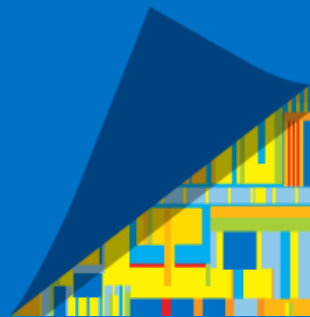
WSTS 2016
Virtualization of Time in Compute Systems

Kevin B. Stanton, Ph.D.
Sr. Principal Engineer
Intel Corporation



Time is anchored firmly in the physical world. Users of computing machines yearn for ever-increasing heights in abstraction. But Compute Virtualization need not be the enemy of time that scales both in accuracy and robustness of availability per application requirements. In this talk we provide an overview of the relevant virtualization models, describe current support for gaining immediate access to precision time in such systems, enumerate gaps, and propose an approach for addressing them.

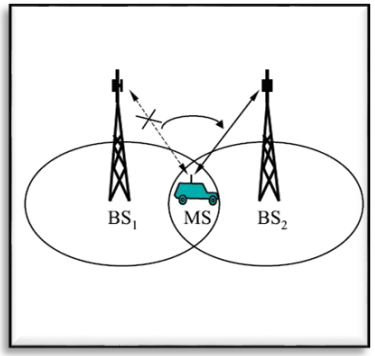
Need for Precision Timekeeping is Growing

Automotive

Conferencing

Realtime A/V

Industrial/
Energy



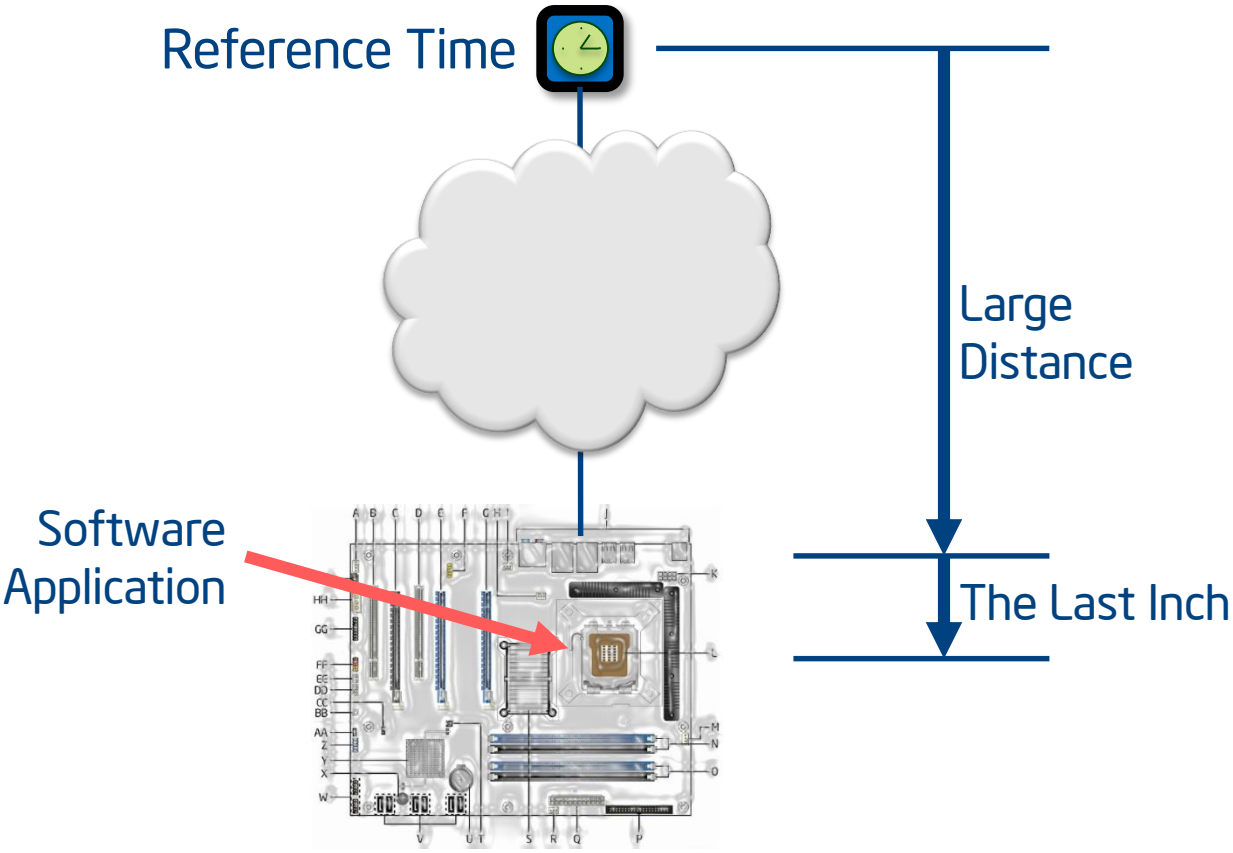
Cellular/Telco

Financial

Cloud/HPC

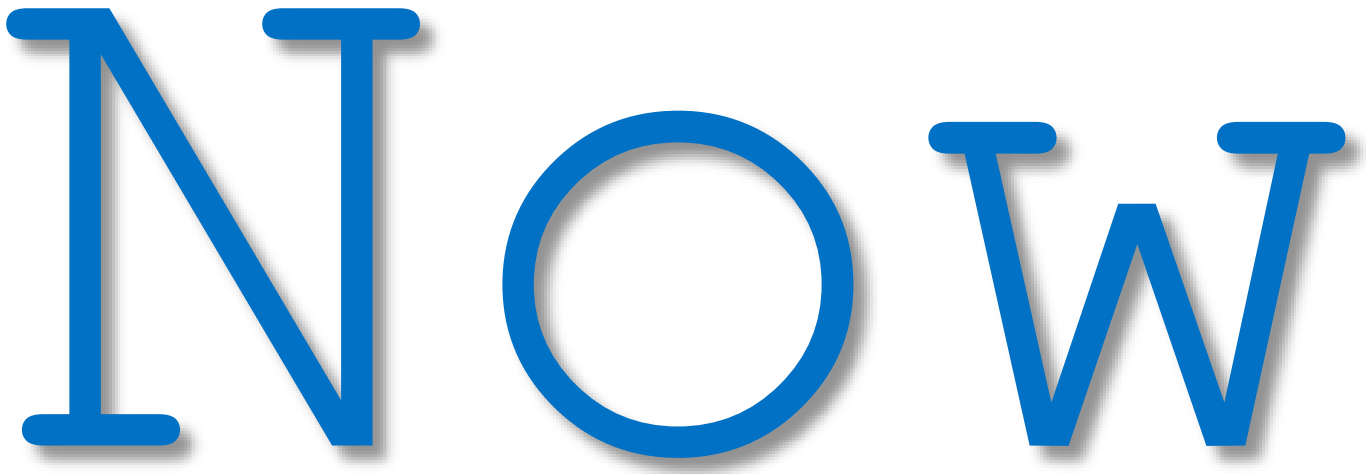
Some Apps Require UTC Traceability
…Some Do Not

The “Last Inch” Challenge



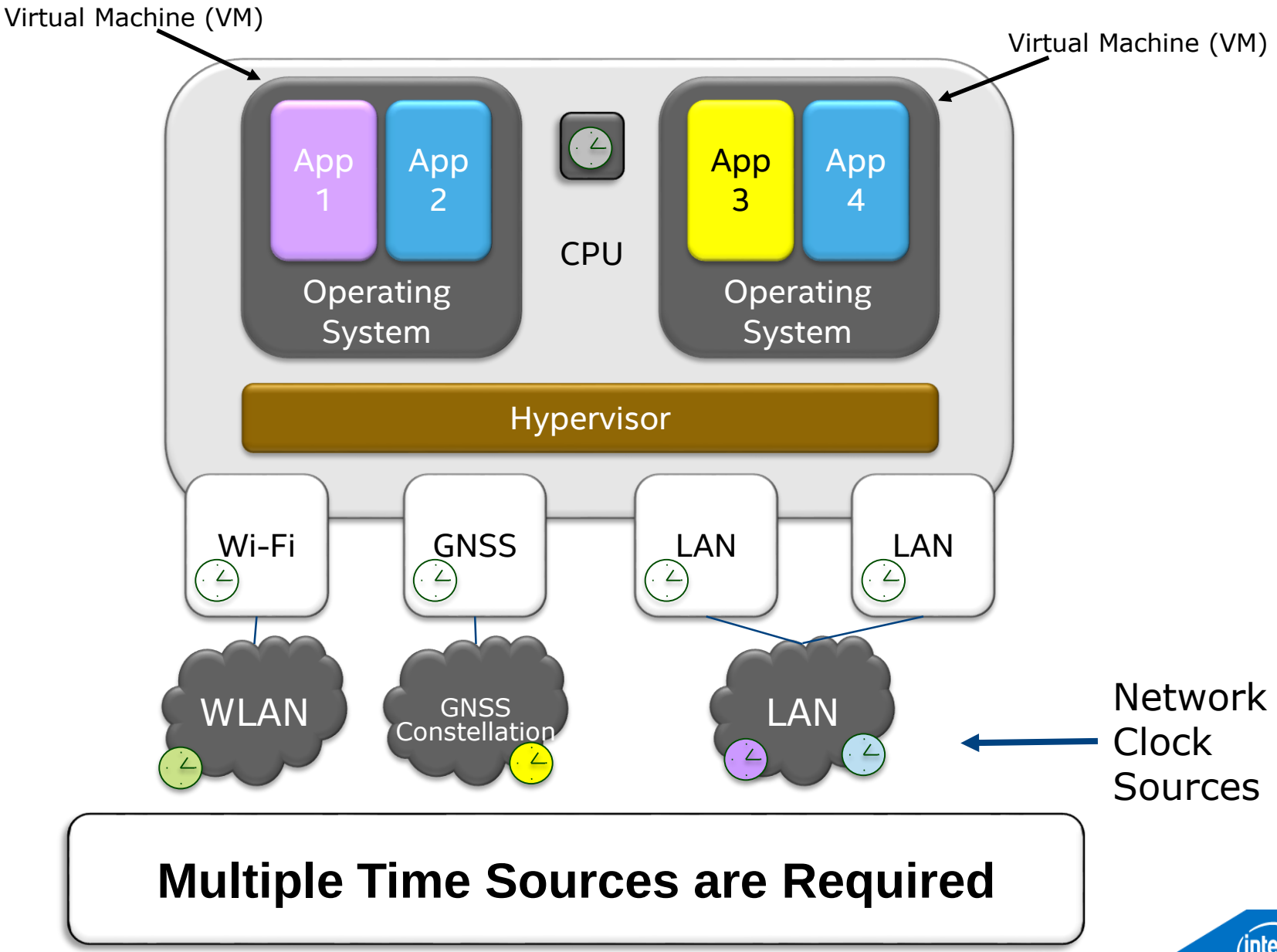
The term “The Last Inch” comes from *Timing in cyber-physical systems: The last inch problem* by John Eidson et al, ISPCS 2015 http://dx.doi.org/10.1109/ISPCS.2015.7324674

Application Software is Separated from Network Time by a Large Chasm

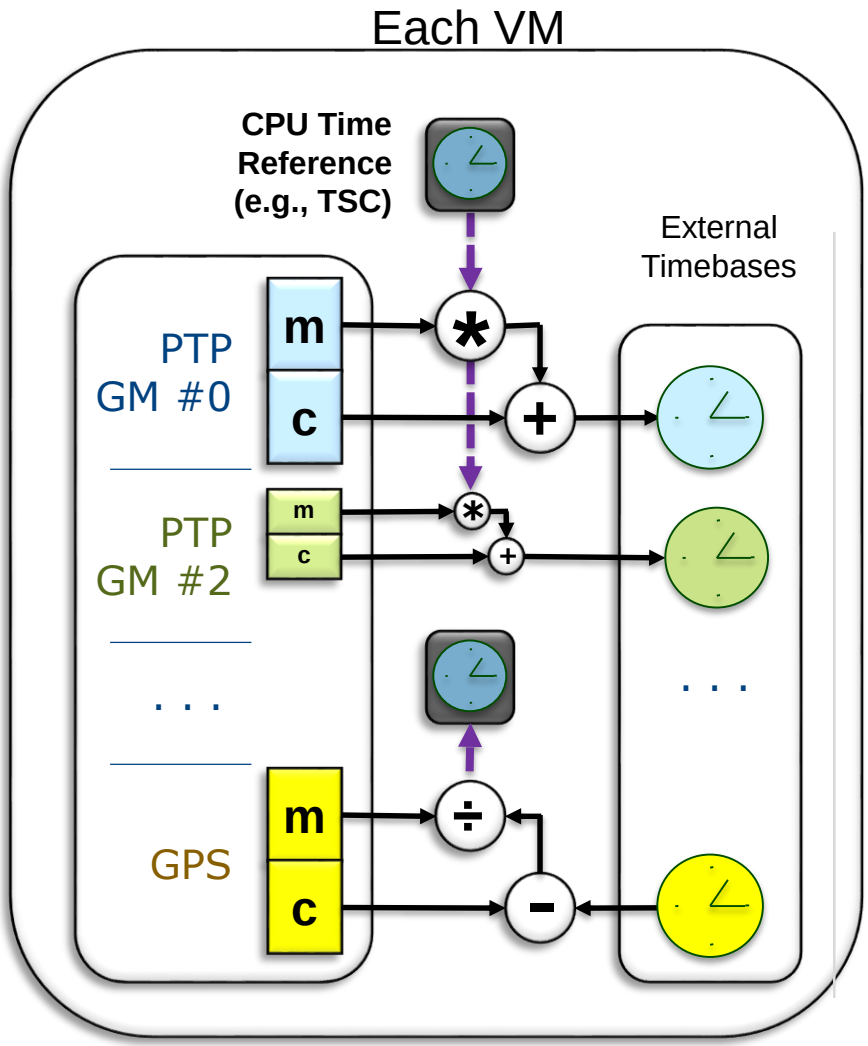


For software, NOW is never really “Now”
It’s always “Recently”

Modern Computer System



Scalable Timebase Representation

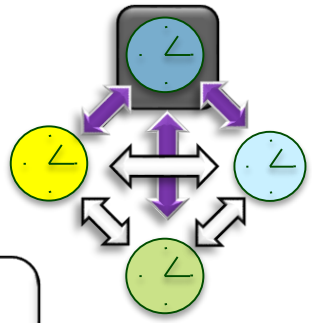


Linear transformation between CPU time and other arbitrary time via y=mx+c

Here's what's needed:

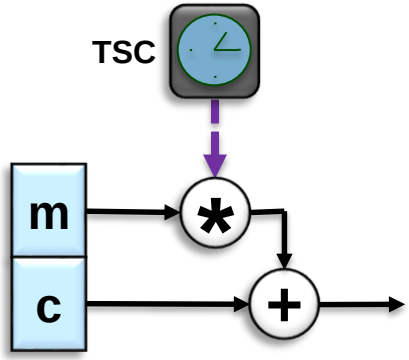
1. A Stable HW Reference
2. Fast * and + Ops
3. Precise estimate of m and c

➔ Any Timebase to/from Any Timebase



Virtualization Need Not Degrade Time

Software access to “Now”




```
clock_gettime(CLOCK_MONOTONIC_RAW,  &now);
```

- Returns current TSC value scaled to nominal nanoseconds

```
clock_gettime(CLOCK_MONOTONIC,      &now);
```

- Returns current TSC value scaled to track TAI, in nanoseconds

```
clock_gettime(CLOCK_REALTIME,       &now);
```

- Returns CLOCK_MONOTONIC + (now-1/1/1970) + leap seconds

POSIX: Piecewise-Linear Clock Model: Y[n]=mx[n]+c

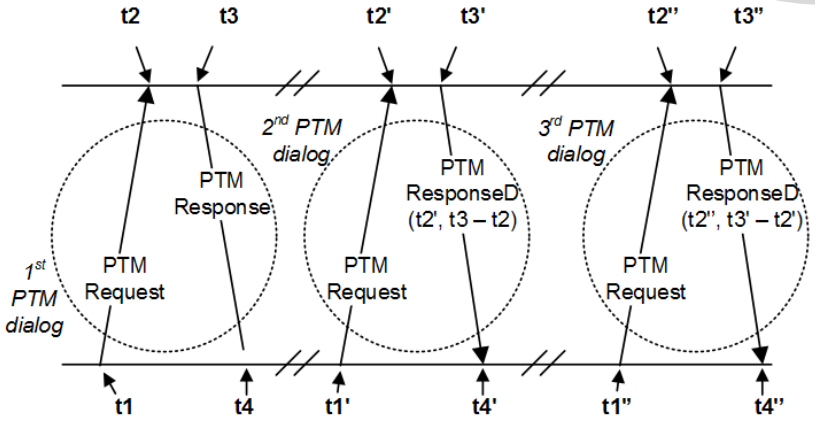
Measuring PTP vs. System Time using PCIe PTM

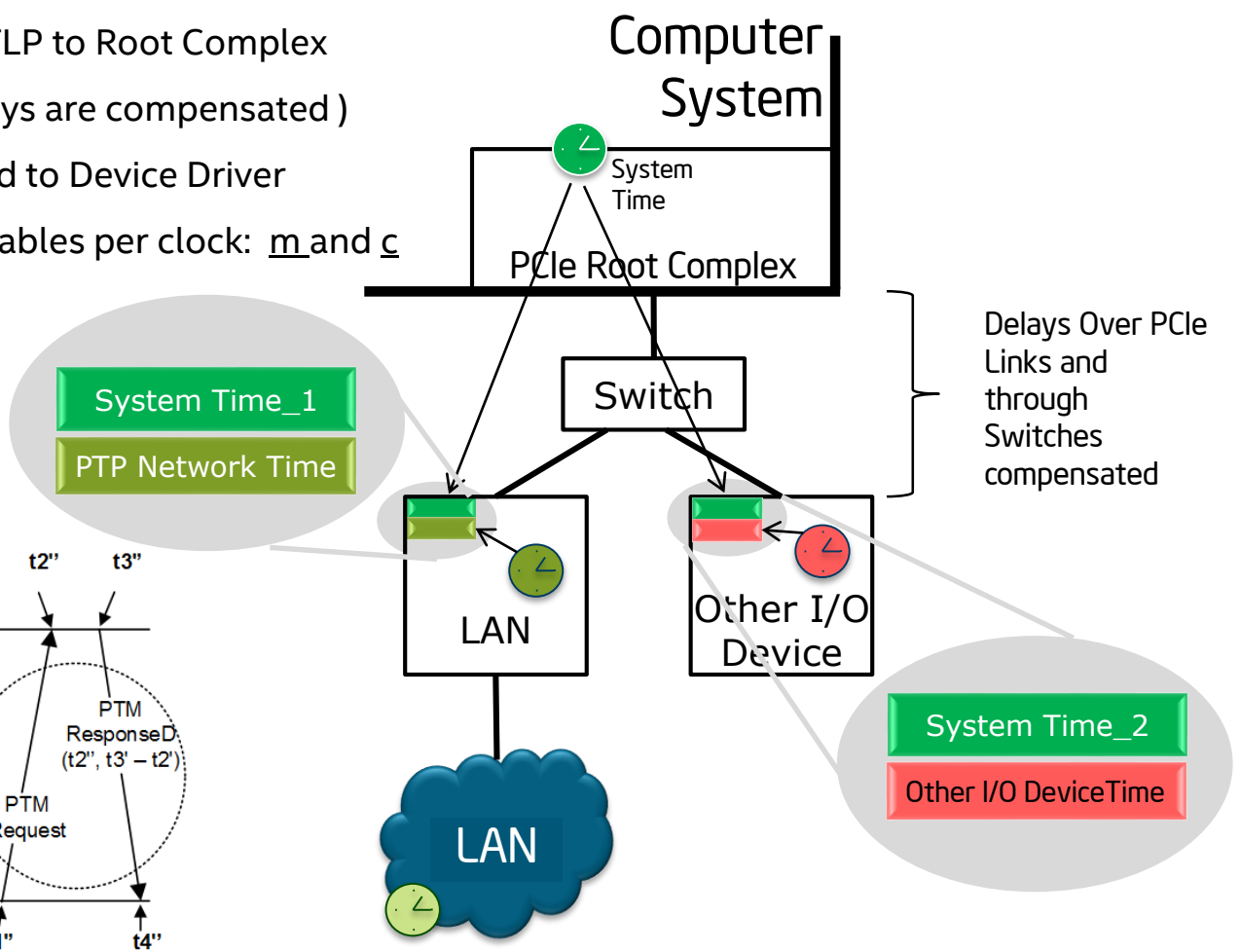
(Precision Time Measurement)

Scenario:

1. Device Driver Triggers Cross-Timestamp
2. Device initiates *PTM Request* TLP to Root Complex
3. System Time is Returned (delays are compensated)
4. (PTM Time, PTP Time) returned to Device Driver
5. Software "disciplines" two variables per clock: m and c

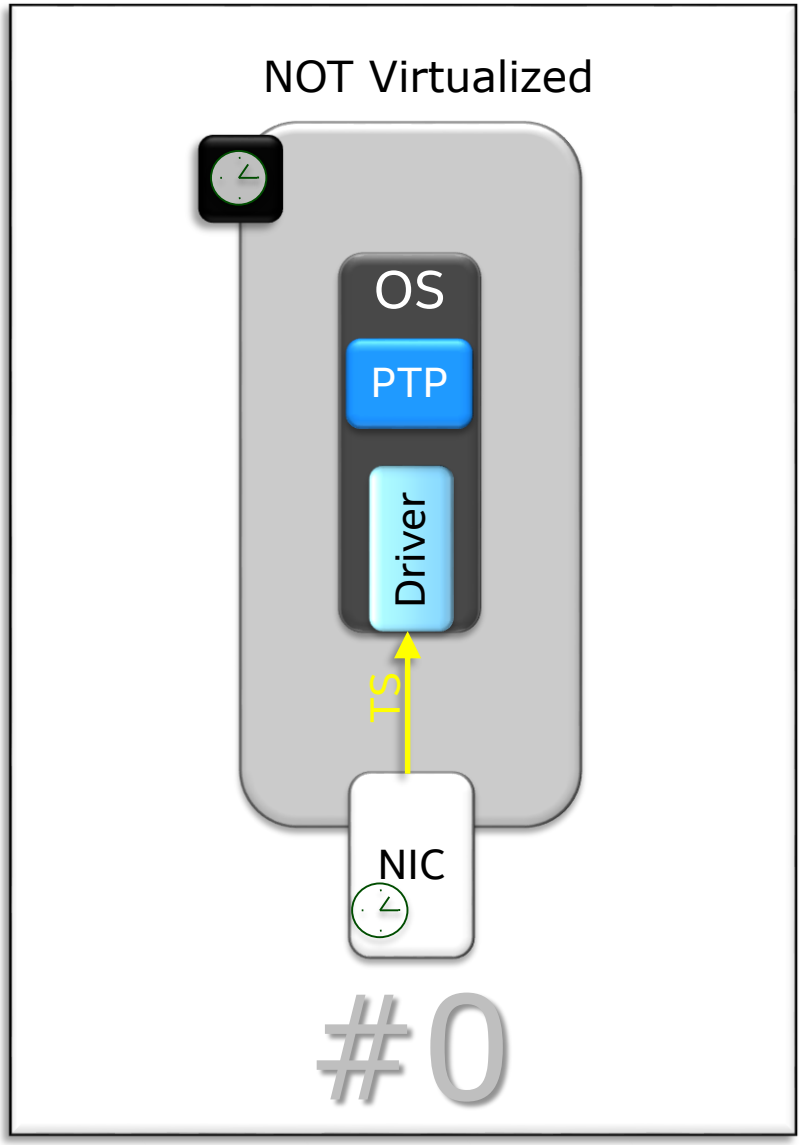
Cross Timestamps, Captured Simultaneously

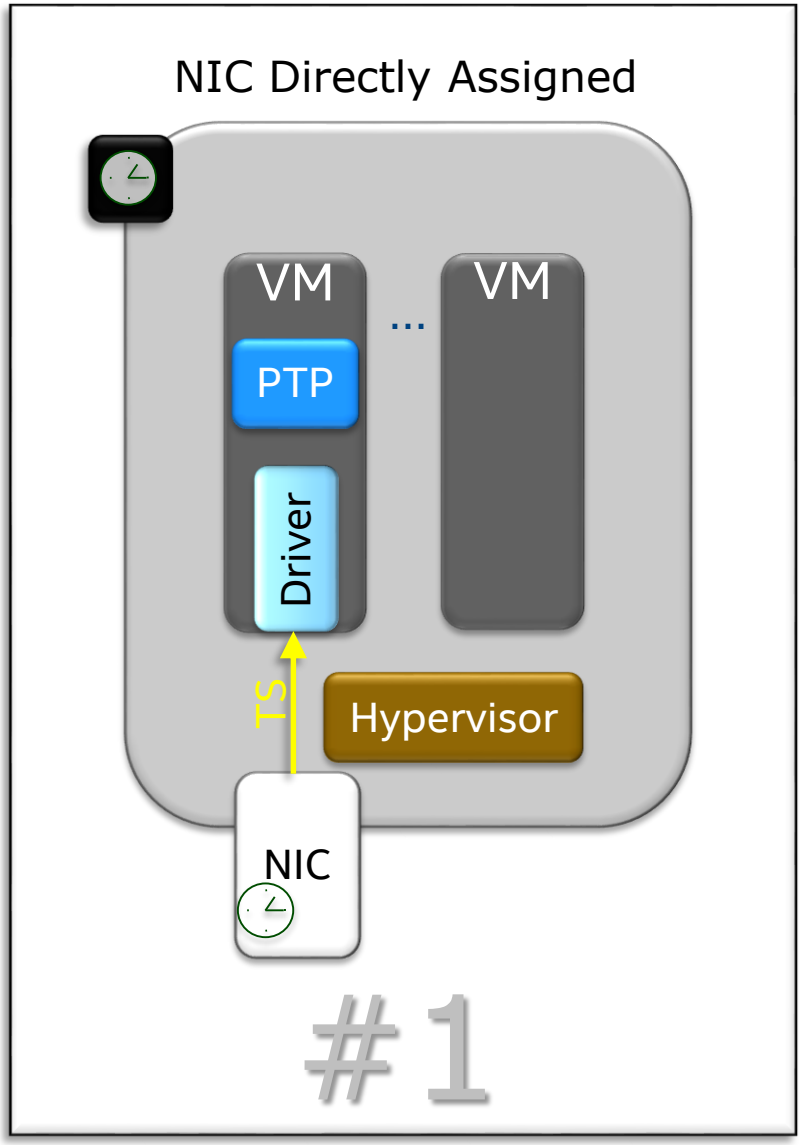




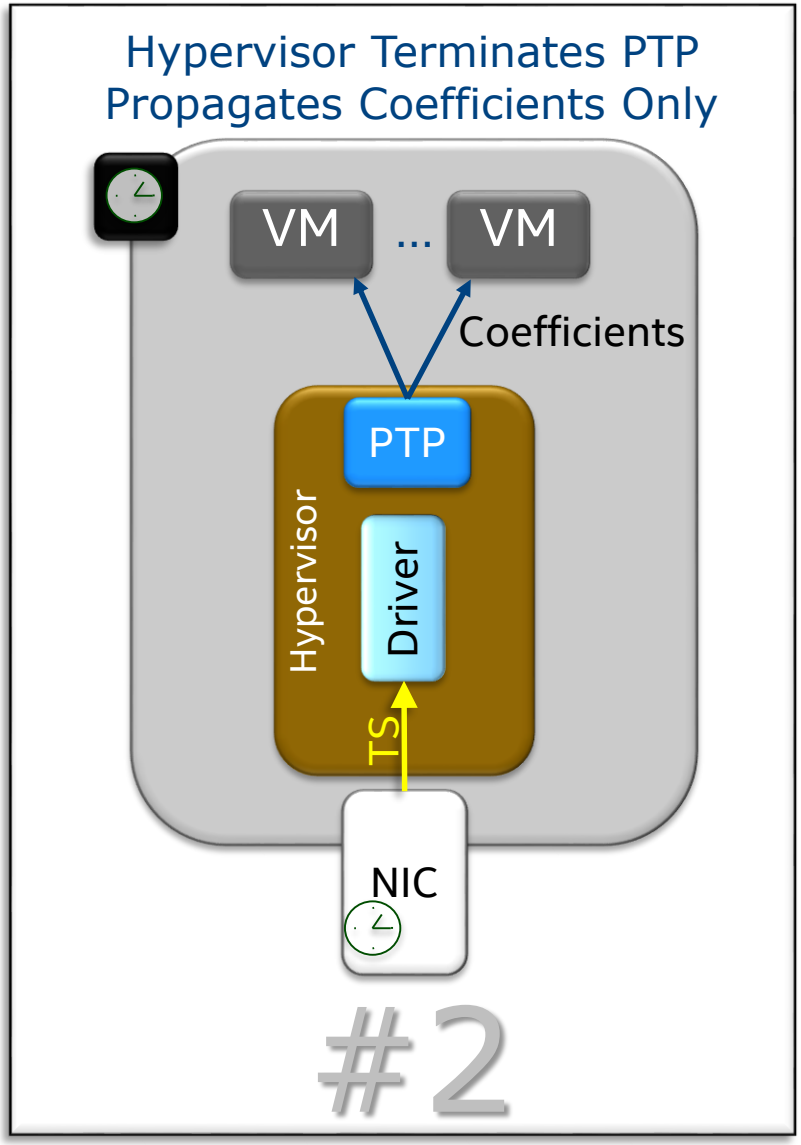
In-System Cross Timestamps ➔ Time Translation Coefficients

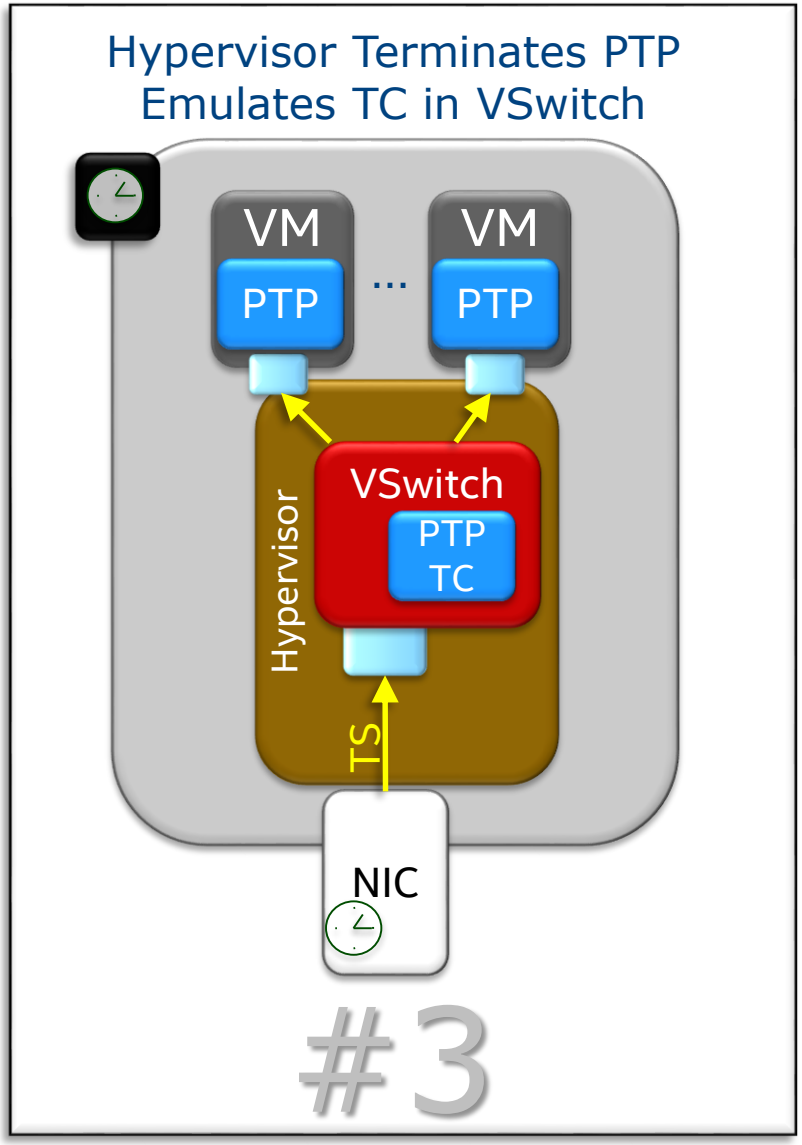
PTP Interface Virtualization



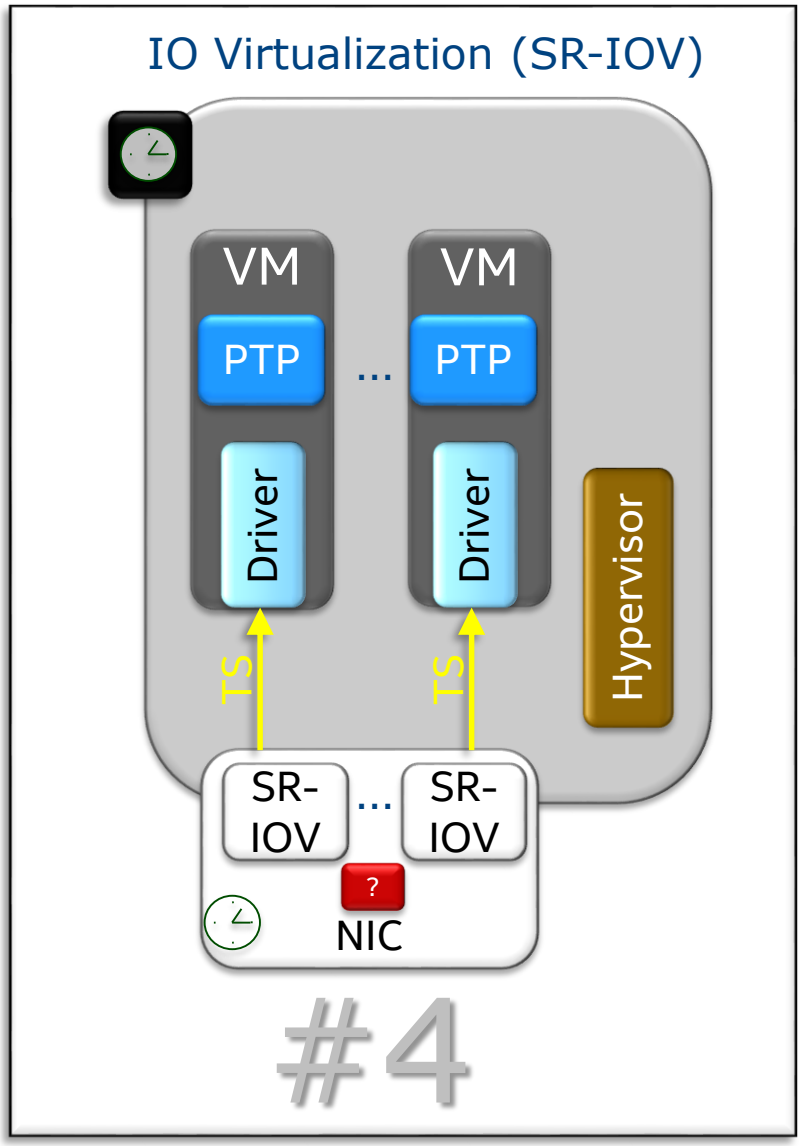


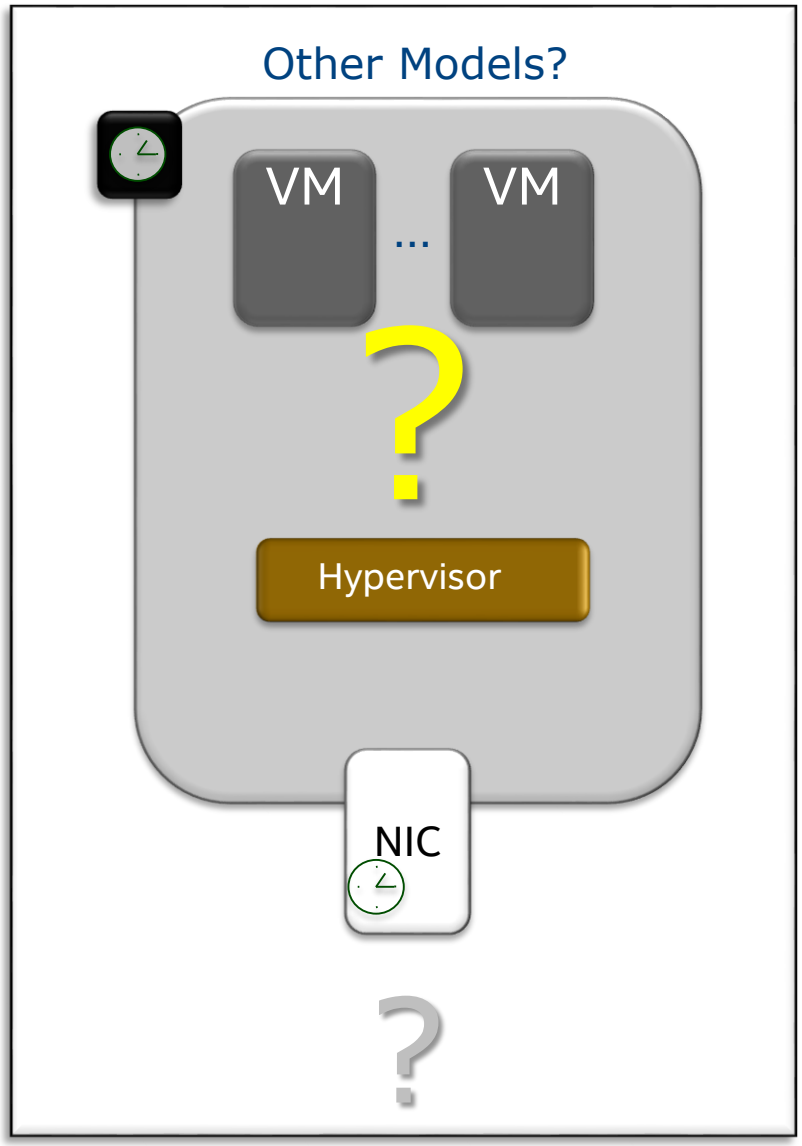
PTP Interface Virtualization





PTP Interface Virtualization





VM Access to PTP Event Message Timestamps

Three models:

1. Ethernet Hardware is owned by one VM instance

2. VMM runs PTP (interacts with PTP hardware directly), provides logical interface or coefficients to VMs

3. VSwitch implements PTP Transparent Clock, defines constant arbitrary residence time (e.g., 1ns)

4. VMs individually run PTP, interact with Ethernet hardware (e.g., via SR-IOV)

Disciplining/Adjusting the PTP Timestamp Counter

Approaches:

1. **Leave the PTP Counter be—let it free run**
   - Improves stability, scales to an infinite # of VMs, scales to infinite # of GMs and PTP Domains
2. Ethernet PTP Hardware Clock (PHC) owned by one Guest/VM instance
   - Implementable today
3. VMM Disciplines separate PHC clocks on behalf of guests
   - Let's not do this
4. Hypervisor virtualizes the adjustment requests by managing mx+c coefficients, and resulting PTP timestamp values
5. VMs individually interact with PTP Hardware Clocks (e.g., via SR-IOV)
   - Requires hardware replication—limited # of PHCs

Why Does PTP Timestamp Counter Need To Contain UTC (if Coefficients are Known)?

Summary

1. Virtualization *need not* degrade timing
2. Coefficients: The best way to map system time (TSC) to universal time
3. Virtualization of timestamping hardware: Multiple options, with pros/cons
   - If you're interested, please join the dialogue
4. The need to virtualize control of PTP counter is limited to a few situations