Distributed, Time-Safe TDL Execution — Concepts, Tools and Run-time Infrastructure

Emilia Coste
Josef Templ

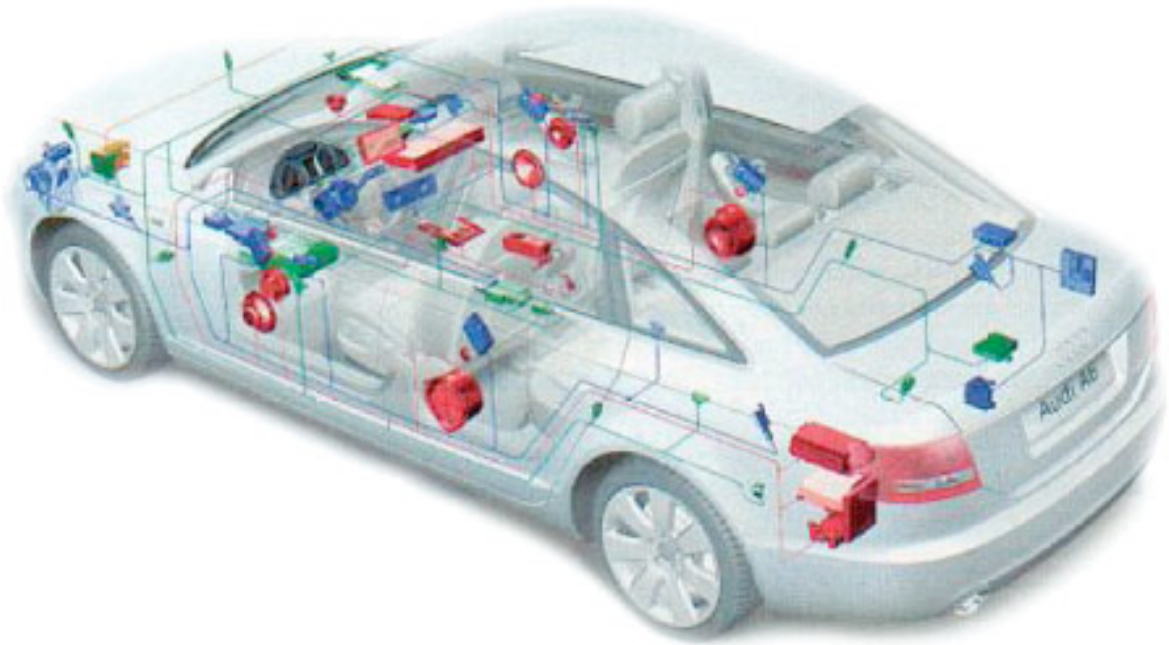
Department of Computer Science
cs.uni-salzburg.at
www.MoDECS.cc

UNIVERSITÄT SALZBURG

Overview

- Motivation
- Transparent Distribution
- Bus Schedule Generation Tool
- TDL Run-time Environment
- Tool Chain

Motivation



MOST-Bus

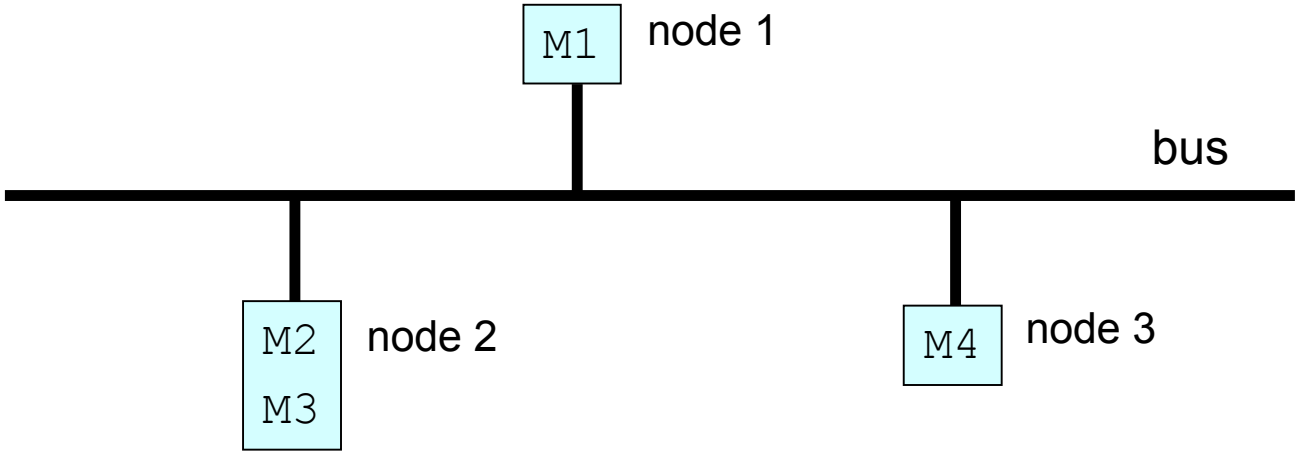
CAN-Bus

Some benefits of distribution:

- Fault tolerance
- Scalability
- Less wiring

UNIVERSITÄT SALZBURG

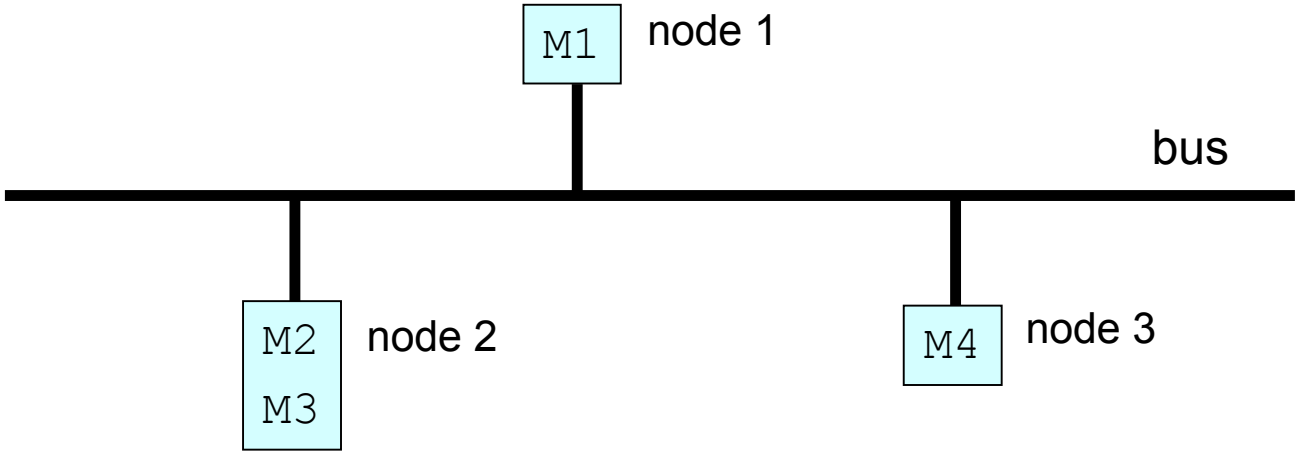
Introduction to Distributed TDL

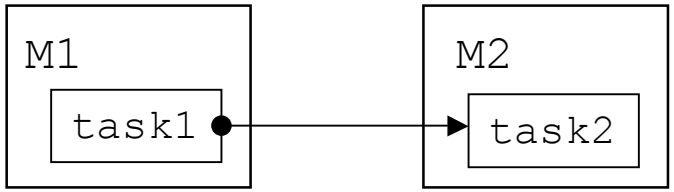


Unit of distribution:	TDL module
Behavior:	as if executed locally
Communication:	via broadcast (bus)
Medium access control:	TDMA (time-slotting)
Cooperation model:	Producer-Consumer (Push)

UNIVERSITÄT SALZBURG

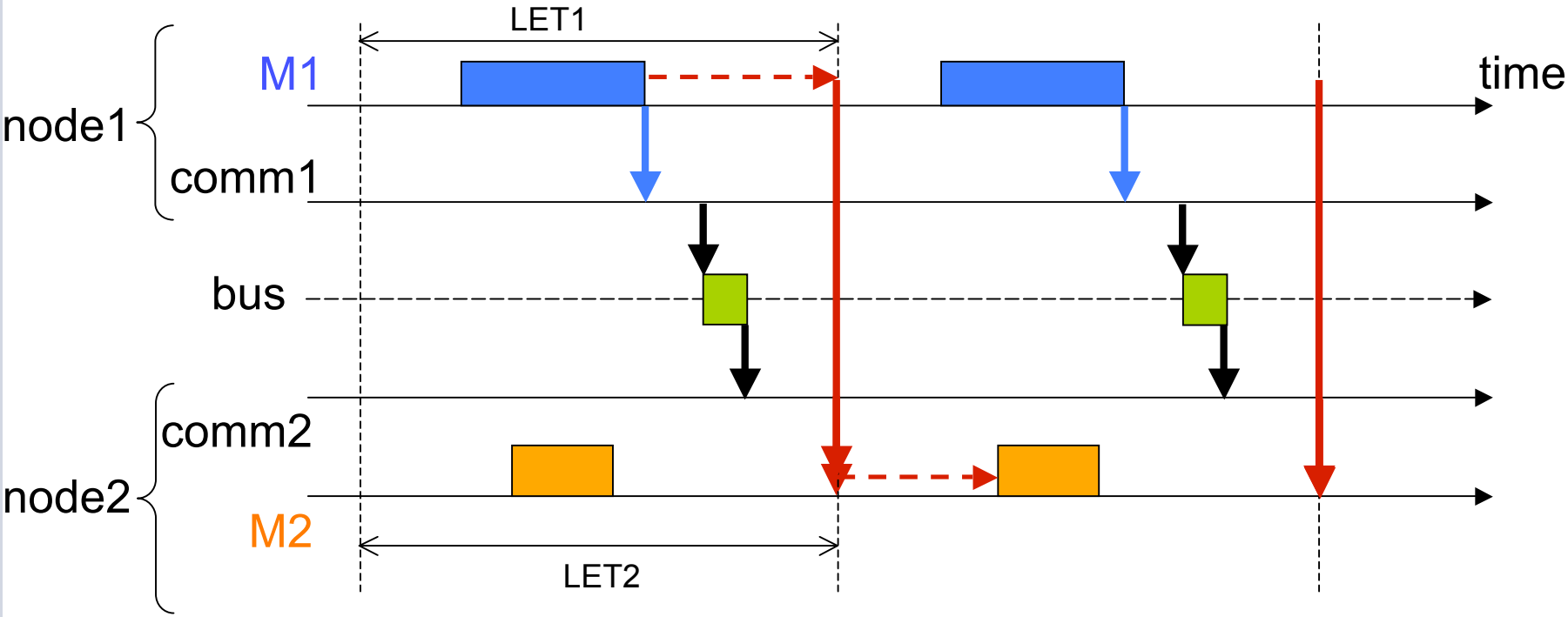
Example of Distributed TDL





UNIVERSITÄT SALZBURG

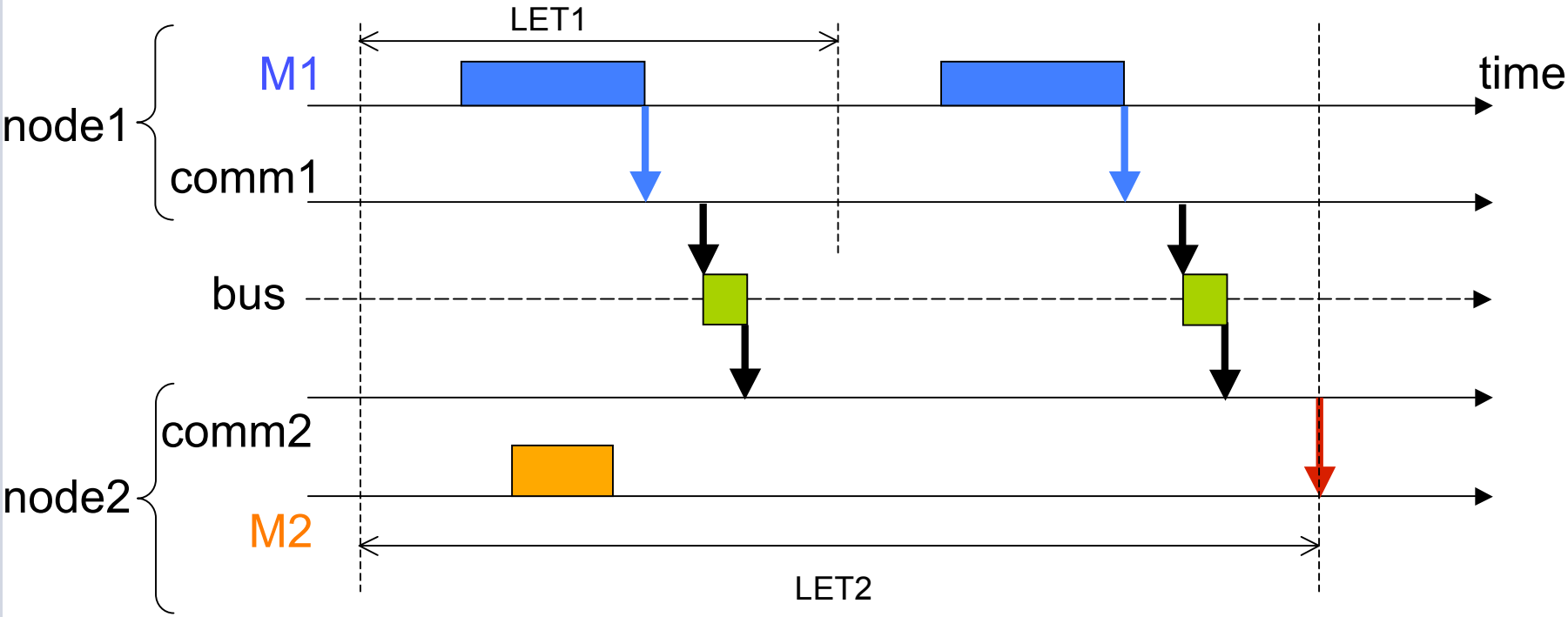
Transparent Distribution



- message sent according to <u>bus schedule</u> (TDMA)

UNIVERSITÄT SALZBURG

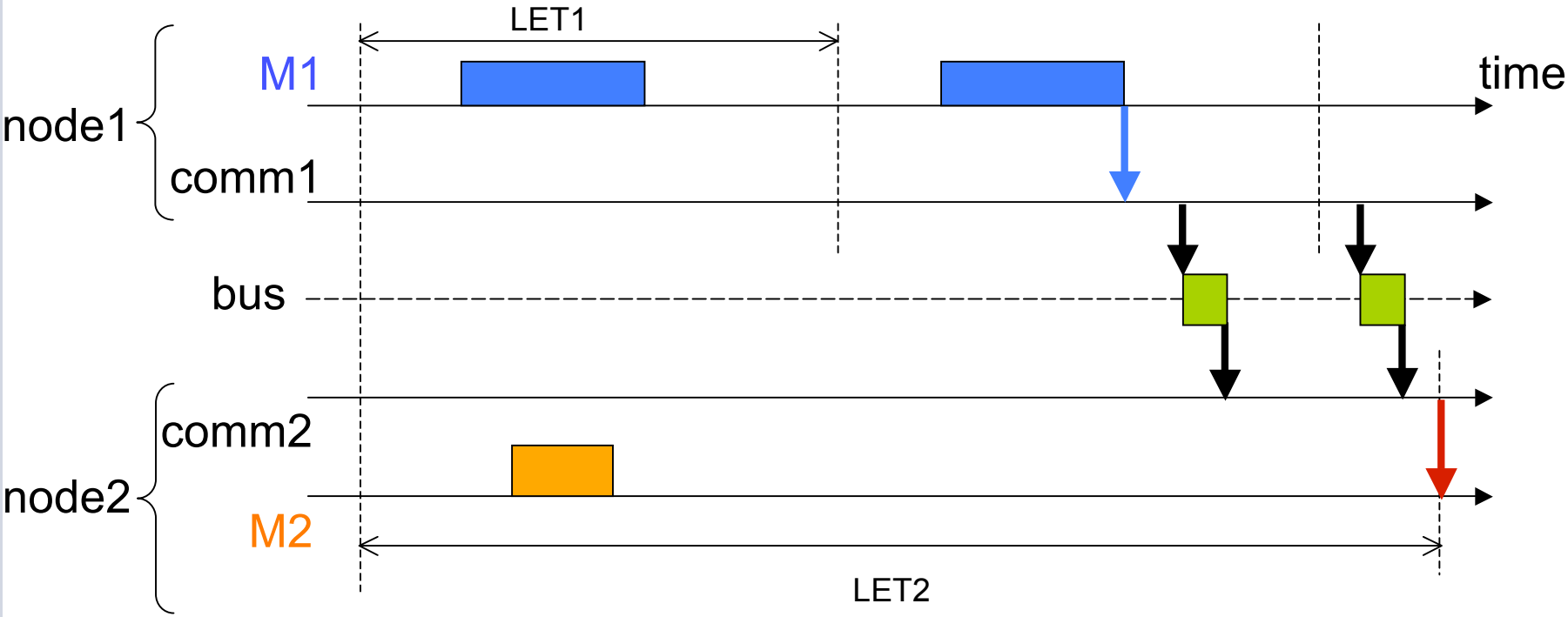
Optimization I



- if the consumer runs slower e.g. by a factor of 2
- redundant message are avoided
- saves bandwidth

UNIVERSITÄT SALZBURG

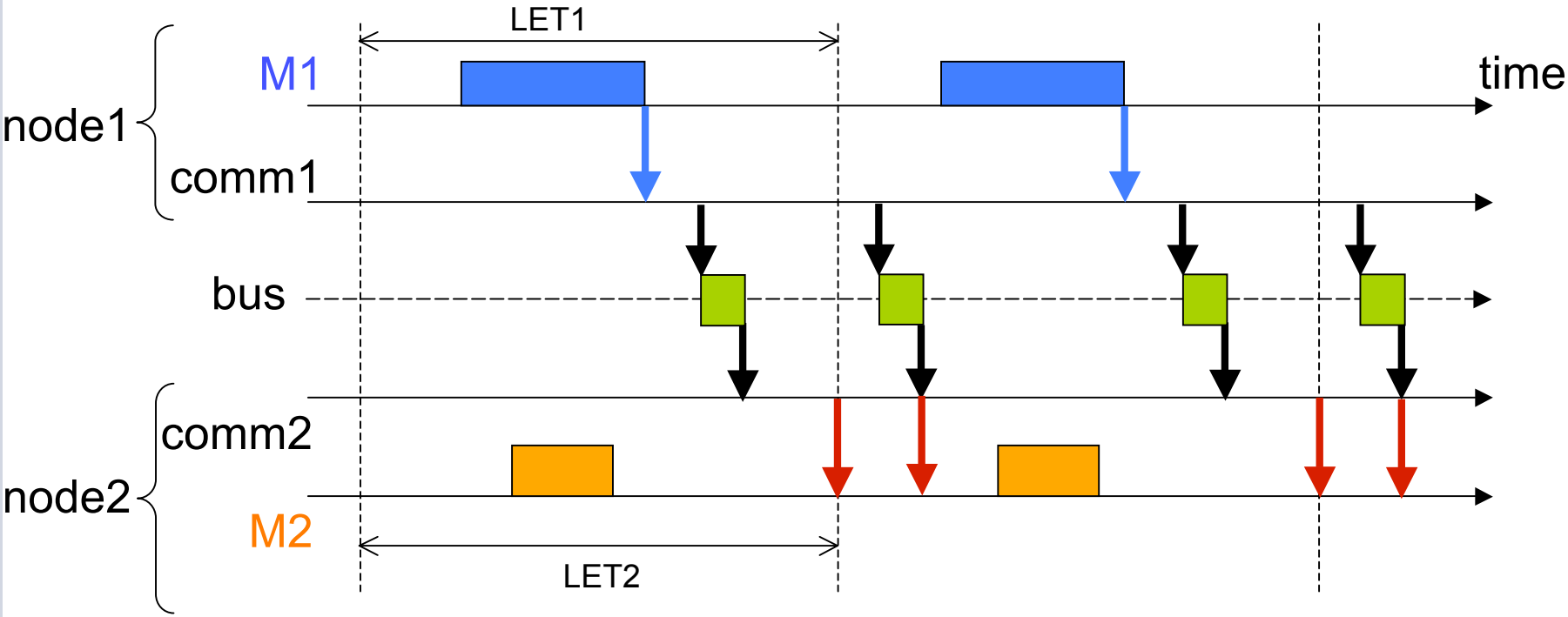
Optimization II



- if the consumer needs a variable later than the producer's FLET

UNIVERSITÄT SALZBURG

Optimization III



- the release of the consumer can be delayed until the message with the input variable is received

UNIVERSITÄT SALZBURG

Bus Schedule Generation Tool

UNIVERSITÄT SALZBURG

What Does the Tool Do?

It generates a global bus schedule file, which contains the following information:

- Which node has to send a packet and when.
- Which nodes have to receive a packet and when.
- The content for bus packets (a corresponding datagram, which has one or more items).

What Does the Tool Need as Input?

- TDL modules
- Platform description file
  - module to node assignment
  - physical bus properties (e.g., bus frequency, protocol overhead, inter frame gaps, min/max payload)

The tool automatically detects:

- Who has to communicate with whom.
- Which messages are needed in a communication cycle (bus period).

Who Has to Communicate with Whom

Result: a set of messages.

- A message has:
  - a Producer
  - one or more Consumers
  - size.

- Producers: sensors, task output ports.
- Consumers: actuators, task input ports, guard arguments.

UNIVERSITÄT SALZBURG

Messages Needed in a Bus Period

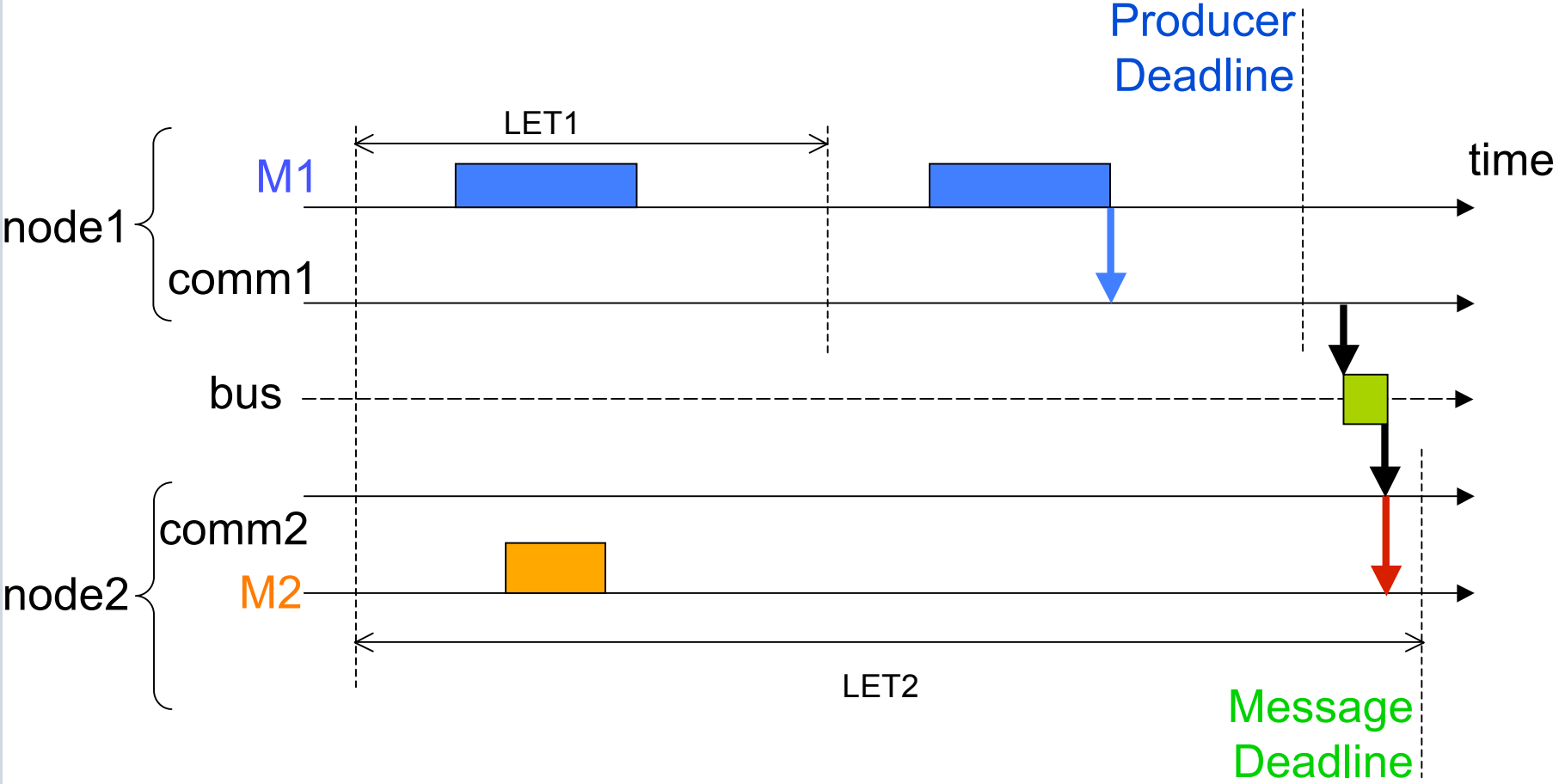
Result: a set of message instances, with individual timing constraints:

- Release Offset
- Deadline

- Basic Producer-Consumer:
  - Send messages with the frequency of the Producer.
  - Message deadline = Producer LET.
  - BusPeriod = LCM(Producer.period)

- Optimized Producer-Consumer:
  - Send messages only when they are needed by the Consumers.
  - Message deadline depends on the optimization (e.g.,= consumer LET).
  - BusPeriod = LCM(Producer.period, Consumer.period)

Message Deadline in Optimization II



UNIVERSITÄT SALZBURG

Message Scheduling

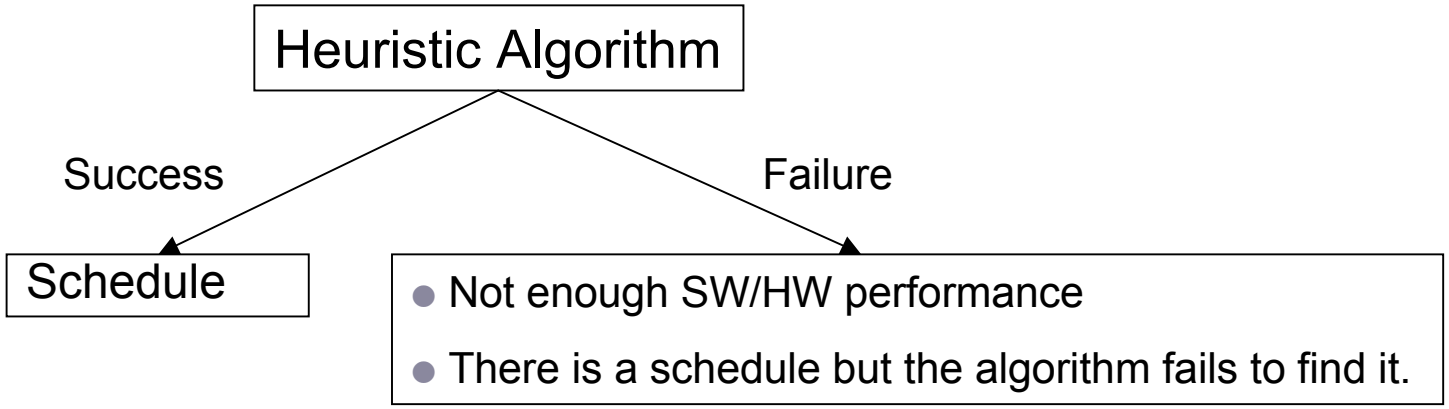
Current approach:

- Scheduling in 2 steps:
  - Schedule first the messages.
  - Schedule then the tasks with deadlines constraints from messages.

- Optimizations:
  - Build bus schedulers which allow more flexibility for the task scheduler.
  - Try several bus schedulers and get feedback from the Time-Safety-Check (TSC) for tasks.
  - Schedule individual messages or merge messages sent from the same node.

Scheduling Algorithms

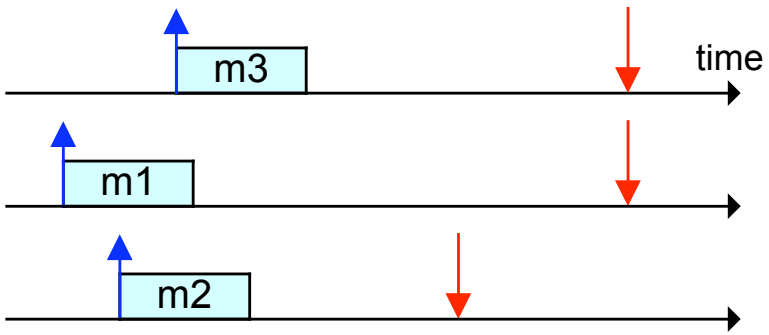
- Heuristic algorithm - “Latest Deadline Last” - LDL
  - Adapted from Reversed EDF (Latest Release Time - LRT) - treats deadlines as release times and vice versa
  - Schedule messages as late as possible



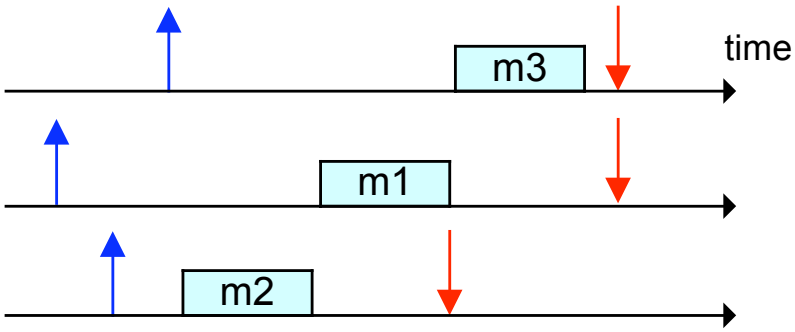
- Optimal algorithm
  - Branch and bound search
  - Exponential complexity in the worst case

UNIVERSITÄT SALZBURG

Latest Deadline Last - Example



Released messages {m1, m2, m3}



LDL scheduling {m2, m1, m3}

UNIVERSITÄT SALZBURG

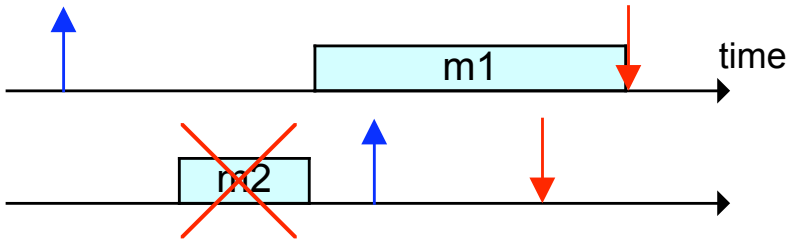
Latest Deadline Last

- Sorts the list of messages by:
  - Key1 = message deadline
  - Key2 = message release time
  - Key3 = producer deadline.

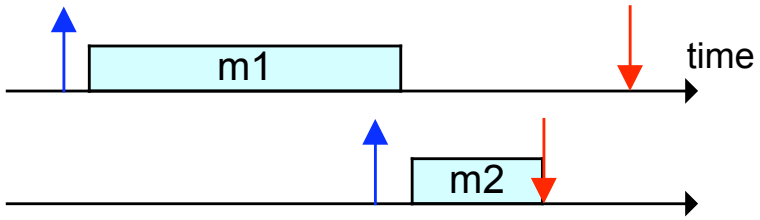
- Bus Scheduler is non-preemptive and just schedules the messages in the resulted order.
  - Starts from the end of the Bus Period and goes backwards.
  - Merges messages if they have to be sent by the same node, and are adjacent.

UNIVERSITÄT SALZBURG

Search Scheduler - Example



LDL scheduling failure {m2, m1}



Search scheduler {m1, m2}

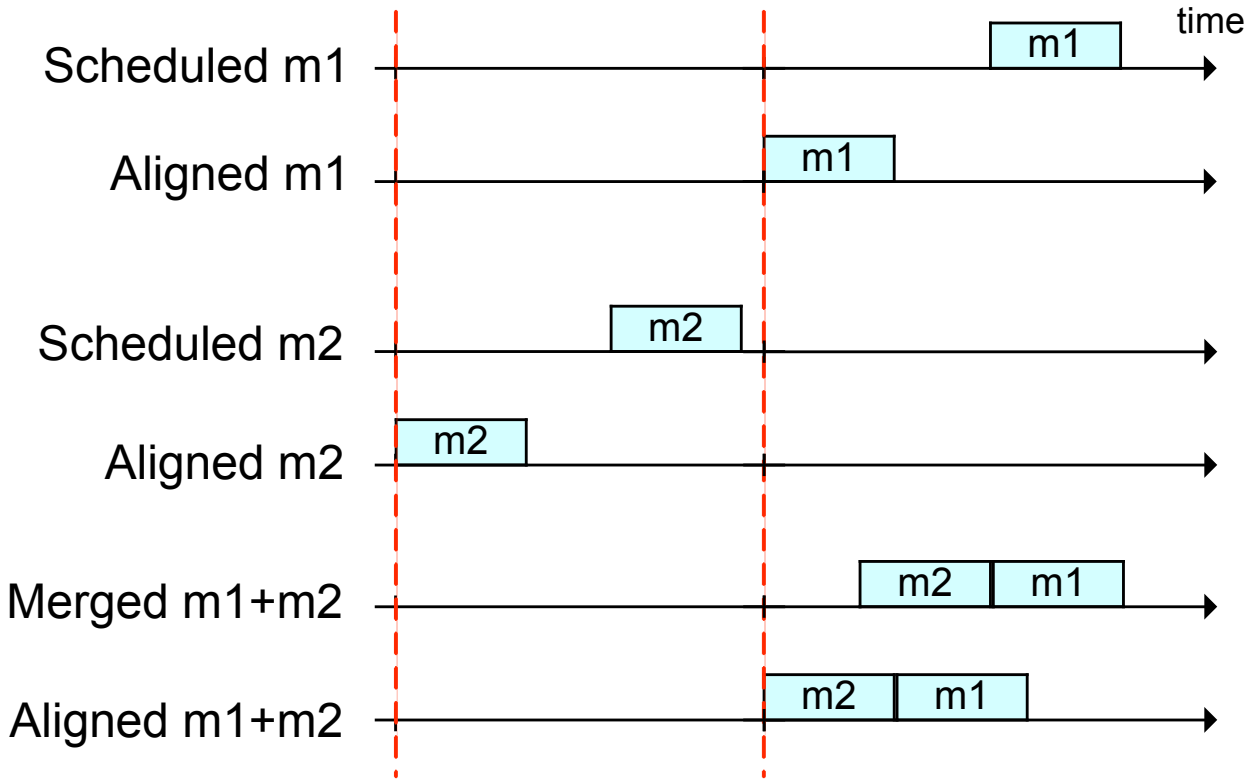
Bus Properties as Constraints

- Relevant for:
  - Merging messages (min/max payload)
  - WCCT (Bps, protocol overhead)
  - Time alignment (inter frame gaps, clock resolution)
  - Control packets (time synchronization)

- Clock Resolution:
  - TDL time unit is microsecond (us).
  - Different platforms have a given clock resolution (e.g., 1ms or 100us).
  - Bus communication is computed in microseconds or even nanoseconds.

UNIVERSITÄT SALZBURG

Merging Messages and Clock Resolution



UNIVERSITÄT SALZBURG

We do Various Measurements as Basis for Optimizations

Metrics relevant for efficient bus utilization:

- Throughput
- Bus utilization
- Average data efficiency
- Maximum and average sending rates
- Maximum and average receiving rates

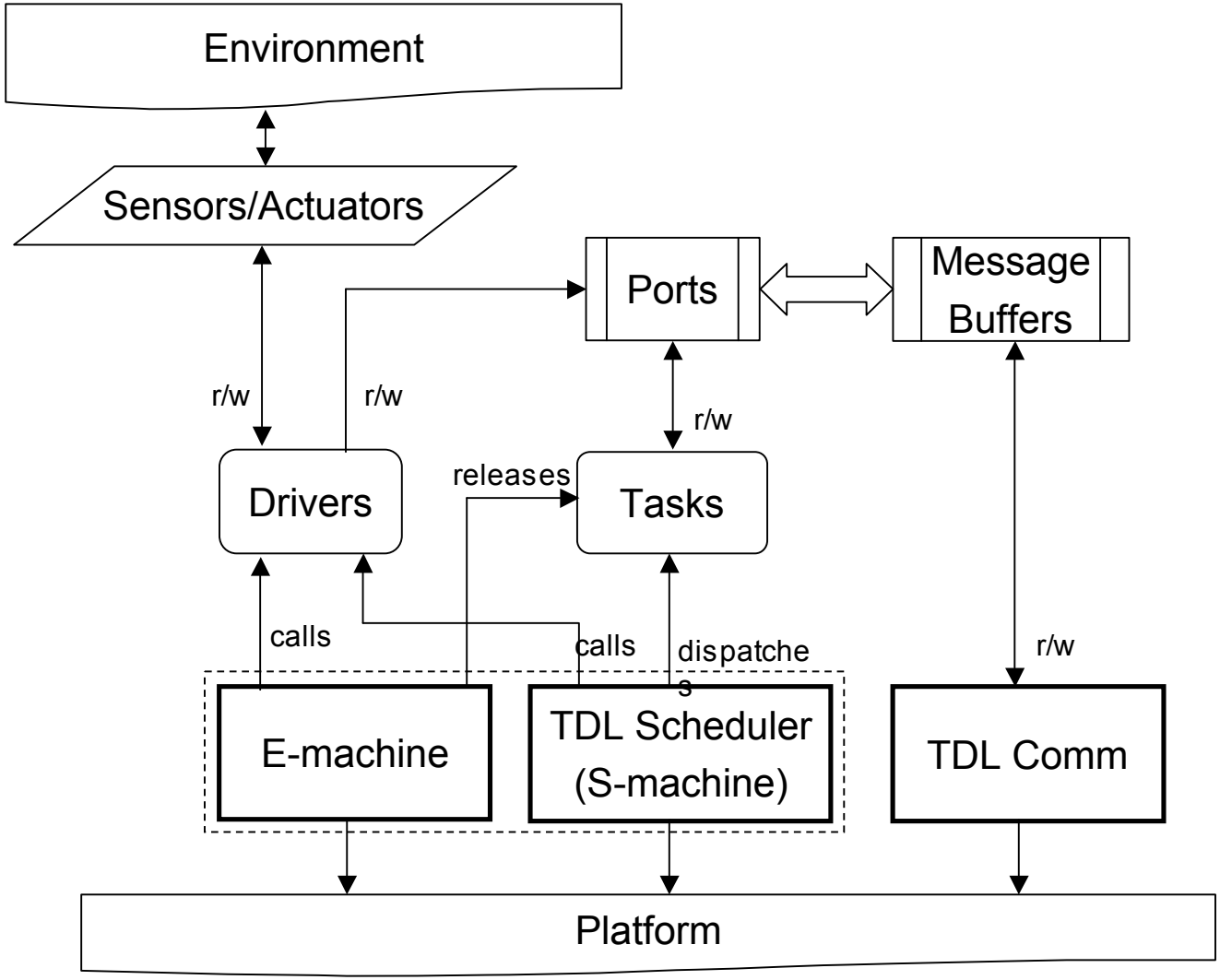
Metrics relevant for flexibility in task scheduling:

- Minimum and average release-send intervals
- Minimum and average relative release-send intervals

TDL Run-time Environment

UNIVERSITÄT SALZBURG

TDL Run-time Environment



UNIVERSITÄT SALZBURG

E-Machine Operation

- Executes E-code instructions at logical time instances
- Implementation is platform dependent (OSEK, InTime, RTLinux, Java)
- It is fast and lightweight (e.g. 8KB for OSEK E-machine).

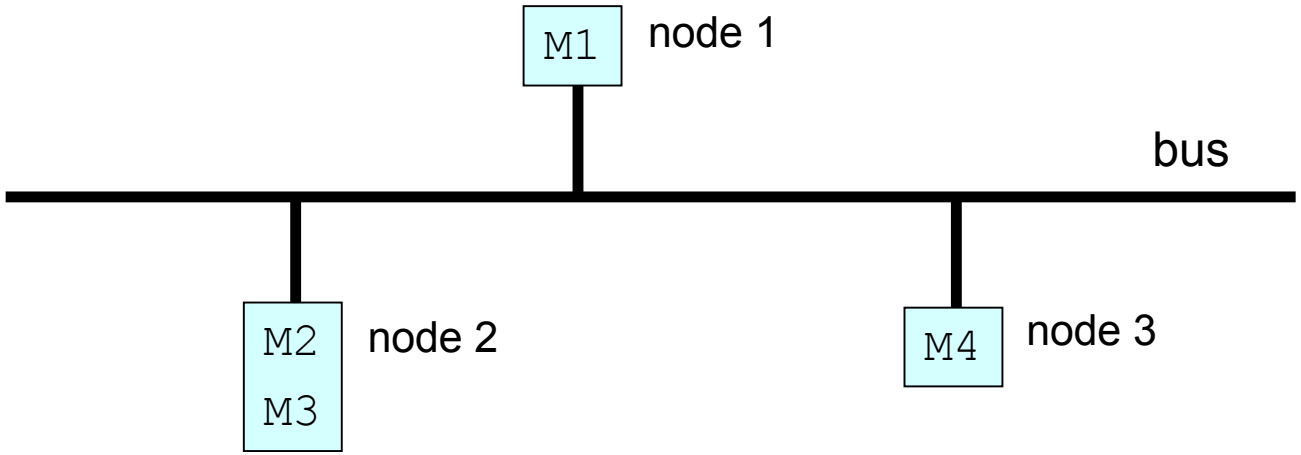
- Supports three kinds of module execution: local, push, and stub.

UNIVERSITÄT SALZBURG

E-Machine

M2 imports M1

M4 imports M1, M3

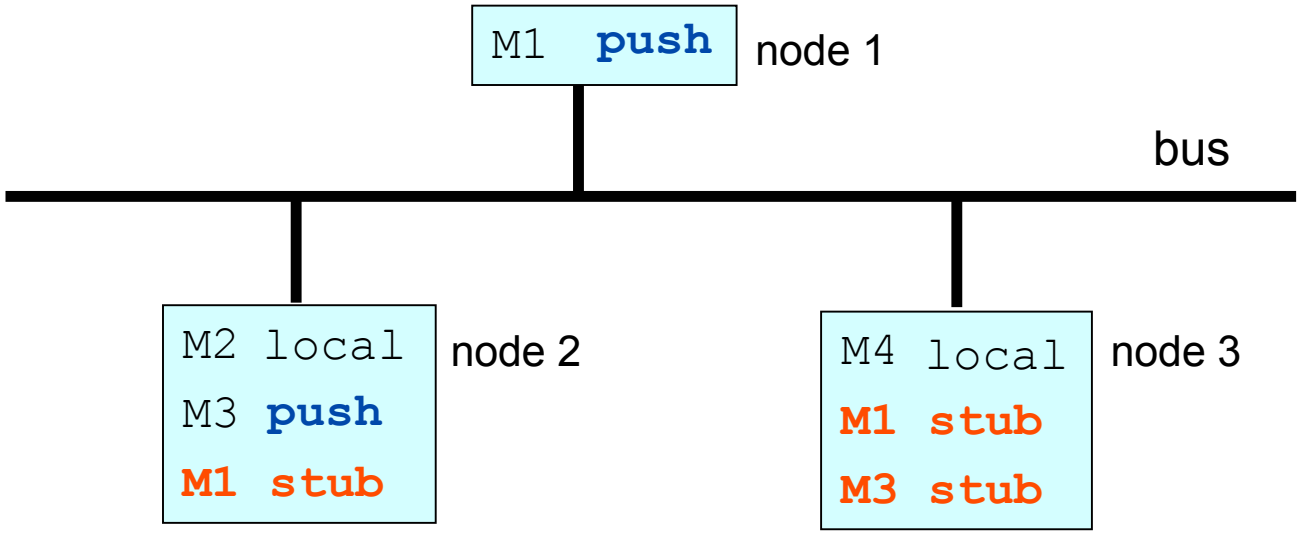


UNIVERSITÄT SALZBURG

E-Machine

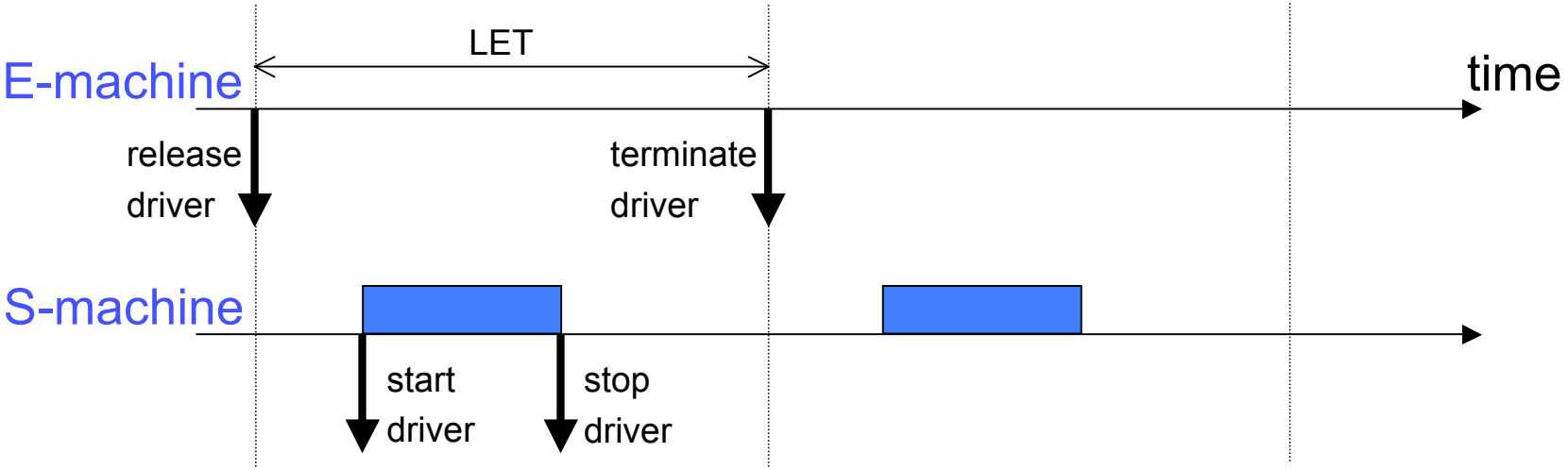
M2 imports M1

M4 imports M1, M3



UNIVERSITÄT SALZBURG

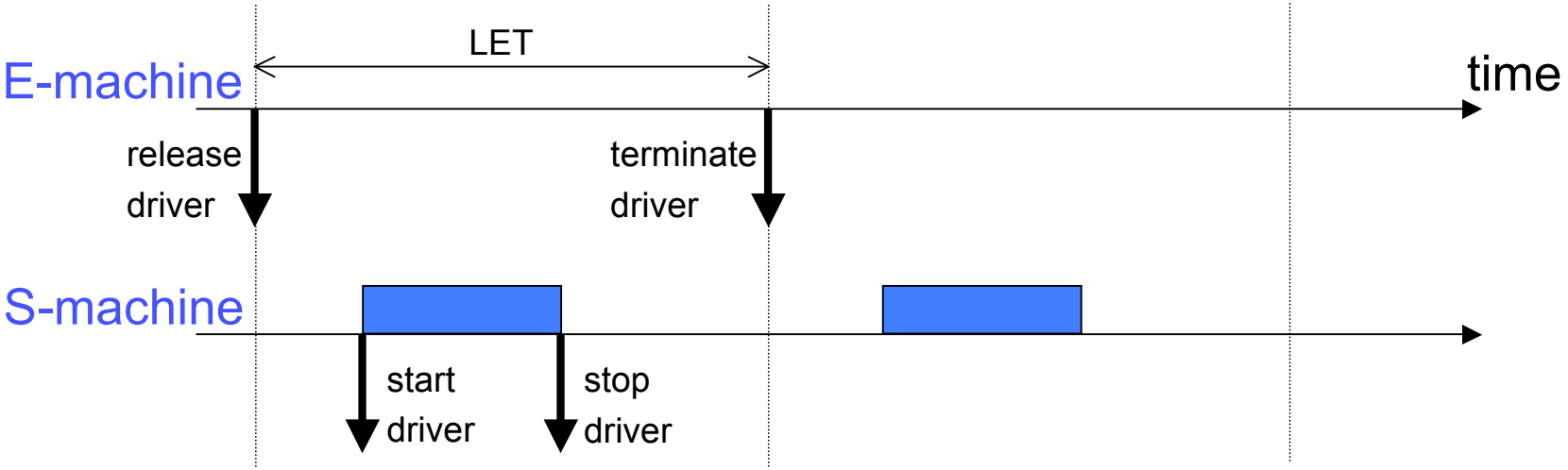
LOCAL



- release driver - copies input arguments
- terminate driver - copies output arguments
- start driver - calls task impl. function
- stop driver - *noop*

UNIVERSITÄT SALZBURG

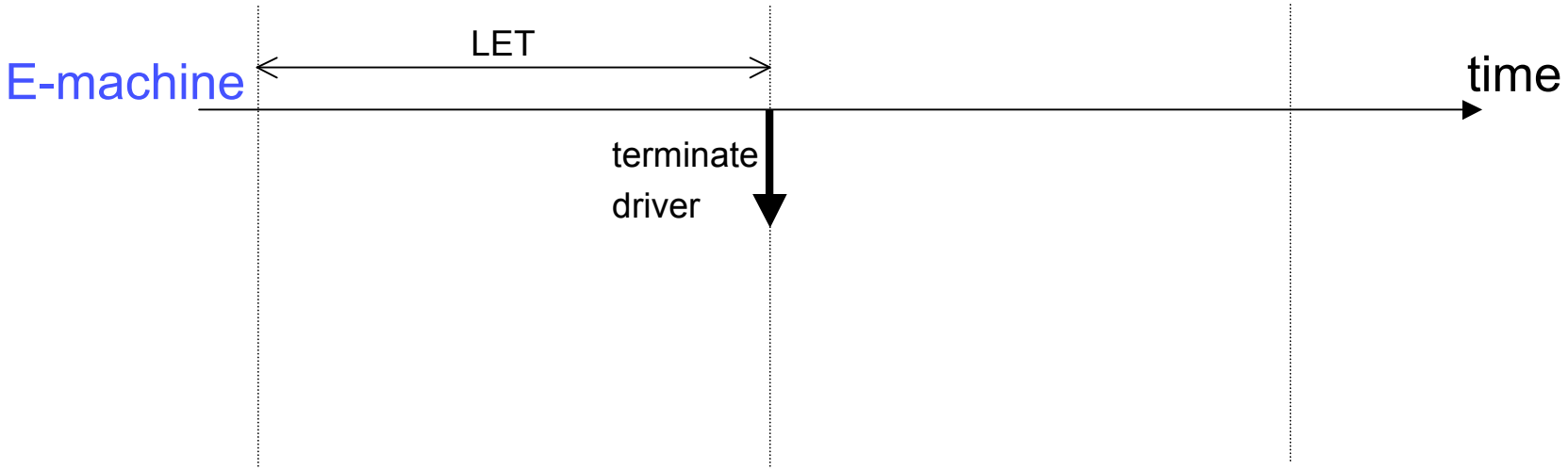
PUSH



- release driver - same as LOCAL
- terminate driver - same as LOCAL
- start driver - same as LOCAL
- stop driver - *copy results to TDLcomm*

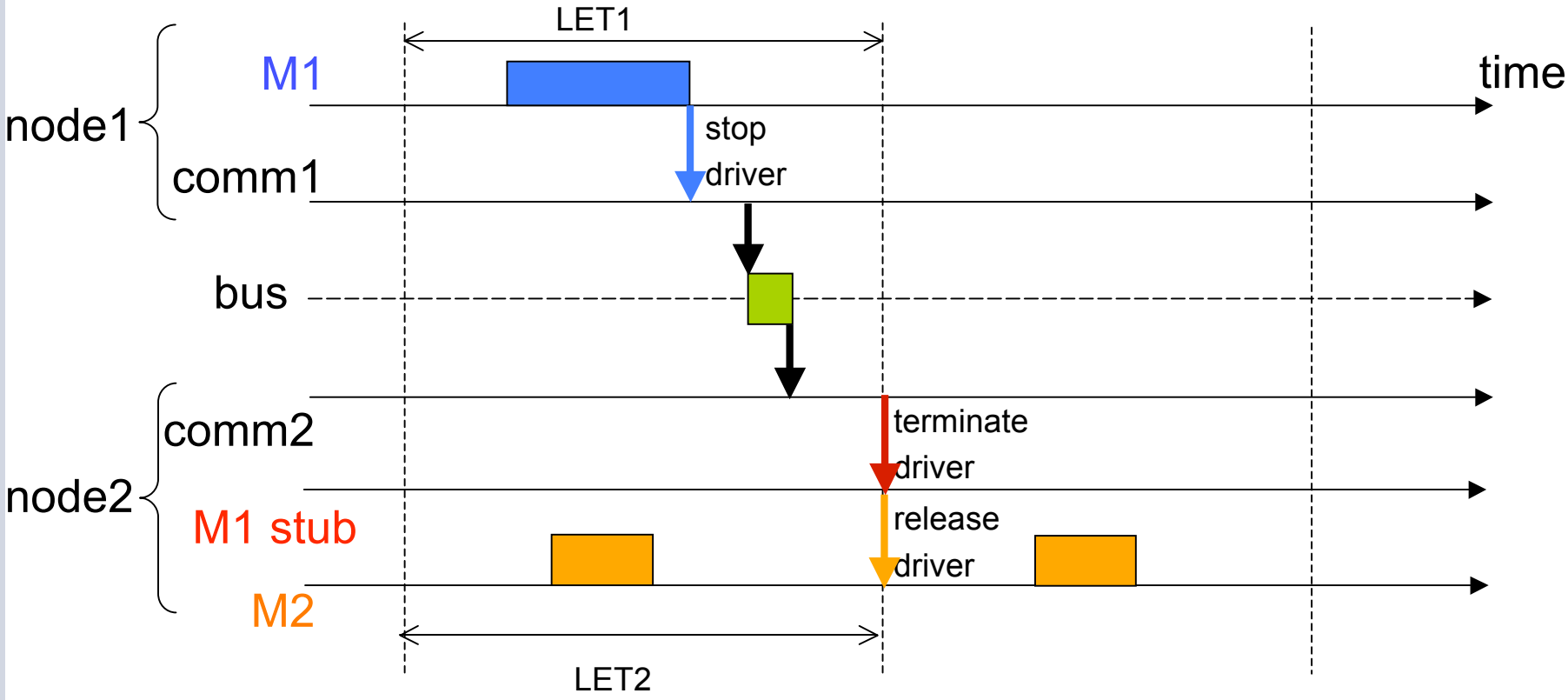
UNIVERSITÄT SALZBURG

STUB

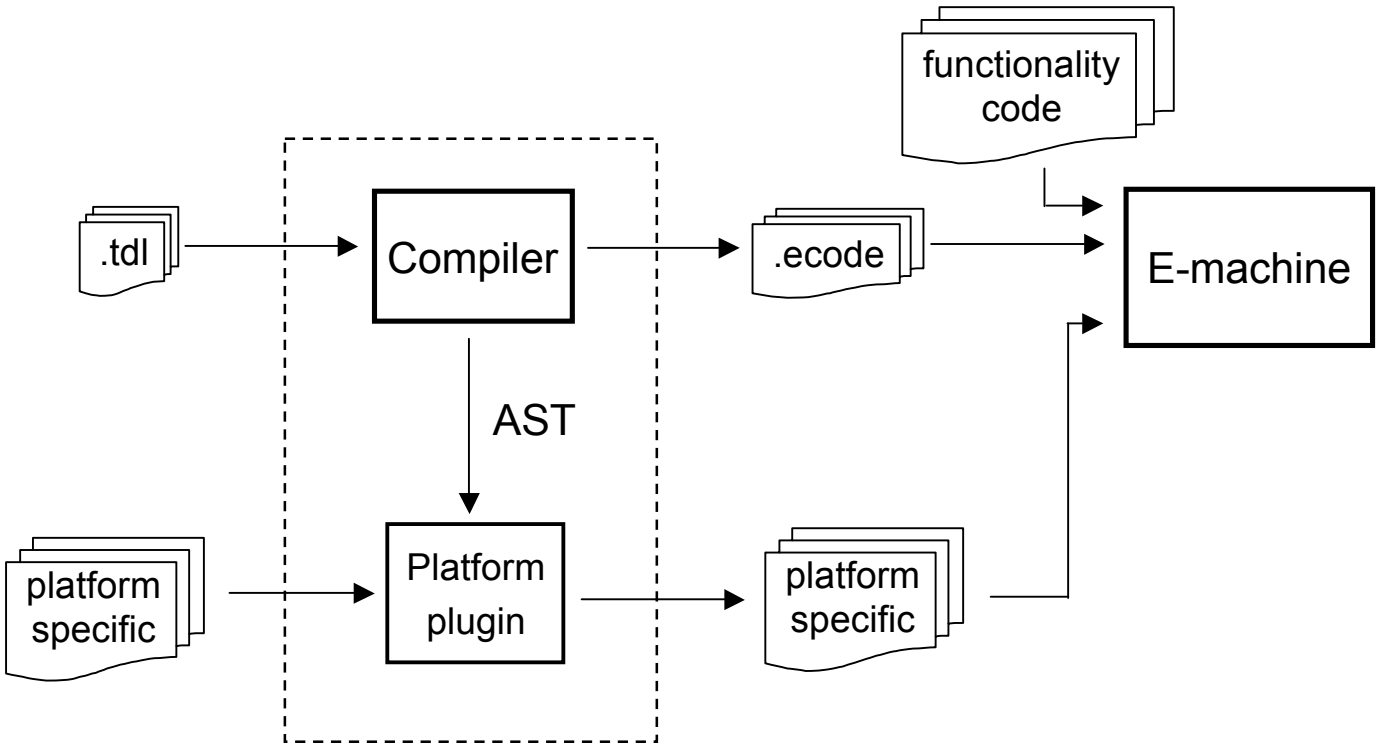


- terminate driver - *copies from TDLcomm to output*
- uses special E-code that contains only terminate driver calls at appropriate time instances => stub mode

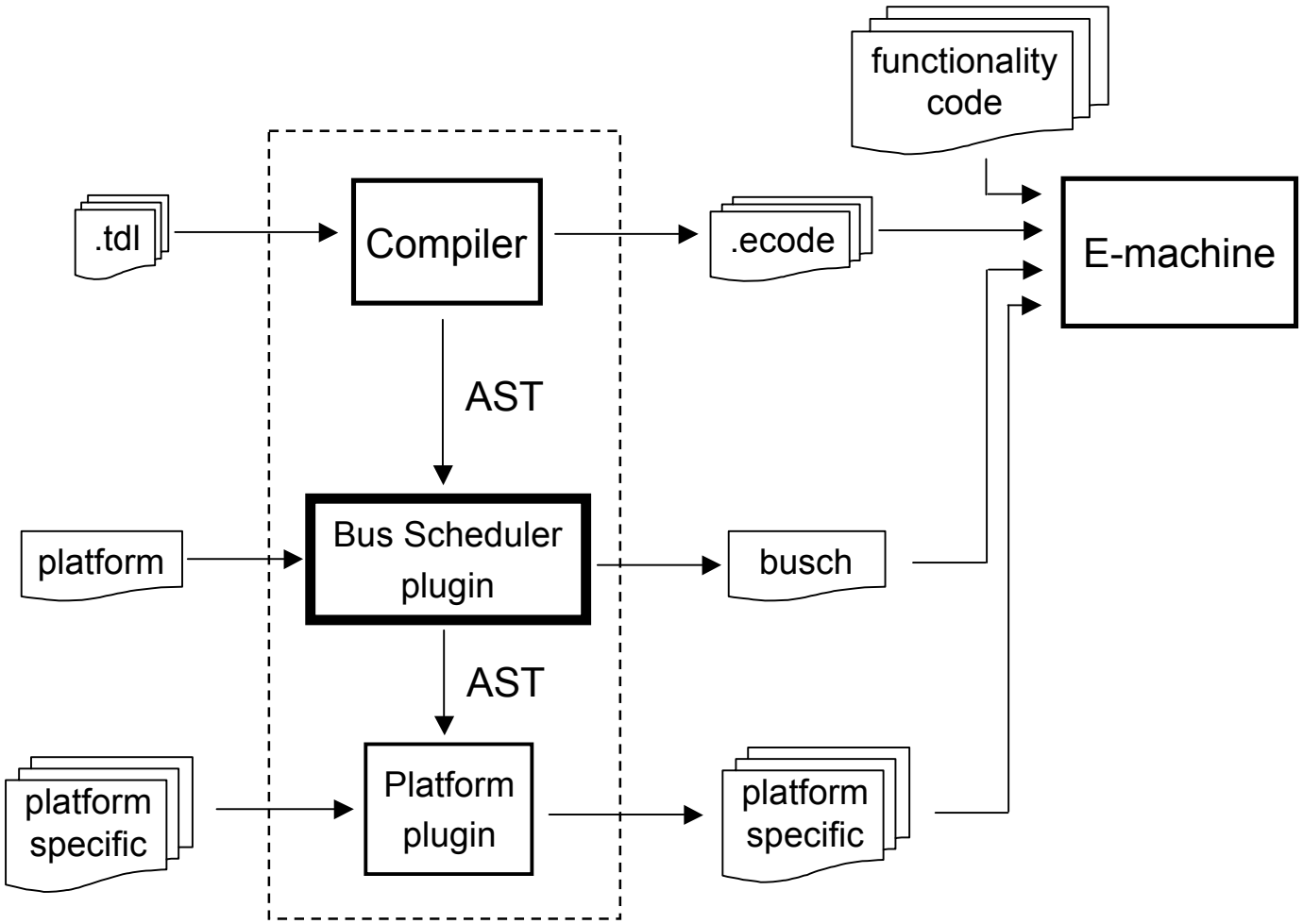
Transparent Distribution



Tool Chain


functionality code
.tdl
Compiler
.ecode
E-machine
AST
platform specific
Platform plugin
platform specific

Tool Chain



UNIVERSITÄT SALZBURG

Thank you for your attention!

UNIVERSITÄT SALZBURG