Discrete-Event Time-Sync Simulation

Dragan Obradovic & Guenter Steindl

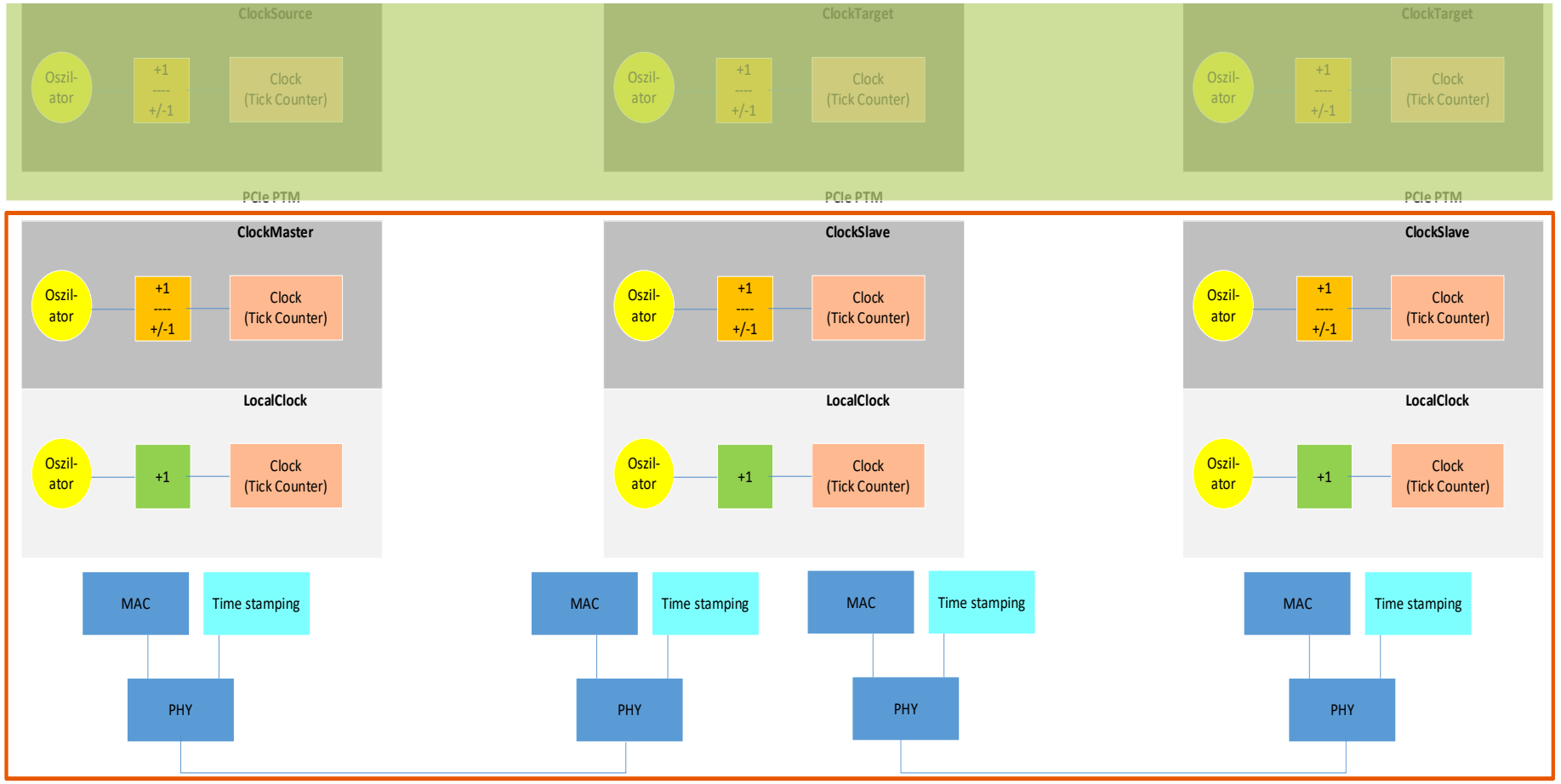
Content

- Time Synchronization in 802.1AS: brief overview
- Time-Sync Simulator requirements
- Time-Sync Simulator implementation as an Event-driven dynamic system
- Simulator Characteristics and a few representative runs
- Examples of the required level of detail for simulation: Modeling of Residence-Time and of pDelay, Delays in the Controller application
- Conclusions

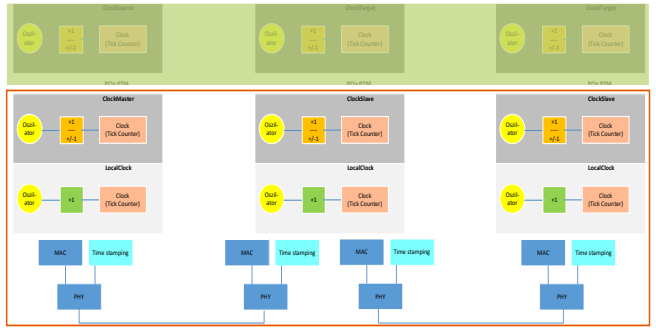
Time Synchronization

Based on:

- **Signaling**: Sync and pDelay messages (protocol defined)
- **Carried information**: Clock Times related to frequencies of available oscillators
- **Estimation and Compensation** of propagation delays
- **SW-based correction/control** of the clocks (tick counters)



Time-Sync Simulation

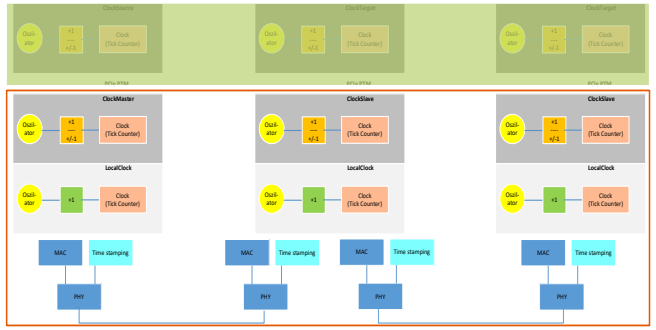


Any Time-Synchronization simulator needs to implement two models:

- MODEL of “PHYSICS”:
  - Fully implement the actual propagation of messages based on the complete knowledge of the delay mechanisms and of the frequency evolution of the underlying oscillators over time. All these information are known in absolute time frame

- MODEL of ESTIMATION AND CONTROL:
  - Conduct estimation of different variables, such as the pDelay, RateRatios, etc., based on the limited knowledge about the delay mechanisms and based on having the observations only in the local time frames (e.g. of the LocalClocks)

Time-Sync Simulation

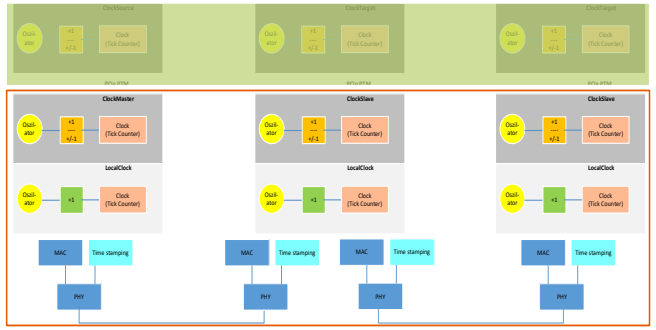


Input for the model of "physics":

- **propagation delays:** sources and values (constant or random)
- **oscillators frequency evolution over time**: sources (temperature, vibrations, etc.) as well as typical time evolution
- **definition and timing of different events (operations, described by state-machines),** triggered by message sending/receiving
- **schedule of messages** (e.g. Sync Messages at GM), which drive the system

→ This "physics" models the dynamics of the underlying information propagation between network elements

Time-Sync Simulation

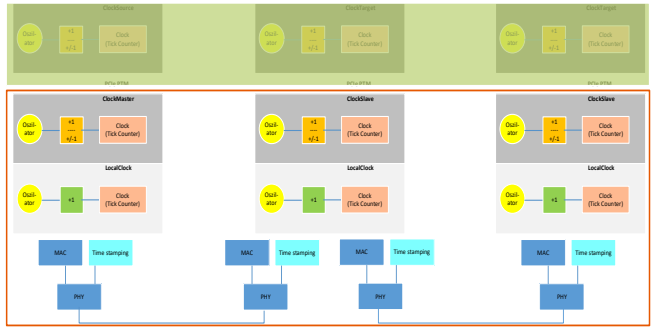


Estimation and Time-Correction (algorithmic part) depends on:

- **Observability of different variables** (i.e. what and when can be measured)
- **Estimation of variables which cannot be directly observed:** i.e. estimation of the frequency RR as a ratio of time-intervals
- **Time-correction mechanisms**: e.g. choice of the control actions (PID controller, linear extrapolation, etc.)

→ This is the algorithmic part, but event "the best" algorithms cannot guarantee the desired performance if they are not provided with the necessary and timely input information!

Time-Sync Simulation Dynamics



What is the dynamics of the time-synchronization process:

1) Clocks (counters) are integrators of the oscillators frequency → they have a continuous dynamics (or essentially time-discrete with dt=1/f)

2) But all important time instances (such as arrival and the departure times of messages) are related to events which do not happen continuously and not in regular time intervals

3) These events are causally dependent, one event triggers other events in a deterministic manner which is defined by state-machines. E.g. the arrival of a Sync Message at the next element triggers the augmentation of the pDelay to the Sync-Message-carried MasterTime, then sending these updated MasterTime to ClockSlave, …

→ **The whole dynamics is described as a sequence of deterministic events triggering each other at well defined time instances in absolute time (known to the simulator).**

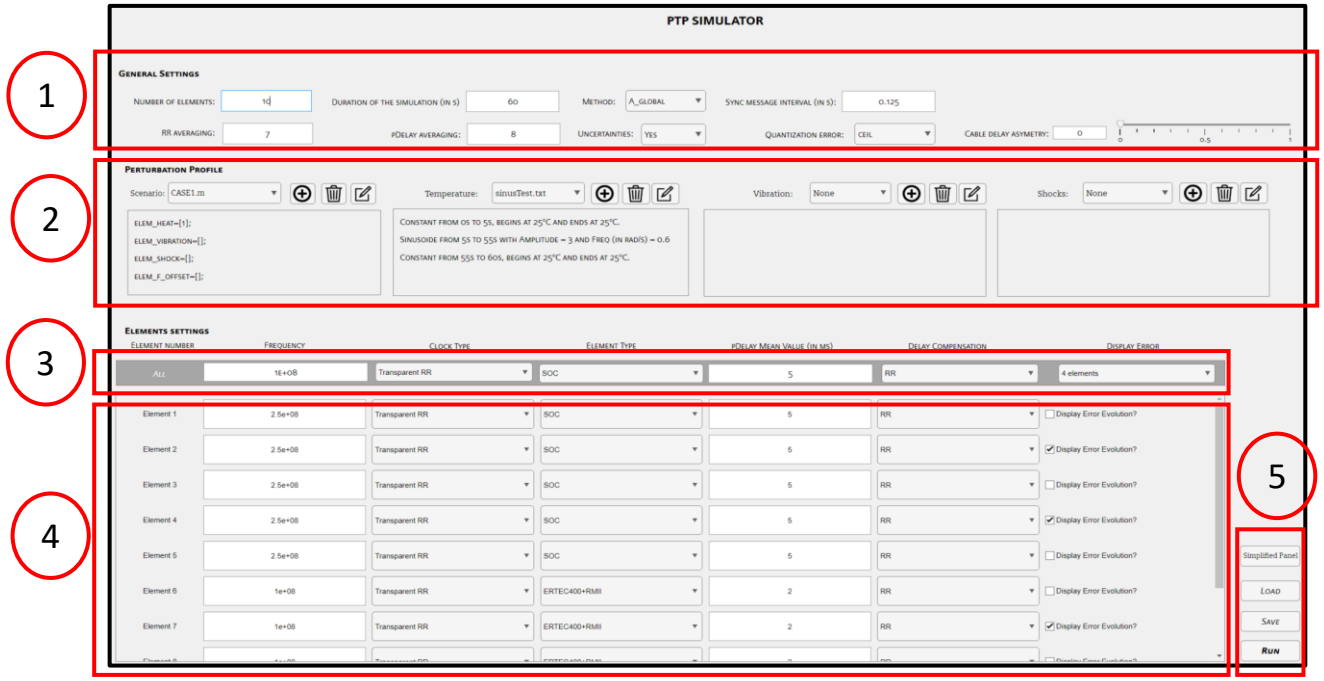
➔ **This is a Finite State Machine triggered by events scheduled in time!**

→ **There is no necessity for a model based on fixed time-step size!**

➔ **A discrete-event simulation suffices for modeling and analysis of the Time-Synchronization process!**

Simulator‘s GUI

1) **General settings of the simulation**. Here you can set up all the general parameters of the simulation such as the duration, the number of elements or the time interval between SyncMessage.
2) **Perturbation settings**. Here you can modify the perturbation profile of the simulation. You can find (from the left to the right) the set up panel for scenarios, temperature changes, vibrations, and shocks.
3) **Elements common settings**. Here you can set a parameter of all the elements of the simulation to one specific value.
4) **Elements individual settings**. Here you can modify all individual parameters for each element.
5) **Control panel of the simulator**. Here, you can find (from top to bottom) the button to the *Simplified Panel*, the *Load* and *Save* button to load and save a simulation configuration, and the *Run* button to launch the simulation.



Simulator Characteristics (selected)

RR calculation methods:

- Product of NRRs, NRRs calculated using pDelay-Messages
- Product of NRRs, NRRs calculated using Sync-Messages (currently proposed amendment to 802.1AS)
- Directly calculated using the MasterTime carried by Sync Messages

Averaging (Filtering) used in calculation of RR, NRR, and pDelay:

- Moving average (mean or median) of the chosen length over non-overlapping windows
- Moving average (mean or median) of the chosen length over overlapping windows

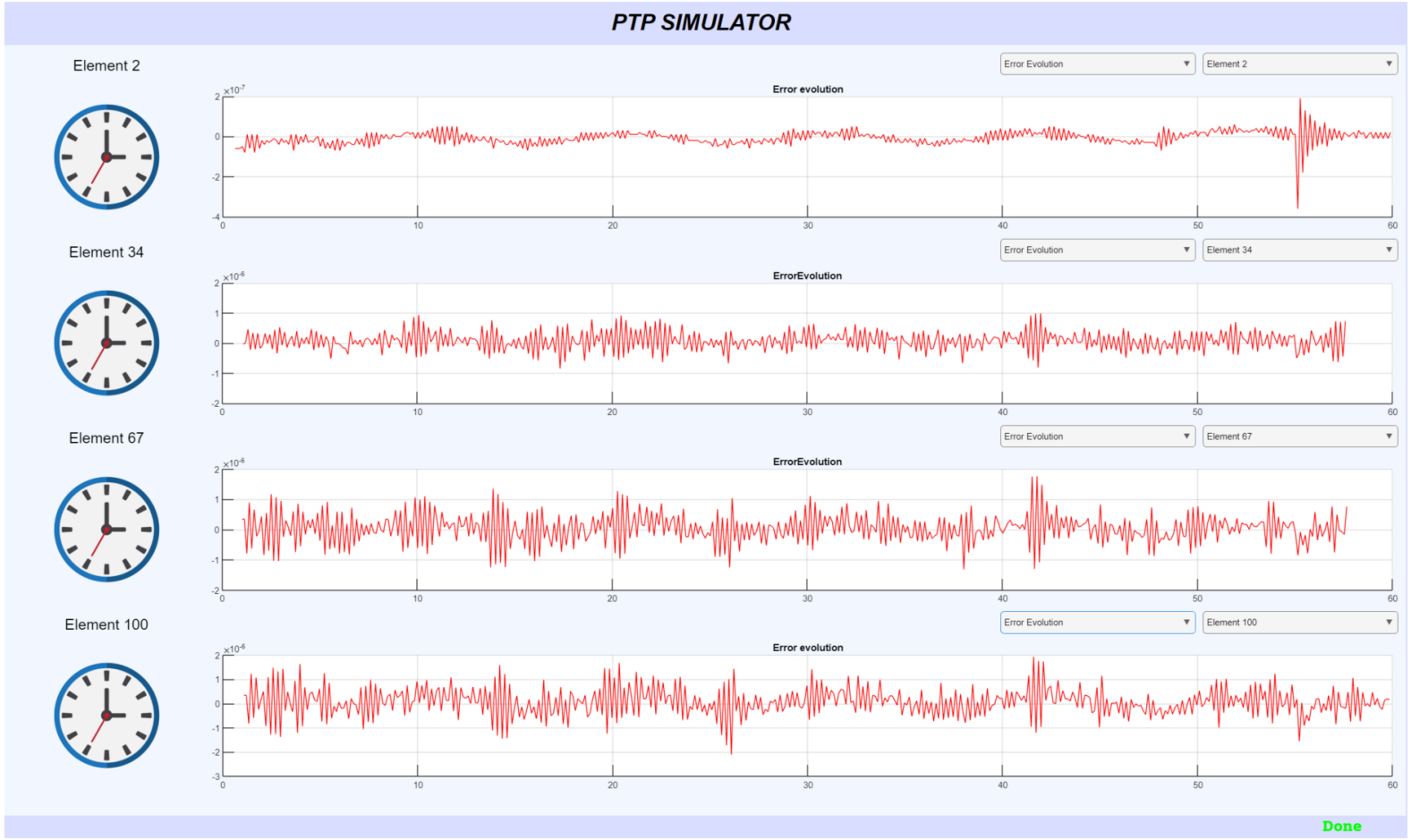
Perturbation definition:

- Temperature profile definition (start time, evolution such as ramp, triangular or sinusoid periodic signal, end time)
- Vibrations (sine function)
- Shocks (impulses)

Controller:

- PID, with limits on control action

Simulation with NRR via Sync and with following parameters: Tsync=125ms, Sine-frequency change with w=0.6rad/s and A=2ppm, ResidenceTime=Gauss(m=5ms, std=0.833ms)



The simulation does not have to wait long „to stabilize“. Even with the first element in the filters buffer (for NRR) we have a good estimate. When the buffer (which is of the filter‘s order) is full, we have fully converged.

→ The buffer is full after a couple of Sync Messages (depending on the filter definition) and their time to propagate to the last element.

→ 8 Sync messages require 1sec and the propagation over all elements is ~500ms

→ Definition of the Error is on the next page

Error Definition visualized in the simulator outputs

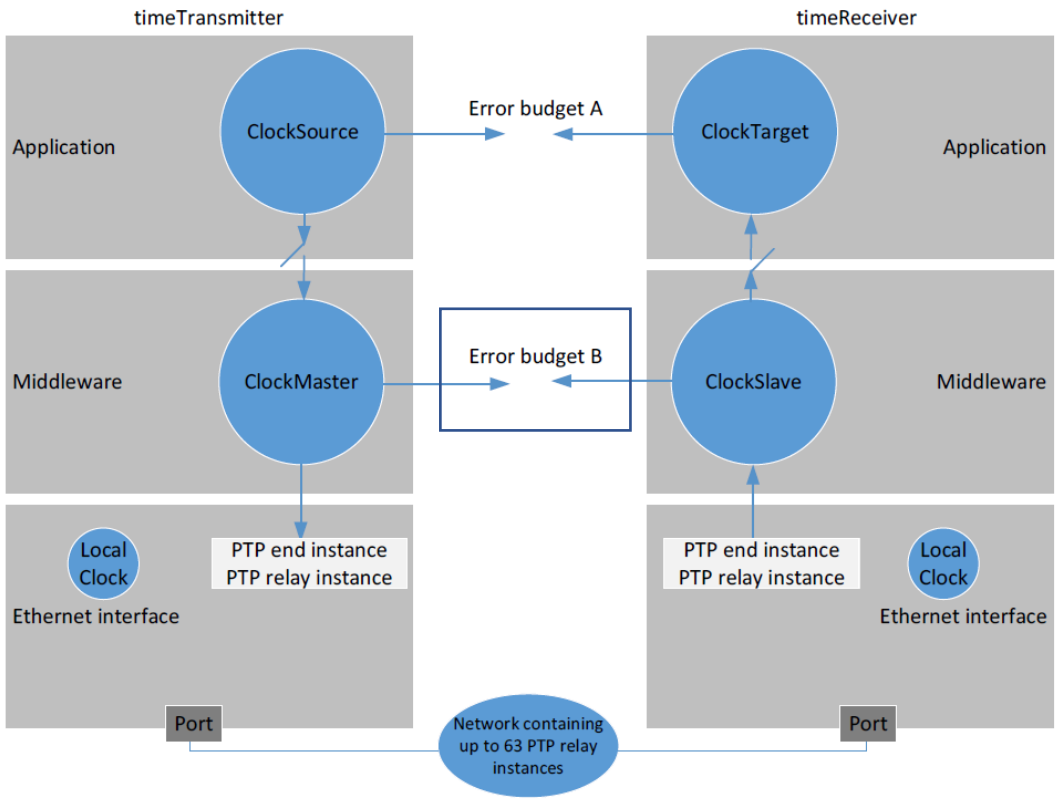
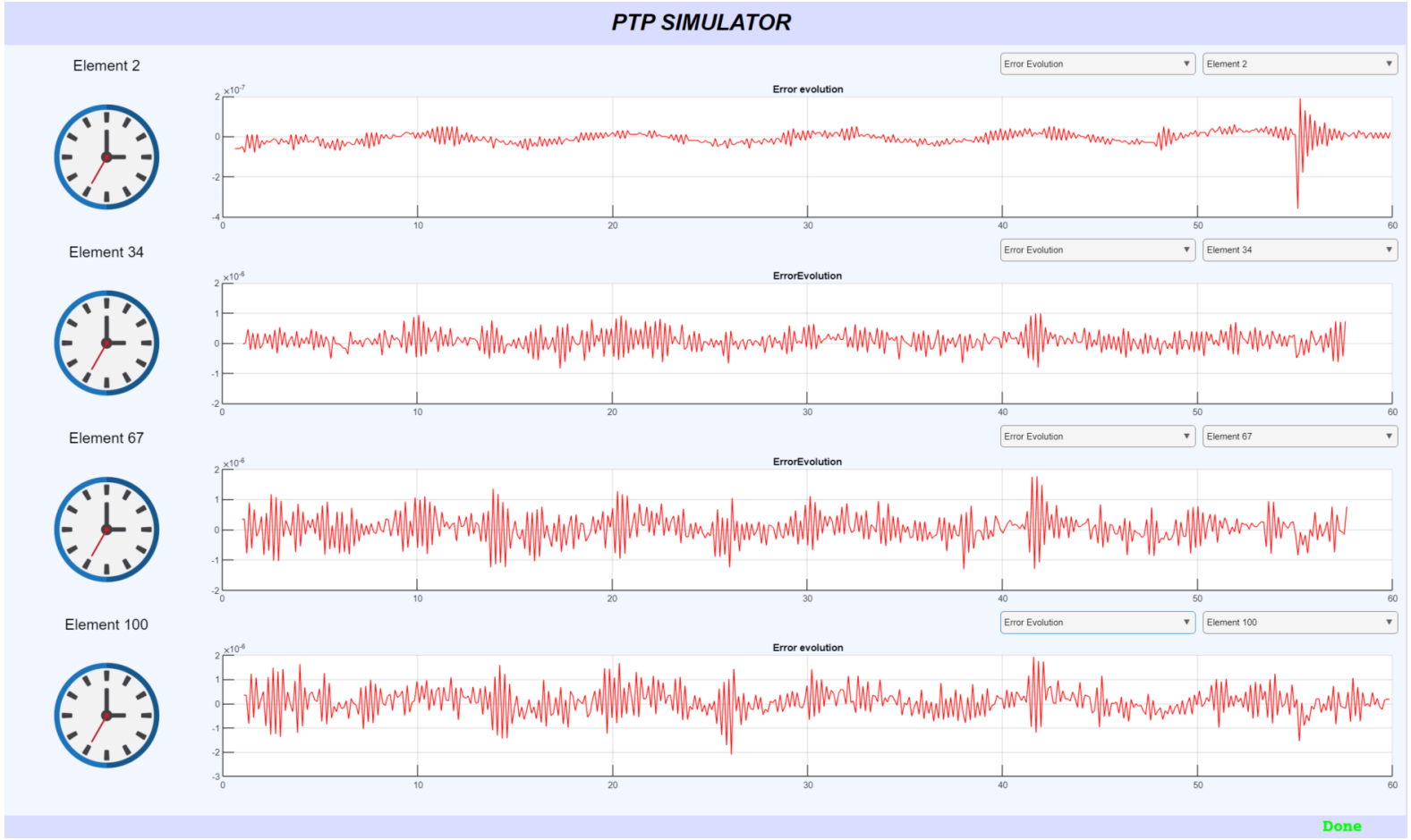


Figure 16 – Error budget scheme

The error presented here is the difference between the MasterTime provided by the controller and the actual MasterTime provided by the ClockMaster

Simulation with NRR via Sync and with following parameters: Tsync=125ms, Sine-frequency change with w=0.6rad/s and A=2ppm, ResidenceTime=Gauss(m=5ms, std=0.833ms)

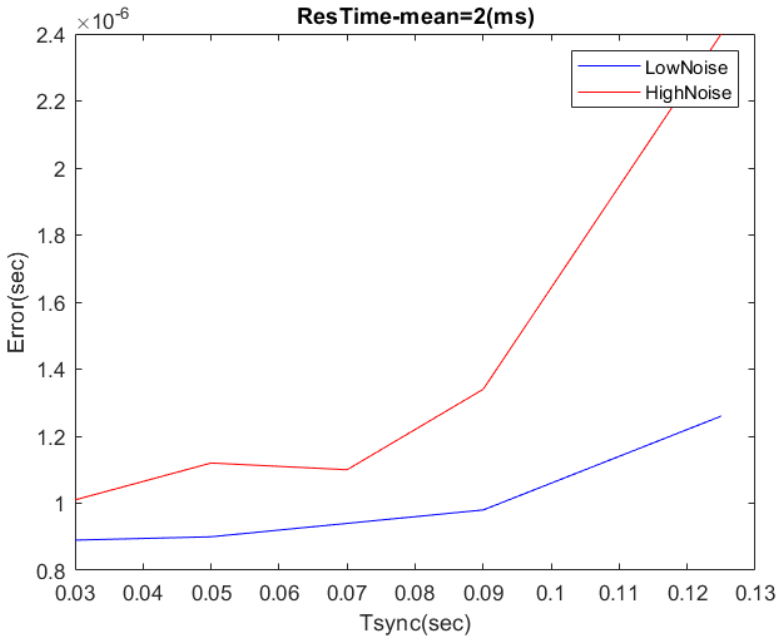


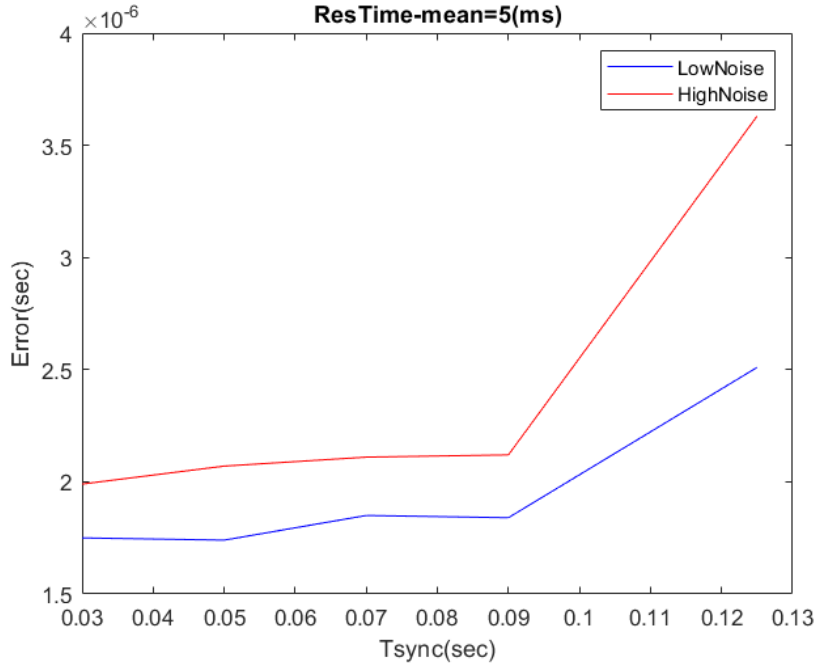
The error is above 1µs

We can see what happens when we vary:

1) Tsync
2) Residence-Time Mean (with a constant std, for simplicity)
3) Rx & Tx jitter level

Time Synchronization Error as a function of different parameters (worst error over 100 elements, with the statistics of 30 repetitions)





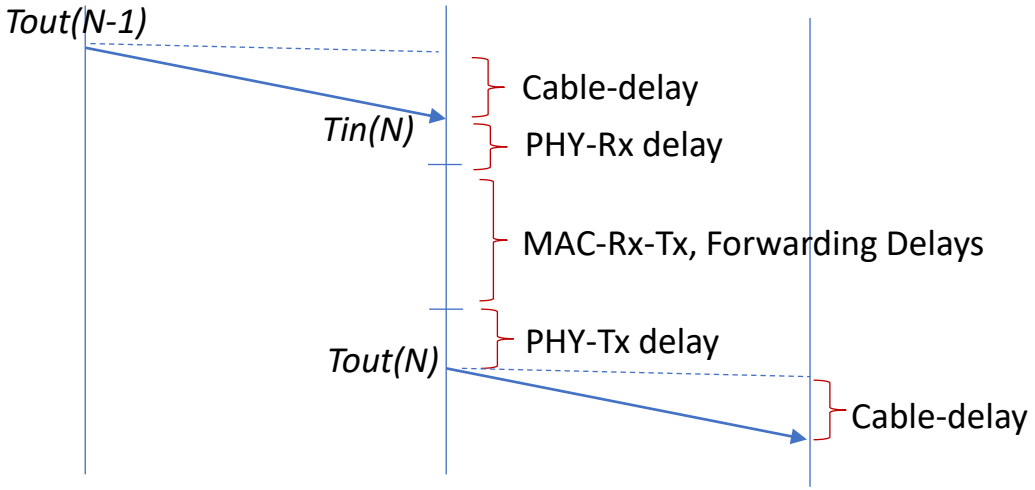
➔ Reducing mean of the Residence-Time and the Tsync helps

➔ But the jitter level (the variability of Tx and Rx delays) has influence

Modeling Example 1: Residence-Time and pDelay in Sync Message Propagation

The simulator needs two inputs:

1) Given Tout(N-1) how to calculate Tin(N) and Tout(N)
2) How to measure/estimate “Residence-Time” and “pDelay”



Receiver Delay (PHY-Rx):

Fixed: 180ns (with 100Mbit/s), known

Jitter: [0:16]ns for standard or [0:4]ns for the „best in class“ switch: random sample

BlockedOutputPort Delay: uniform between LB and UB, known (part of FORWARDING)

LB=(64*6)/f → for f=100MHz LB = 5µs

UB=(1518*8)/f → for f=100MHz LB = 125µs

Transmitter Delay (PHY-Tx):

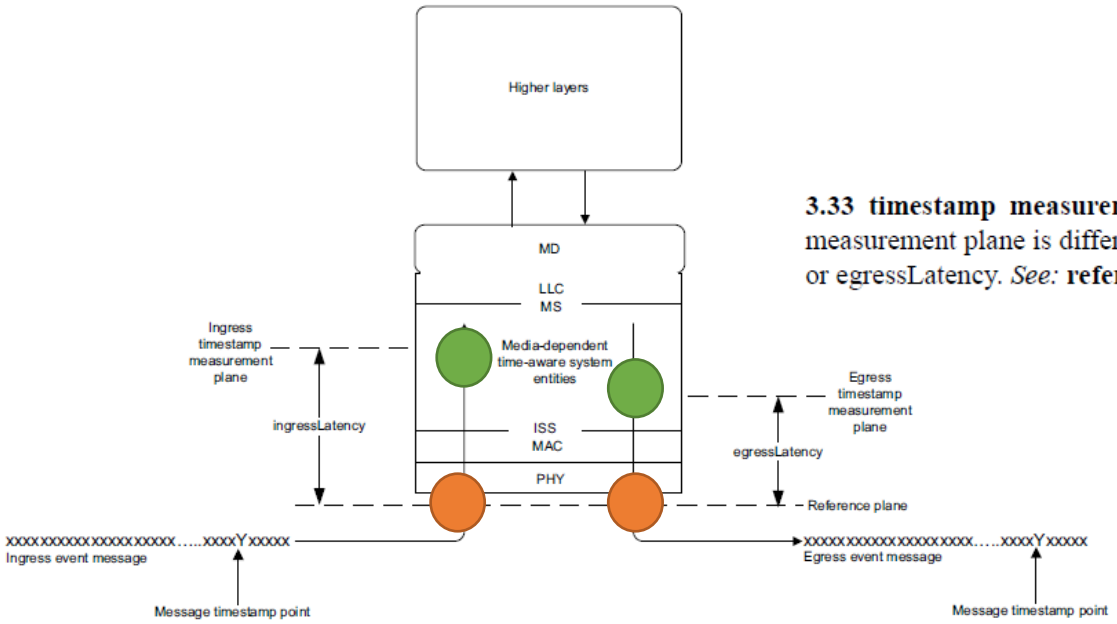
Fixed: 100ns (with 100Mbit/s), known

Jitter: [0:32]ns for standard or [0:4]ns for the „best in class“ switch: random sample

Cable Delay:

5ns/m (length dependent)

Time Stamping in 802.1AS



3.33 timestamp measurement plane: The plane at which timestamps are captured. If the timestamp measurement plane is different from the reference plane, the timestamp is corrected for ingressLatency and/or egressLatency. *See:* **reference plane**.

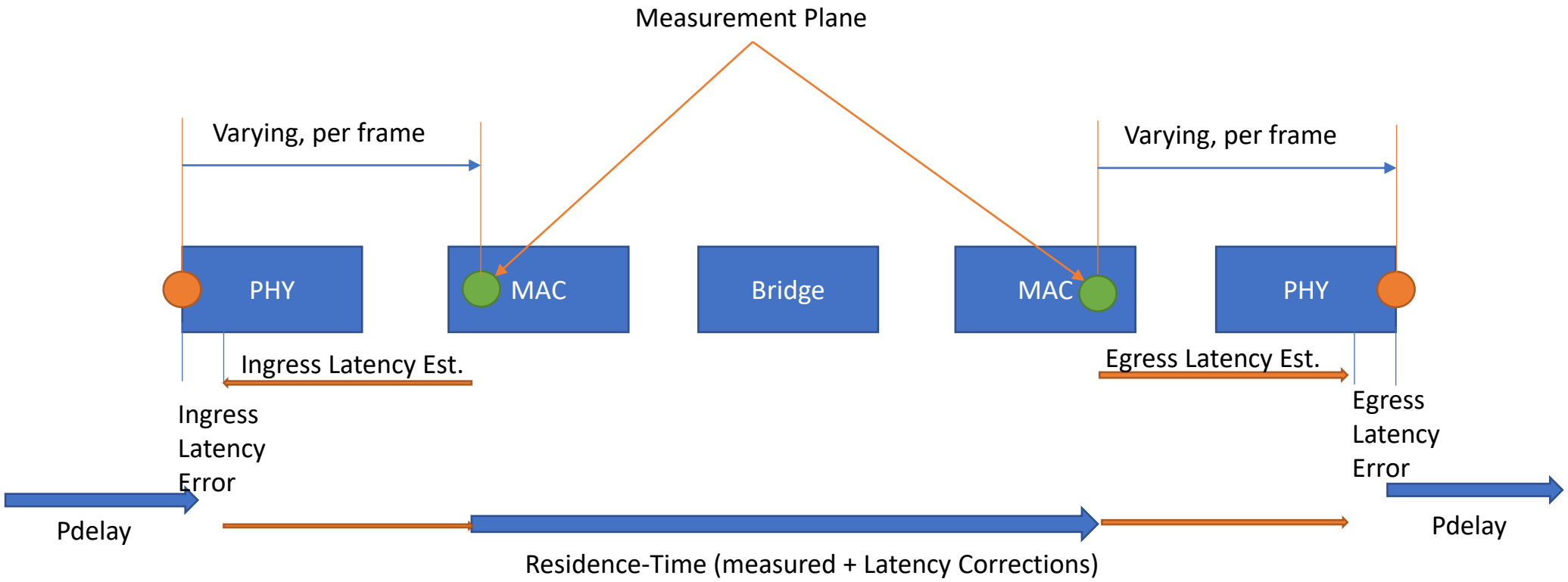
Figure 8-2—Definition of message timestamp point, reference plane, timestamp measurement plane, and latency constants

egressTimestamp = egressMeasuredTimestamp + egressLatency

ingressTimestamp = ingressMeasuredTimestamp − ingressLatency

Question: How is the Residence-Time defined, i.e. what is meant by the statement: ResTime=Gaussian (mean=5ms, std=0.833ms) limited within [1,10]ms ?

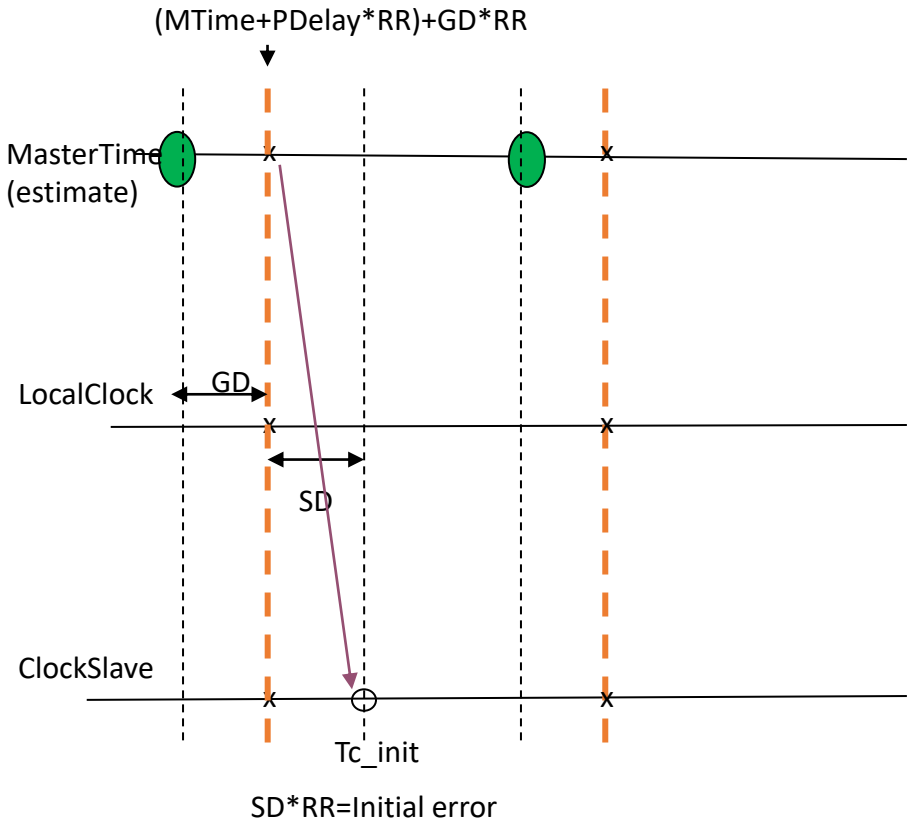
Modeling Example 1: Residence-Time and pDelay in Sync Message Propagation



The Simulator needs enough information to determine the true Master/LocalClock times at both pairs of green and orange circles!

The statement: „ResTime=Gaussian (mean=5ms, std=0.833ms) limited within [1,10]ms“ does not suffice, regardless if it describes the true or the estimated ResTime! More information like on page 14 needed!

Modeling Example 2: Delays in the Controller Application



Question: which values, if at all, have been used in the simulations so far?

There are different delays in the controller application:

1) GD ("GetDelay") – the time needed to provide the controller with the best MasterTime after receiving the latest SyncMessage (this is the value the controller should track).
   We use GD=1ms for one step or GD=1ms+2StepDelay, where the latter is uniformly distributed in [1,2]ms

2) SD (Delay of Setting) – The delay of the implementation of the control action

- MTime: Master Time carried by the last Sync Message
- (MTime + pDelay*RR): Master Time at receiving instance of the Sync Message
- (MTime+PDelay*RR)+GD*RR: The MasterTime when it is provided to the controller as the desired output
- But the controller needs time (SD*RR) to start implementing the calculated control action

Conclusions

- Simulators of Time Synchronization require a full model of the „physics“ (i.e. the actual delays of messages) and of the estimation methods

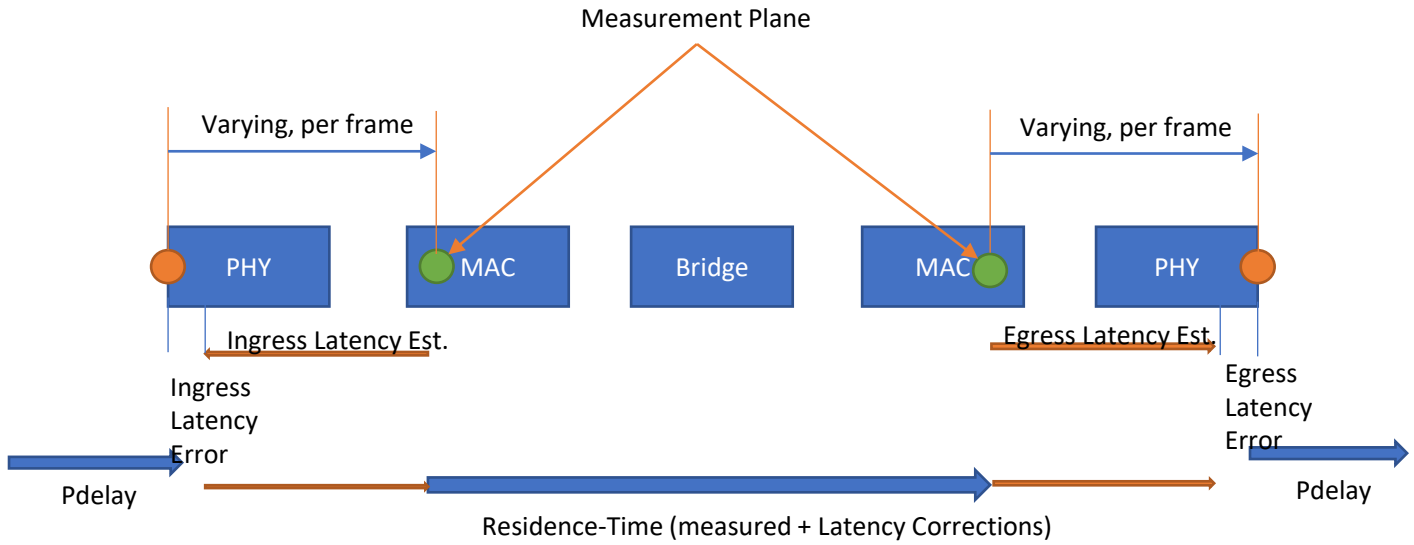
- These details have to be presented in order to compare different simulators (as shown in the example of defining and estimating the Residence-Time and the pDelay)

- We have an event-driven Time-Sync simulator with a collection of methods to calculate/estimate different variables (including the NRR calculation via Sync Messages)

- Basic simulation confirms that Tsync and the ResidenceTime have a large influence on the performance (as known). The same stands for the jitter (at Tx & Rx)

Extra:

What is the influence of the Rx and Tx jitter?



- The time-varying part of the Ingress and Egress Latencies (e.g. uniform [0 : 32]ns) cannot be measured and completely compensated
- Their mean will be (partially) compensated by the pDelay estimation process, but the variation will stay.
- The error due to the non-observable variable latency affects the accuracy of both Residence-Time and in the pDelay estimates, as well as of the NRR estimate

→ NRR is not affected much because of large Tsync interval (125ms) >> 32ns, Pdelay will (with averaging) compensate the mean

→ The total error in (Pdelay_est + ResTime_est) will have an error of [-32:32], but with a triangle as pdf. Over the successive elements there will be a superposition of these errors which will converge to zero mean but with a growing std.