Distributed TDL Execution

Emilia Coste
Claudiu Farcas

Department of Computer Science
cs.uni-salzburg.at

UNIVERSITÄT SALZBURG

Overview

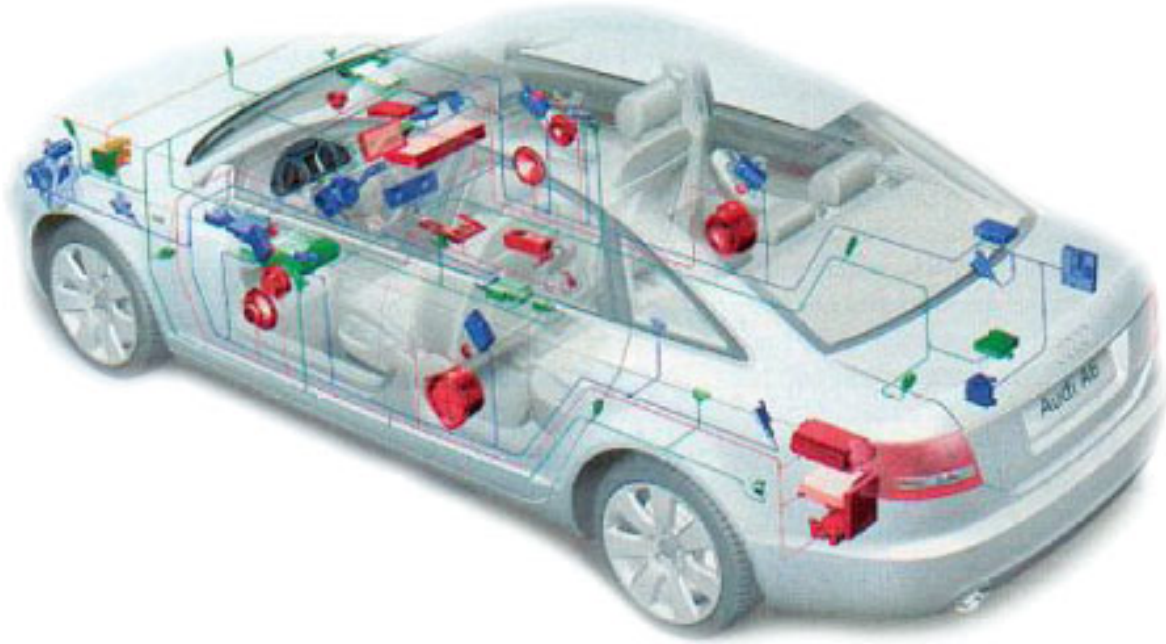
- What is TDL?
- Motivation
- Transparent Distribution
- Tool Chain
- Case Study

This presentation summarizes some results of the MoDECS (Model-Based development of Distributed Embedded Control Systems) Project: www.modecs.cc

What is TDL?

- Successor of Giotto
  - Syntax refinement
  - Simplifications (e.g., implicit drivers for task invocation, mode switches, and actuator updates)
- Component architecture:
  - Based on the notion of a *module*
  - Defines module *import*
  - Modules may run in parallel on the same CPU, and may switch mode independently of other modules.

Motivation



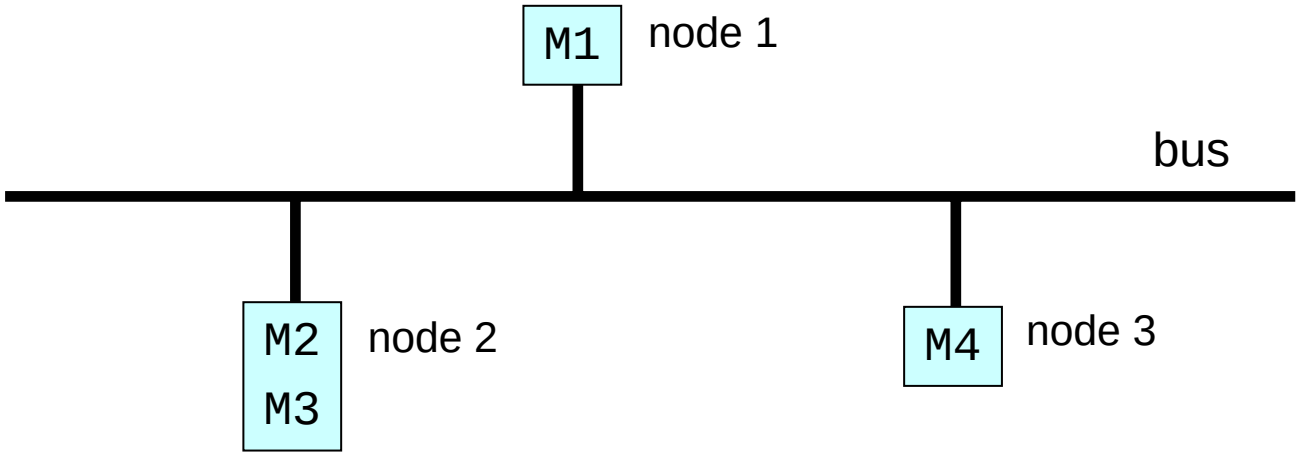
MOST-Bus

CAN-Bus

Some benefits of distribution:

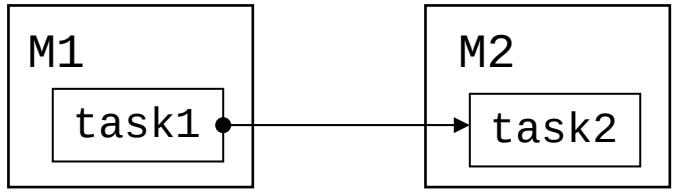
- Fault tolerance
- Scalability
- Less wiring

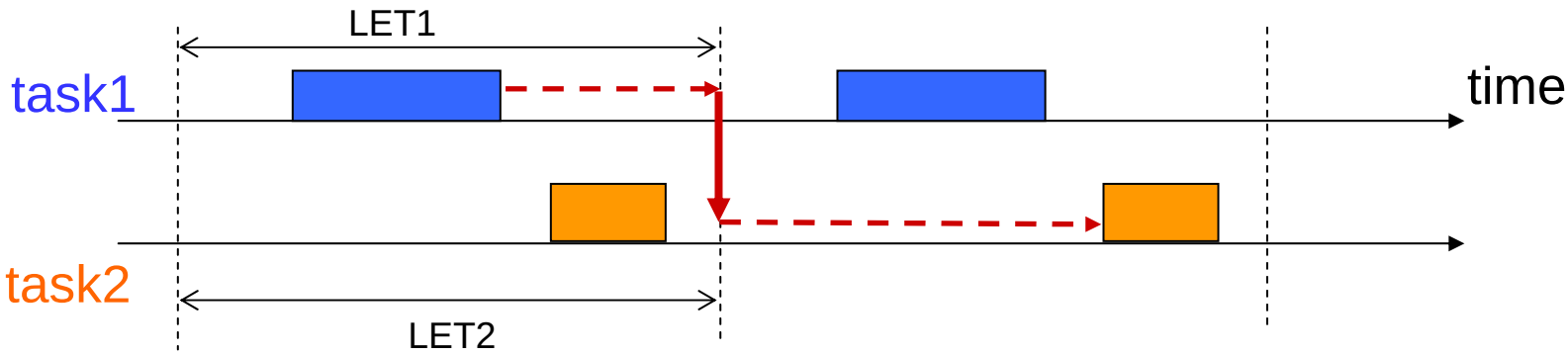
Introduction to Distributed TDL



Unit of distribution:	TDL module
Behavior:	as if executed locally
Communication:	via broadcast (bus)
Medium access control:	TDMA (time-slotting)
Cooperation model:	Producer-Consumer (Push)

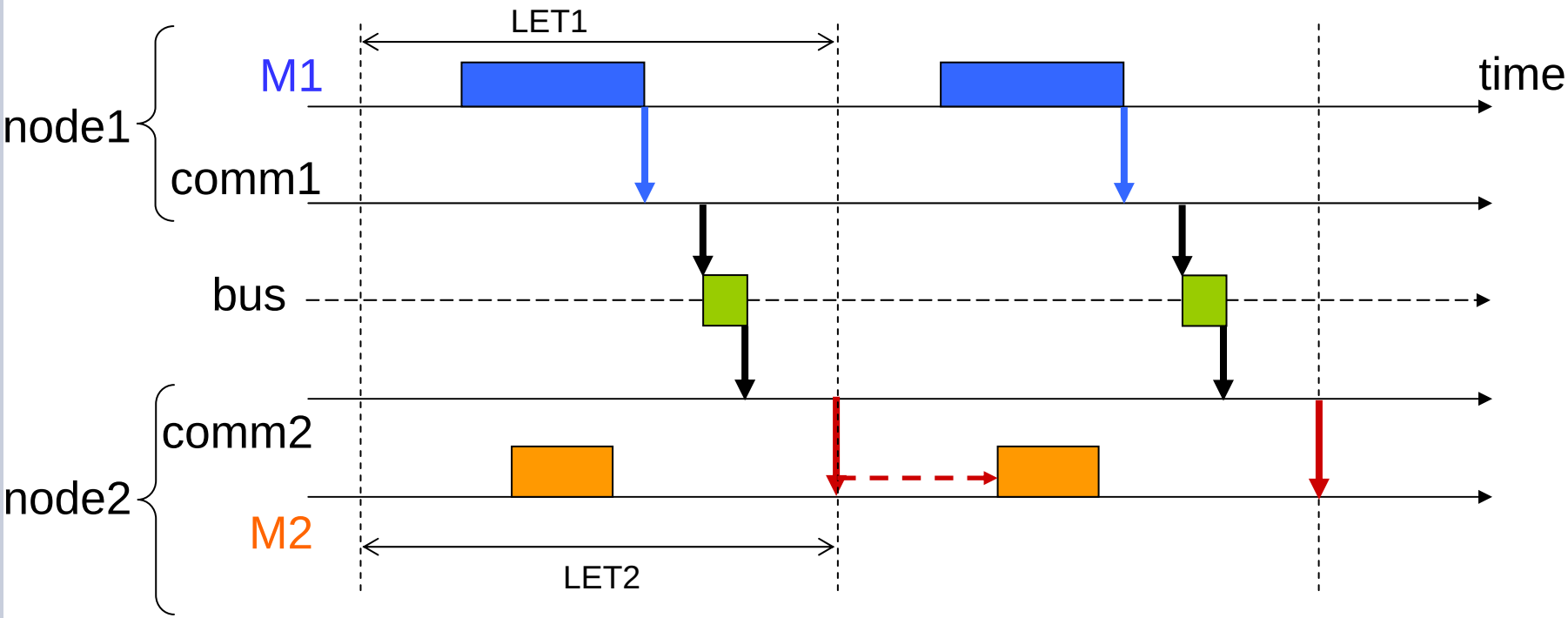
Transparent Distribution





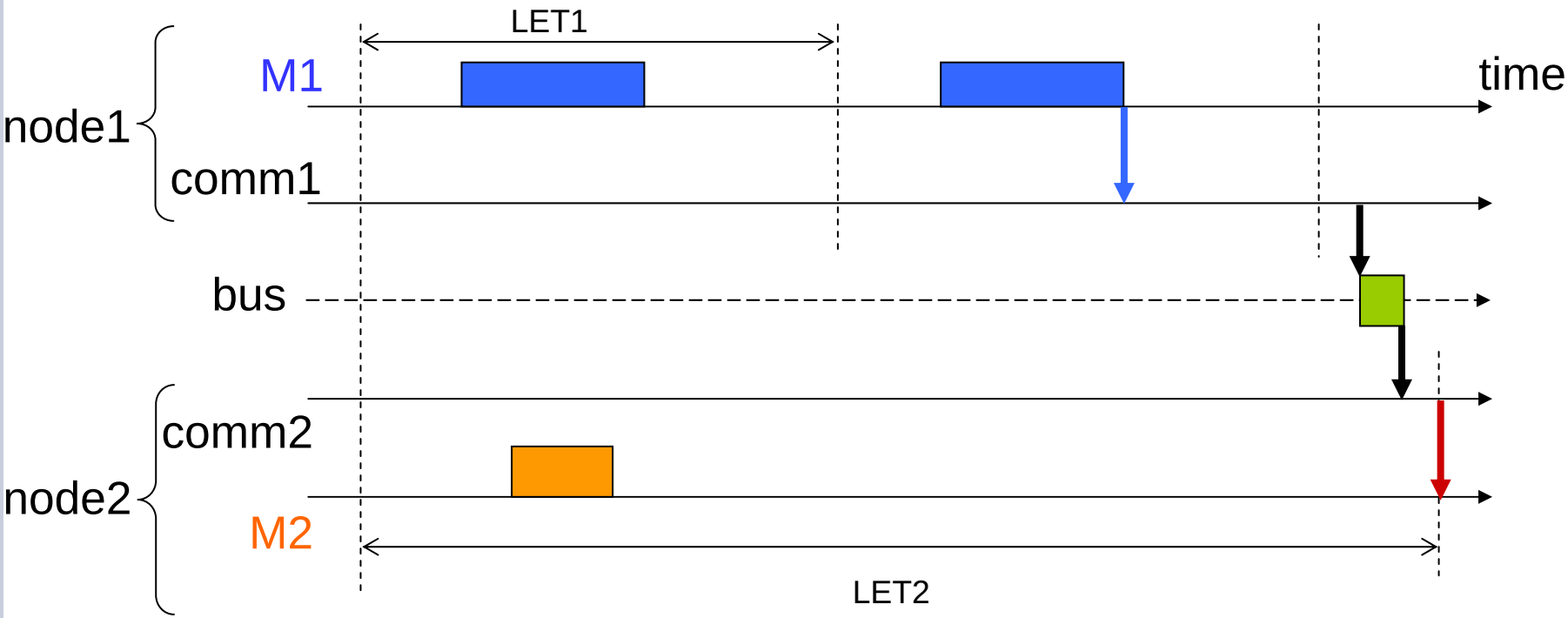
UNIVERSITÄT SALZBURG

Transparent Distribution



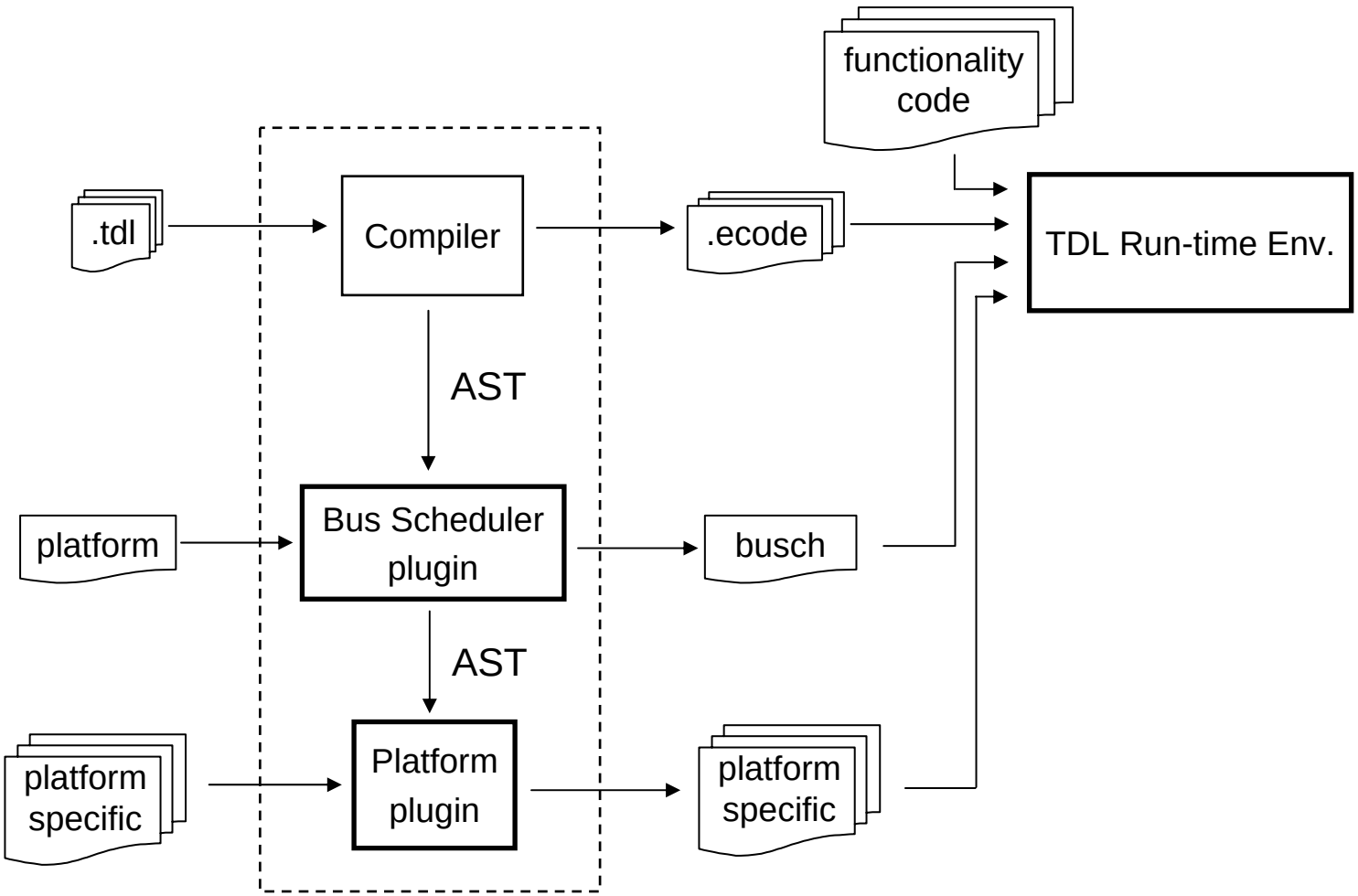
- message sent according to bus schedule (TDMA)

Optimizations



- if the consumer runs slower, redundant message are avoided
- if the consumer needs a variable later than the producer's LET, the message send can be delayed

Tool Chain



Bus Schedule Generation Tool

Need as input:

- TDL modules
- Platform description file: module to node assignment, and physical bus properties.

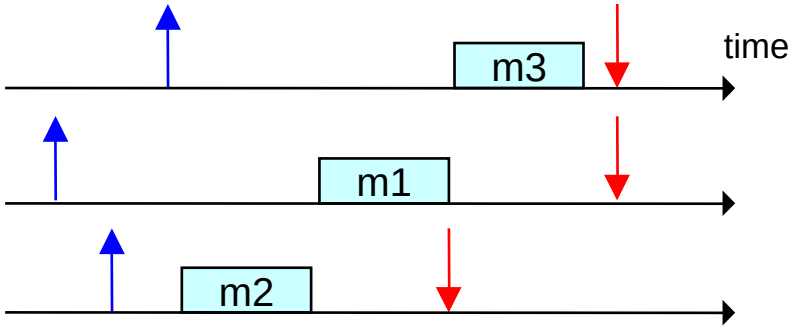
The tool automatically:

- Detects who has to communicate with whom
- Computes which messages are needed in a communication cycle.
- Schedules messages as late as possible (variant of Reversed EDF).

Provides as output a *global bus schedule* file, which contains:

- Which node has to send/receive a packet and when.
- The content for bus packets

Message Scheduling - Example



Rev. EDF scheduling {m2, m1, m3}

- Sorts the list of messages by message deadline and then release time.
- Bus Scheduler is non-preemptive and just schedules the messages in the resulted order, starting from the end of the bus period and going backwards.
- Bus Properties are constraints for the scheduler

TDL Run-time Environment for OSEK

- E-machine (8KB) – multimode and multi-module capable
- Dispatcher – uses OSEK system calls for handling tasks

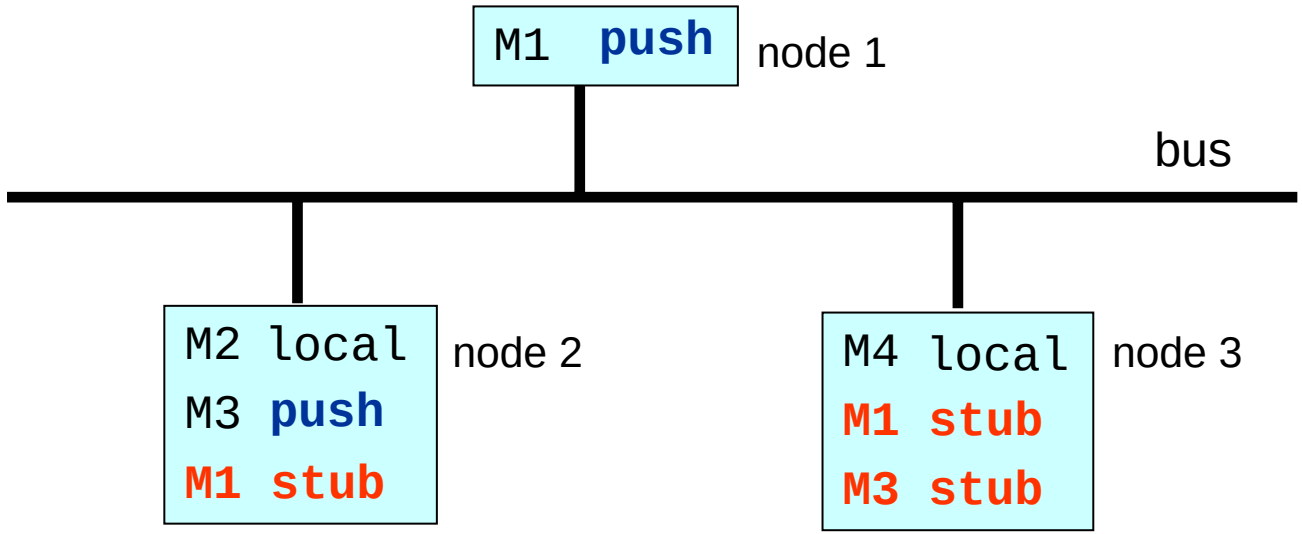
- For distribution
  - TDL-comm - provides transparent communication
    - Port mapping - dependencies are solved at compile time
  - Global clock sync
    - sync frame every network round
    - synchronizes OSEK system time on different nodes
  - Bus schedule – generated at compile time

E-Machine

Supports three kinds of module execution: local, push, and stub.

M2 imports M1

M4 imports M1, M3



UNIVERSITÄT SALZBURG

Case Study

- Two modules on a single node/two nodes system
- M1 implements a ramp signal generator with two modes
  - One input sensor as mode switch trigger
  - Two actuators for two signals (rising/falling)
- M2 imports M1
  - Two inputs from the M1 output ports
  - One output as the sum of the two input signals

- Goal: same behavior on both systems (single/dual node)
- Requirements: transparent distribution

Thank you for your attention!