Synchronization

Distributed and Pervasive Systems

Christian Fischer Pedersen
cfp@eng.au.dk

Section of Electrical and Computer Engineering
Department of Engineering
Aarhus University

Revised on February 25, 2018

Outline

Clock synchronization

Logical clock: Lamport time-stamps

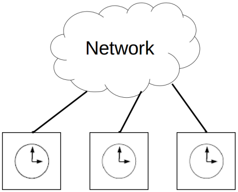
Real clock: Precision Time Protocol

References

Outline

Clock synchronization objectives

- **Common** notion of local or global, real or logical, **time**
- Coordination of **logical** or **real** clocks in a distributed system
- **Logical and real** clocks: Able to obtain logical event ordering
- **Real clocks:** Ability to meet wall-clock deadlines
- **Logical clocks:** Ability to meet deadlines relative to events



Synchronization

How can you implement a synchronous algorithm in an asynchronous environment?

Clock synchronization challenges

Logical clocks

- Achieve chronological local and global event ordering in a distributed system without synchronous real clocks
- Initial and on-going corrections are required

Real clocks

- Precise synchronization of clocks in nodes
- Each node's clock tend to drift due to instabilities in source oscillators and environmental conditions such as temperature and mechanical wear and tear
- Initial and on-going corrections are required

Clock synchronization solutions for logical clocks

Examples

- Lamport time-stamps
- Vector/matrix clocks
- Version vectors
- ...

Clock synchronization solutions for real clocks

NTP (Network Time Protocol) and SNTP (Simple NTP)

- Synchronize clocks on the millisecond level

RBIS (Reference Broadcast Infrastructure Synchronization)

- Synchronize clocks on the microsecond level

IEEE 1588 Precision Time Protocol (PTP) v1/v2

- Synchronize clocks on the nanosecond level

And many more ...

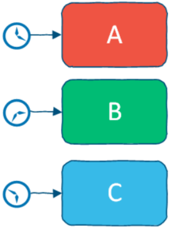


Figure: Local clocks: skew+drift

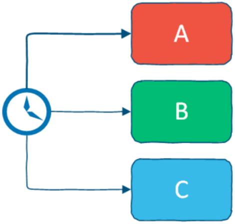
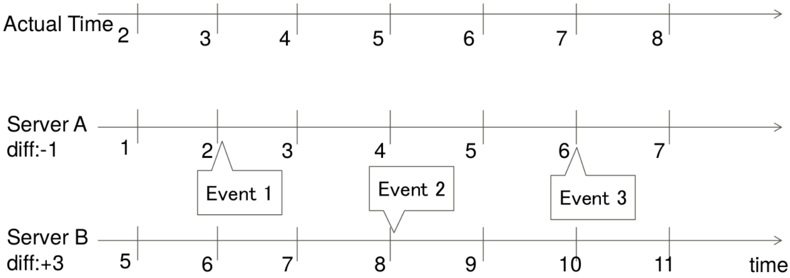


Figure: Global clock: Accurate

Example: Event ordering: Wrong order



Event Ordering based on Actual Time

Time	Event
3	Event 1
5	Event 2
7	Event 3

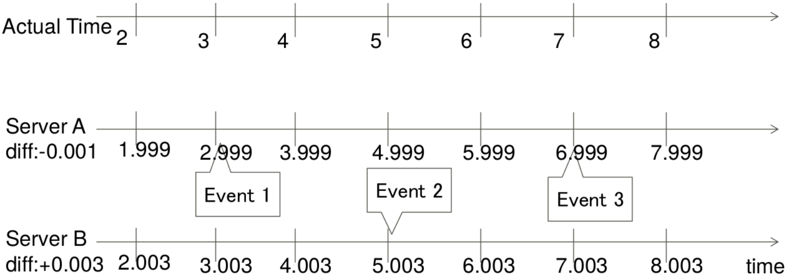
Event Ordering based on Timestamp

Time	Event
2	Event 1
6	Event 3
8	Event 2

reverse!

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

Example: 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).

Outline

Clock synchronization

Logical clock: Lamport time-stamps

Real clock: Precision Time Protocol

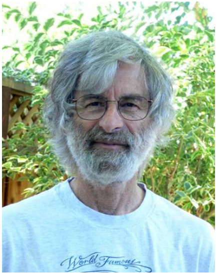
References

Logical clocks in general

- Achieves chronological event ordering across nodes in a distributed system
- Especially useful in absence of synchronous real clocks
- Often it is not necessary to know when (real clock) an event happened, but rather the logical ordering of events
- Each process keeps track of logical local and global time
- We need to keep the logical clocks synchronized
- Typically, a logical clock algorithm updates
  - logical local time after each local event
  - logical global time after each data exchange

About Lamport time-stamps

- (Leslie) Lamport time-stamps, 1978. Turing Award, 2013
- Algorithm to determine partial ordering of events in a distributed system
- A Lamport logical clock is basically an incrementing software counter maintained by each process



Time, Clocks, and the Ordering of Events in a Distributed System

Leslie Lamport
Massachusetts Computer Associates, Inc.

Lamport time-stamps pseudo code

- A process increments its counter before each local event
- The counter value is included with the message when sending

Send

```
time = time+1;
time_stamp = time;
send(Message, time_stamp);
```

- A process updates its counter when receiving a message to the greater of current counter and received time-stamp
- When the counter incremented, the msg is considered received

Receive

```
(message, time_stamp) = receive();
time = max(time_stamp, time)+1;
```

The Happened-Before Relation 1/2

Events $\{e_i\}_{i=1}^{E \in \mathbb{Z}_{++}} \subset \mathcal{E}$ are time-stamped by $C(e_i) \subset \mathbb{Z}_{++}$ such that

1. $e_1 \rightarrow e_2 \Rightarrow C(e_1) < C(e_2)$ (clock consistency)
2. thus, if $C(e_1) \nless C(e_2)$ then $e_1 \not\rightarrow e_2$ (contrapositive)
3. $e_1 \rightarrow e_2 \Leftrightarrow C(e_1) < C(e_2)$ (strong clock consistency)

where $\rightarrow$ denotes *happened-before*.

- Lamport clocks: 1, 2
- Vector/matrix clocks: 1, 2, 3 (strong clock consistency)

The Happened-Before Relation 2/2

The following holds for the *happened-before* relation $(\rightarrow)$

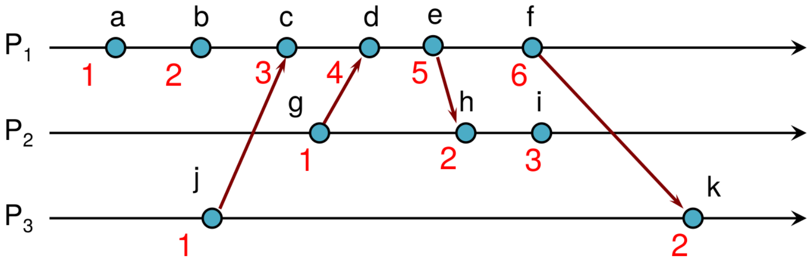
- Transitive:
  $\forall e_1, e_2, e_3$, if $e_1 \rightarrow e_2$ and $e_2 \rightarrow e_3$, then $e_1 \rightarrow e_3$
- Irreflextive:
  $\forall e_i : \quad e_i \not\rightarrow e_i$
- Antisymmetric:
  $\forall e_1, e_2$, where $e_1 \neq e_2$, if $e_1 \rightarrow e_2$, then $e_2 \not\rightarrow e_1$.

Example: Event counting: The setup

Consider the following

- Processes: P_1, P_2, P_3
- Events: $a, b, c, ..., k$
- Local event counter in each process
- Clock is advanced between consecutive events in same process
- Processes communicate
- Each message carries a time-stamp of sender's logical clock

Example: Event counting: Wrong ordering

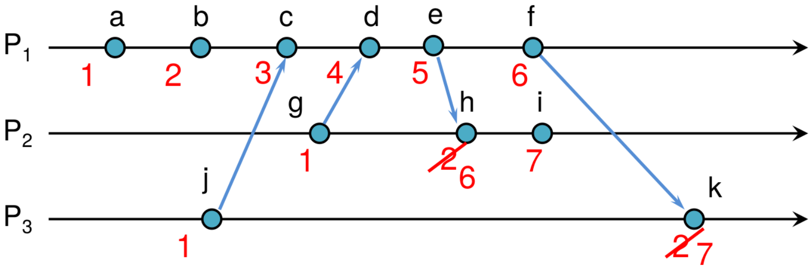


Wrong ordering

- $e \rightarrow h$ but $5 \geq 2$
- $f \rightarrow k$ but $6 \geq 2$

Courtesy of P. Krzyzanowski

Example: Event counting: Corrected ordering



Corrected ordering

- $e \to h$ and $5 < 6$
- $f \to k$ and $6 < 7$

Courtesy of P. Krzyzanowski

Outline

Clock synchronization

Logical clock: Lamport time-stamps

Real clock: Precision Time Protocol

References

IEEE

IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems

1588™

IEEE Instrumentation and Measurement Society

Sponsored by the
Technical Committee on Sensor Technology (TC-9)

IEEE
3 Park Avenue
New York, NY 10016-5997, USA

24 July 2008

IEEE Std 1588™-2008
(Revision of
IEEE Std 1588-2002)

Figure: Accessible via IEEE Xplore (289 pages).

Clock synchronization by 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

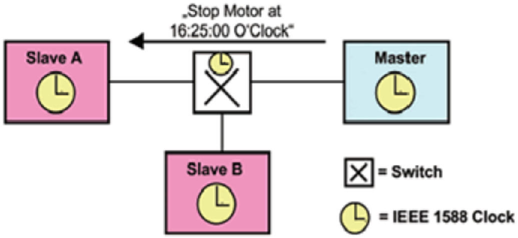


Figure: Clock synchronization by the Precision Time Protocol

Precision Time Protocol: Timeline

- 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: E.g. application area

Industrial automation

- Plant floor control and management of sensors and actuators

Telecommunication

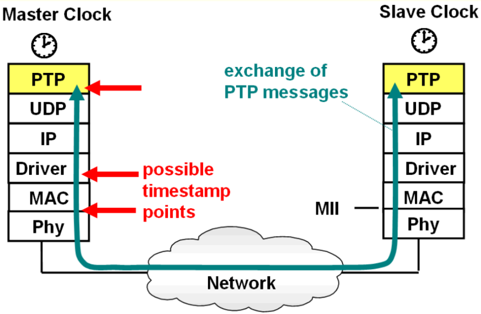
- Control voice, video, and text services in beyond 3G arch.

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
- Synchronization and data transfer on the **same** network
- Minimal admin, network, software and hardware requirements
- Most precise clock synchronizes the other clocks



(Weibel, 2009 [2])

Time stamping for clock synchronization

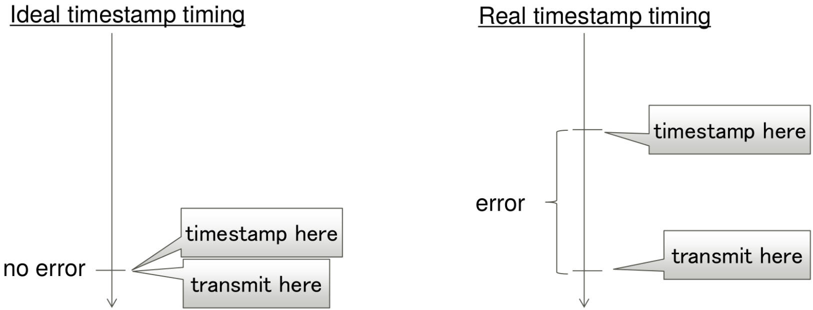


Figure: Minimize the time (error) between time of stamping and transmitting or receiving (Fujitsu).

HW vs. SW time stamping

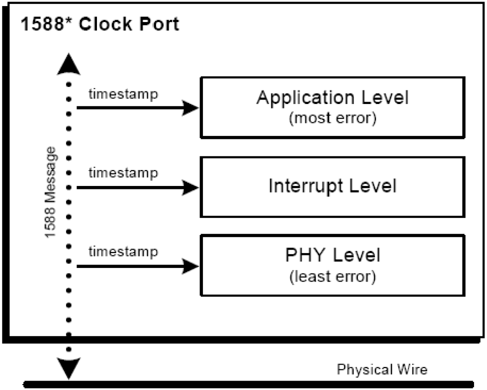


Figure: Hardware vs. software time stamping (Intel).

Synchronization element with Time Stamp Unit

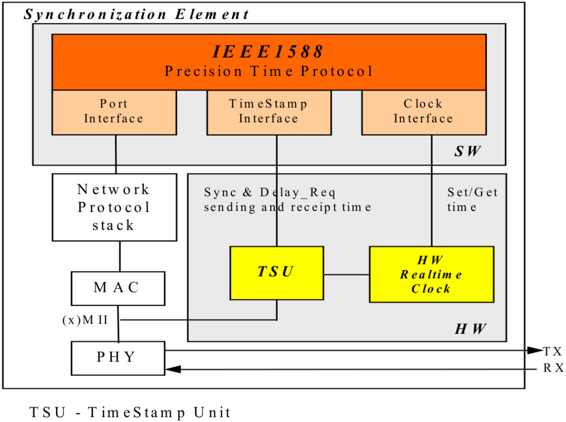


Figure: Synchronization element with Time Stamp Unit (TSU) and Real-time Clock in hardware. The rest of the protocol can be in software as timing requirements are lower (Mohl, 2003 [1]).

Hardware support for IEEE PTP

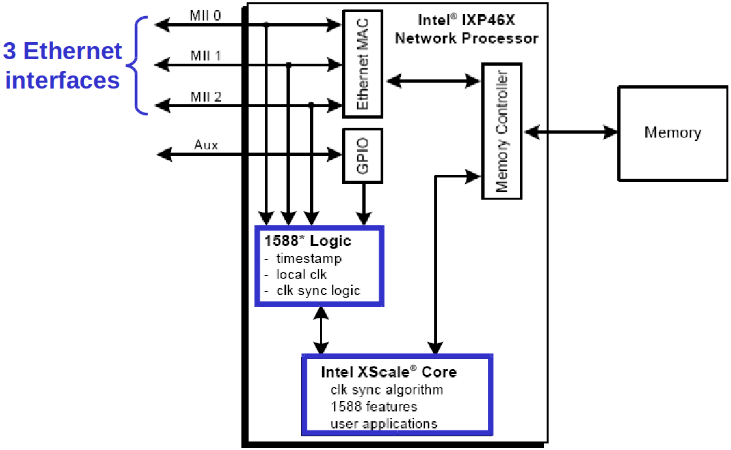


Figure: Hardware support for IEEE PTP, e.g. Intel IXP46X Network Processors (Intel).

PTP: Phase 1: Offset correction

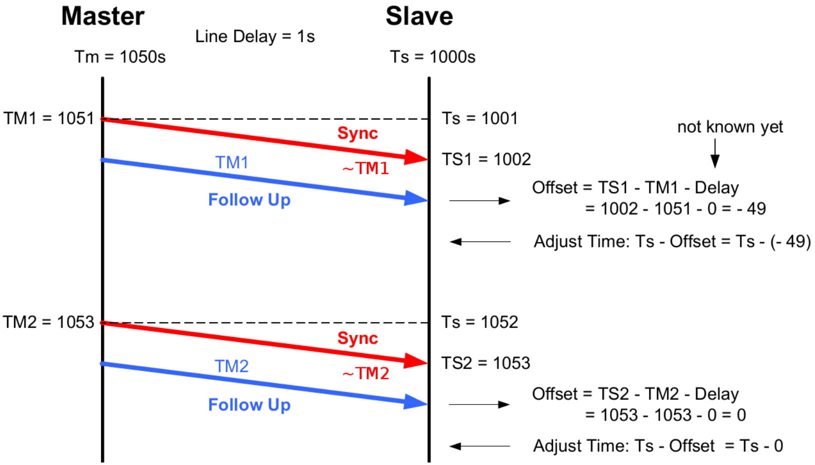


Figure: In offset correction, the line delay is assumed to be 0s. The actual delay (1s) is measured and added in the next phase (Mohl, 2003 [1]).

PTP: Phase 2: Delay correction

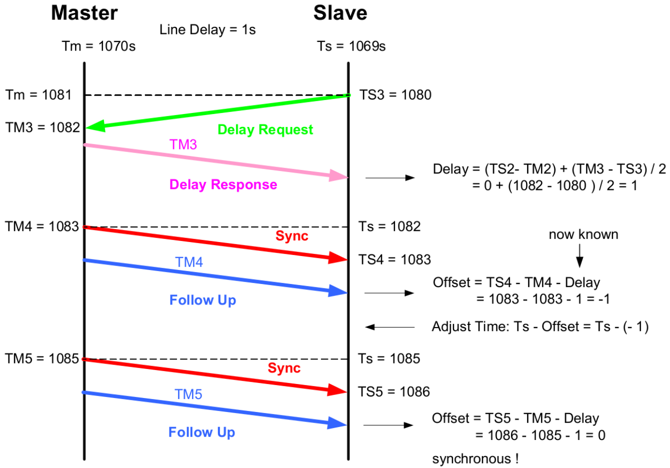


Figure: Known: M is ≥ 0s ahead. Assumed: Symmetric delay (Mohl, 2003 [1]).

E.g: Two subnets and a boundary clock

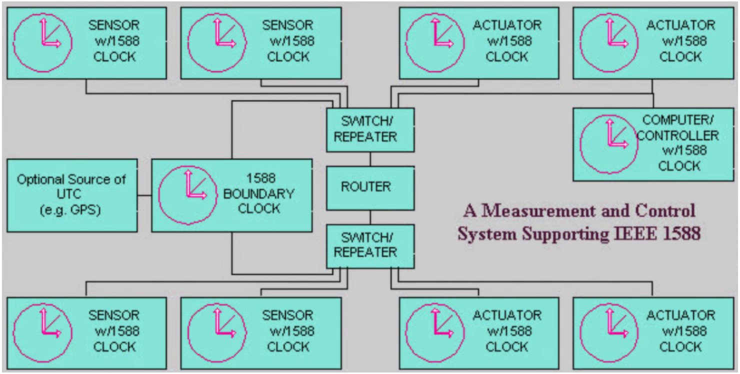


Figure: Measurement and control system with IEEE PTPv2 (www.nist.gov).

Switch with boundary clock 1/2

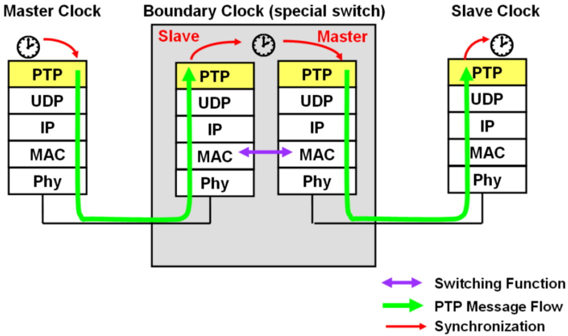


Figure: Boundary clocks synchronize subnet nodes' clocks across switches and routers (Weibel, 2009 [2]).

Switch with boundary clock 2/2

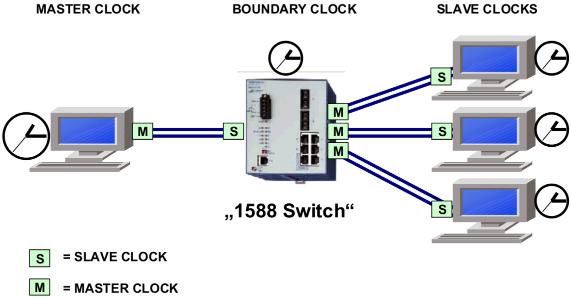


Figure: Switch with boundary clock (Mohl, 2003 [1]).

Synchronization hierarchy

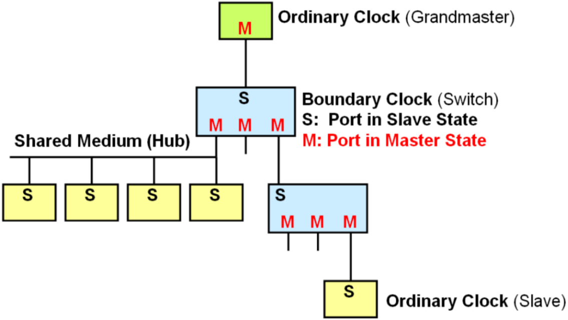


Figure: The most precise clock, i.e. the Grandmaster, is selected by the Best Master Clock Algorithm. Grandmaster is root of the hierarchy (Weibel, 2009 [2]).

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

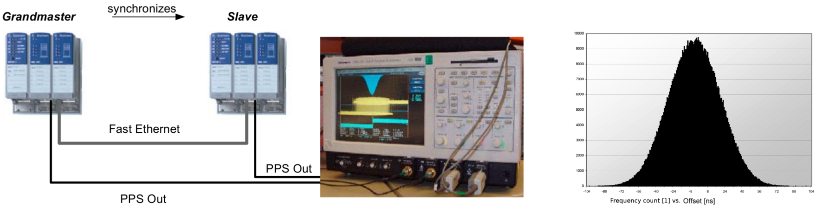


Figure: Master/Slave clock setup with Ethernet packet generator

(Mohl, 2003 [1])

Normal distribution - How errors often distribute

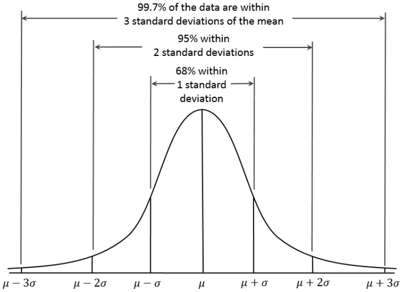


Figure: Normal dist: values less than one std, σ, away from the mean, μ, account for 68.27% of the set; while 2σ from μ account for 95.45%; and 3σ account for 99.73%. Fig.: Courtesy of Dan Kernler

$$\text{PDF: } f(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Outline

Clock synchronization

Logical clock: Lamport time-stamps

Real clock: Precision Time Protocol

References

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] 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.