TIMICO

Precision Time Protocol

Christian Fischer Pedersen
Assistant Professor
cfp@eng.au.dk

Section of Electrical and Computer Engineering
Department of Engineering
Aarhus University

October 29, 2014

Outline

Outline

Real-time systems

It is all about deadlines

Real-time systems

- Logical **correctness** in **due time**
- Must respond to an event before **deadline**
- Not about **"fast"** but **timely** systems
- Deadline is either relative or absolute

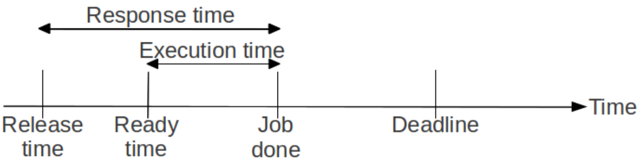
Deadlines

- Relative to an event:
  $T_{\text{Deadline}} = T_{\text{Event}} + T_{\text{Allowed response time}}$
- Absolute according to synchronized clock:
  $T_{\text{Deadline}} = T_{\text{Synchronized clock}}$

Real-time systems

Timings

- Release time: Job is released
- Ready time: Job can start executing
- Execution time: Time it takes to execute job
- Response time: Time elapsed from job released to job done
- Deadline: Job **must** be done **before** deadline



Real-time systems

Importance of meeting deadlines

Hard real-time

- Missing a deadline is highly critical
- Severe degradation of quality of service

Firm real-time

- Infrequent deadline misses are tolerable
- Some degradation of quality of service

Soft real-time

- Some deadline misses are tolerable
- Minor degradation of quality of service

Real-time systems

Examples

Hard real-time example

- Flight control or train control systems
- Drive by wire

Firm real-time example

- Industrial robots, e.g. for spray painting cars:
  Hopeless painting and damage of cars

Soft real-time example

- Online reservation or streaming multimedia applications
- Slight occasional delays cause minor inconveniences

Determinism

An overview

Determinism

- In every state of a computation the next state is **unique**

Nondeterminism

- In every state of a computation **several** next states may exist

Determinism of real-time systems

- Real-time systems **must** respond before a **deadline**
- This can **not** be guaranteed if task is nondeterministic

Example: Standard Ethernet

- Nondeterministic, i.e. **not** suited for real-time systems
- Challenge: Make Ethernet suitable for real-time systems

Outline

Automata in model checking

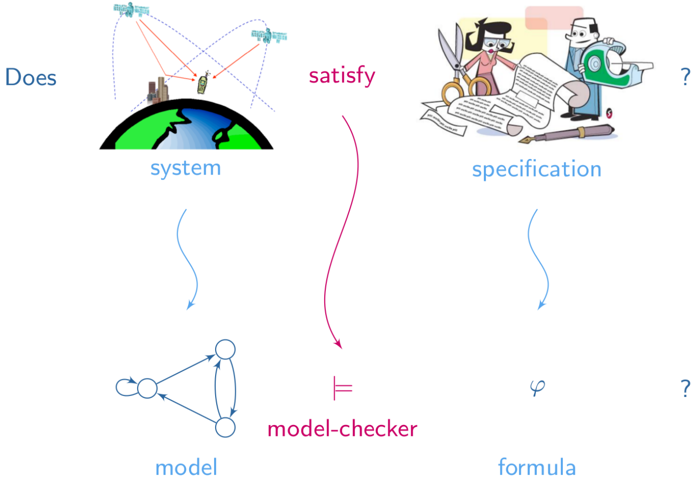


Figure : Automatically prove that the system model satisfies the mathematical representation of the specification (Nathalie Bertrand).

Deterministic finite automaton

The definition

A DFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set of states
- Σ is a finite alphabet
- $\delta : Q \times \Sigma \to Q$ is the transition function
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of accept states

Deterministic finite automaton

An example

- $Q = \{q_0, q_1\}$
- $\Sigma = \{0, 1\}$
- $\delta : Q \times \Sigma \to Q$
- $F = \{q_0\}$

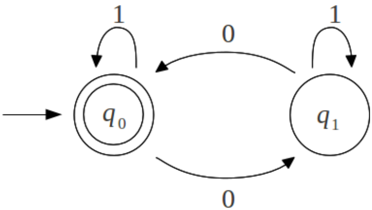


Figure : DFA that accepts strings given by the regular expression $1^*(0(1^*)0(1^*))^*$. The Kleene star, $(^*)$, means *zero or more*.

Nondeterministic finite automaton

The definition

Introduce

- $P(Q)$ is a power set, i.e. a set of all subsets of Q
- $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ where ε is the empty string

A NFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set of states
- Σ is a finite alphabet
- $\delta : Q \times \Sigma_\varepsilon \to P(Q)$ is the transition function
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of accept states

Nondeterministic finite automaton

An example

- $Q = \{q_0, q_1, q_2, q_3\}$
- $\Sigma = \{0, 1\}$
- $\delta : Q \times \Sigma_\varepsilon \to P(Q)$
- $F = \{q_3\}$

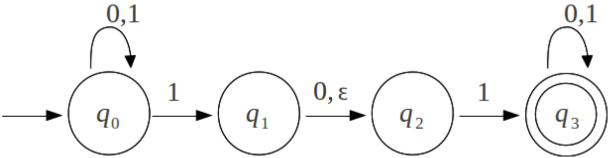


Figure : NFA that accepts strings with substrings 101 or 11.

Timed Finite Automata

The definition

A TFA is a tuple $(L, L_0, L_F, \Sigma, C, E, I)$ where

- L is a finite set of locations (states)
- $L_0 \subseteq L$ is the set of start locations
- $L_F \subseteq L$ is the set of final (accept) locations
- Σ is a finite alphabet
- C is a finite set of clocks, c, where $c \in \mathbb{R}_+$
- $E \in L \times \Phi(C) \times (\Sigma \cup \{\epsilon\}) \times 2^C \times L$ is the set of transitions
- $I: \ L \mapsto \Phi(C)$ assigns local clock invariants to locations
  - $\Phi(C)$ is the set of clock constraints, δ, defined inductively by: $\delta := x \sim q \mid x - y \sim q \mid \neg\delta \mid \delta_1 \wedge \delta_2 \mid true$ where
    - $q \in \mathbb{Q}$ and $\sim \in \{=, <, >, \leq, \geq\}$

Timed Finite Automata

An example

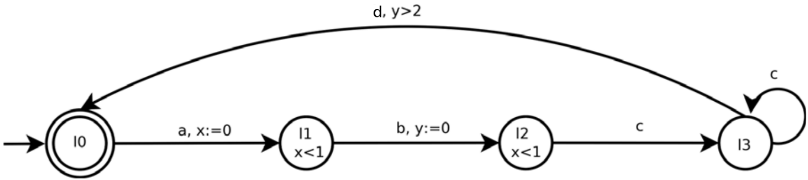


Figure : A TFA with two clocks x and y (Waez et al., 2013 [2]).

- When a transition occurs the associated clocks are reset
- Time elapses in locations
- Transitions are instantaneous

Automata in model checking

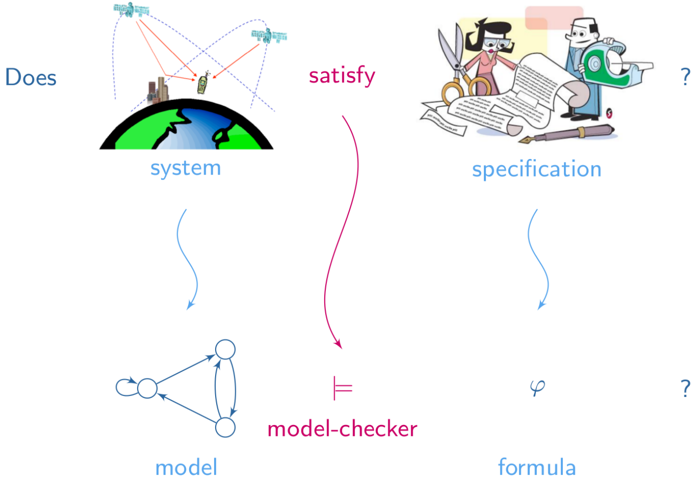


Figure : Automatically prove that the system model satisfies the mathematical representation of the specification (Nathalie Bertrand).

Outline

Determinism and timeliness of real-time systems

Automata – Some underlying formalisms

Precision Time Protocol

References

Clock synchronization

To meet deadlines

Real-time systems

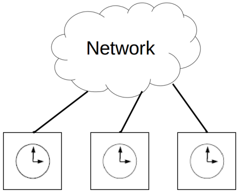
- Need to meet **dead-line guarantees**

One way to support this in a distributed system

- Introduce a **common** notion of distribution-wide time

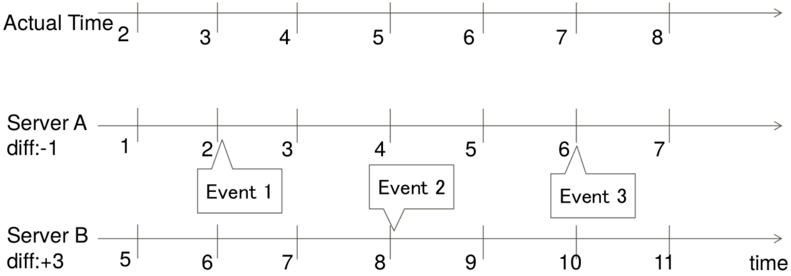
This requires

- Precise **synchronization** of clocks in nodes



Clock synchronization

Event ordering: Wrong order



<u>Event Ordering based on Actual Time</u>

Time	Event
3	Event 1
5	Event 2
7	Event 3

<u>Event Ordering based on Timestamp</u>

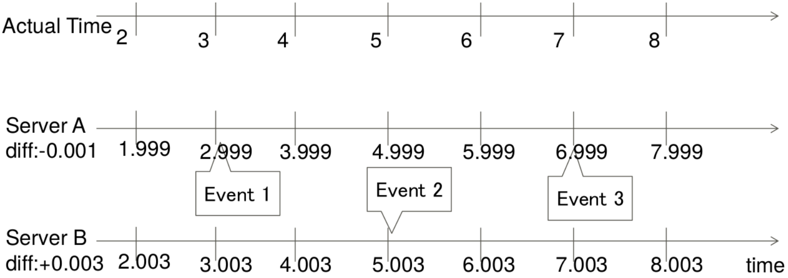
Time	Event
2	Event 1
6	Event 3
8	Event 2

reverse!

Figure : Reversed events due to offset clocks (Fujitsu).

Clock synchronization

Event ordering: Right order



Event Ordering based on Actual Time

Time	Event
3	Event 1
5	Event 2
7	Event 3

Event Ordering based on Timestamp

Time	Event
2.999	Event 1
5.003	Event 2
6.999	Event 3

correct!

Figure : Properly ordered events due to non-offset clocks (Fujitsu).

Clock synchronization

Time stamps

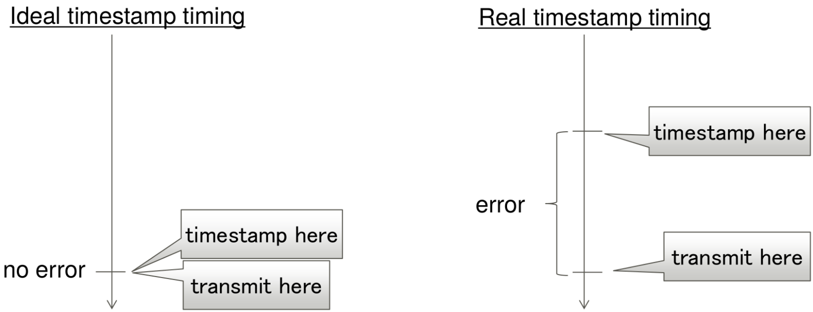


Figure : Minimize the time (error) between time of stamping and transmitting or receiving (Fujitsu).

Clock synchronization

Challenges

Each node's clock tend to **drift** due to

- Instabilities in source oscillators
- Environmental conditions such as:
  Temperature, air circulation, mechanical stress, vibration, ...

Clock synchronization

- **Initial** and **on-going** corrections are required

Clock synchronization

A solution: The Precision Time Protocol

- To synchronize clocks in nodes on a computer network
- Achieves **sub-μs** clock synchronization accuracy
- Suitable for distributed real-time systems
- Works on packet switched networks, e.g. Ethernet and LTE

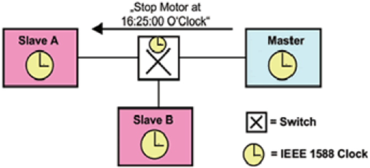


Figure : Clock synchronization by the Precision Time Protocol

Clock synchronization

Related protocols

NTP (Network Time Protocol)

- Typically synchronize clocks within range of ± 50 ms
- Hours to reach precision

SNTP (Simple NTP)

- Typically synchronize clocks within range of ± 50 ms
- Approximately 1 minute to reach precision

IEEE 1588 (PTP v1/v2)

- Typically synchronize clocks within range of ± 50 ns
- Less than 1 minute to reach precision

Precision Time Protocol

History

- Originally defined in the IEEE 1588-2002 standard (PTPv1)
- Revised in 2008 to IEEE 1588-2008 (PTPv2)
- PTPv2 improves accuracy, precision, and robustness
- PTPv2 is **not** backwards compatible with PTPv1
- Current information may be found at nist.gov and ieee.org
- Conferences on IEEE 1588 PTP held in 2003, 2004, ...

Precision Time Protocol

Application area examples

Industrial automation

- Plant floor control and management of sensors and actuators

Telecommunication

- Control voice, video, and text services in beyond 3G arch.

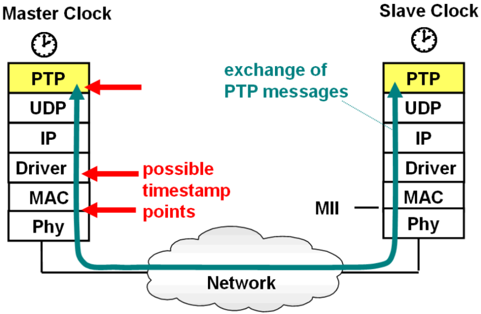
Audio and video bridging (IEEE AVB 802.1)

- Time-sensitive interoperability between multimedia devices

Precision Time Protocol

Overall mechanics

- For packet switched networks, e.g. Ethernet and LTE
- Synchronization and data transfer on the **same** network
- Minimal admin, network, software and hardware requirements
- Most precise clock synchronizes the other clocks



(Weibel, 2009 [3])

Precision Time Protocol

Time stamping: Potential points

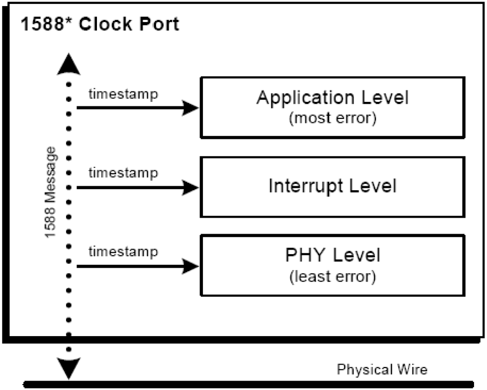


Figure : Hardware vs. software time stamping (Intel).

Precision Time Protocol

Time stamping: Synchronization element

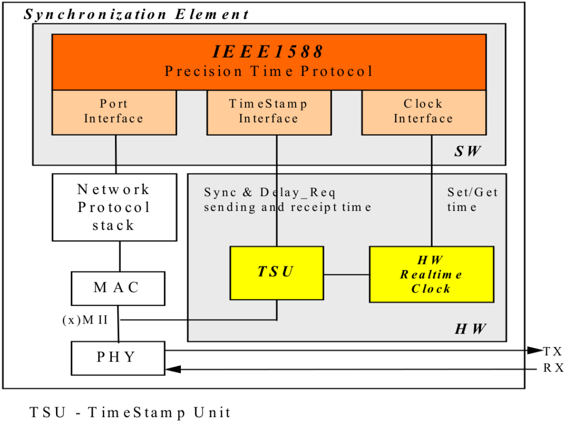


Figure : Synchronization element with Time Stamp Unit (TSU) and Real-time Clock in hardware. The rest of the protocol can be in software as timing requirements are lower (Mohl, 2003 [1]).

Precision Time Protocol

Time stamping: Intel IXP46X Network Processors

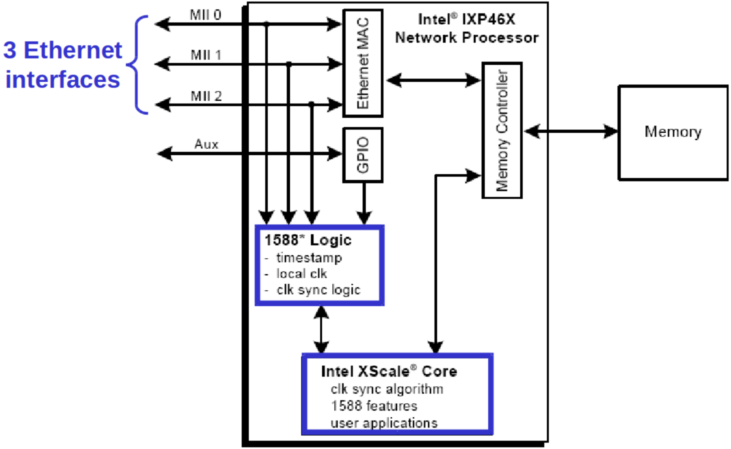


Figure : Hardware support for IEEE PTP (Intel).

Precision Time Protocol

Phase 1: Offset correction

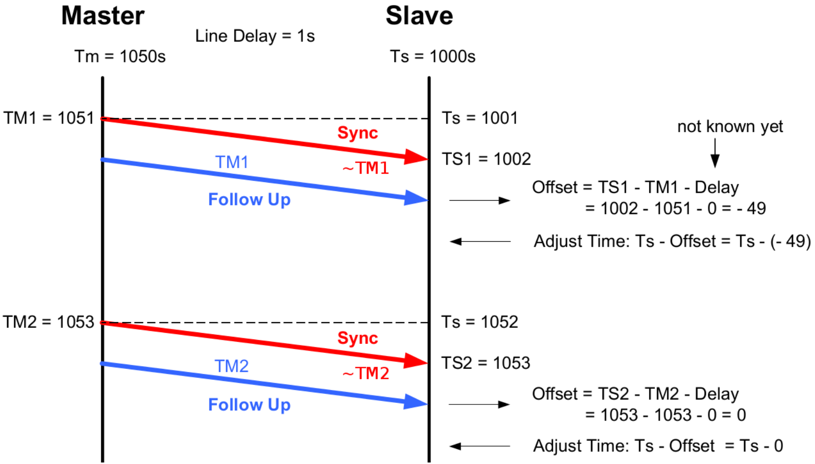


Figure : In offset correction, the line delay is assumed to be 0s. The actual delay (1s) is measured and added in the next phase (Mohl, 2003 [1]).

Precision Time Protocol

Phase 2: Delay correction

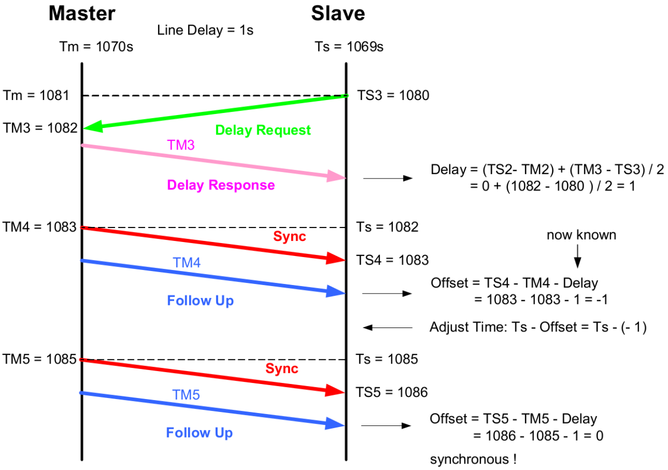


Figure : Known: M is ≥ 0s ahead. Assumed: Symmetric delay (Mohl, 2003 [1]).

Precision Time Protocol in networks

A generic distributed system with two subnets

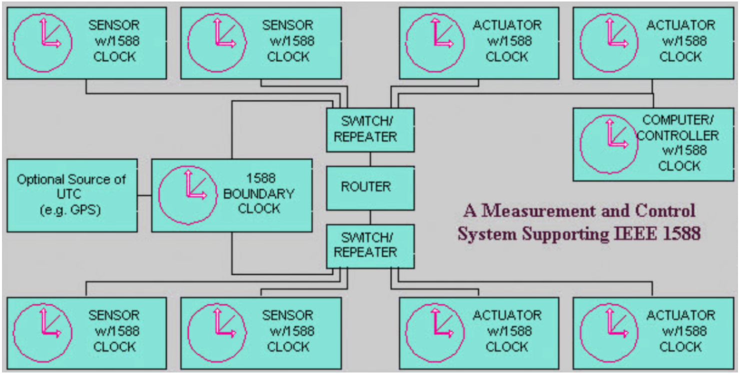


Figure : Measurement and control system with IEEE PTPv2 (www.nist.gov).

Precision Time Protocol in networks

Boundary clock

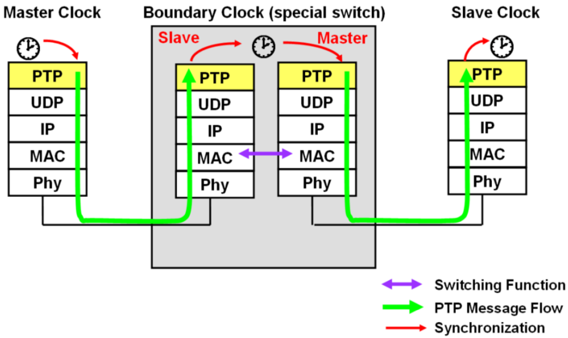


Figure : Boundary clocks synchronize subnet nodes' clocks across switches and routers (Weibel, 2009 [3]).

Precision Time Protocol in networks

Boundary clock

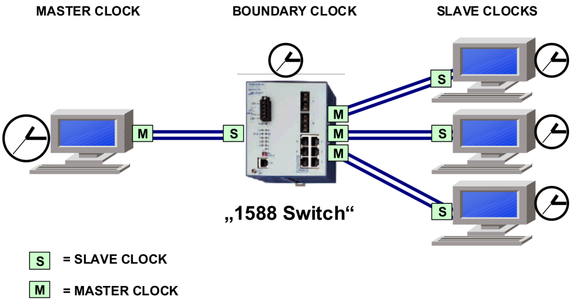


Figure : Switch with boundary clock (Mohl, 2003 [1]).

Precision Time Protocol in networks

Synchronization hierarchy

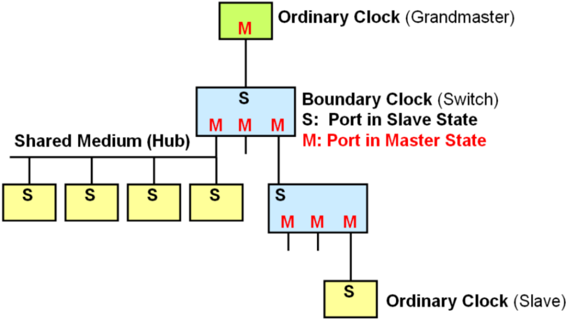


Figure : The most precise clock, i.e. the Grandmaster, is selected by the Best Master Clock Algorithm. Grandmaster is root of the hierarchy (Weibel, 2009 [3]).

Master/Slave synchronization test (PTPv1)

- Network modules with pulse output (PPS: Pulse Per Second)
- Oscilloscope measures deviation between master and slave
- Offset mean, max, and std.dev.: (-4.25, ±100, 23.95) ns

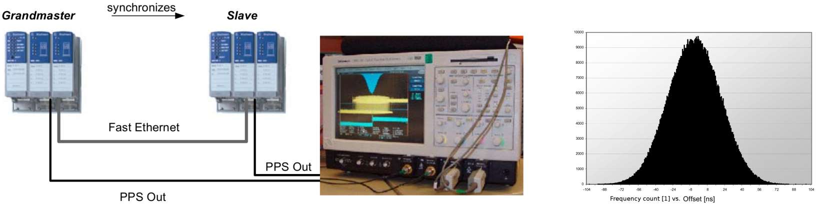


Figure : Master/Slave clock setup with Ethernet packet generator

(Mohl, 2003 [1])

Outline

References I

[1] Mohl, D. (2003). IEEE 1588 - Precise time synchronization as the basis for real time applications in automation. Technical report, White Paper, Industrial Networking Solutions, pp. 1–8.

[2] Waez, M., J. Dingel, and K. Rudie (2013). A survey of ttime automata for the development of real-time systems. *Computer Science Review* (9), 1–26.

[3] Weibel, H. (2009). Technology update on IEEE 1588: The second edition of the high precision clock synchronization protocol. Technical report, Zurich University of Applied Sciences, Winterthur, Switzerland.