

# Packet Based Time Synchronization Using the IEEE 1588 Standard

Tamás Kovácsházy, PhD



Méréstechnika és  
Információs Rendszerek  
Tanszék

# Introduction

Details about representing time and using it in computers...

# Time

- One of the seven base units of the SI system
  - Very special, it grows continuously with a constant speed
  - It is strictly monotonous and continuous
- How we use it in computer systems
  - Ordering events based on time (timestamping)
  - Measuring time between events (difference of timestamps)
  - Measuring other physical units based on time
    - Example: Measuring speed based on distance and time ( $v = s/t$ )
  - Etc.
- The most frequently used physical unit, though rarely think about it in computer science...
- But that is changing...

# Measurement of time

- Temporal measurement, or chronometry
- Two fields:
  - Calendars
    - Epoch : The starting “event” of a calendar
      - UNIX clock...
    - Splitting time to special, human units or intervals (quite different from other units), there is a lot of problem due to it
    - 1 minute is 60 seconds, 1 hour is 60 minutes, 1 day is 24 hours
    - 1 week is 7 days, **1 month is 28, 30, 31, or sometimes 29 days**
    - **1 year is 365 or 366 days**
    - **GPS leap second, and other compensation**
    - **Stellar periods (human concept)**
    - Calendars: Gregorian calendar, but there are other national or religious calendars
  - Clocks
    - Physical devices to measure time

# Calendars

- Gregorian calendar is used now
  - It was first used on 4th of October, 1582 in some part of the world, but gained wide scale use later
    - Most of the European countries joined later
    - Russia changed to it only in 1918
  - The Julian calendar was used before it
  - All of this is due to some stellar irregularities (how Earth rotates around the Sun)
    - The Gregorian calendar will be OK for the next 3000 years
- It is very hard to determine when a past event happened (Russian 1917 October revolution happened in November)
- Lot of countries use different calendars...
- This calendar mess is a real issue from the point of view of algorithms...

# Coordinated Universal Time, UTC

- French and English people cannot make an agreement on the abbreviation...
- Based on the International Atomic Time (TAI)
  - 35s difference (December of 2013)
  - Leap Second are introduced at approximately 18 month
    - The Earth rotates slower and slower due to various energy losses (except some rare situations)
  - TAI does not take into account this, while UTC does take into account the rotation of Earth and other stellar events
- We (people) tend to use UTC:
  - All the other time formats are based on localization...
  - Summer/winter time is not an issue (it is only localization also)
    - UTC does not depend on it, only local time
  - However, TAI seems to be better for computers
    - Leap second does not exists in TAI
    - It also a human “artifact”
    - It is a strictly monotonous...

# Some additional info about UTC

- Monotonous
  - A minute can be 60 s, but sometimes it can be 59 or 61 seconds...
    - It was never 59s, but it was 61s 35 occasions (December of 2013)
      - When these events were? It is fundamental to handle time, there is a table for it...
      - Even a major earthquake can influence UTC (Japan 2011 EQ did it)
    - The computer knows this table (it is received in patches, leap seconds are inserted as they are needed)
    - This is a mess, some better solutions are under research...
- After a certain precision there are relativistic effects (twin paradox, atomic clocks on GPS satellites)
  - Keeping time in space a science
  - Other formats are used in astronomy (Barycentric Dynamical Time, TDB)
- It is not a simple nor a transparent system (also a mess)...

# Clocks

- Physical device to measure time
- They show time from an epoch (starting point in time)
- Components:
  - Impulse source (oscillator)
    - Provides impulses with a given frequency
  - Counter
    - Count impulses from the epoch
  - Display
    - Shows time in a predefined format
  - Management
    - Setting/maintaining the clock...





# Describing clocks

- Clock properties:
  - Stability (how much the frequency of the oscillator changes with time)
  - Precision (how much the shown time differs from a reference clock, that is typically UTC/TAI)
  - Resolution (resolution of time shown on the display)
- The clock integrates the oscillator impulses (the frequency)
  - Precision and stability are interrelated
- Time domain description:
  - Time series plots
  - Allan variance/deviation : Two sample variance for time  $\tau$
- Frequency domain description
  - Spectra of the jitter series (ideal versus actual event)
  - Phase noise : frequency domain representation of rapid, short-term, random fluctuations in the phase of a waveform

# Why clocks are inaccurate?

- Erroneous initial setting
  - We cannot set the clock when it starts properly according to the reference clock (delays in perception and action)
  - **Setting the clock against a strictly monotonous and continuous time principle!**
  - It means that a clock can be set when it is not used to check time
    - otherwise all time bases processes may fail
- The frequency error of the oscillator (offset)
  - Production error (difference from the nominal value)
  - Frequency drift
    - Temperature, movement, mechanical forces influence the frequency
    - Electrical variation (e.g. supply voltage, EMI, etc.)
    - Ageing
  - The unit of frequency error is PPM (parts per million) or PPB (parts per billion)
    - 1 PPM : Can see a 350-400 m object on the Moon (e.g. Empire State Building in New York)
    - 1 PPB : Can see a 0.35-0.4 m object on the Moon (e.g a medium size dog in standing position)
    - Moon-Earth distance is 350 000 km – 400 000 km

# Long term effects of the inaccuracy

- Frequency error accumulates in the counter
  - The clock is late or in hurry
  - The frequency must be measured and corrected
  - **The strictly monotonous and continuous time principle cannot be violated!**
    - If the clock is late, we run it faster to catch up with the reference time
    - If the clock is in a hurry, we run it slower to let the reference time to reach it
    - Otherwise the local time will wonder around the global time
      - The actual properties of this behavior must be also investigated
        - » Both in time domain (Alan variance/deviation)
        - » Frequency domain (Phase noise)
      - These are also interesting properties for non synchronized clocks...
    - If the reference is not available: Holdover

# Clock hardware in computers

- Oscillators
  - Real-Time Clock (RTC) based on a 32 kHz quartz
    - A small IC used for off-line timekeeping
  - HW System Clock, system tick (Timer IT) and the system clock derived from it
    - Provides clock for all digital circuits including the CPU
  - NIC clock (for all network interfaces)
    - Timestamp unit for hardware times tamping (receive/send)
  - Clock of the sound card
    - How long a sound is played?
  - Graphics card (frame, line and pixel frequency)
    - How long a video is played?
  - External time sources: GPS receiver, DCF77, IRIG, NTP or IEEE 1588 network clock, etc.
- Which one is taken into account?
  - Clock ensemble is the best, but hard to do technically
  - Synchronization of clocks...
    - Sound and video in sync is a major issue itself!
    - Especially in distributed systems
      - This is why we have the Audio Video Bridging standard (later)

# Typical HW and SW architecture, RTC

## ■ Real-Time Clock

- Measures time while the computer **is switched off**
- Low power, battery based operation
- Properties:
  - Inaccurate, especially when the battery is low
  - Medium temperature dependence (e.g., charging the main battery of a portable computer influences timekeeping properties)
  - Slow access (typically connected by a slow bus such as I2C/SMI)
  - Capable of waking up the OS on a given time (most cases, not all)
- The counter uses very “strange” data structure
  - Binary coded decimal numbers
  - In other words, it uses a human form, not a machine form (binary)

# Typical HW and SW architecture, Sysclock

- HW clock driving the whole system including the CPU, etc.
  - Typically using a clock generator chip with multiple PLLs
  - Phase noise, jitter?
    - Spread-spectrum (artificially increased random jitter) to limit EMI...
- System tick and derived system time
  - Initialized at startup from the RTC
  - At shutdown it is written to the RTC (can be also periodically updated to the RTC)
  - The stability of the oscillator and the accuracy of the clock primarily depends on the machine temperature
    - So it depends on the machine load and the environmental parameters
    - It may be also used to detect malfunctions of FANs in the machine (overheating)
  - Construction:
    - HW counter:  $N \cdot 1\text{MHz}$  clock divided to a 10-20 ms clock tick, which requests an interrupt (binary counter)
    - SW counter for low resolution clock (binary counter)
    - Subdivision: The HW counter or some other counters (Time Stamp Counter) may be accessed for increasing the resolution (us or ns resolution is required today)
    - SW timers are derived from the clock tick also (SW timeout, time based scheduling, etc.)

# Example: Linux timer

- Jiffies (system tick): Kernel dependent (100 Hz, 1000 Hz, 250 Hz, 300 Hz)
  - Can be changed by changing one constant in a header file in the kernel source and recompiling the kernel
  - Defines the resolution of the system clock also if no subdivision is used
- High resolution timer (since kernel 2.6.21): It depends on the available HW
  - `clock_getres()` returns resolution (if supported)
  - Tasks waiting for timers are stored in a binary tree
- If you want to know more about timers in your Linux machine:
  - `cat /proc/timer_list | less`
- More than one system clocks are available in Linux:
  - Settable system clock : **CLOCK\_REALTIME**
  - Monotonous, non settable: **CLOCK\_MONOTONIC**
  - Process and thread clock for time domain scheduling information, etc.
- Clock synchronization
  - `adjtimex` – synchronize the system clock to external reference clock
  - RFC 5905 (Network Time Protocol)
  - Tunes the oscillator of the clock (virtually, not really, hardware tuning is not supported on the hardware)
  - It implements a software Phased-locked or Frequency-locked loop by changing the division ratio of the HW part of the system clock

# Typical errors in common hardware

- PCs and other devices use quartz crystals (cheap ones) for oscillator frequency determination:
  - Specification: 200 ppm max. error: the clock is maximum  $\pm 17.28$  s off a day
    - It adds up to a minute in less than 4 days!
    - Quartz for Ethernet is allowed to have 50 ppm frequency error
  - Average error : 70-80 ppm (NTP based measurement of thousand of computers)
  - Temperature dependence:
    - 0.5-1 ppm /°C typical
  - Better oscillators are drastically more expensive nor they solve the problem (the clocks will be off slower)
    - TCXO 1-5 ppm max. error, but costs 3-5 USD in large orders
    - OCXO 1-10 ppb max. error, but costs around 100 USD or more
    - Rubidium or Cesium clocks (under 0.001 ppb)
      - Chip-Scale Atomic Clock : 1500 USD (considered low cost)
      - Large physical size and power consumption (even for the CSAC)
- Embedded systems may have only a RC oscillator...
  - 1000 ppm error or more error
  - Strong temperature dependence



# Consequences

- The errors are too large
  - The clocks must be synchronized to the reference time (to reference clocks)
  - Clock synchronization : The clocks advance with the same rate and show the same time
    - Requires a Phase-Locked Loop (PLL) to be implemented
  - Clock synchronization : The clocks advance with the same rate, but they may start from a different epoch
    - Requires a Frequency-Locked Loop (FLL) to be implemented
  - Connection to the reference clock?
    - Solutions:
      - Out of band : GPS, DCF77, IRIG timecode
      - In band : Network Time Protocol (NTP), IEEE 1588

# Out of Band

- A dedicated communication infrastructure for clock synchronization
- Global Positioning System (GPS)
  - Localization is based on the knowledge of precise time
    - An extremely accurate estimation of UTC/TAI is available in GPS receivers
  - Interface:
    - Timecode (time in UTC), typically through an asynchronous serial port
    - Pulse Per Second signal (for clock synchronization)
      - Typically under 1  $\mu$ S accuracy...
      - GPS modules with 10 ns accuracy (stationary location required) are available!
- DCF77 (Germany), similar service exists in other countries (e.g. USA)
  - Long-wave (77.5 kHz) radio station transmits the reference time
    - Quite inaccurate due to wave propagation
    - Availability is limited in Hungary (we are too far away from the transmitter)
  - Primarily for setting clocks, watches used by people
- IRIG (Inter-range instrumentation group) timecode
  - Professional distributed measurement
  - Developed in the USA for military and aerospace use but widely used everywhere
  - Dedicated cables are used to transmit the time information

# In band

- We use the regular communication channel also to transmit time
- Major problems:
  - Delay, delay asymmetry, jitter
- Typically over TCP/IP
  - Over Ethernet or WI-FI it is also possible (less complex)
- Network Time Protocol (NTP)
  - Hierarchical clock synchronization
    - Stratum 0 (reference clocks, GPS, atomic clocks, DCF77 with limitations, etc.)
    - Stratum 1 (NTP servers connected to reference clocks)
    - Stratum N (Level N. in the clock hierarchy)
  - Redundant (multiple servers can be used to minimize errors)
  - Optimized for Internet, and precise for human use (100 ms-10 ms offset to the reference time is possible)
  - With local Stratum 1 servers and special setting 1 ms is achievable in LANs
- IEEE 1588 Precision Time Protocol
  - Master-slave protocol for LANs
  - High precision (under 1 us is not a problem, under 100 ns is possible)
  - Hardware timestamping on the participating hosts and network instruments must be used for that precision

# Summary

- Time is a strictly monotonous, continuous physical unit growing with the same pace
- Clocks: Oscillator + counter + display + management
- Clocks are inaccurate
  - Initial setting is erroneous, frequency offset and drift
- Setting the clock
  - The clock jumps, dangerous in applications using time
- Synchronizing the clock
  - Clock is monotonous and continuous
  - We tune the frequency of the clock
- NTP and IEEE 1588 clock synchronization protocols are available to solve the problem

# In band time synchronization

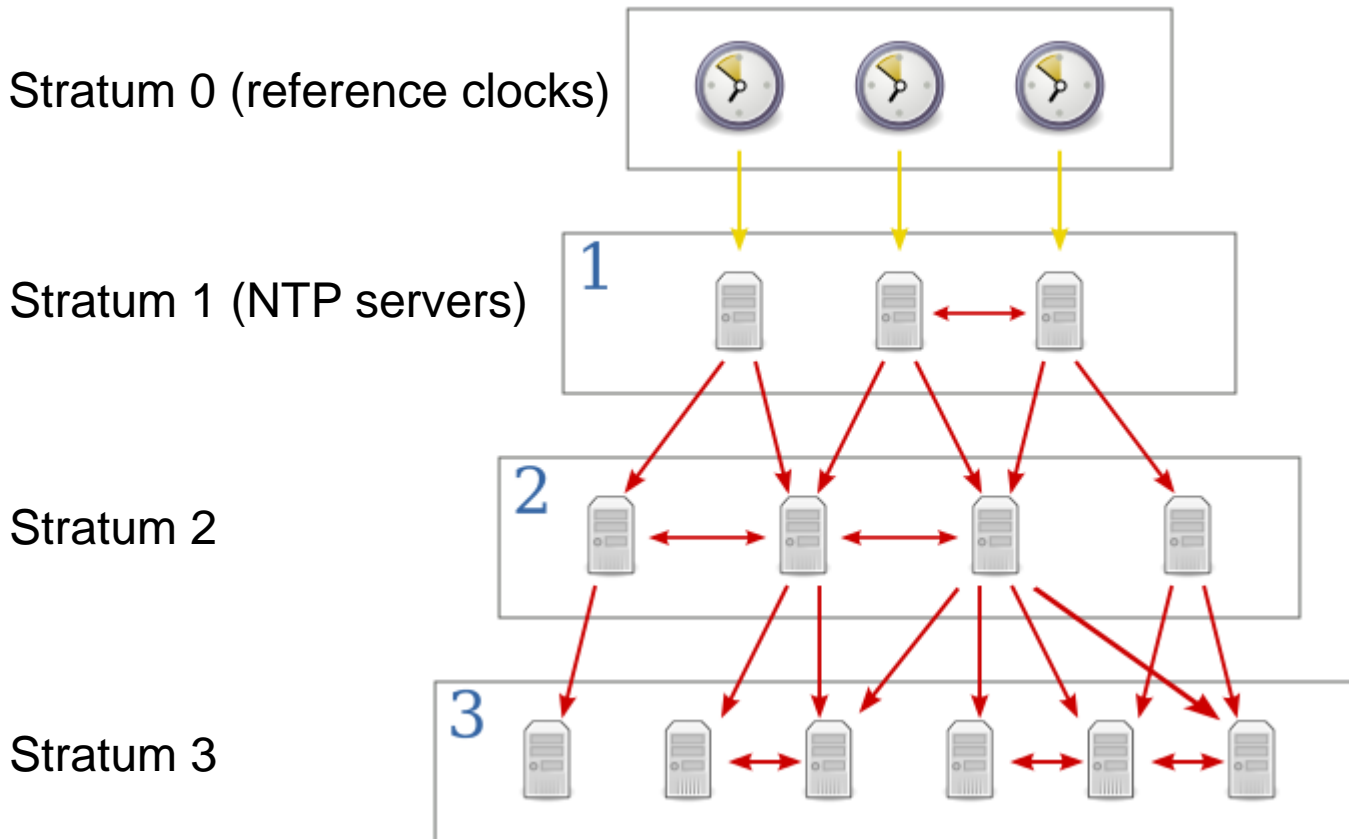
How NTP and IEEE 1588 works...

# NTP (short introduction)


- Inventor and main developer: David L. Mills
- Fundamental idea:
  - Ask the a “time server” about the current time and take network and processing delays into account...
    - You know when you asked the server about time (request, t0 timestamp on the client)
    - You know when the server received you message, it tells you in the reply (receive request, t1 timestamp on the server)
    - You know when the server replies, it is also in the reply (reply, t2 timestamp on the server)
    - You know when the message from the server arrived (receive reply, t3 timestamp on the client)
  - Assume that the message transmission time for the request and the reply is identical (the network delay is symmetric)
    - Then the estimation of the offset between server and client can be computed as:
$$t = ((t1 - t0) + (t2 - t3)) / 2$$
    - The time difference between the client and server does not effect the offset computation (if it can be considered constant during the measurement)
  - Do not set the local time with offset, but change the rate of the local oscillator to eliminate the offset...

# NTP

- Hierarchical semi-layered architecture
- Practically: P2P architecture, SNTP is client only



# The problem with NTP

- The message transmission time for the request and the reply is **NOT** identical!
- Contributing factors for delay:
  - Client OS and network hardware latency
  - Network latency from client to server
  - Server OS and network hardware latency for request processing
  - Server OS and network hardware latency for reply processing
  - Network latency from server to client
  - Client OS and network hardware latency
- OS latency has a high variability due to OS scheduling and queuing
- Network hardware latency is asymmetric due to:
  - Speed asymmetries (ADSL, wireless links, etc.)
  - Load asymmetries (store and forward delay, processing delay, queuing)
- Using QoS reduces the asymmetry only
  - One maximum sized packet can block any queues
  - Any number of timing packets can block each other in network hardware...
- NTP tries to reduce it by advanced filtering method, with limited success
  - It cannot be done properly (there are fundamental limits)
  - The only option is to measure the asymmetry
  - Active (time aware) devices (master, slave, network) could do it  **IEEE 1588**



# IEEE 1588

- NTP cannot provide required precision for distributed measurement and control...
- NTP is designed for the Internet, the application domain is LAN specific
- Solution:
  - **IEEE 1588-2002**, *Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, 2002*
  - **IEEE 1588-2008**, *Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, 2008*
    - Modified based on the practical experiences gained by using IEEE 1588-2002
    - 289 pages (not big for a standard)
  - The standard is also called Precision Time Protocol (PTP)
  - Here we consider IEEE 1588-2008 only
    - The only exception if IEEE 1588-2002 is explicitly referred

# Fundamental idea of IEEE 1588

- NTP cannot handle asymmetric delay
- IEEE 1588 is designed to measure one-directional delay components on the participating nodes...
  - Can work on any HW, but works well on HW supporting it
  - Hardware timestamping...
    - IEEE 1588 Event messages are timestamped
    - IEEE 1588 General messages are not timestamped
- Main asymmetric delay sources:
  - Client SW and HW
  - Server SW and HW
  - Network nodes (switches, routers, wireless access points, etc.)
    - Store and forward delay, queuing delay, processing delay, etc.
- Negligible asymmetric delay sources
  - Cable delay

# Facts about IEEE 1588 1.

- Hierarchical master-slave architecture
  - Single master (Grandmaster): Best Master Clock (BMC) algorithm selects it
- Support global time (reference clock) or local time
- Designed for the LAN
  - Primarily Ethernet medium
    - It supports some field buses also
  - But wireless (e.g. IEEE 802.11 or IEEE 802.15.4) may also be used
  - Ethernet (Ethertype : 0x88f7) or UDP (IPv4 or IPv6) transport
  - Primarily multicast communication, but may use unicast also
- One-step or two step clocks
  - One step clock: timestamps are included in HW into packets used during synchronization
    - The packet must be rewritten under the MAC layer (practically, in the PHY)
    - The layered architecture of communication is not (fully) compatible with it
  - Two-step clock: timestamps are sent in follow up messages
    - Second message conveys the real send timestamp of the synchronization message
    - Timestamp measured and sent to the software, which generated the follow up message

# Facts about IEEE 1588 2.

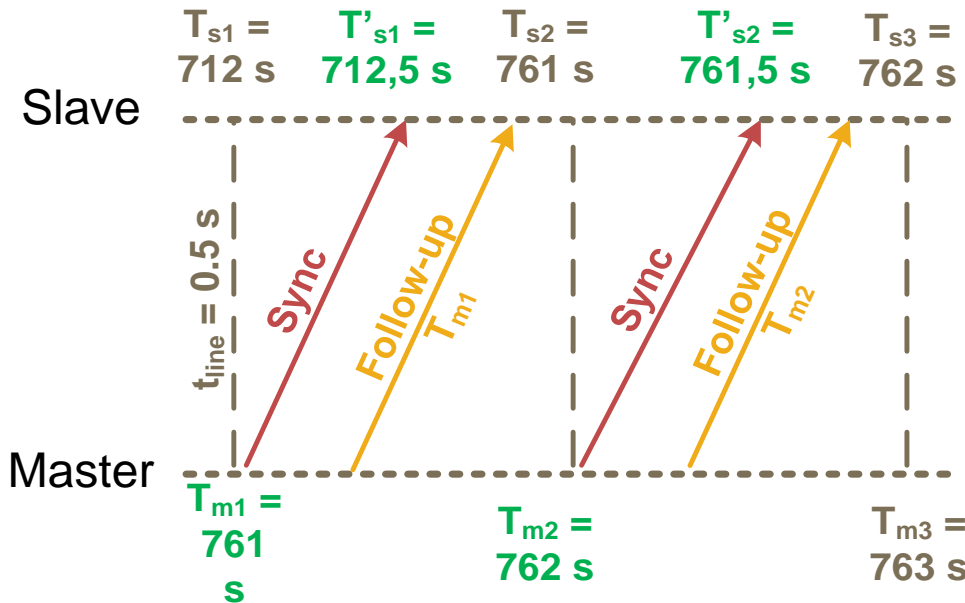
- Profiles: The standard can be extended
  - IEEE 802.1AS (802.1AS-2011 - IEEE Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks)
    - Profile for Audio Video Bridging (AVB) support
    - Layer 2 (Ethernet) transport
    - Mandatory P2P transparent clock support
  - Power profile for IEC 61850 (Substation automation)
    - Layer 2 (Ethernet) transport
    - Mandatory one step clock support

# IEEE 1588 components

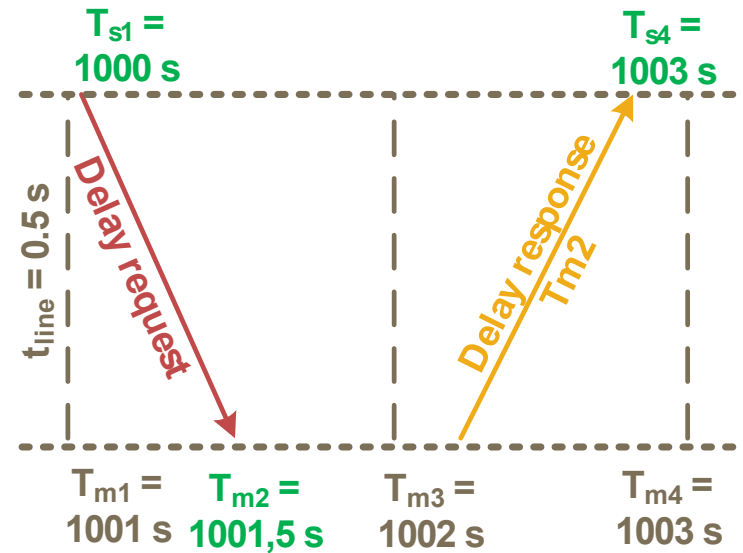
- **Clock domain**
  - A single logical group of clock synchronizing to each other using IEEE 1588
- **Ordinary clocks**
  - Master or slave clocks
  - Single port supporting IEEE 1588 in a clock domain
- **Boundary clock**
  - Multiple port device acting as slave on one port, and as master on at least one other port
- **End to End (E2E) Transparent clock**
  - Multiple port device (switch, router, etc.) supporting End to End delay measurement mechanism (client to grandmaster)
  - It measures how much time the event message spends in the device (residence time)
- **Peer to Peer (P2P) Transparent clock**
  - Multiple port device (switch, router, etc.) supporting Peer to Peer delay measurement mechanism (neighbor to neighbor)
  - It measures the residence time + peer delay
  - Better scaling and better handling of communication faults (e.g. STP/RSTP actions)
    - Local communication only
  - Functionally equivalent to boundary clocks (It is shown in a paper.)
- **Some notes about E2E and P2P transparent clocks**
  - They can be one-step or two-step
  - They cannot be mixed in a clock domain (E2E or P2P can be used)
  - IEEE 802.1AS requires P2P or Boundary clocks, E2E is not supported

# IEEE 1588 two-step clock

## Sync + Follow-up



## Delay measurement



General message



Event message

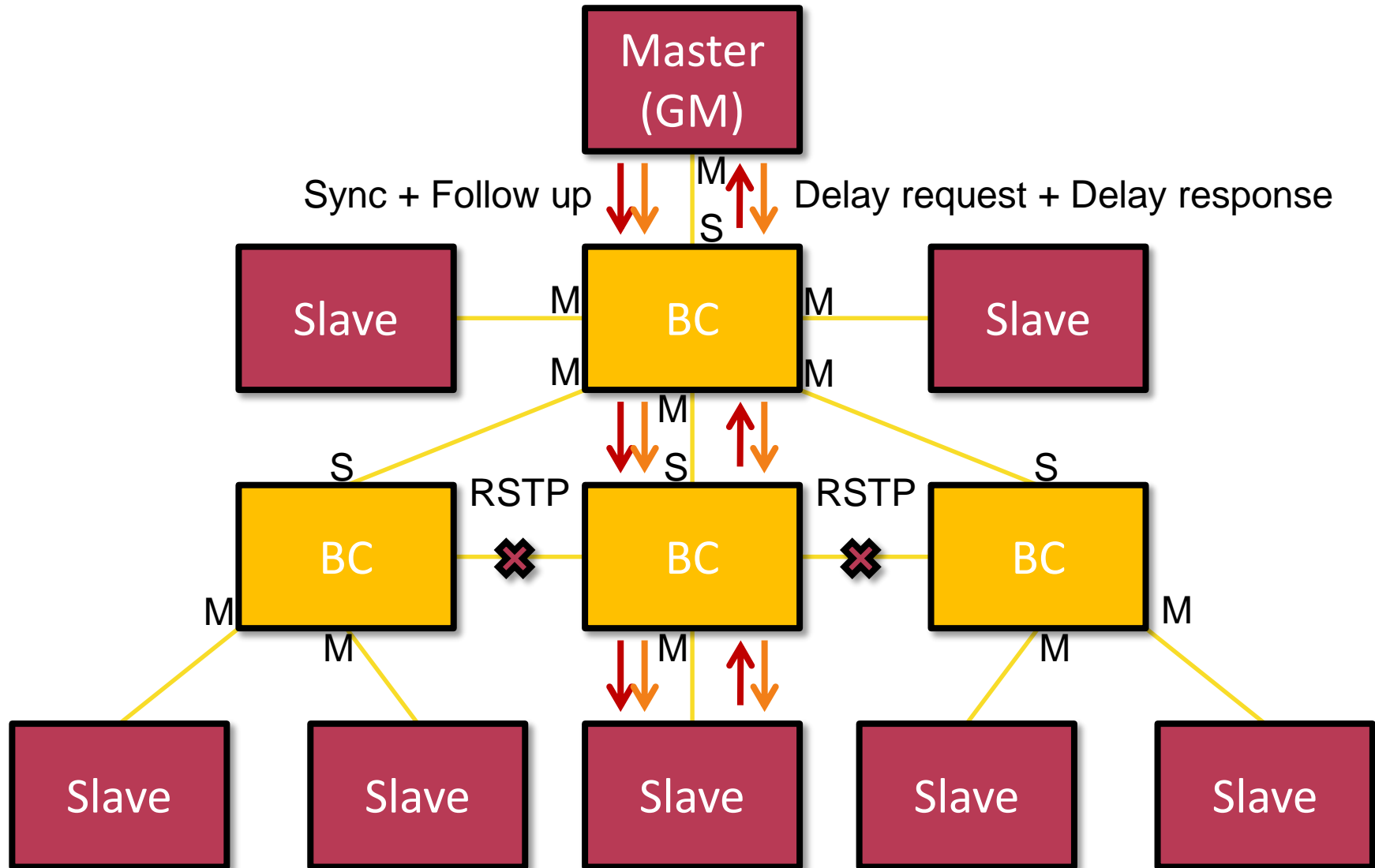


E2E transparent clock: master-slave Delay messages

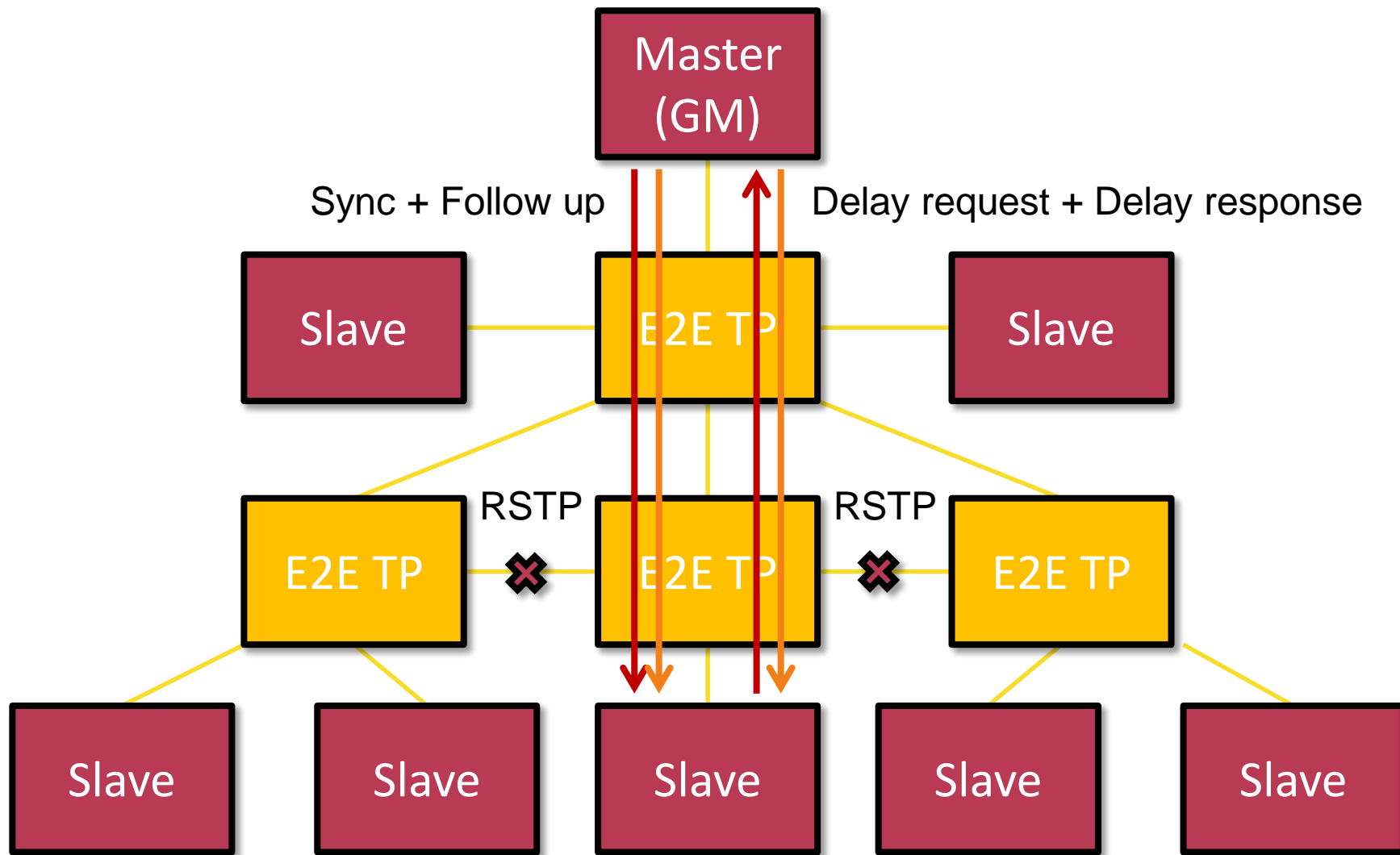
P2P transparent clock: Peer-delay messages

(answered by the nearest device, i.e. master or P2P transparent clock)

# Architecture with two-step Boundary clocks

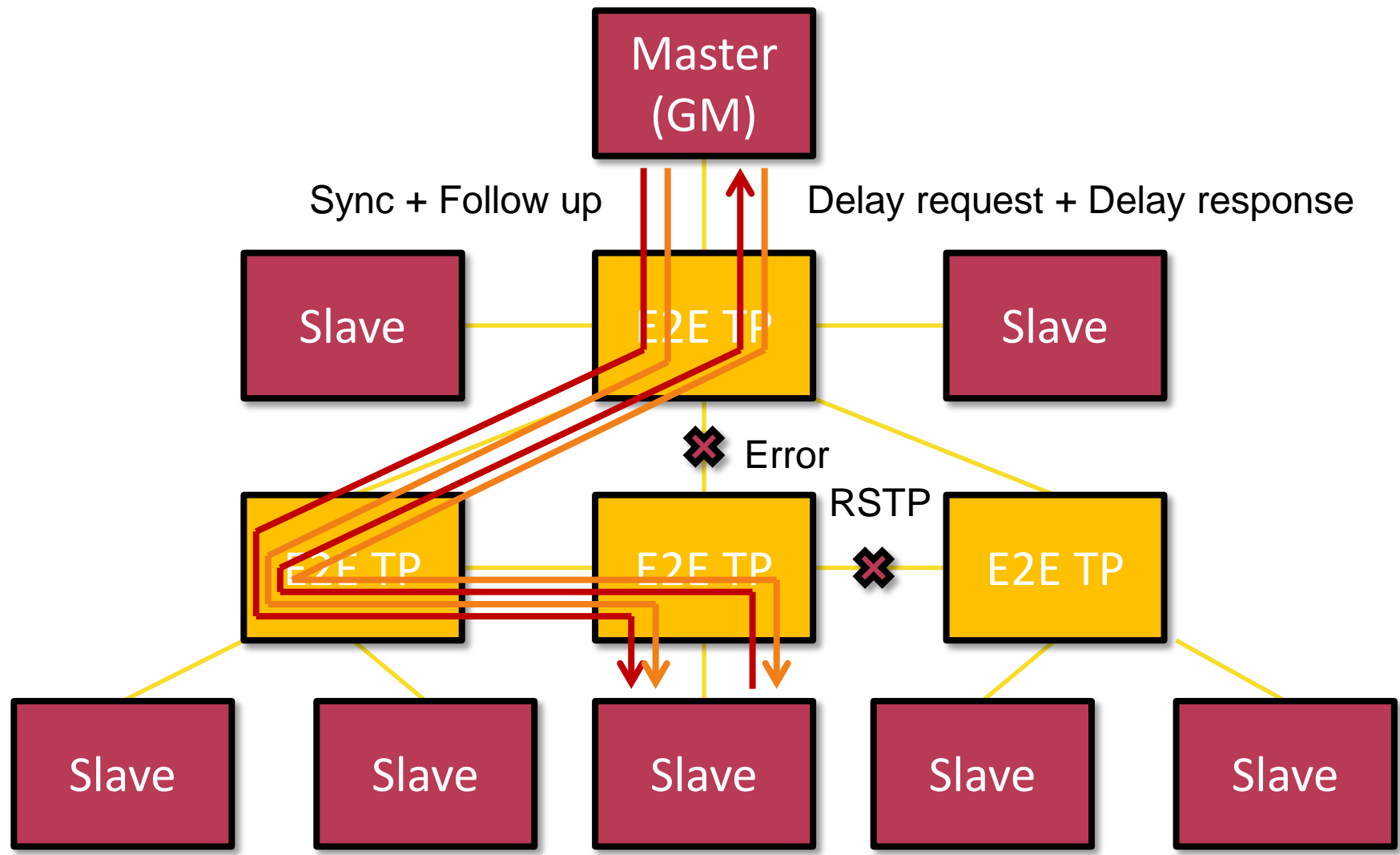


# Architecture with two-step E2E transparent clocks

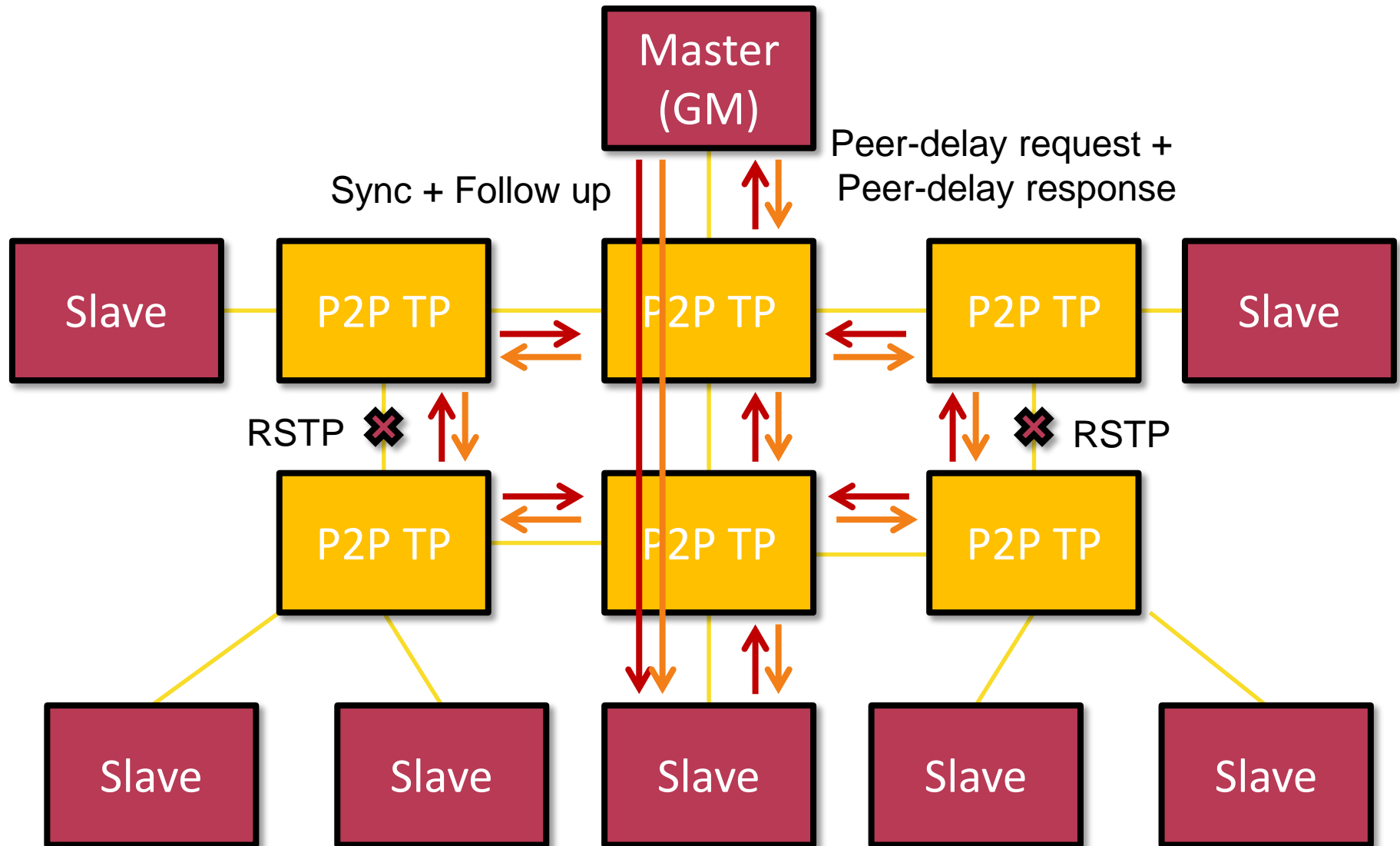




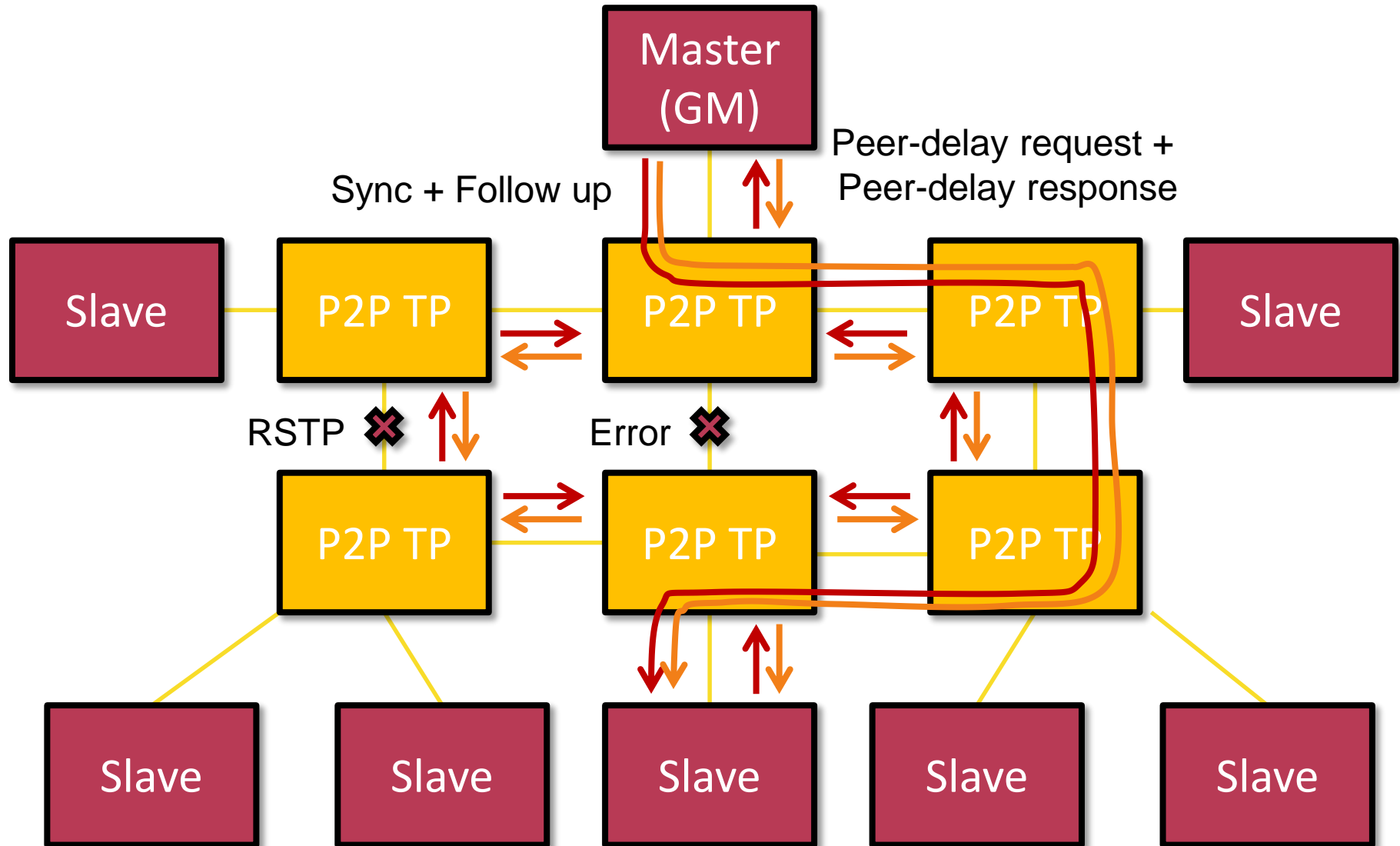
# Architecture with two-step E2E transparent clocks (after link error)



# Architecture with two-step P2P transparent clocks



# Architecture with two-step P2P transparent clocks (after link error)



# Client implementation

- Hardware timestamping is required for sub 1 us precision
- PHY or MAC timestamping?
  - PHY timestamping can be more precise (less jitter and asymmetry) but PHY to SW communication is problematic
  - MAC timestamping is simpler but less precise
- SEND timestamping is generally more complicated
  - RECEIVE is simple, timestamp can be attached to the packet as a part of a pseudo header
  - In case of sending, the packet and the timestamp move into opposite direction

# Timestamping unit

- Clock + capture unit
  - On IEEE 1588 event messages the clock is captured
  - There must be a HW filter for it...
    - All packets cannot be timestamped
    - Especially on the SEND side
  - Other capture units are also nice features (to capture external events)
    - External signal generation (1 PPS, N PPS)
    - External event timestamping
    - Testing of the precision of the clock (timestamp the clock on reference clock 1 PPS signal)
- HW Clock format
  - Synchronized IEEE 1588 standard format (truncated TAI64N)
    - ```
struct Timestamp {  
    UInteger48 secondsField;  
    UInteger32 nanosecondsField;  
};
```
    - The secondsField is the low 48 bits of TAI64 format (UInt64)
    - The nanosecondsField member is always less than  $10^9$
    - TAI64NA format exists also (attosecond resolution)
      - 32 bit nanosecond count field, 32 bit attosecond count field (seems to be a bad idea, why not attosecond only)
  - Proprietary format (synchronized or not)
    - Cannot be used in one-step clock
    - Clock transformation between the protocol SW and the HW (can be a real mess)

# HW clock synchronization

- Tuning the oscillator of the clock
  - Voltage controller oscillator (VCO), expensive...
  - PLL/FFL type operation, the loop is closed through the whole IEEE 1588 solution
- Fractional clock rate control (arithmetic clock)
  - All digital solution
  - $2^{-32}$  nS resolution typically
    - Specifies how much of the period of the oscillator (nominal 125 MHz clock, nominal 8 ns period typically) must be reduced or extended to reach the reference 8 ns
  - Intel i210: Increment Attributes Register
  - TI DP83640 : PTP Rate Low/High Registers

# Time domain interface of IEEE 1588

- Special HW or SW interface:
  - Current time with seconds resolution (e.g. UART with NMEA format or proprietary format)
  - Pulse Per Second (PPS) signal
  - Some systems: Unlocked signal
- IEEE 1588: Virtual wire between the reference time source and the slave devices using this time domain interface

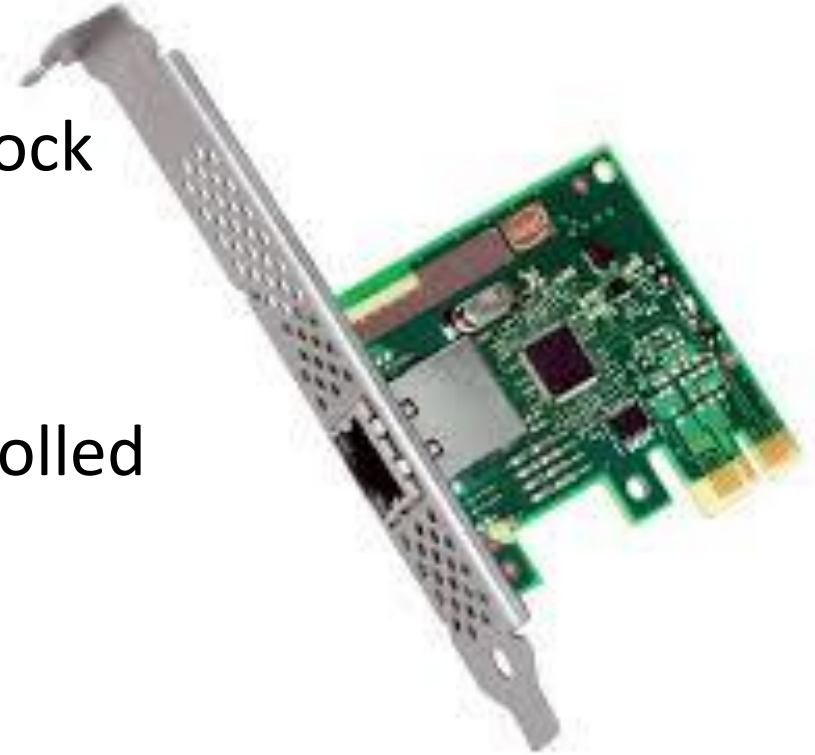
# IEEE 1588 Linux support

Using it on Linux...



# IEEE 1588 compliant NIC

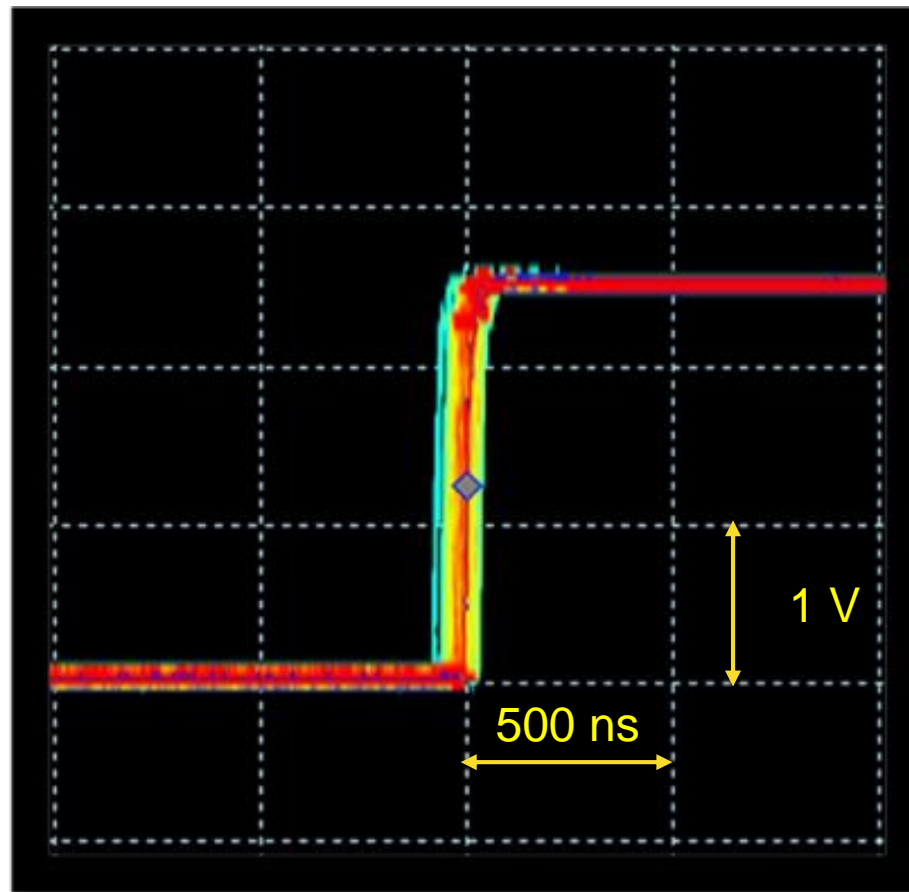
- Intel Server Network Interface Cards support it for years
  - Current ones: Intel i210 and Intel i350
  - IEEE 1588 compliant NIC clock
  - IEEE 1588 compliant timestamping unit
  - Software Defined Pin controlled capture and compare units
  - Designed for real-time networking



# IEEE 1588 support in Linux

- PTPd daemon
  - SO\_TIMESTAMPING is required for HW support from the driver (from kernel 2.6.30)
    - For Intel NICs, it is developed by BME-MIT
- linuxptp
  - Enhanced implementation based on the lessons learned from PTPd
- Linux PTP Hardware Clock (PHC) API
  - kernel 3.x, some Linux distributions have compiled in, but for most of the distribution you have to compile your own kernel
  - Intel NIC support is developed by BME-MIT
- Intel NIC + Linux is whole solution?
- Question: If yes, how precise the solution is?

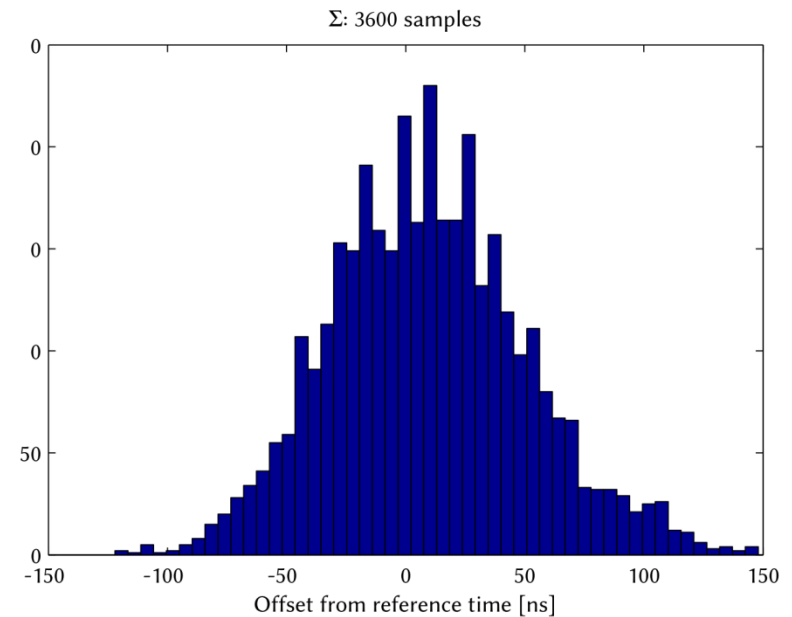
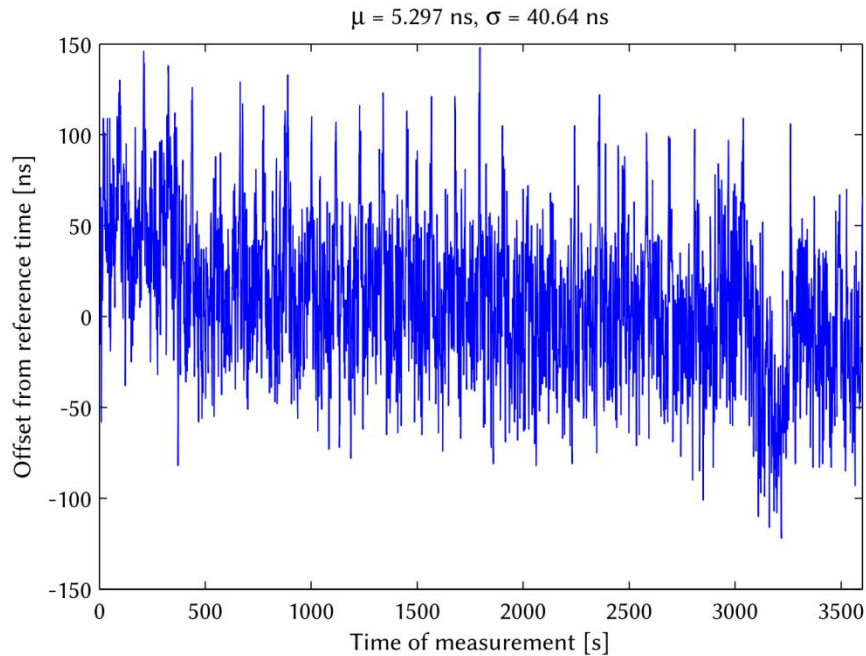
# Results: „EYE” diagram of the slave



Eye diagram of the Slave Clock's PPS output for 3600 test cases  
Triggered to the PPS output of the reference clock  
(1 hour test,  $dX = 500 \text{ ns}$ ,  $dY = 1 \text{ V}$ )

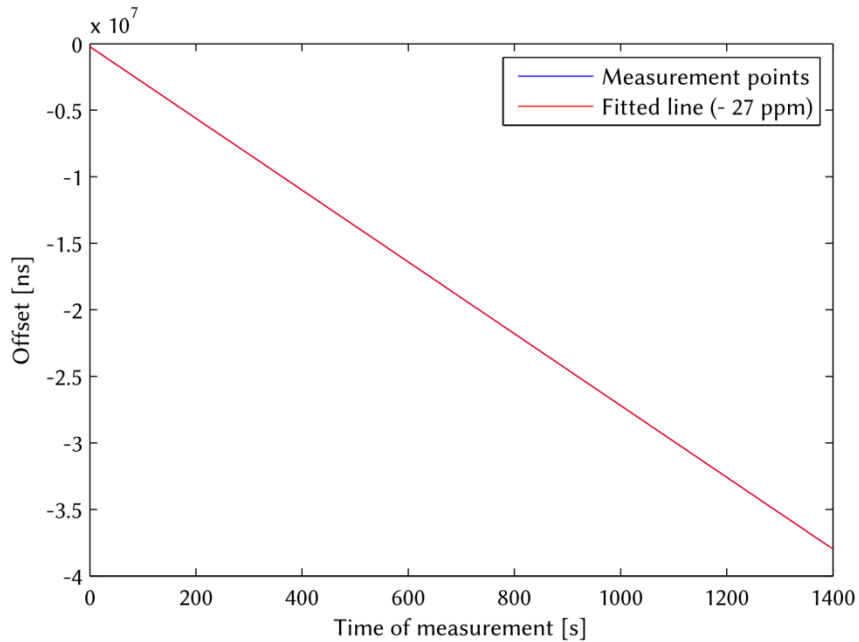
# Results: Slave clock performance

- High load on the network and the slave generated by iperf!

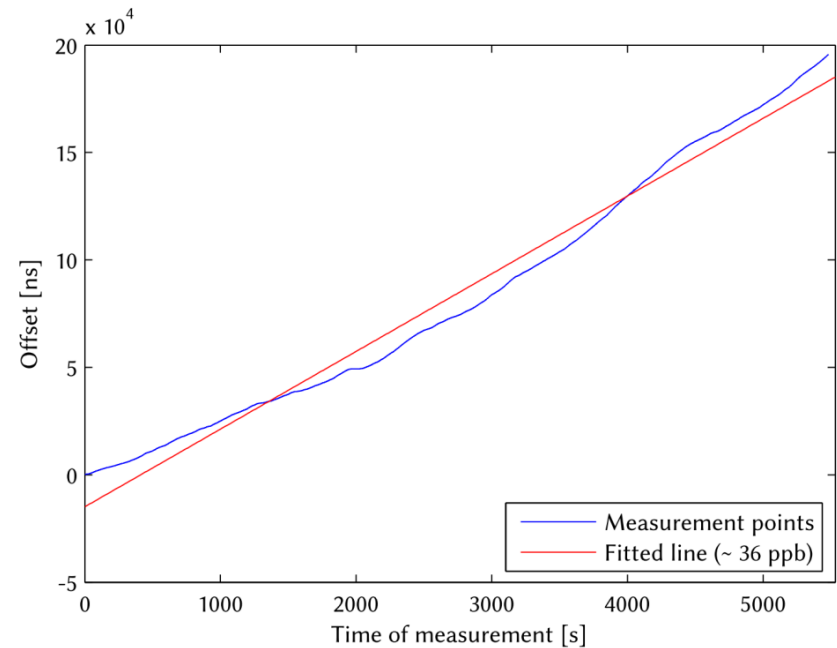


# Results: Holdover (on lab. temperature)

## Unsynchronized system



## Synchronized system losing communication to the master



# IEEE 1588 Compatible Intel HW

The devices and how they work

# Gigabit Network Interface Cards

- Intel is supporting IEEE 1588 since 2009, but device features are developed continuously
  - Intel® 82576 Gigabit NIC (launch 2009)
    - Initial IEEE 1588 support, sufficient for system clock synchronization
  - Intel® 82580 Gigabit NIC (launch Q1'10)
    - IEEE 1588 fully supported, better NIC internal clock
    - Initial I/O pin support for HW synchronization
  - Intel i350 Gigabit NIC (launch Q2'11)
    - I/O pin support for HW synchronization (Software Defined Pins)
  - Intel i210 Gigabit NIC (launch Q4'12)
    - One step clock support
      - Layer 2 transport, and Layer 4 transport for IPv4 with no UDP checksums
      - IPv6 is not supported as IPv6 specifies mandatory UDP checksums
    - SDP pins are available on production cards
    - AVB support
    - Send on time

# 10G Network Interface Cards

- X520 and X540 supports IEEE 1588 based on the available information
  - The X540 is a specialized version of the X520 for datacenter applications with minor modifications
  - The X540 is quite similar to i210 regarding IEEE 1588 support, except:
    - No one step clock support
    - No SDP pins on the production cards (chip supports them)
    - Limited AVB support (only 802.1as)
    - Only twisted pair cable support (no external PHY)
      - No 100 Mbps timesync support
  - The X520 is the predecessor of X540
    - Some IEEE 1588 functionalities are less polished...
    - No AVB support
    - External PHY module support (optics, etc.)



# i210 features, NIC clock 1.

- i210 has an IEEE 1588 NIC clock built in for timestamping events
  - The clock runs with the physical clock of the NIC
    - Nominal 125 MHz
    - Max. 50 ppm error in the full operating temperature range
      - 0-85 °C for commercial SKU
      - -45-105 °C for industrial SKU
  - It keeps time in the standard in a nearly IEEE 1588 “wire” format
    - Format:
      - SYSTIMR register (32 bit): sub ns fraction
      - SYSTIML register (32 bit): ns fraction (0 to 999,999,999 decimal value)
      - SYSTIMH register (32 bit): seconds from the PTP epoch
      - SYSTIMTM register (16 bit):  $2^{32}$  seconds from the PTP epoch
        - » Set by software, assumed to be 0, a potential year 210x (2106?) problem
    - SYSTIML+SYSTIMH+SYSTIMTM is the IEEE 1588 time format
    - It is required for one step clock operation
    - It makes software implementations easy
  - It is an arithmetic clock, not a counter
    - The operation of the clock is fractional, defined by the TIMINCA and TIMADJ registers

# i210 features, NIC clock 2.

- It is an arithmetic clock, not a counter
  - TIMINCA (frequency compensation)
    - $SYSTIM = SYSTIM + 8 \text{ nsec} +/- TIMINCA.Incvalue * 2^{-32} \text{ nsec.}$
    - 0-30 bits: Increment value (Incvalue). Value to be added or subtracted (depending on ISGN value) from 8 nS clock cycle in resolution of  $2^{-32} \text{ nS}$
    - 31: Increment sign (ISGN).
      - Value of 0b = Each 8 nS cycle add to SYSTIM a value of  $8 \text{ nS} + Incvalue * 2^{-32} \text{ nS.}$
      - Value of 1b = Each 8 nS cycle add to SYSTIM a value of  $8 \text{ nS} - Incvalue * 2^{-32} \text{ nS.}$
  - TIMADJ (monotonic offset compensation)
    - $SYSTIM = SYSTIM + 8\text{ns} +/- 1 \text{ nsec}$  as long as  $TIMADJ > 0$ 
      - $TIMADJ = TIMADJ - 1$
    - 0-29 bits: Time Adjustment Value (defined in ns units). The TADJL field can be set to any non-zero value smaller than 999,999,900 decimal (slightly below 1 second)
    - 30 bit: Always zero
    - 31 bit: Sign, 0b is addition, 1b is subtraction

# i210 features, RX timestamping

- Packet filter
  - Select incoming packets to be timestamped
    - IEEE 1588 or other protocol specific
      - IEEE 1588 V1 or V2 capture
      - Ethernet type in case of L2 transport
      - UDP port based case of L4 transport
      - RX queue must be specified (where the received packet is transported)
    - All receive packets are timestamped
      - Only descriptor based timestamp communication is supported
  - The packet must be received at least partially (Ethernet type or UDP port) to timestamp based filtering
    - There is a capture latency compared to the Message Timestamp Point (First octet after SoF on the wire)
    - The actual minimum, maximum and average latency can be found in the datasheet
      - There is a statistical variation in the capture process...
- RX timestamp communication to SW
  - Advanced Receive Data Descriptor structure pushed into host memory (TS flag) with the timestamp included
    - SYSTIML and SYSTIMH are put into the packet descriptor
  - The timestamp is read from RXSTMPL/H register (TSIP flag)
    - Interrupt may be requested on RX timestamp

# i210 features, TX timestamping

- TX timestamp is problematic:
  - Packet and timestamp info handled differently
    - Packet goes out on the port
    - The timestamp needs to be processed by local software
  - One step clock (HW packet rewrite) does not help also to simplify things...
- It is not advised to timestamp all TX packets as possible for RX (lot of communication)
  - Only the packets requested by software are timestamped
  - One step clock Sync:
    - Software can request timestamping by setting bit Advanced Transmit Data Descriptor bit 1STEP\_1588
    - Timestamp is inserted into the packet matching the filter with a given offset, Ethernet CRC recalculated
    - IPv4 UDP is supported with no UDP checksum
    - IPv4 with UDP checksum and IPv6 (mandatory UDP checksum) are not supported
  - Two step clock Sync and all Delay\_Req packets:
    - Software can request timestamping by setting Advanced Transmit Data Descriptor bit 2STEP\_1588
    - The transmit timestamp can be read by SW from TXSTMP
    - The software can send the proper IEEE 1588 message using any transport method

# i210 features, SDP timestamping

- SDP pins configured
- “CAPTURE” peripheral on microcontrollers
- 2 Auxiliary timestamp registers (AUXSTMP)
  - Latches in SYSTIMH and SYSTIML on specified SDP pin change
    - The actual time can be recorded for external events based on the local SYSTIM clock
    - Can be used for master clock reference input, local clock accuracy measurement (compared to reference clock), etc.
  - Interrupt can be requested upon a write to AUXSTMP

# i210 features, SDP as output

- External events based on SYSTIM
  - 2 TRGTTIM registers (TRGTTIM0 and TRGTTIM1)
  - “COMPARE” peripheral on microcontrollers
  - If the value of SYSTIM reached the value of TRGTTIM state change on a configured SDP pin can be requested
  - An interrupt can be also requested
  - Time based events for external components
- External clocks based on SYSTIM (phase locked)
  - 2 FREQOUT registers (FREQOUT0 and FREQOUT1)
  - The output frequency must be an integer multiply of 1 Hz
    - The requirements are somewhat relaxed if the half cycle time of the clock is under 70ms
  - The HW uses TRGTTIM internally (by auto-incrementing it)

# i210 features, Interrupt on time

- Time SYNC Interrupts
  - SYSTIM Wrap around
  - Receive timestamp on timestamp loaded into RXSTMP
  - Transmit timestamp on timestamp loaded into TXSTMP
  - Target Timer 0 trigger
  - Target Timer 1 trigger
  - Auxiliary Timestamp 0 taken
  - Auxiliary Timestamp 1 taken
  - Time adjust is done (TIMADJ)

# ACKNOWLEDGMENT

- Intel Corporation has supported this work.