DETERMINISM IN ROBOTICS SOFTWARE

– WHEN THINGS BREAK *SOMETIMES* AND HOW TO FIX IT –

INGO LÜTKEBOHLE

BOSCH

“It worked for the video”
(Anonymous)

Determinism in robotics systems
Goals

- Introduce the issue and its consequences
- Give you a better understanding of
  - ROS internals
  - Common Practices
  - Useful packages
- Show how you can measure what's going on

Agenda

1. Introduction
    a) Definitions
    b) Motivation
    c) Background
2. Patterns
    a) Input timestamping
    b) Sampling effects and how to avoid them
    c) Multi-node/thread cases
3. Measurement
4. Summary / Discussion

INTRODUCTION

A deterministic system will always produce the same output when starting conditions and inputs are the same.

*Deterministic **execution** will always run computations in the same **order** when starting conditions and inputs are the same.*

*Deterministic **communication** is when, for a message from A → B, communication is guaranteed to be complete either always before or always after B uses it.*

Note: One way of achieving this is by waiting for the *right* data before use.

Determinism in robotics systems
Example...

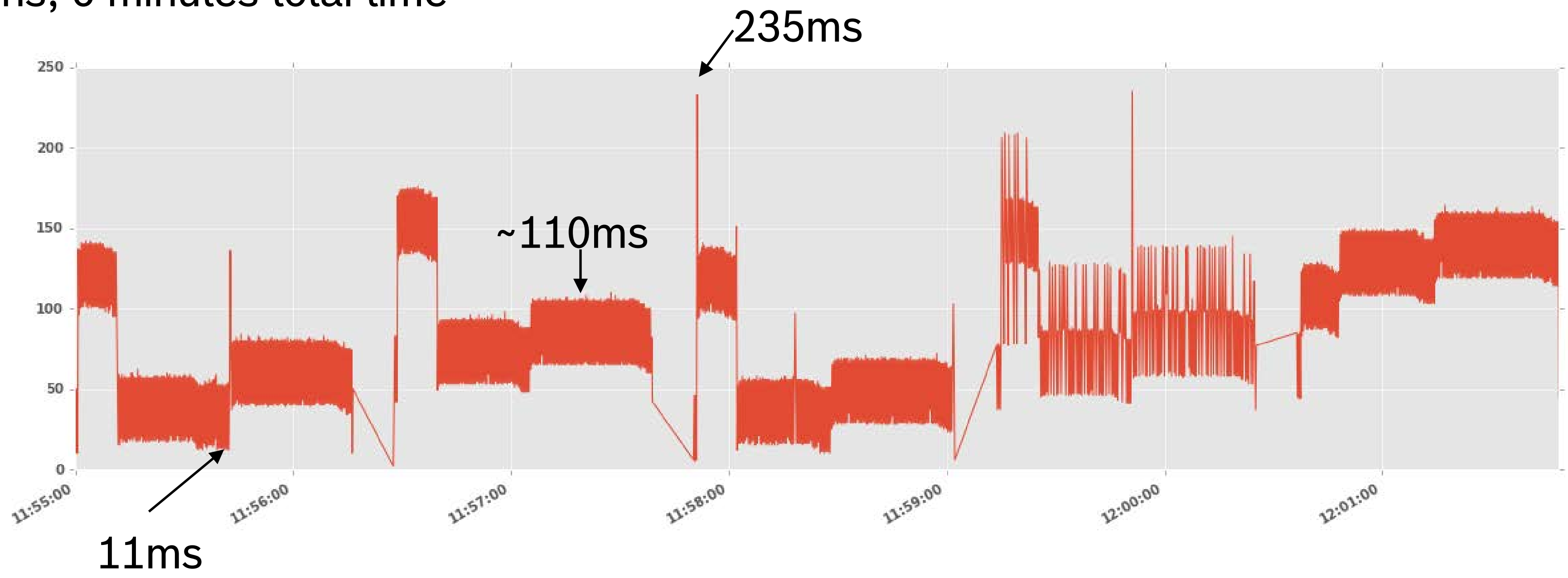
Determinism in robotics systems
Motivation

- Q: How can I be certain my robot won't hit anybody and reaches the goal?
- A: Use reactive obstacle avoidance
- Ingredients
  - Timely and reliable sensor information
    - How often and how fast do I get sensor information?
  - Environment models
    - Fusing multiple sensors, and prior information
  - Replan
    - Find an optimal, safe path, quickly
  - Reliable actuation
    - How fast can my robot brake/change course?

- Q: Is all this combined in a reliable and trustworthy manner?
  - Deterministic Execution

Determinism in robotics systems
Reaction time in standard move_base node

- Time between /scan input and (corresponding) /cmd_vel output
- Five runs, 6 minutes total time



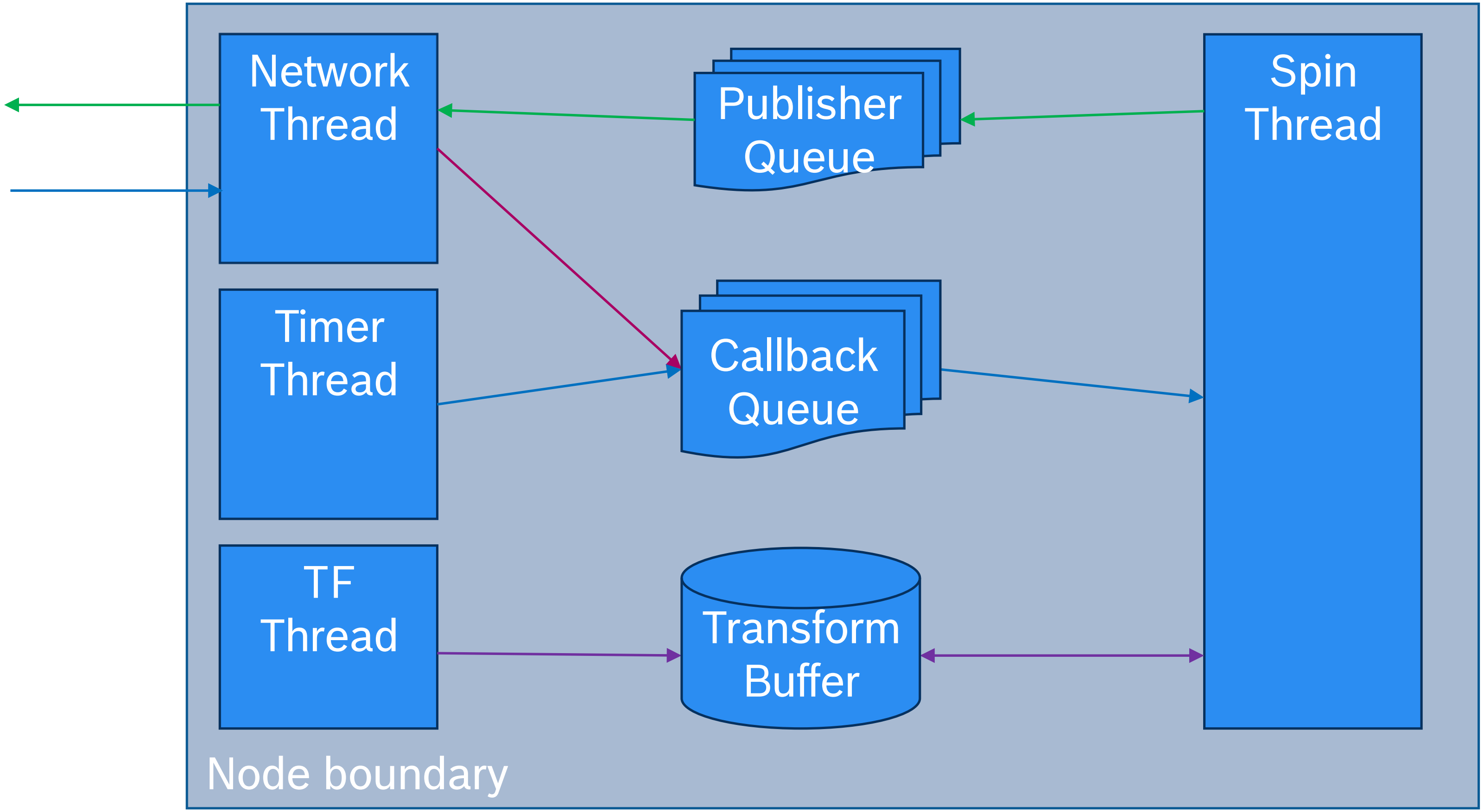
Determinism in robotics systems
Root causes for lack of determinism

- Sensor data processing
  - Timestamps are often wrong
  - Failure to ensure real-time leads to lost data
- Sampling effects
  - Asynchronous communication combined with periodic processing leads to sampling effects
  - Sampling effects cause systematic shifts in response time
  - Sampling effects also cause different results

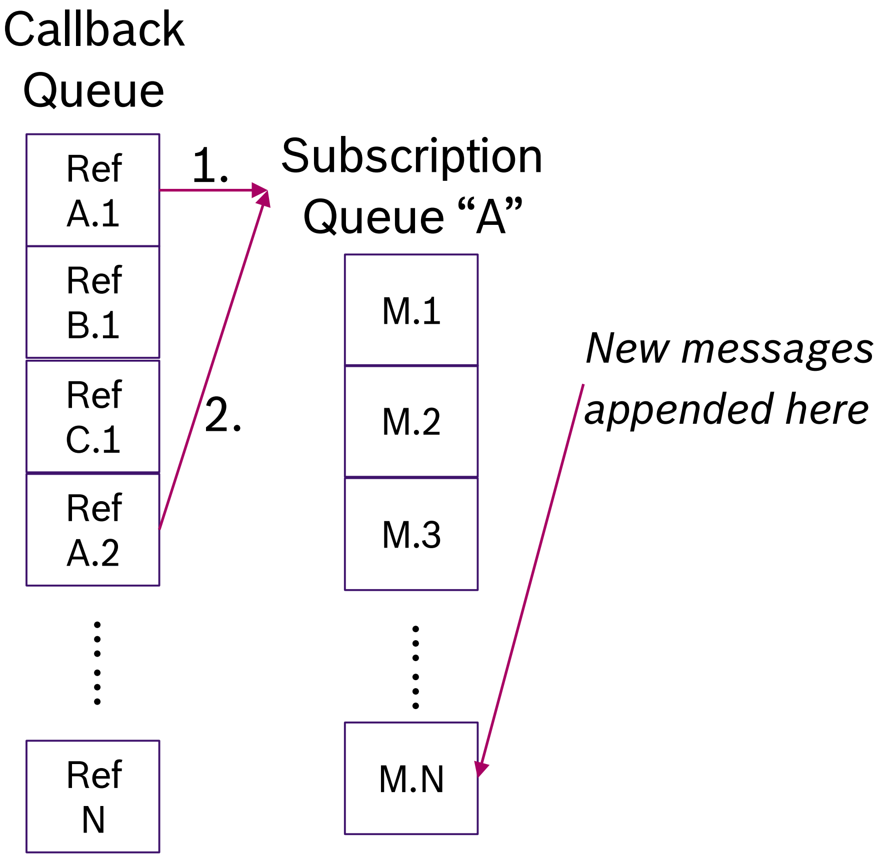
Determinism in robotics systems
Assumptions in the remainder of this talk

- Data dependency
  - Outputs depend on inputs
  - The environment changes significantly between each sensory reading
    - Depends on sensor rate and speed of motion
  - The change is difficult to predict
    - Particularly true for partially unknown and/or dynamic environments
- Typical ROS system
  - Multiple, communicating nodes or nodelets
  - Running on Linux
  - Devices connected via USB, Ethernet, CAN-Bus or serial links
- Focus on the *critical loop*
  - Not all parts of the system need the same attention

Determinism in robotics systems
Roscpp internals: Framework and user threads



Determinism in robotics systems
ROS Internals: Handling of messages to topics



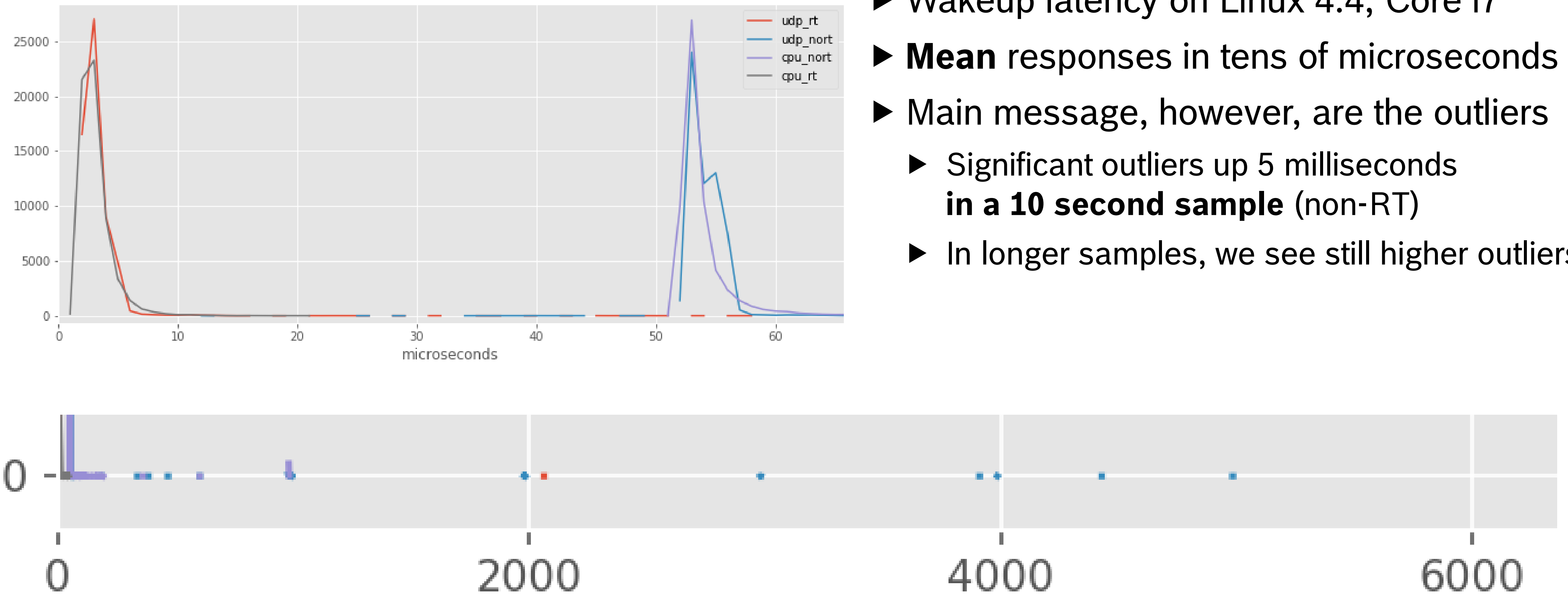
- roscpp maintains one queue per subscriber callback
  - Size depends on queue argument in subscribe
  - When called by spin thread, calls *first element* only
- Append behavior has two options
  - If subscription queue is not full
    - Append message to subscription queue
    - Append new subscription queue ptr to callback queue
  - If subscription queue is full
    - Drop oldest element, then append message

  →New messages to a subscription whose queue is full can "jump the queue"
- Two callbacks to the same topic are processed in order of subscription (and have separate queues)

Determinism in robotics systems
Scheduling approaches in Linux

- Hierarchy
  1. rt_schedPriority based runqueue
  2. fair_sched
     - Attempts to allocate an equal (“fair”) amount of time to every task
       - Weighted by nice level
     - Tasks which run rarely and/or require little CPU time are implicitly prioritized
     - Balances interactive and CPU intensive computational tasks
  - idle_sched
- rt_sched is *not really real-time* in stock kernel
  - Internal kernel tasks can block real-time processes

Determinism in robotics systems
Linux fair scheduling and standard real-time



- Wakeup latency on Linux 4.4, Core i7
- **Mean** responses in tens of microseconds
- Main message, however, are the outliers
  - Significant outliers up 5 milliseconds **in a 10 second sample** (non-RT)
  - In longer samples, we see still higher outliers

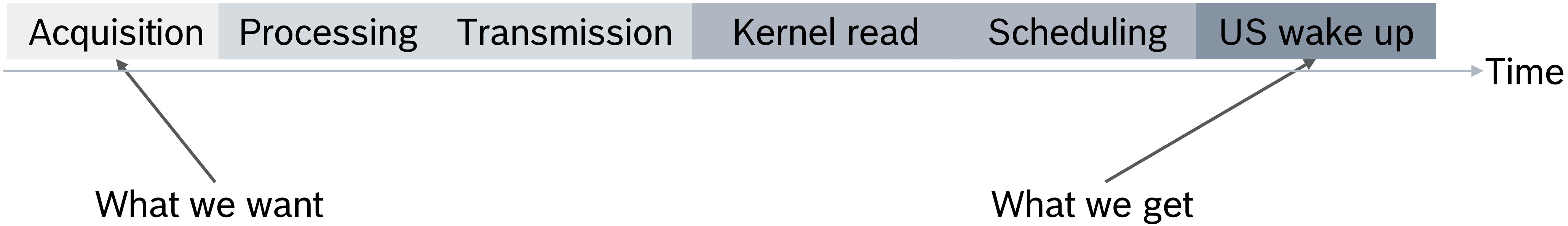
Determinism in robotics systems
Linux with RT-PREEMPT

- Vanilla Linux is not fully preemptive
  - Certain kernel sub-systems can block even RT processes
- Stress-testing of certain sub-systems
  - 10 second test
  - UDP: ~5ms delay for RT,
  - Procfs: ~60ms max delay for RT, 35 delays >5ms in testing period
- RT-PREEMPT patch fixes that
  - Available from https://wiki.linuxfoundation.org/realtime/start
  - We're currently using version for Linux 4.4
- Calling for community
  - A RT kernel package would ease adoption
  - Merging RT patch and Ubuntu kernel patch is straightforward, but annoying
  - Repeats every week...

HANDLING INPUT

Determinism in robotics systems
Typical timestamping for sensor data

- Many drivers do the following
  - Wait for sensor data
  - Read sensor data
  - Use current ROS time as header timestamp
  - (Sometimes), subtract some fixed offset to account for “transmission delay”
- But…



- And… any of these can be variable length

Determinism in robotics systems
Alternative 1: Sensor with clock

- Principle
  - Sensor contains a clock chip, or a clock line
  - Data sent includes a timestamp
- Crucial info
  - Which point in time does the timestamp refer to?
  - How is the clock synchronized / what’s the clock line connected to?
- Typical challenges
  - Clock sync
  - Handling clock misalignment and drift
  - Synchronizing multiple sensors

Determinism in robotics systems
Synchronizing clocks

- Distributed machines use the Network Time Protocol (NTP). See Mills, 1989
  - Achieves at best millisecond accuracy
- With fast, local connections use the Precision Time Protocol (IEEE 1588)
  - Typically achieves microsecond accuracy
- For devices on the same hardware board, a shared clock line may be possible
- Real-Time bus systems sometimes include a clock message

BOSCH

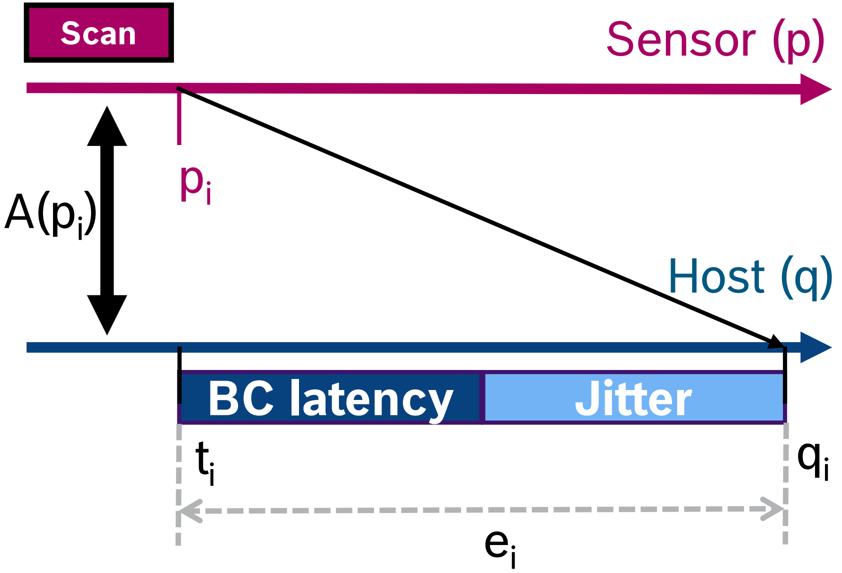
Determinism in robotics systems
Clock misalignment and drift

- Clock misalignment (aka *skew*) is a fixed offset
  - Caused by lack of exact synchronization
- Clock drift is the change in the offset over time
  - RTC's are never exactly accurate, manufacturing differences, temperature and voltage changes all play a role
  - These are *systematic* errors that add up
- Is this relevant?
  - Olson's example: Consider a robot turning 90°/s and an object in 10m distance
    Clock misalignment of 10ms → ~16cm location error
  - An uncompensated RTC can drift by 10ms in just a few minutes

BOSCH

Determinism in robotics systems
Passive clock offset and drift estimation

- Method due to Olson (2010): "*A Passive Solution to the Sensor Synchronization Problem*"
- Compute offset A as best-case (lowest) offset ever observed
  - Be sure to account for lost data when using this online
- Drift usually needs observation over a longer time period to estimate – measure at operating temp!



BOSCH

Determinism in robotics systems
Alternative 2: Explicit Trigger

- Principle
  - Host sends a signal when data acquisition should start
  - Either via software (when sensor protocol supports it) or using a hardware trigger
- Crucial info
  - For SW trigger: What’s the communication delay?
  - What’s the delay after triggering (should be in the data-sheet)
  - Is there the potential for loss of frames?
- Typical challenges
  - Software trigger requires real-time process
  - Hardware trigger usually needs custom hardware
    - Can often be added even to existing boards
  - Different sensors may have different trigger commands

BOSCH

Determinism in robotics systems
Merging sensor data alternatives

1. Receive, store, compute on condition
   - Store data upon receipt
   - Invoke computation on condition
     - Any new data, all/matching new data, new data on trigger channel, periodically
2. Request, then compute
   - Causes blocking
3. Common in ROS: Merging on timestamp
   - Easiest approach is using message_filters::Synchronizer
   - For merging with pose, use TF::MessageFilter

Determinism in robotics systems
Understanding the TF message filter

- Purpose: Make sensor data available after pose information is available
  - Requires target frame
  - Takes sensor frame and stamp from message
- Version 1 (tf::MessageFilter)
  - Works by registering a 50Hz timer for checking

  → minimum 20ms delay, if pose not immediately available
- Version 2: (tf2_ros::MessageFilter)
  - Registers a callback checked within TF – in principle, ideal solution
  - However: Large overhead at setup and run-time
    - For some reason, callbacks are registered on every incoming *sensor* message
    - If you know a reason for that, please let me know
- Conclusion: Neither is ideal, we're working on it

http://wiki.ros.org/tf/Tutorials/Using%20Stamped%20datatypes%20with%20tf%3A%3AMessageFilter

http://wiki.ros.org/tf2/Tutorials/Using%20stamped%20datatypes%20with%20tf2%3A%3AMessageFilter

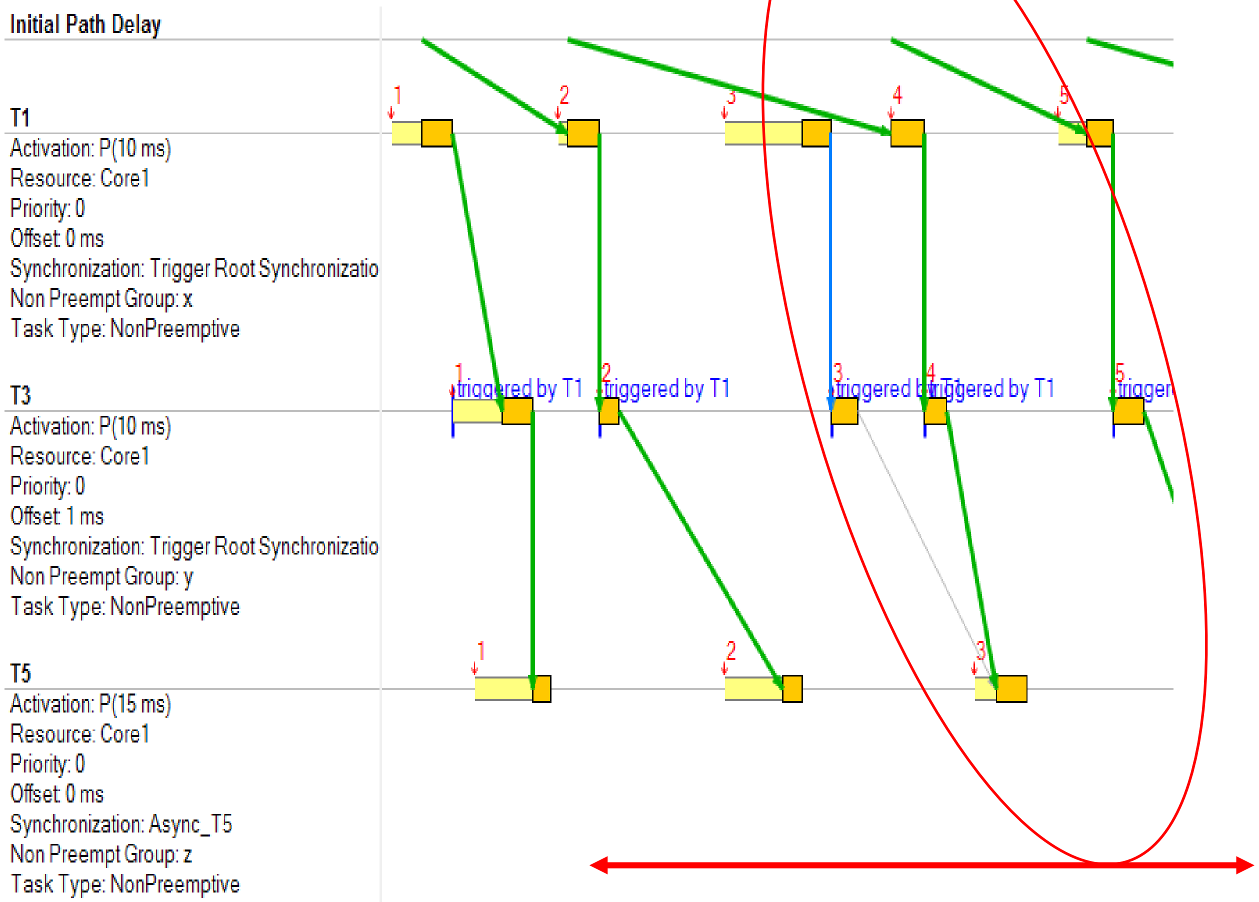
SAMPLING EFFECTS IN THE MOVE_BASE

Determinism in robotics systems
Introduction to Sampling and Sampling Effects

- “sampling” → discrete measurement of a (often continuous) signal at regular intervals
  - The resulting item is then often called “a sample”
  - Term originates with systems that actually measure in that instant
- When data acquisition and data use are *separated*, communication becomes important
  - Coupled: Measuring process hands over measurement directly to user
  - Decoupled: Measuring process puts sample into storage, user process looks at storage independently
- In decoupled systems, the process rates are crucial
  - Mismatched rates can lead to effects similar to undersampling a continuous signal (“sampling effect”)
- Rates are affected by scheduling jitter

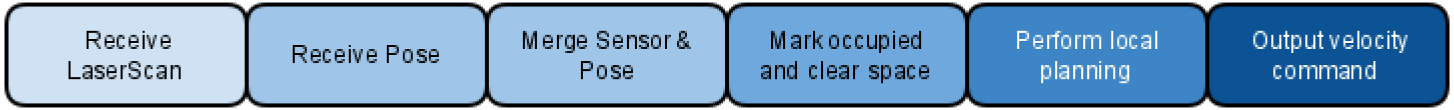
Determinism in robotics systems

Timing Analysis Example

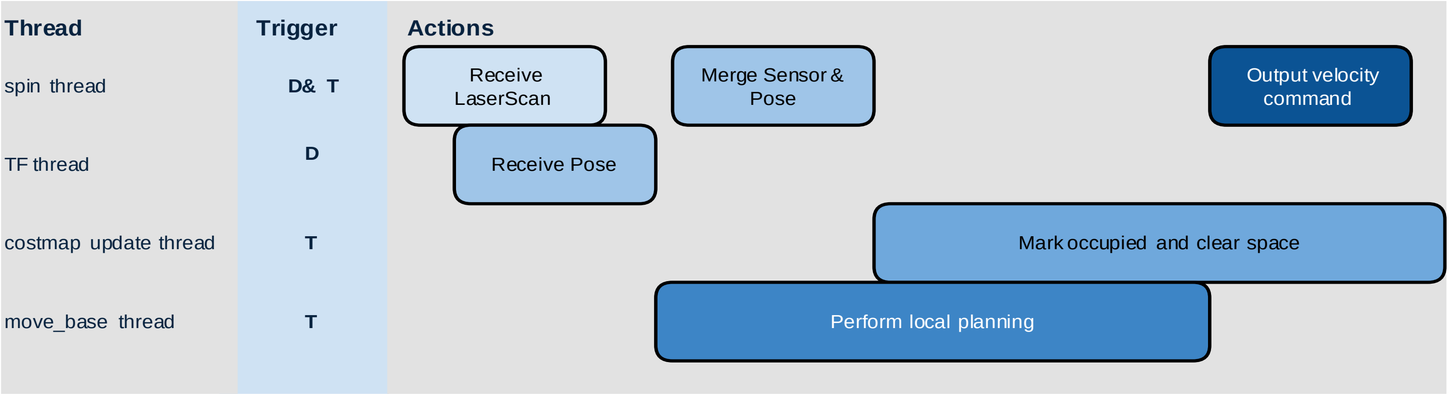


Determinism in robotics systems
move_base’s internal pipeline

- Recap: Basic processing stages

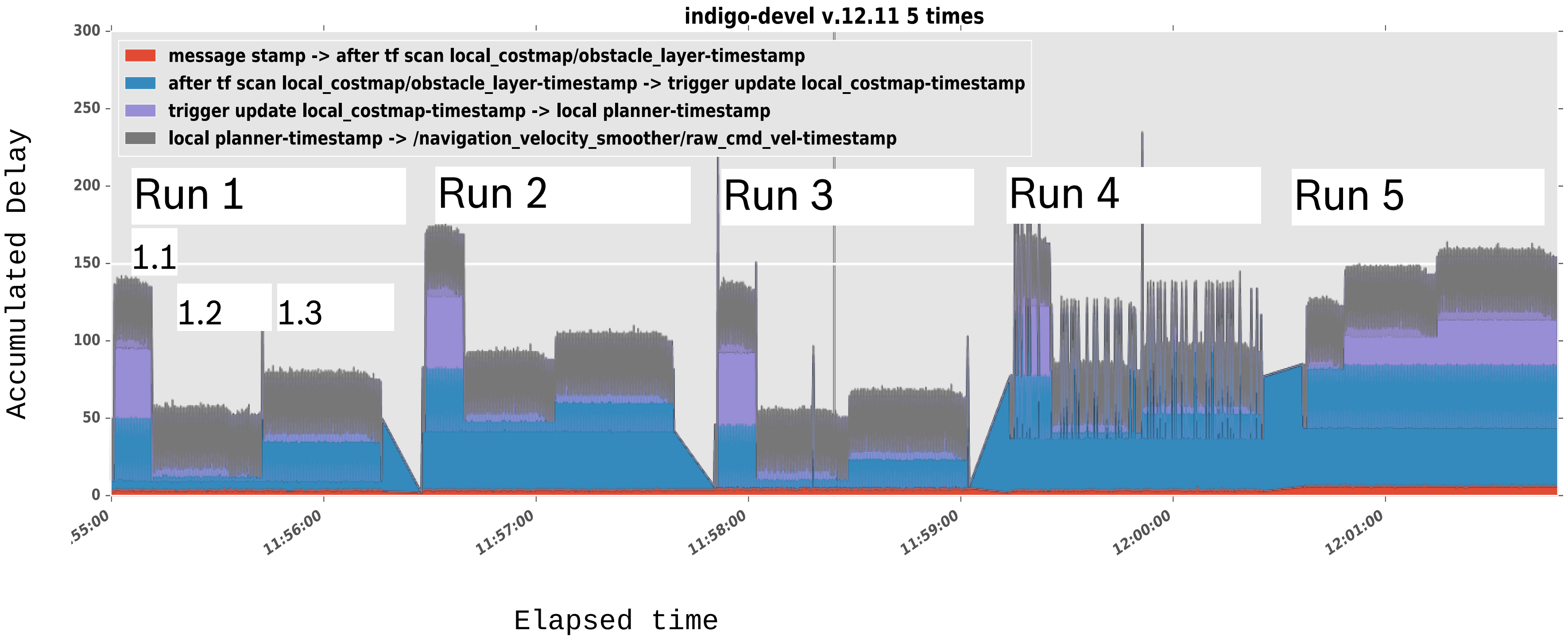


- Move_base realization: 4 threads, not synchronized



Bosch Corporate Research | 8/16/2017

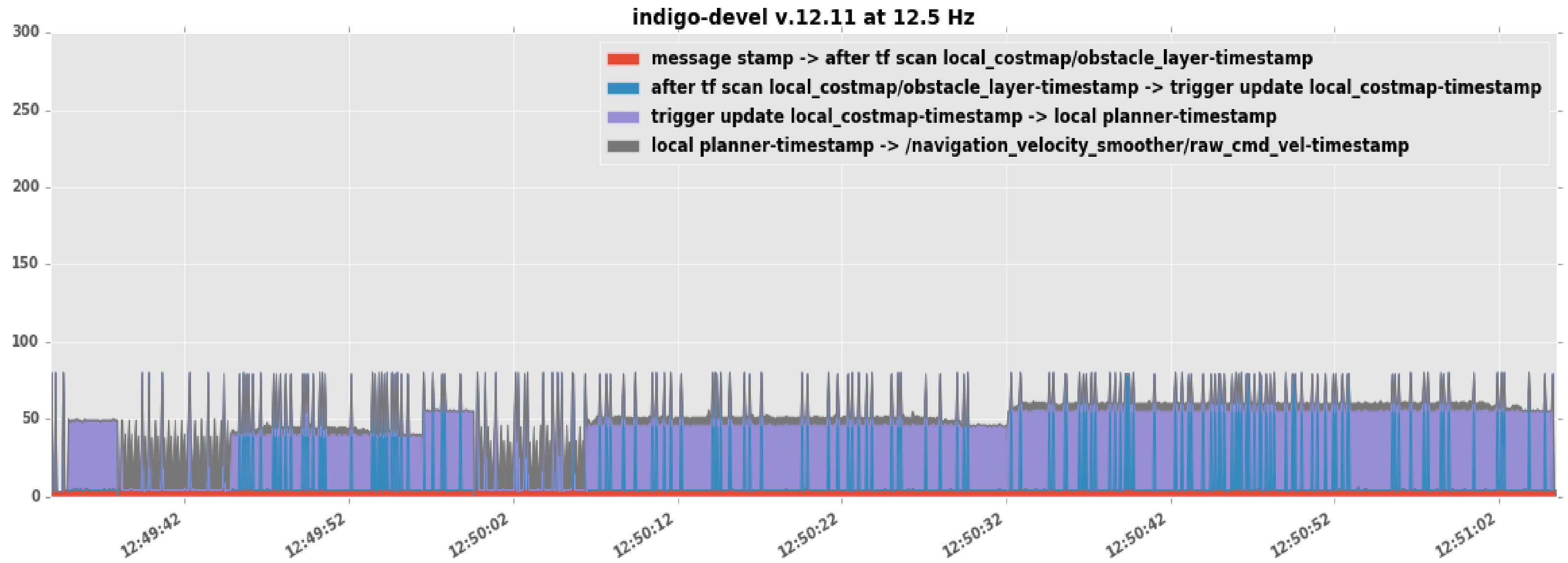
Determinism in robotics systems
Part 1: Default behavior



Not just delays, but also unpredictable change in execution order

BOSCH

Determinism in robotics systems
Part 2: Run everything at sensor rate

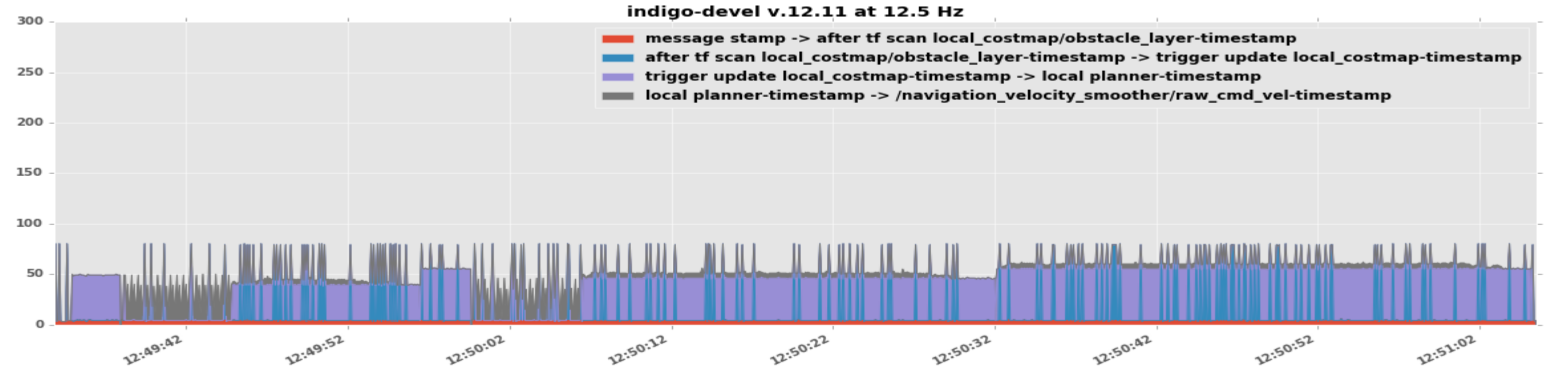


- Observation: Overall lower response time, but **very jittery**
- Cause: Slight differences in activation cause **execution re-ordering**

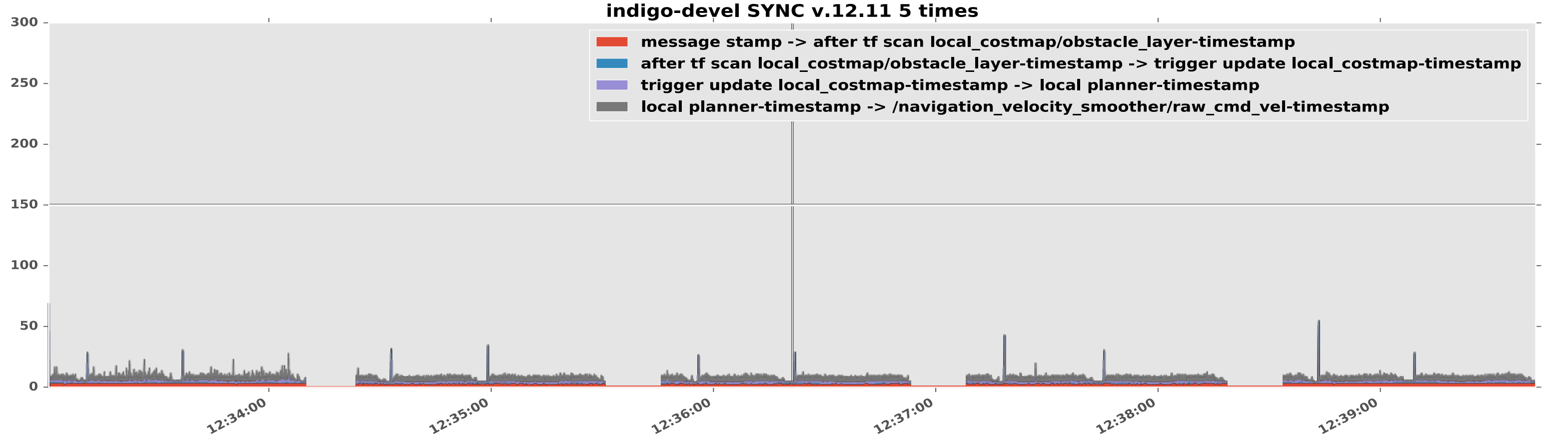
BOSCH

Determinism in robotics systems
Refactored timings, comparison

before

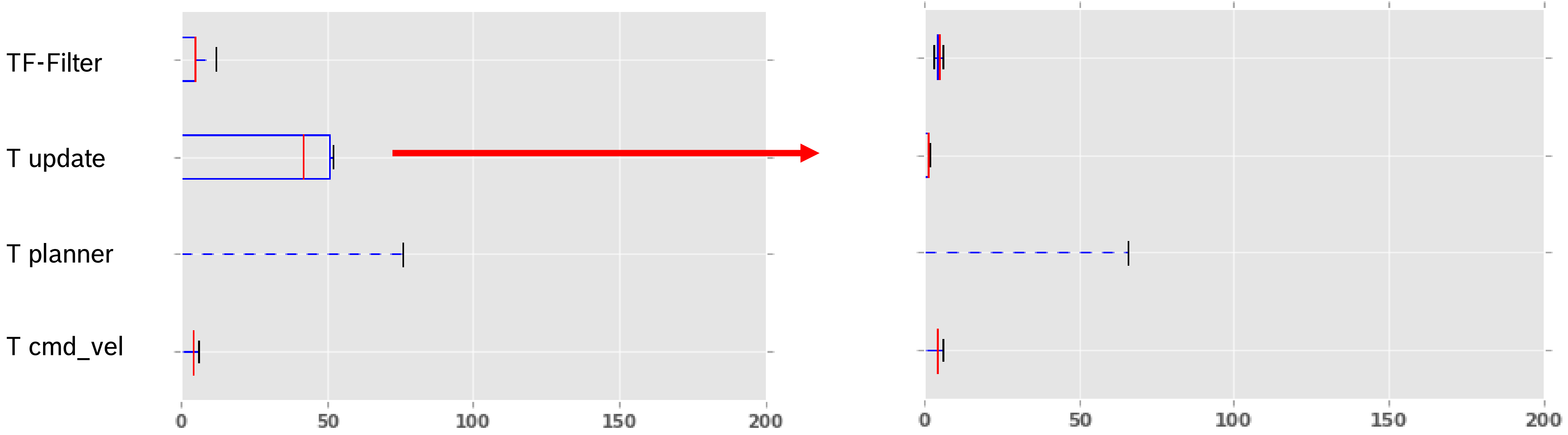


after



BOSCH

Determinism in robotics systems
Timing Boxplots



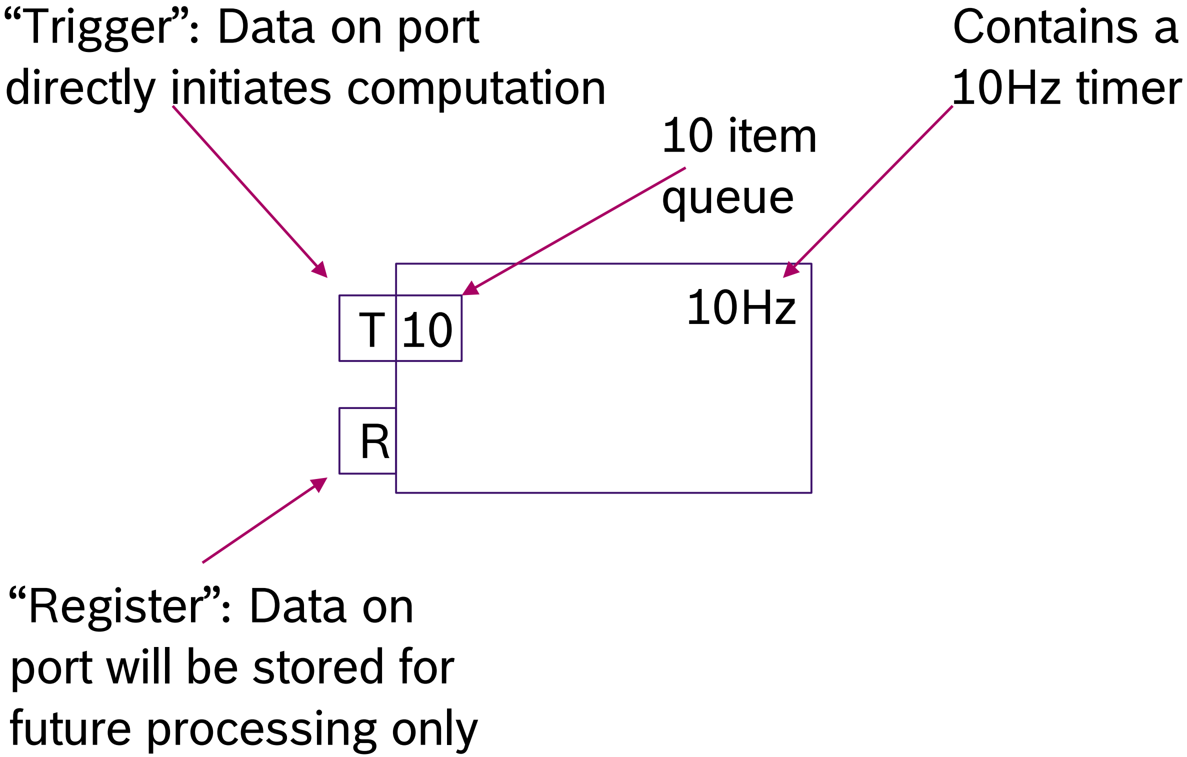
- Drastically reduced mean response time (mean 85ms → mean 9ms)
- Huge jitter reduction (60ms → 5ms std)
- **Bounded** max delay

ARCHITECTURAL CHOICES

Determinism in robotics systems
Intro to activation semantics

- Data-triggered
  - Computations are performed directly in data callback
- Time-triggered
  - Data-callbacks (if any) only store data
  - Computation is performed in time-triggered callbacks
- Time-trigger patterns
  - Timers with spin
    - Data-callbacks are called for every incoming message
    - Data is processed within Timer-driven callbacks
    - Allows combining data-driven and timer-driven processing
  - Loop with spinOnce
    - Usually loop is driven by blocking on hardware or through ros::Rate
    - spinOnce is used, which will invoke data-callbacks only up to queue-size times
    - Mainly useful for timer-only use-cases, and for interfacing with asynchronously integrated hardware

Determinism in robotics systems
A simple notation (for design)



Determinism in robotics systems
Pattern 1: Data-triggered pipelines

- Each computation is triggered by an input and directly followed by computation and output
  - Or it does nothing
- External data is often the start of the pipeline
- A data-triggered pipeline minimizes delay
- ...but it maximizes jitter
  - Any differences in computation time will be passed on
  - Any jitter from scheduling will be passed on

Determinism in robotics systems
Impact of queue-sizes

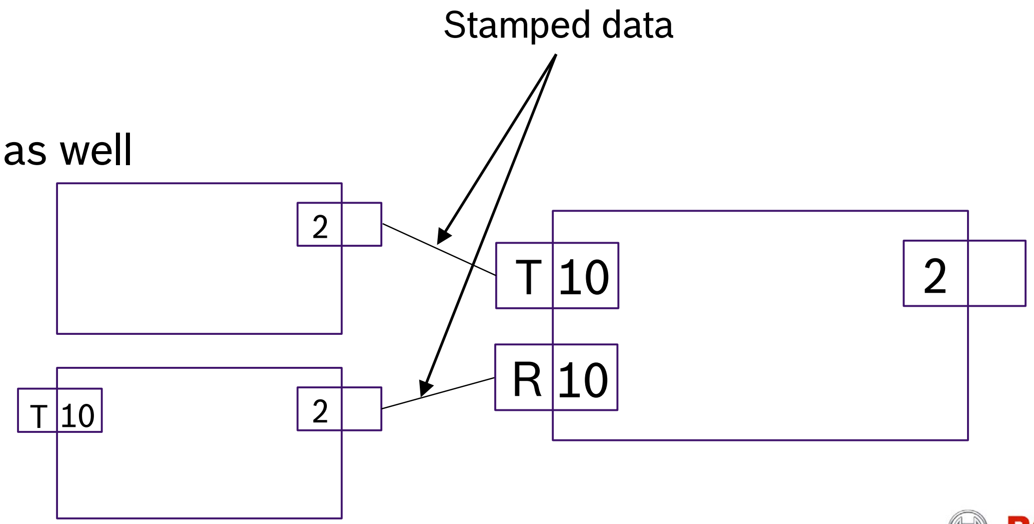
- Remember, data is coming in in parallel with the spin thread
- What happens to it is determined by the queue size
- 0
  - If you really must have everything
  - Make sure your processing is way faster than “necessary”
- 1
  - Keeps latest item only, maximizes reaction time
  - Handles temporary overloads by dropping data
- Larger, but not unlimited
  - Most cases where you prefer not to lose data but could tolerate it, and don’t prioritize reaction time
  - **Only safe when you can, most of the time, process faster than data comes in**

Determinism in robotics systems
Approach 2: Synchronized starts

- Use Cases
  - The ros::Timer has its 0 point when it is created (or started)
  - Hardware devices often start when drivers initializes communication
  - Independent starts will have offsets to each other
    - And of course to other nodes
- This may be fine, but if it is not, use synchronized starts
- Synchronized starts
    - Make start dependent upon an external signal
      - But signals could have a delay, too
    - Use a signal that contains a timestamp in the future, then sleep until
      - std_msgs/Time makes a useful synchronization signal
- Mainly useful for synchronized sensor start-up
- We're considering to suggest this as an addition to the Lifecycle

Determinism in robotics systems
Approach 3: Explicit synchronization

- Across processes: Data-based
  - Sequence number or timestamp in message
  - Utilize knowledge of rates to match up
  - Lifecycle management can help simplify this (by starting sequences in the same instant)
- Across threads: Synchronization objects
  - Simplest approach is a condition variable
    - that’s what we used in the move_base example
    - Wait before data use, notify on (new) data availability
  - All the usual concurrency management tools apply as well



Determinism in robotics systems
Measuring timing

- Approach
  - Use roscpp instrumentation to get general information about timings in nodes
    - Use this identify target nodes for further analysis
  - Insert dynamic or static tracepoints into the target for more detailed analysis
- We have instrumented roscpp using the Linux Tracing Toolkit ng (LTTng)
  - Tracepoints: Message entry/exit, callback entry/exit, queue delay, message_filters
- Tracetools
  - Generic wrapper, could integrate other tracers
  - Model for trace data
  - Generic experiment running and analysis using Python Pandas
- I'm currently pushing this out to Github, talk to me if you'd like early access

Determinism in robotics systems

Example of using the tracetools

```
# module for path manipulation
import os
# import experime
import tracing.experiment as er
# import Pandas support
import tracing.trace_pandas as tp
# import ROS-specific support
from tracing.rosmapping import map_roscpp

# define an experiment using ROS
exp1 = er.ROSTraceExperiment(
  workspace=os.path.expanduser("~/turtlebot_ws"),
  package="turtlebot_stdr",
  launch_file="perf_sim.launch"],
  userspace_events=er.ROSCPP_TRACE_EVENTS)

# set up tracing, run (waits for completion)http://pandas.pydata.org/
exp1.create()
# run
exp1.run()
# convert LTTng data to Python POD, with raw entries
raw_data    = exp1.collect_data()
# convert data to our generic model
trace_data = map_roscpp(raw_data)
# convert the model to Pandas dataframes for analysis
data       = tp.ti2pd(trace_data)

# sample query: plot callback queue delay for the move_base
data.delays.join(data.tasks, rsuffix='t').
  query("node_name=='move_base'").
  groupby("task_id")["delay"].
  plot.line(logy=True, figsize=(20,6))
```

Hint: double-click to select code

Bosch Corporate Research | 8/16/2017

BOSCH

Determinism in robotics systems
Summary

- Deterministic behavior is not automatic
  - Non-deterministic systems can spontaneously fail and/or perform worse
  - Jitter and misalignment are commonplace, not rare
- At the inputs, make sure you have proper timestamps
- Use common merging patterns for data
- For the overall architecture, several simple patterns help a lot
- Design, measure, repeat

Determinism in robotics software
Next steps

- We could use a Kernel package with RT-PREEMPT
  - Bosch does this internally
  - We could make this a community resource
- Lets build a community around system engineering tools for ROS
  - Multiple people have mentioned tracing
  - Analyzing trace data is not easy, but fortunately very repetitive
    - Sharing analysis scripts would make things easier
  - Static analysis of code is very useful
    - ROS-specific analyzes are not so difficult and would make this much more useful
  - We need more models, both for analysis and for construction
- If you’re interested to help out, come to me after the talk, or come to the Bosch booth

THANK YOU

ingo.luetkebohle@de.bosch.com