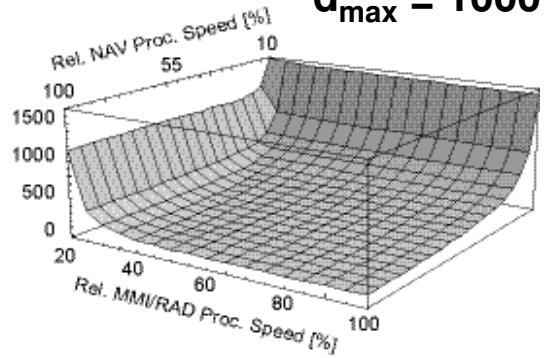
Vol A2V Delay [ms]

d_{max} = 200

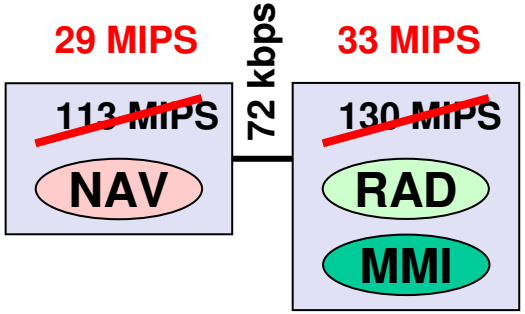


Addr Delay [ms]

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)
- “light weight” formal method

Future Work

- Add more functionality to explore system properties (step-wise curves, context dependant execution of tasks)
- Explore compositionality in more detail
- Quantative comparison to other techniques
- Validation (comparison to measured data)
- Enhance Tool Support
- From UML to Performance Model...