Precision Time Synchronization in Data Centers

Research Scientist, Meta
Ahmad Byagowi

Why to have precision time synchronization in Data Centers?

How to have precision time synchronization in Data Centers?

Background

- Speed of light and speed of electricity are finite!
  - Speed of Light is slower in fiber (2.14 x 10^8m/s) than in vacuum (2.99 x 10^8m/s)
  - Latency in data transfer over the network
- Hyperscale cloud services serve people across the globe
  - Geographical distribution of customers
  - Necessity of distribution due to safety, robustness and regulations
- The demand for hardware resources is always increasing
  - Increase in users, data and multi dimensional contents
  - Interactivity with the data like multi-user and VR
  - Big data and AI

Solution to Address the Hyperscale Demand

- Horizontal Scaling
  - Expansion of resources

- Geographical expansion
  - Reduction of latency

Challenges with Expansion

- Distributed Systems
  - Consistency vs Availability
  - Propagation of Information
  - Need for Redundancy
- Distribution of Clock
  - A common reference between all machines
  - Latency vs Clock Skewness

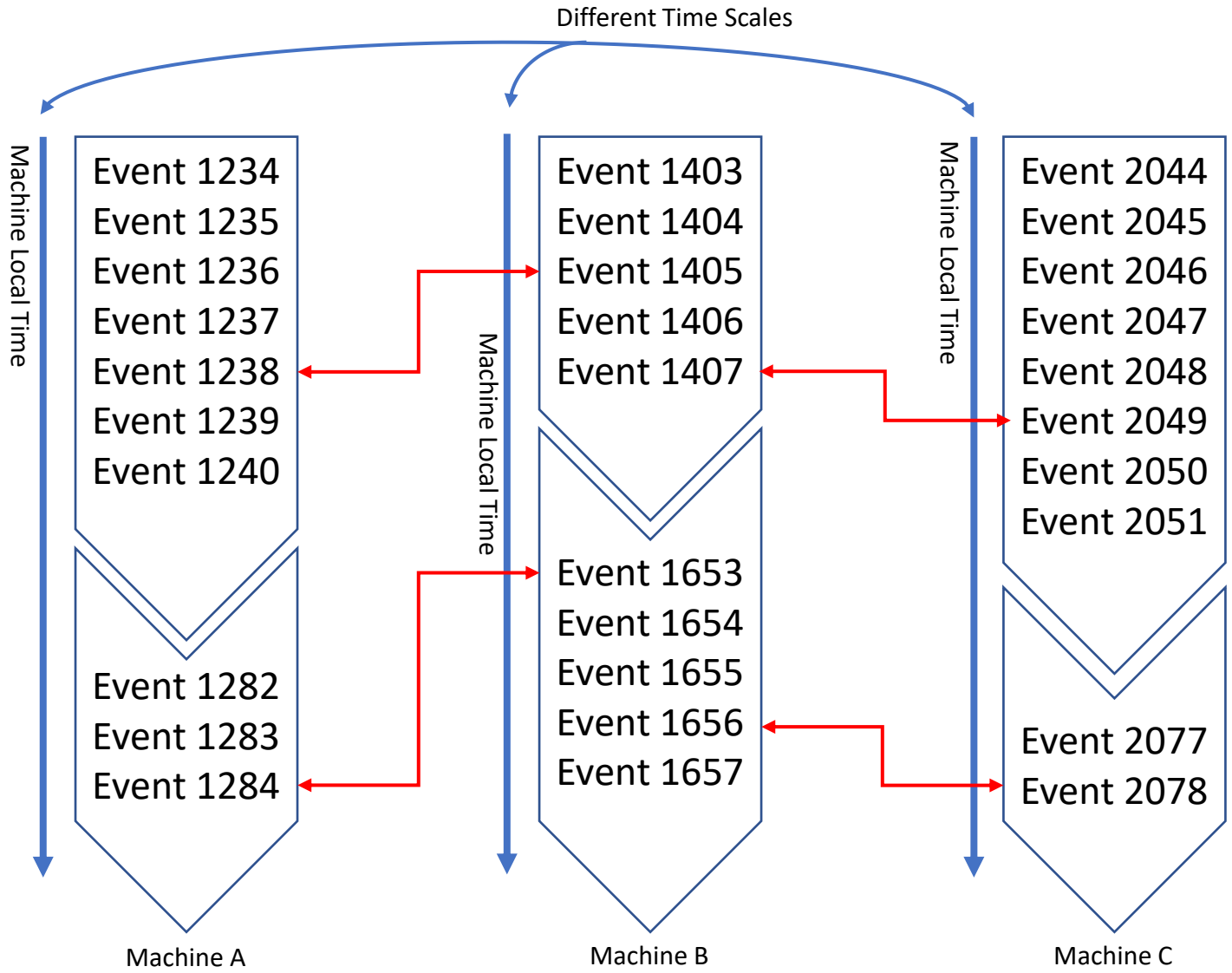
Categories of use cases

Use cases of Precision Time Sync in Distributed Systems:

- Synchronization (Phase)
  - Active (1)
    - Running Events at specific time
      - Sync or desync
  - Reactive (2)
    - Measure latency, time or intervals
- Syntonization (Frequency)
  - Active (3)
    - Calibrate speed and align runtime to reduce tail latency
  - Reactive (4)
    - Measure heterogeneity or provide binning

	Phase	Frequency
Active	1	2
Reactive	3	4

Associate Events Between Multiple Machines



Precision Requirement

- CPU level
- OS (Kernel) level
- Distributed System level

Precision Requirement at CPU level

- Nyquist sampling theorem
  - Sampling interval required to avoid aliasing
  - Sampling frequency should be at least twice the highest frequency contained in the signal
- Frequency in event occurrence
  - Instruction Latency
  - Instruction Throughput

// mov = 1 CPU cycle

// xchg = 3 CPU cycles

// rdtsc = 1 CPU cycle

A CPU with a clock speed of 3.2 GHz executes 3.2 billion cycles per second
That is a period of about 310ps

Precision Requirement at OS level

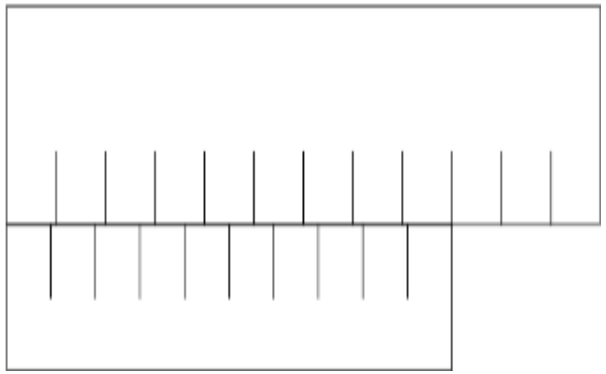
- dmesg

```
[52603.373642] {38}[Hardware Error]: event severity: corrected
[52603.373643] {38}[Hardware Error]:  Error 0, type: corrected
[52603.373644] {38}[Hardware Error]:   section_type: PCIe error
[52603.373644] {38}[Hardware Error]:   port_type: 4, root port
[52603.373645] {38}[Hardware Error]:   version: 3.0
[52603.373645] {38}[Hardware Error]:   command: 0x0547, status: 0x0010
[52603.373646] {38}[Hardware Error]:   device_id: 0000:b7:01.0
[52603.373647] {38}[Hardware Error]:   slot: 255
[52603.373648] {38}[Hardware Error]:   secondary_bus: 0xb8
[52603.373648] {38}[Hardware Error]:   vendor_id: 0x8086, device_id: 0x352a
[52603.373649] {38}[Hardware Error]:   class_code: 060400
[52603.373649] {38}[Hardware Error]:   bridge: secondary_status: 0x0000, control: 0x0013
```

System Logging is based on clock_boottime (clock_minotone_RAW) with a quanta on 1us
Events occur faster than the quanta of 1us (aliasing)

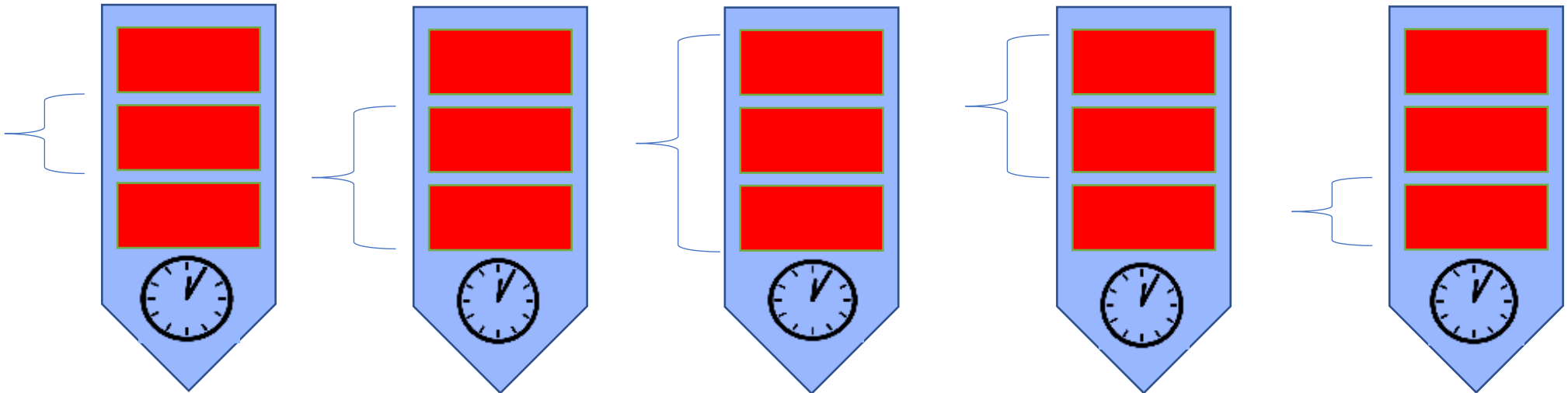
Challenges and the Precision Requirement

- Vernier acuity

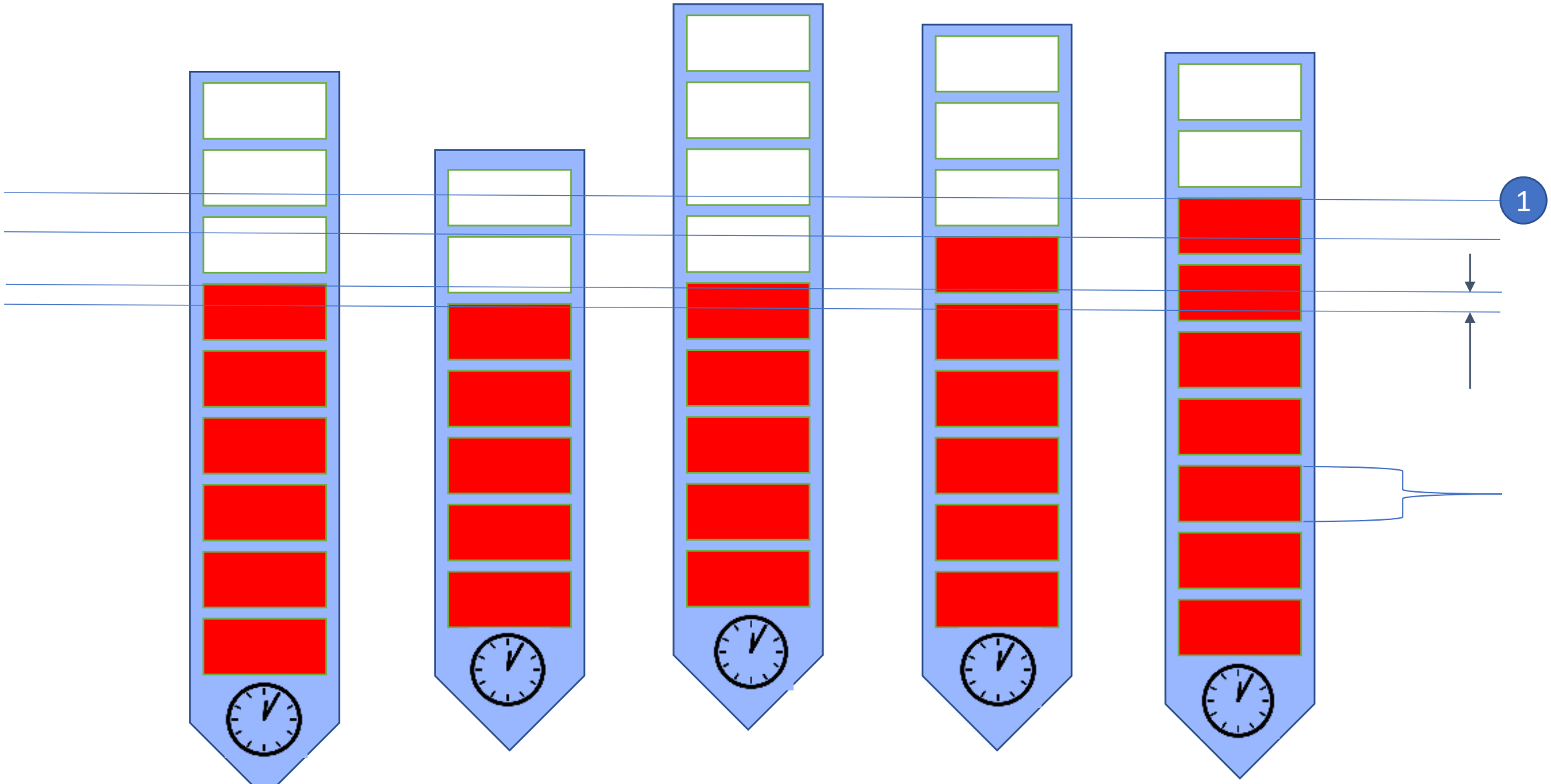


- Compounding of Events

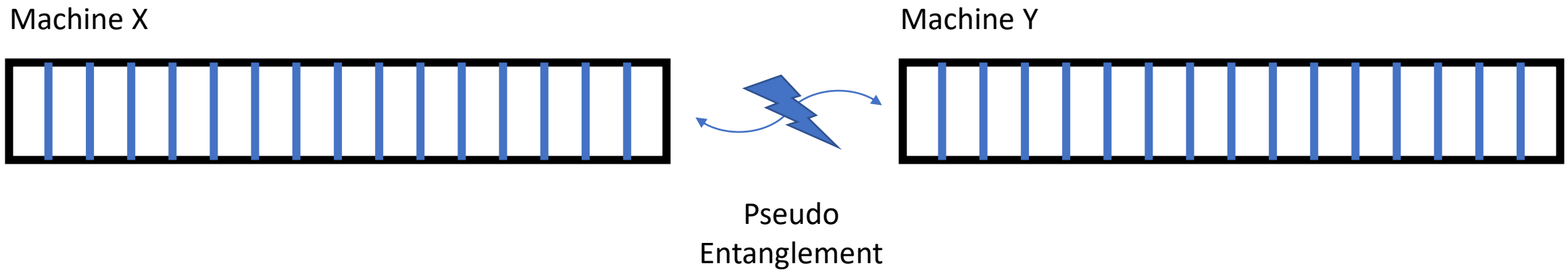
$$1^n = 1$$



Precision Requirement for Distributed Systems



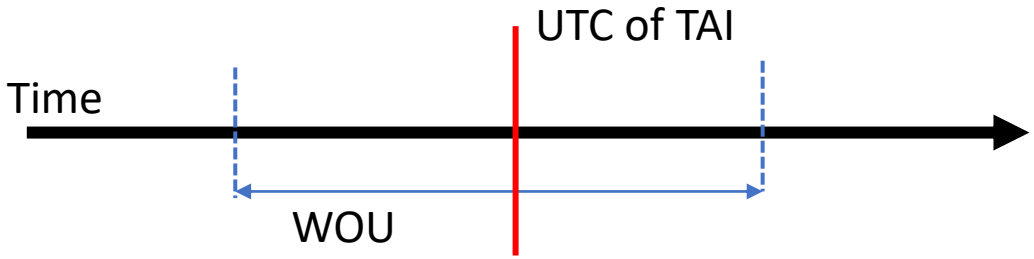
What comes out of Precision Time Sync?



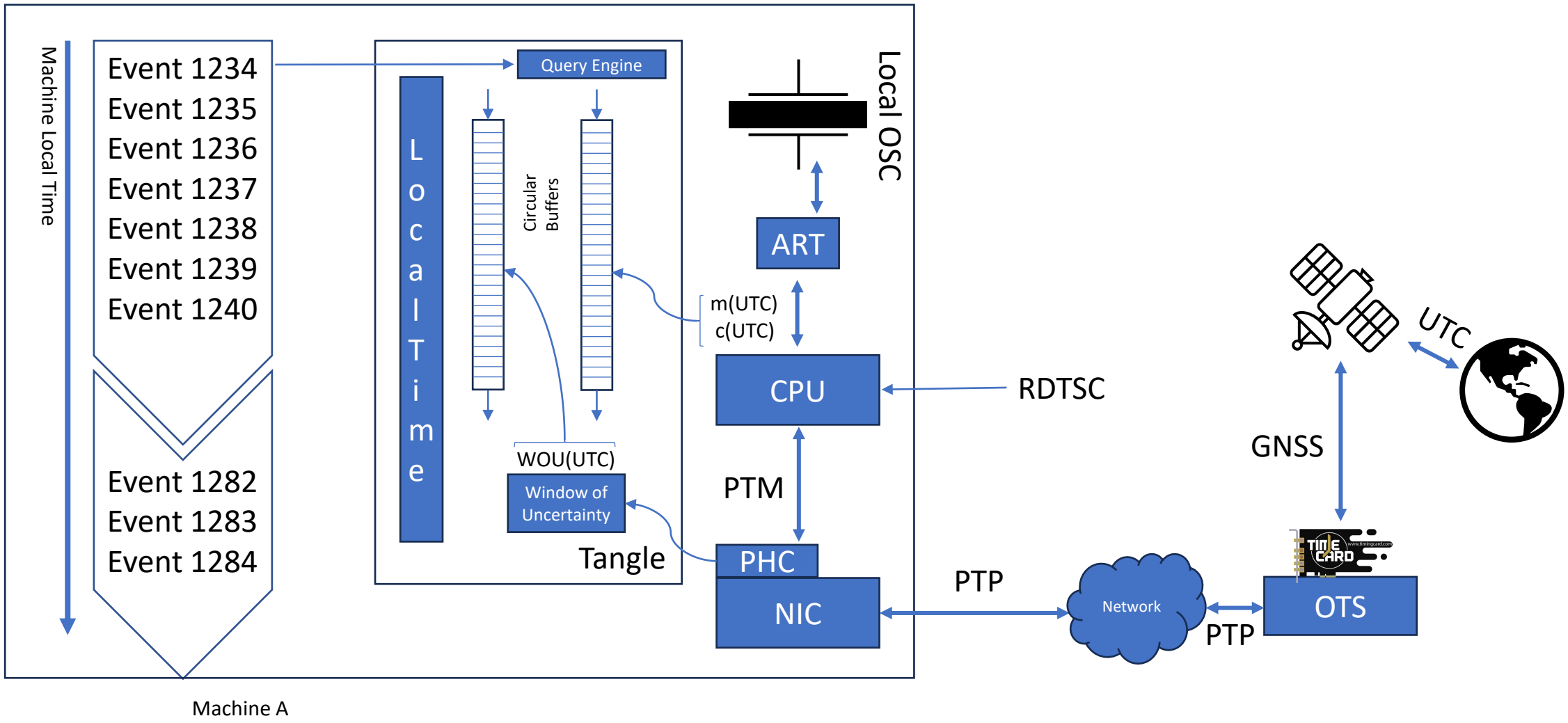
Machine X and Y are any two machines across the globe or inside a local network

Pseudo Entanglement: Probabilistic Entanglement of two Registers (Machine Y and Y) within the Windows of Uncertainty

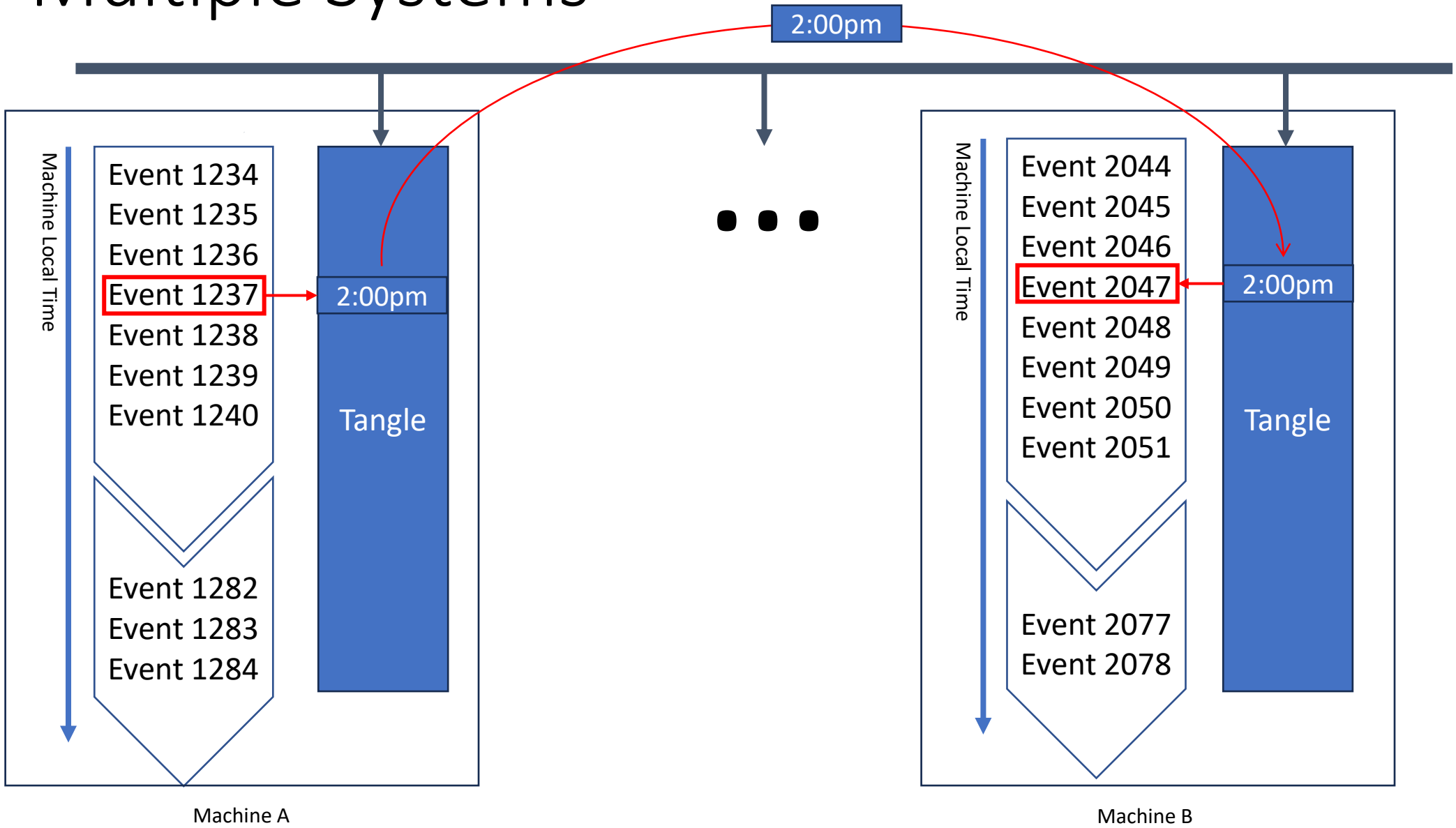
Window of Uncertainty: An ongoing estimation of a time interval that UTC (or TAI) sits inside it (with a given probability)



Tangle


Machine Local Time
Event 1234
Event 1235
Event 1236
Event 1237
Event 1238
Event 1239
Event 1240
Event 1282
Event 1283
Event 1284
Query Engine
Local Time
Circular Buffers
WOU(UTC)
Window of Uncertainty
Tangle
Local OSC
ART
m(UTC)
c(UTC)
CPU
RDTSC
PTM
PHC
NIC
PTP
Network
PTP
OTS
GNSS
UTC
Machine A

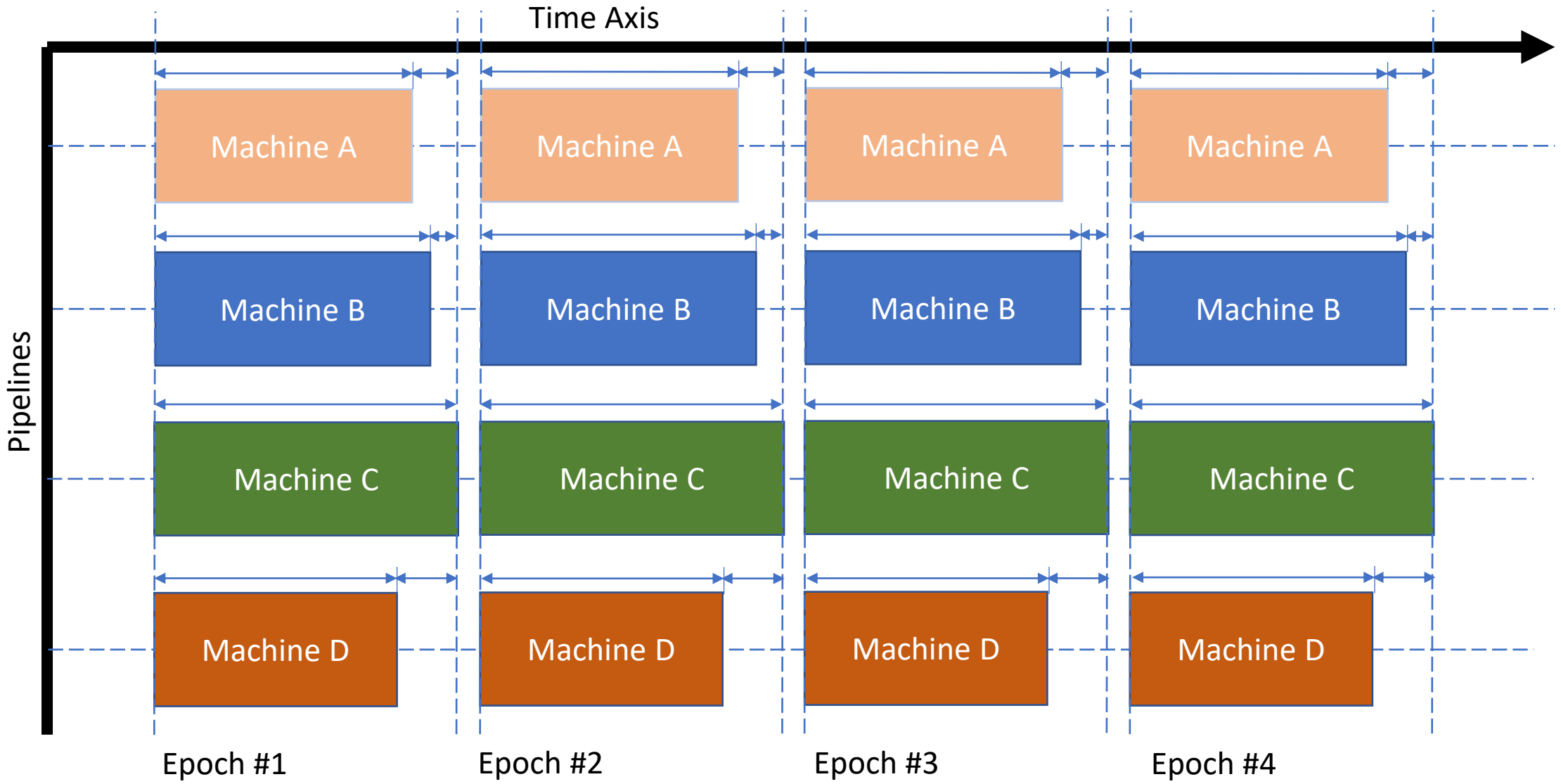
Multiple Systems



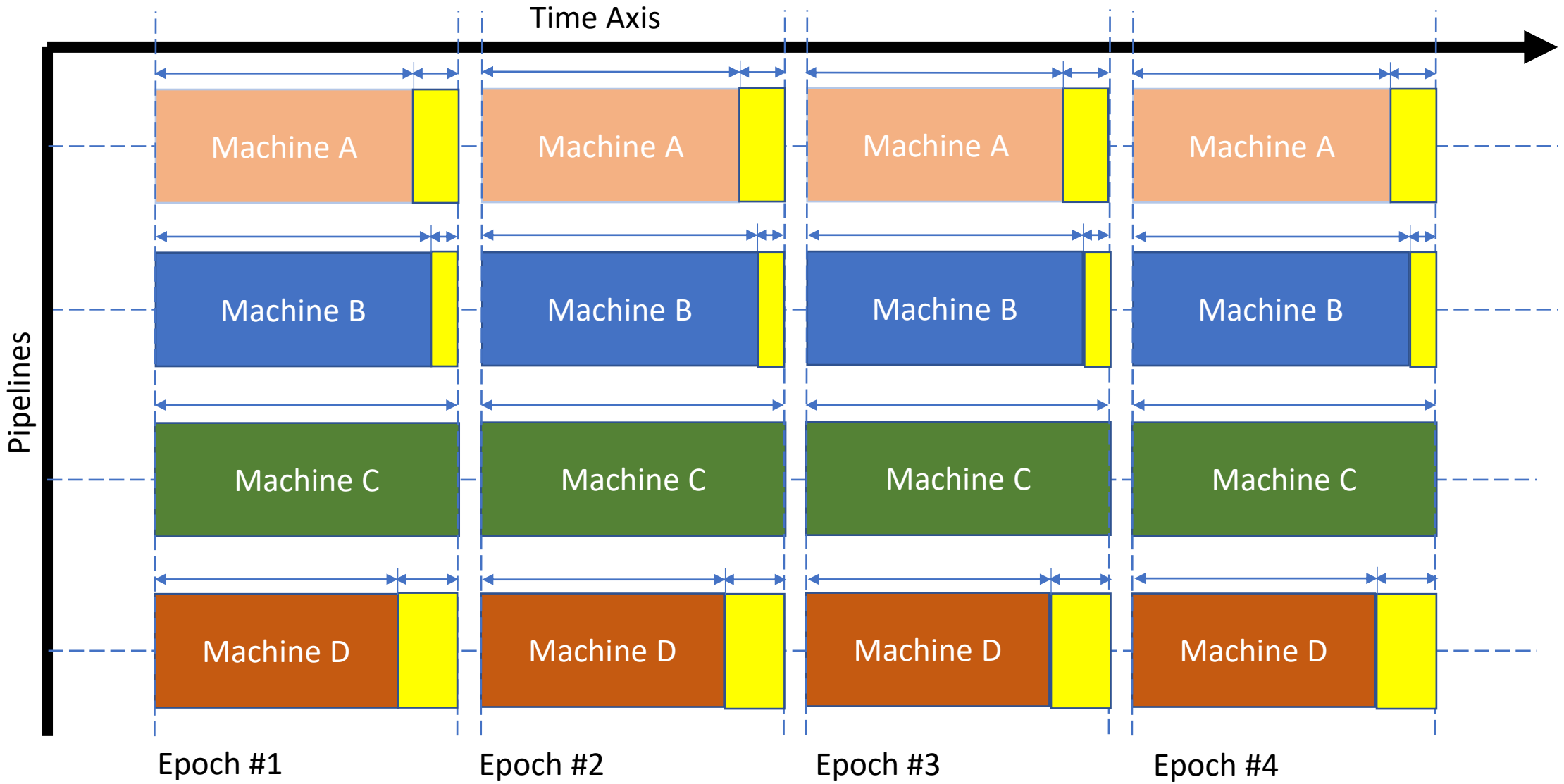
Functions

- Identify concurrent event in another machine[s]
- Find the timestamp of an event in another machine[s]
- Chronologically Rank a given event across machines
- Measure the one-way-latency between machines
- Identify concurrent events with one-way-latency consideration
- Trace chronological order for sequence of events
- Benchmark machines by precise runtime measurement
- Directly utilize RDTSC for maximum precision in event timestamping

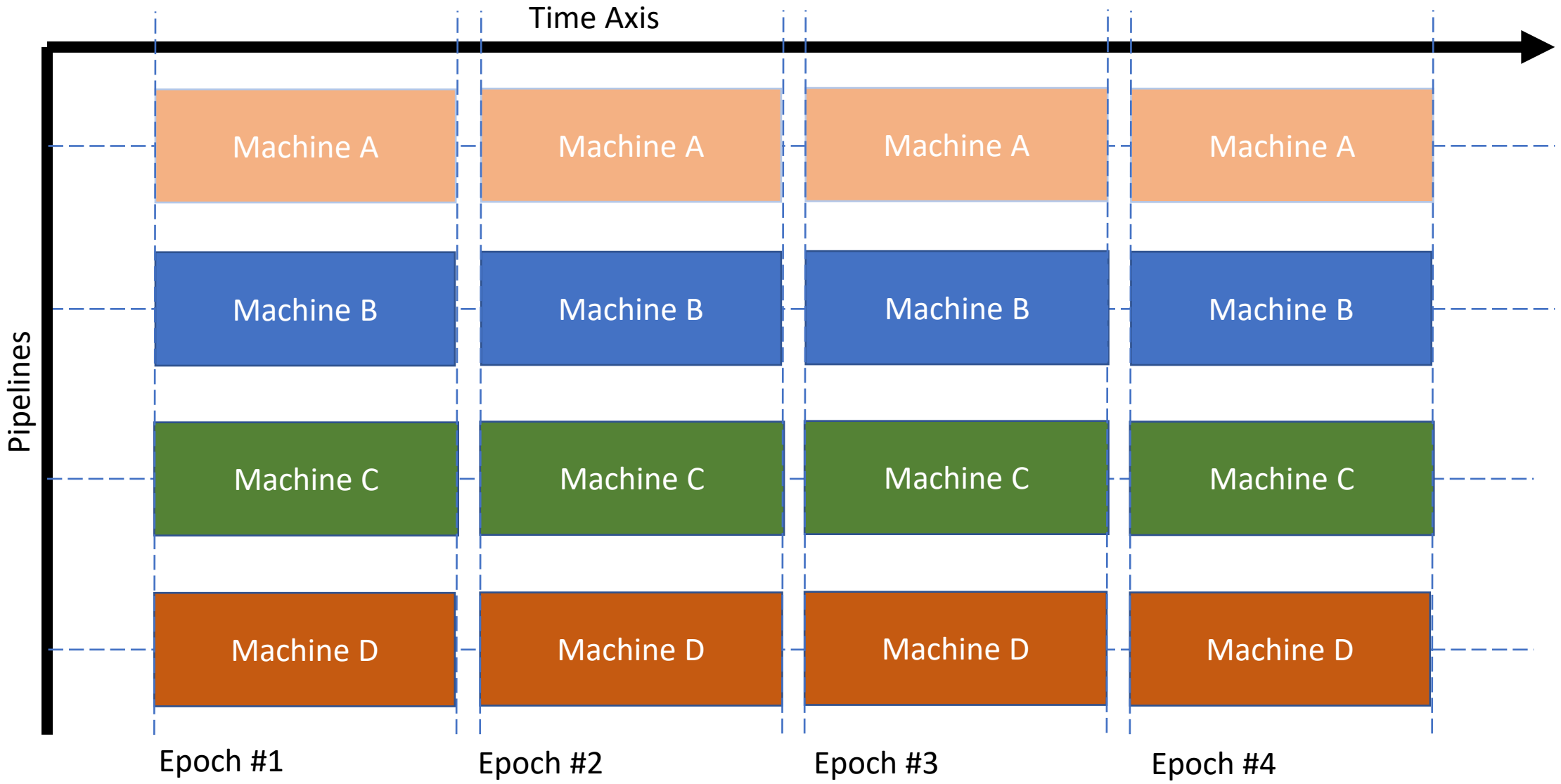
Runtime Difference in a Pipeline



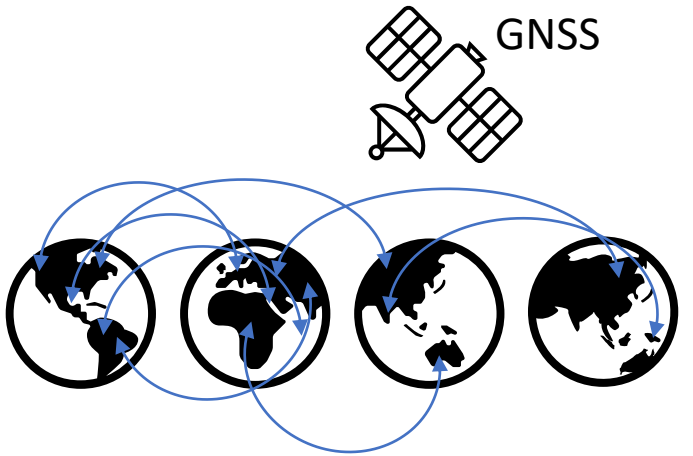
Runtime Difference in a Pipeline



Align and Calibrate Machines in a Pipeline

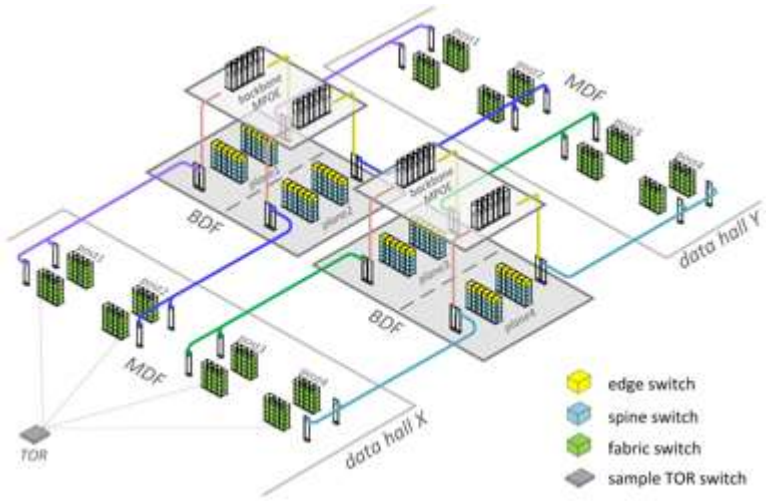


Precision Time Sync Across Different Domains



IEEE P3335

Global



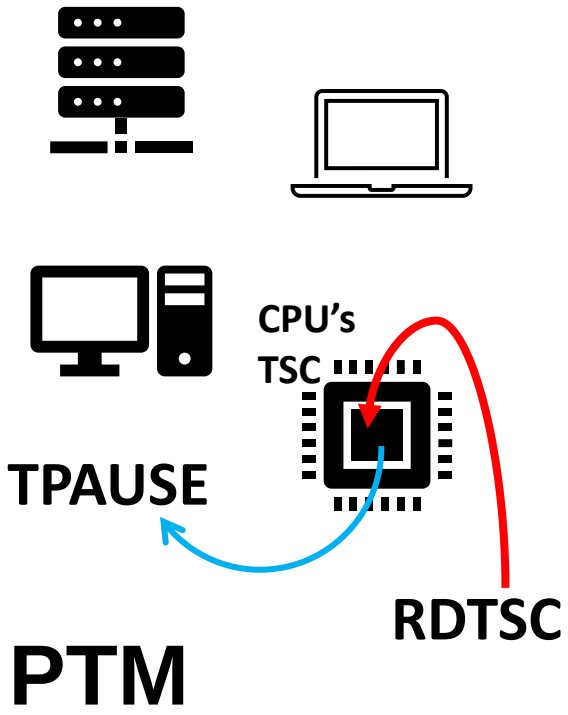
PTP

Data Center IEEE 1588

Local Network

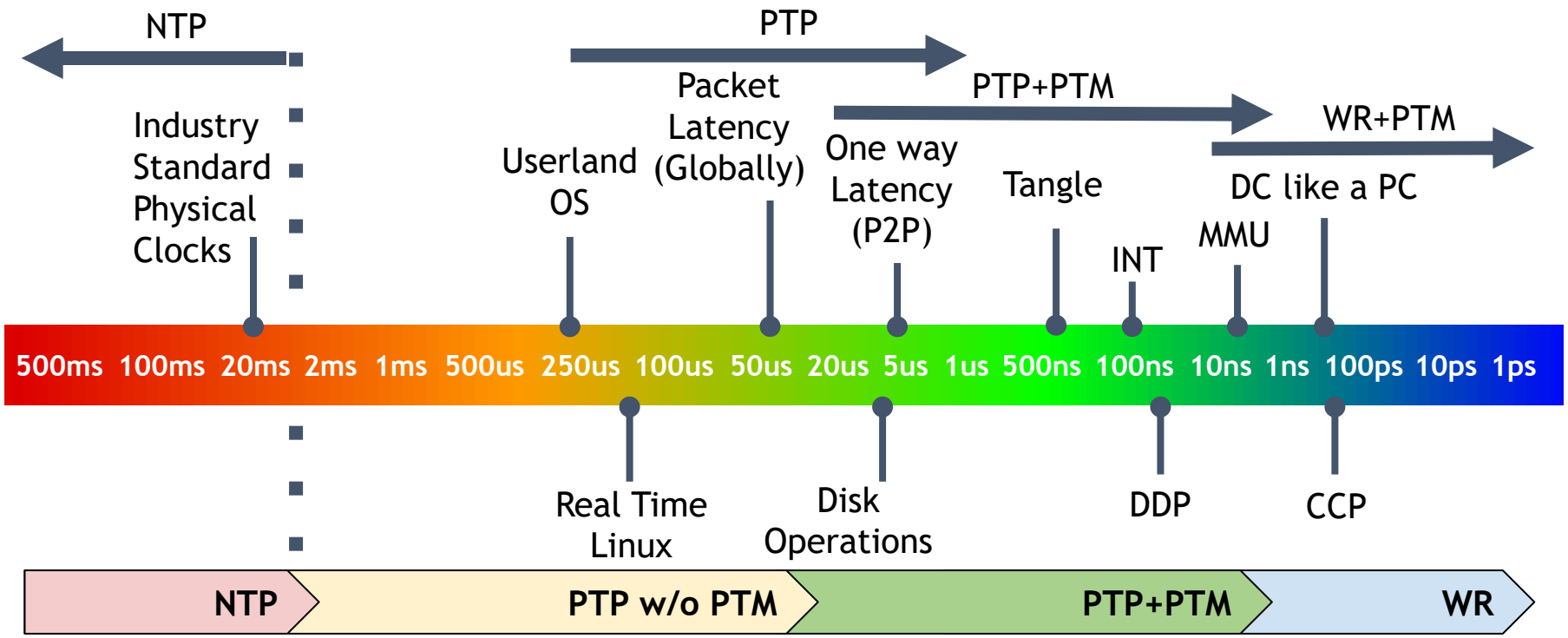
RDTSC — Read Time-Stamp Counter

Opcode*	Instruction	Op/En	64-Bit Mode	Compat/Leg Mode	Description
0F 31	RDTSC	ZO	Valid	Valid	Read time-stamp counter into EDX:EAX.



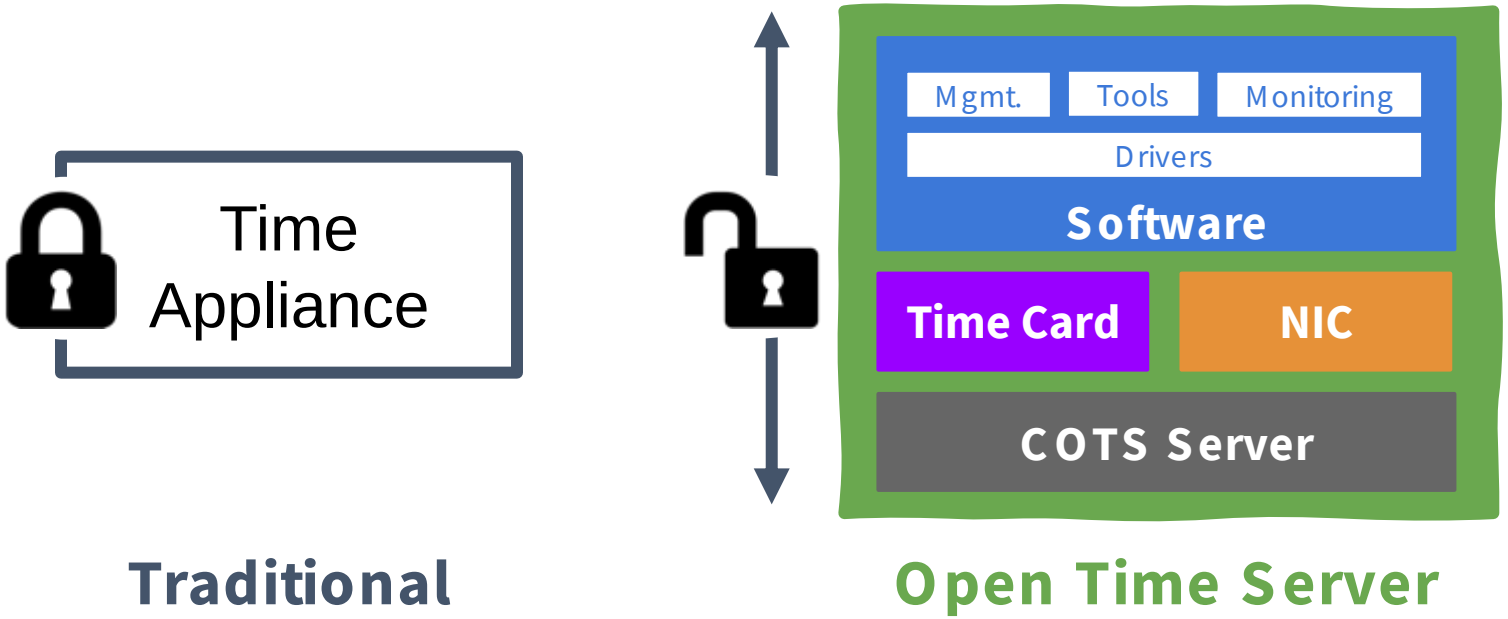
Inside Machines

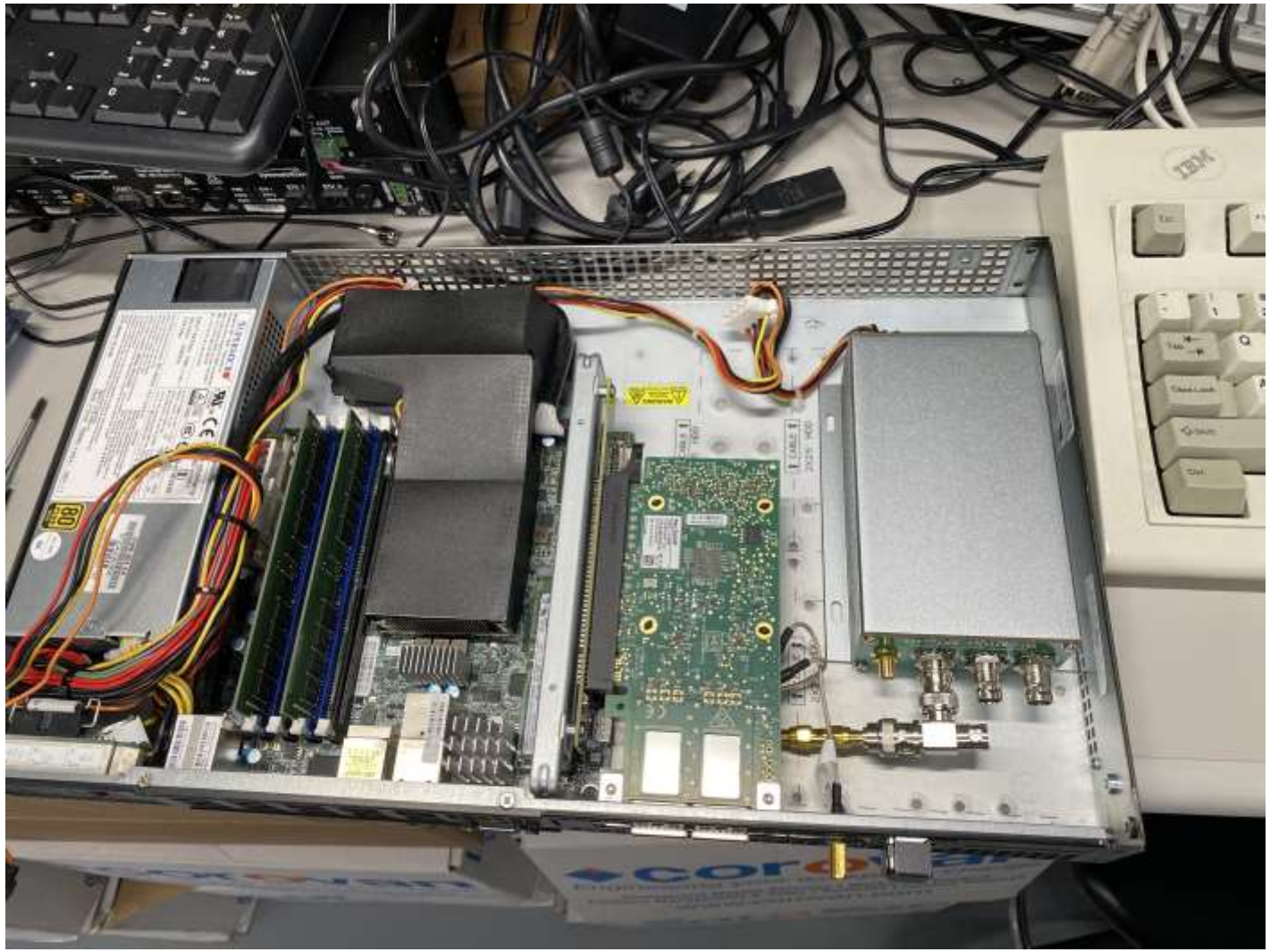
Time Precision and Applications



Open Time Server

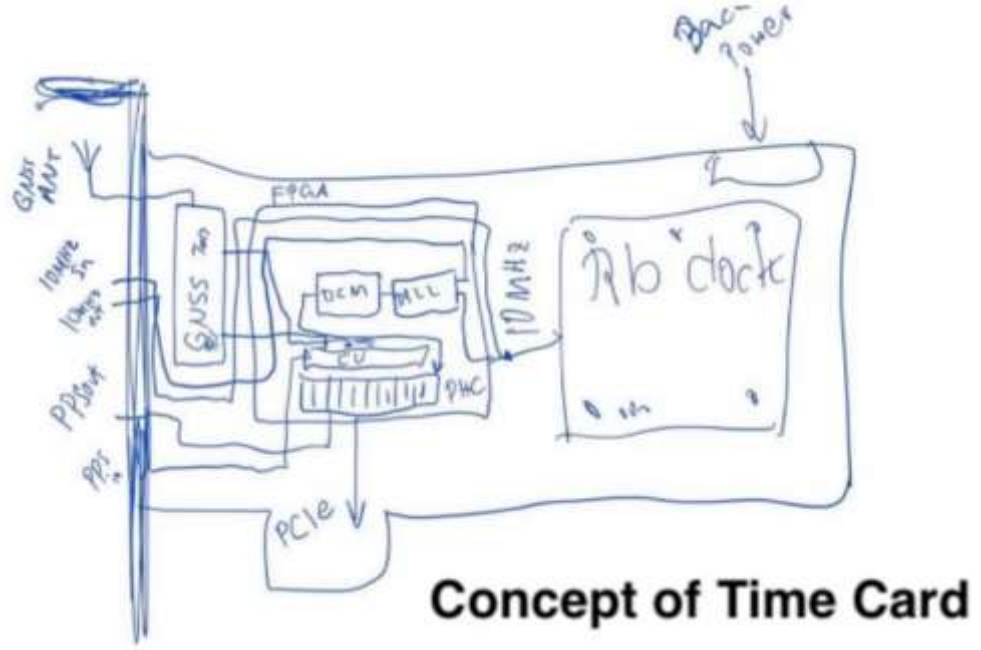
How to sync a Datacenter?



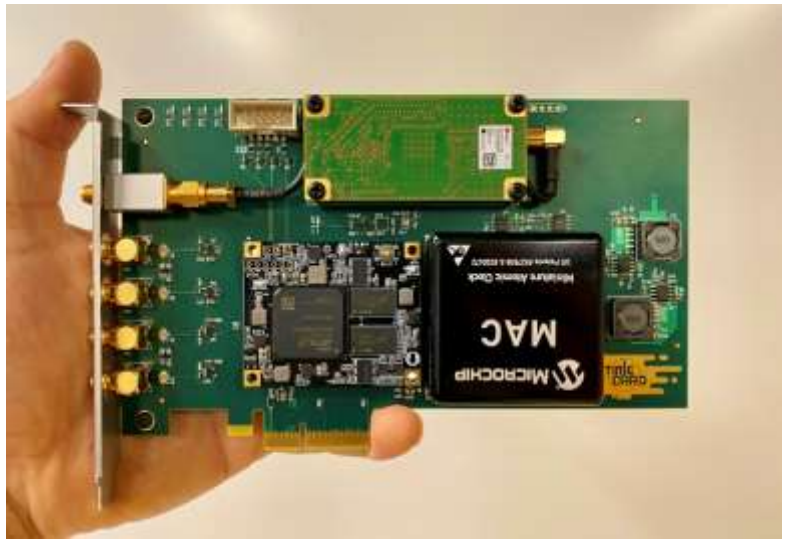


Time Card

- Concept (2020)
- First Prototype (2021)
- Industry Adoption (2022)



Concept of Time Card



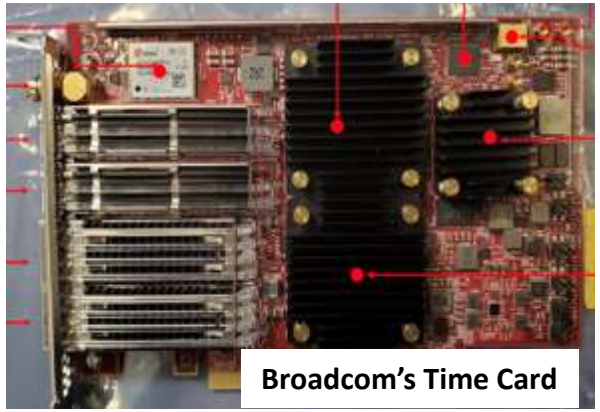
Time Card Family

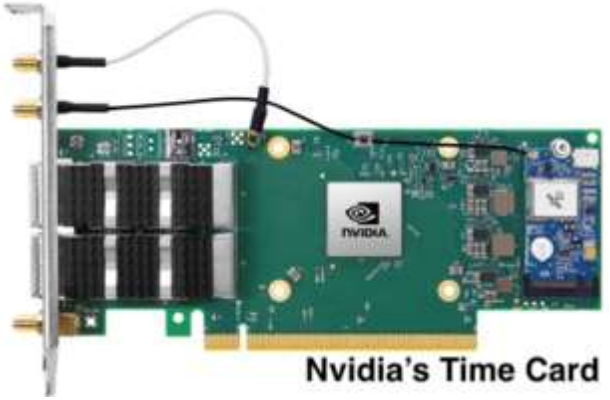

Orolia's Time Card


ADVA's Time Card

Celestica's Time Card


Intel's Time Card


Broadcom's Time Card



Nvidia's Time Card

Latest Time Card

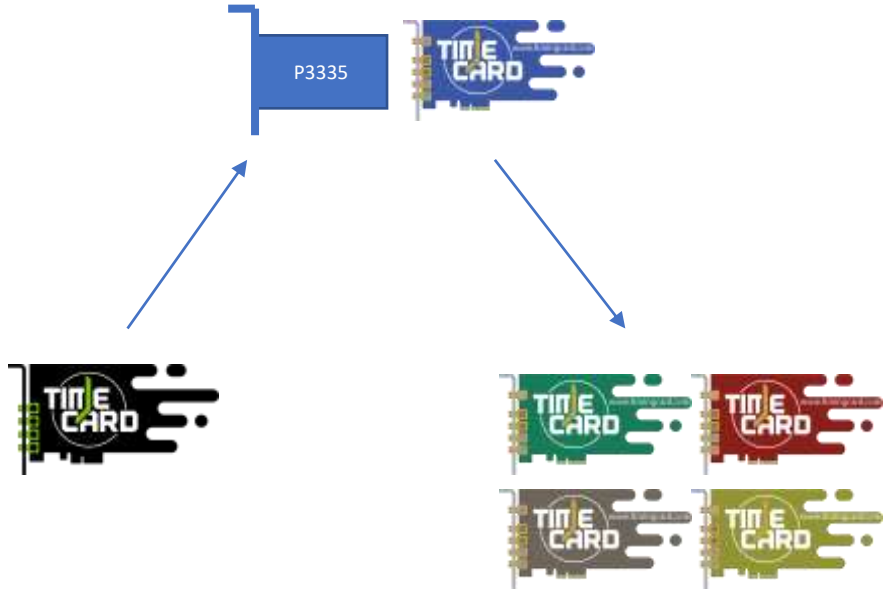
Simplified Objectives for P3335

Produce a guide to improve interoperability, compatibility and comparability between Time Cards manufactured by different vendors.

- Set a structure to define various sub systems
- Set some sub standards for plug-in modules (like GNSS and OSC)
- Define interfaces (hardware: memory addresses, drivers …)
- Suggest testing methods and comparable benchmark metrics
- Provide blueprints to assist further development of Time Cards
- Do all this work in a structure that maximizes people’s engagement

Proposed Approach

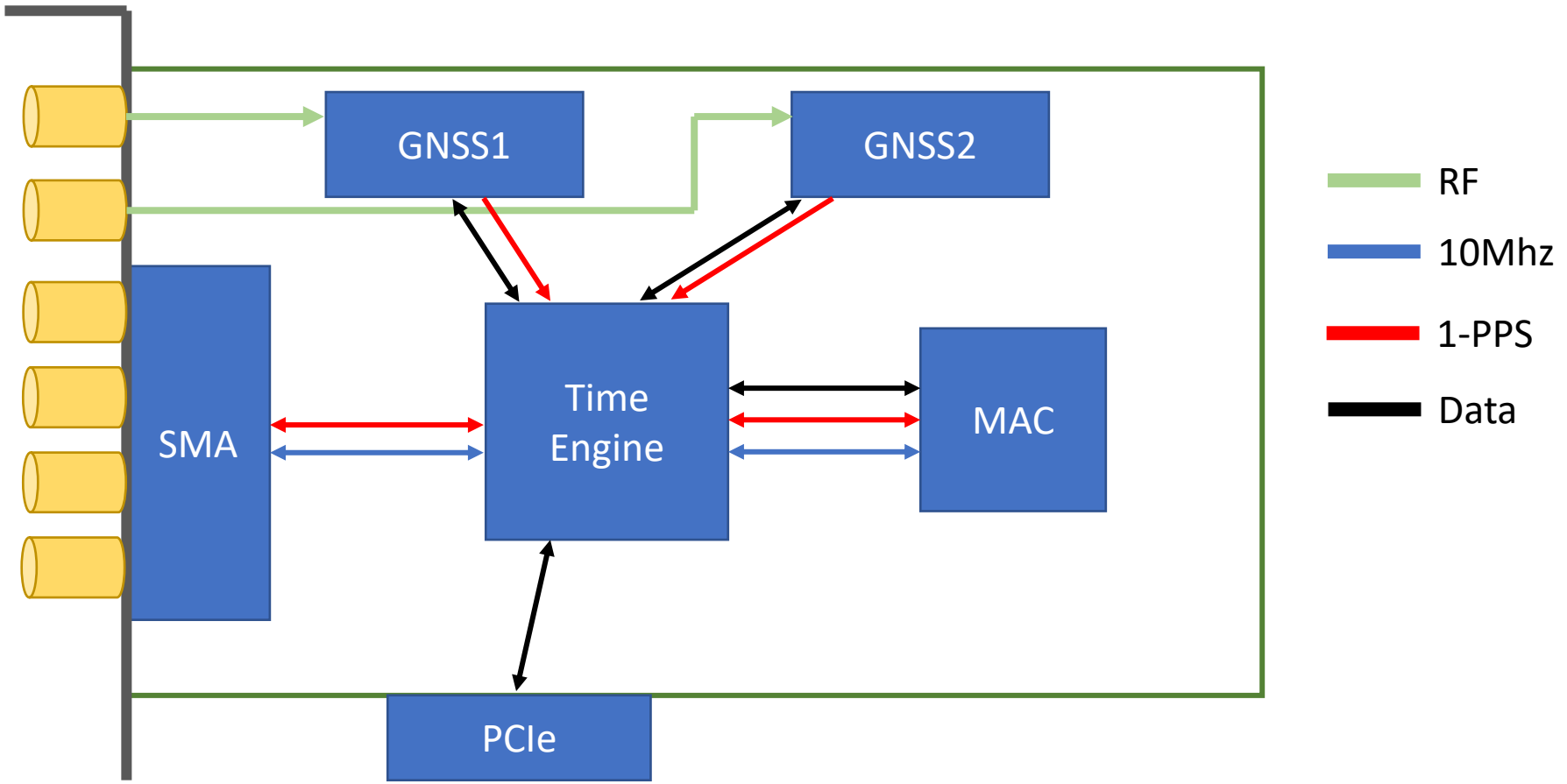
- We already have something that works (various Time Cards)
- Turn the Time Card to a general case
- Study the general case
- Dissect the work in subgroups (Diverge)
- Work in parallel
- Bring things together (Converge)
- Combing through the work (coherence)



Goal: Embrace Diversity, Ensure Interoperability

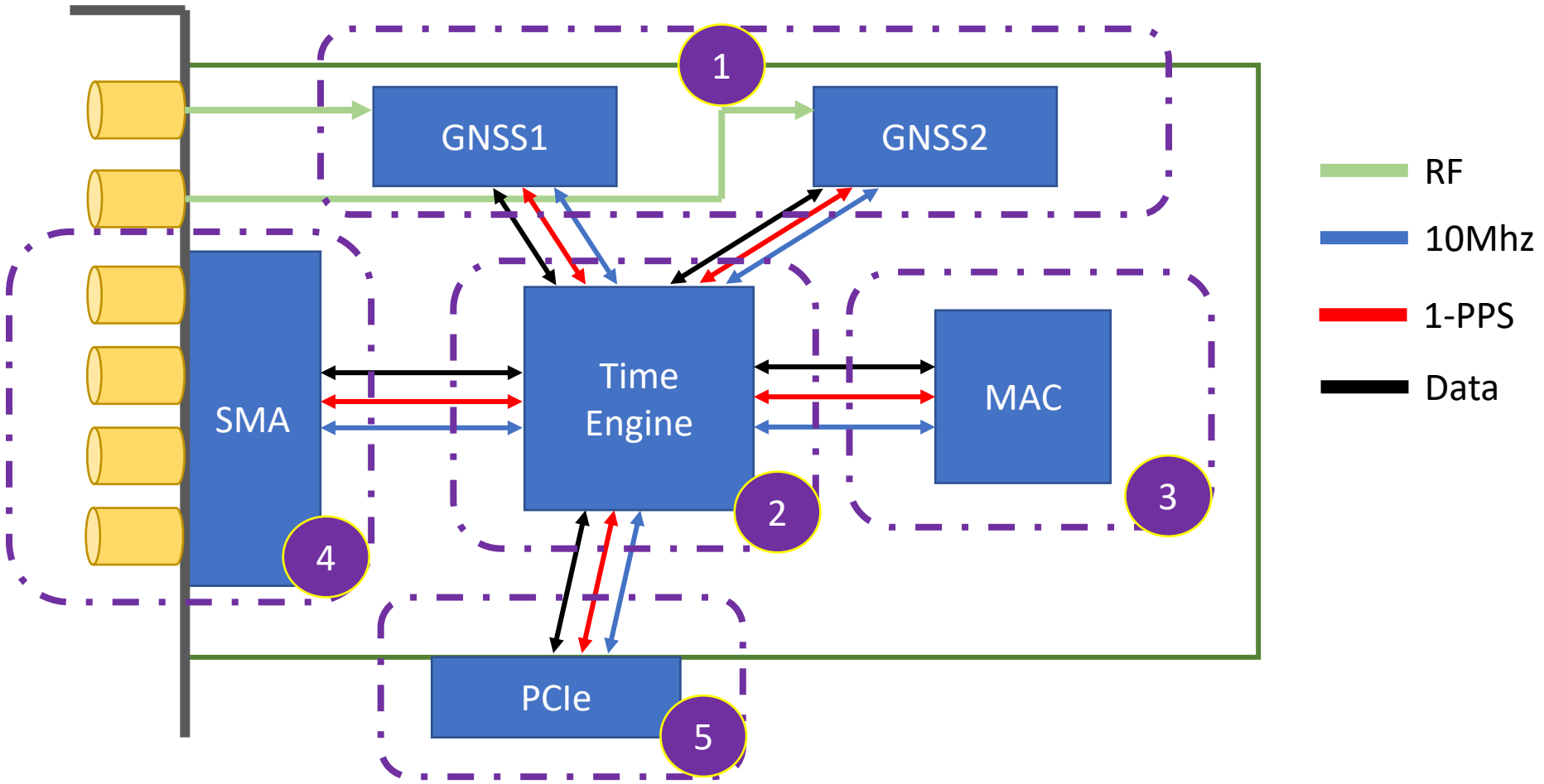
Current Time Card

- Dual GNSS PCIe card with a MAC

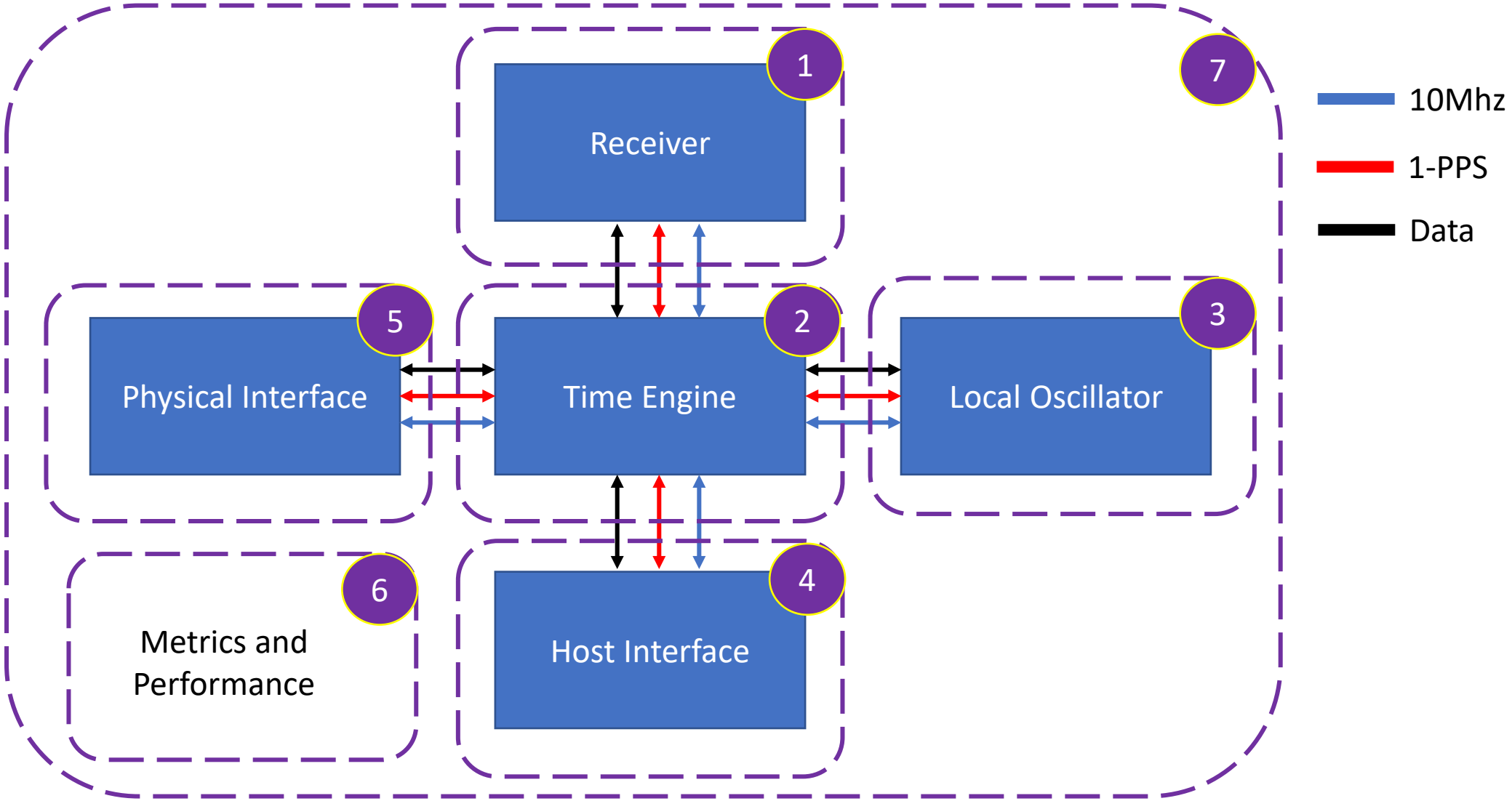


Time Card Concept

- All possible pathways



Focus Areas


1
Receiver
7
10Mhz
1-PPS
Data
5
Physical Interface
2
Time Engine
3
Local Oscillator
6
Metrics and
Performance
4
Host Interface

Thank you

Find out more on:

www.ocptap.com
www.timecard.ch
www.timingcard.com
www.opentimeserver.com