Delivering Better Time-of-Day Using Synchronous Ethernet and 1588

Yaakov (J) Stein, Alon Geva, Gabriel Zigelboim
RAD Data Communications

The problem

I want discuss the use of 1588 (or NTP for that matter)
for time of day (wall-clock) distribution
in conjunction with Synchronous Ethernet (Sync-E)

Sync-E is a physical layer ***frequency*** distribution mechanism
What does Sync-E have to do with for ***time*** distribution ?

In practice, frequency and time distribution
are often inextricably intertwined

So, time distribution protocols (1588 or NTP)
always do frequency distribution as well

When don't we need a frequency distribution protocol for time-of-day ?

If we have a

- high accuracy local frequency source
- high time update rate
- Non-stringent time accuracy requirement

Then we don't need to distribute frequency to obtain time

If, for example,

- local oscillator is an OCXO with a lifetime accuracy of 200 ppb
- time update rate is 10 per second
- time accuracy requirement is 50 nanoseconds

Then it's OK not to update of the local frequency reference
since the time error due to the frequency offset
is always less than 20 nanoseconds

When do we need a frequency distribution protocol for time-of-day ?

The problem arises when we don't have accurate local frequency

We can increase the time distribution update rate

but that increases the (network, computational) resource drain

If the local reference is stable (but not accurately calibrated)

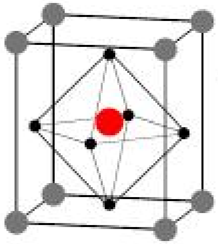
then we can use a frequency distribution protocol to set it

For example,

an inexpensive crystal's frequency accuracy might be 1 ppm

but after frequency lock it is within 1 ppb

and thus its drift is only 1 nanosecond per second



So frequency distribution greatly reduces the resource drain

Frequency distribution protocols vs. Sync-E

But using a **higher layer** time distribution protocol
for locking frequency has its drawbacks

In particular, it

- increases convergence time (time for error < required)
- increases steady state time error

Sync-E is a **physical layer** frequency distribution mechanism

Sync-E requires special hardware
but puts no further demands on (BW or computation) resources

If we have Sync-E

already installed for applications that require phase lock
especially installed since less expensive than better oscillator

then we don't need to use higher layer frequency distribution

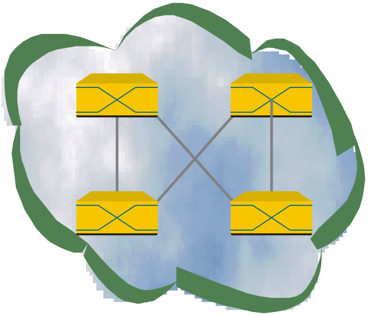
Advantages and disadvantage

By augmenting 1588 with Sync-E, we can obtain:

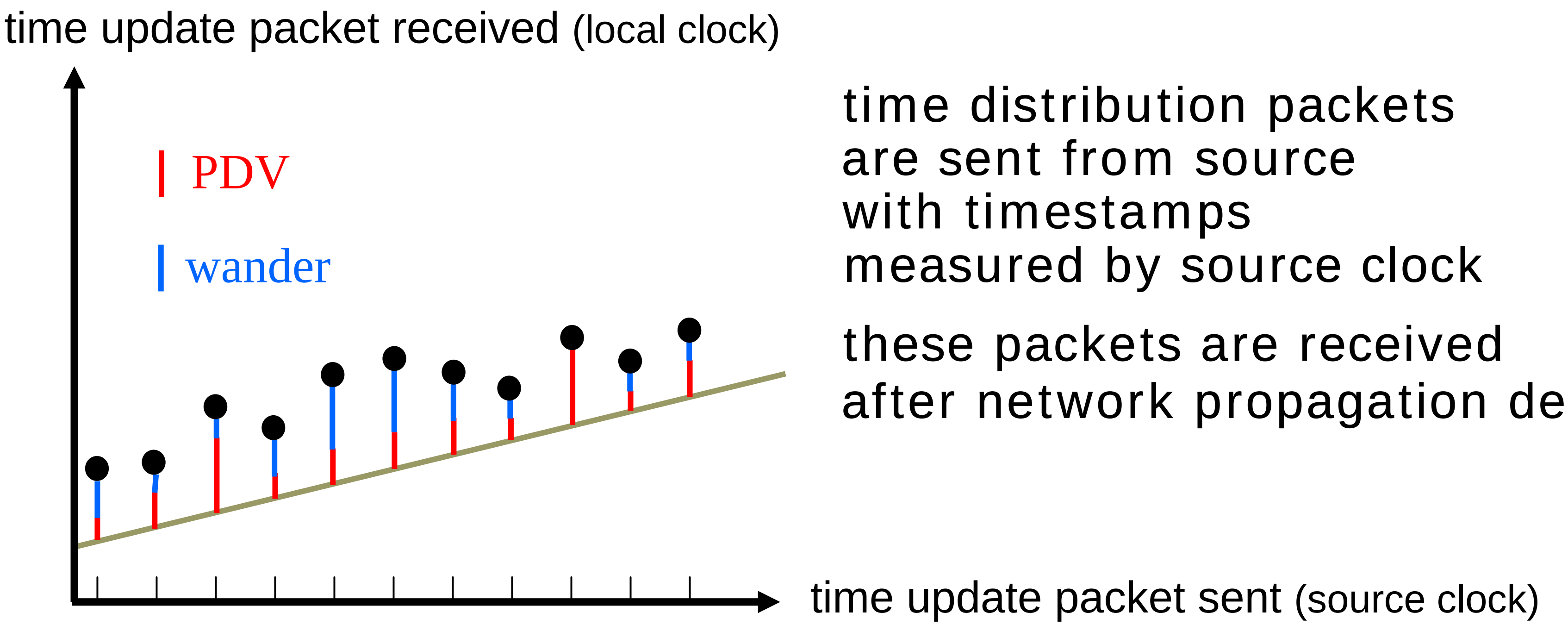
- better steady-state performance
- faster convergence to the desired time accuracy
  - capability of on-demand time transfer
  - possibility of using lower update rates during peak hours

But, if Sync-E is not already installed:

- may require a forklift upgrade of all switches along the path
  - may thus be prohibitively expensive



Time distribution over packet switched network



time distribution packets are sent from source with timestamps measured by source clock

these packets are received after network propagation delay

Without queuing delay, all packets received after minimal delay otherwise there is Packet Delay Variation (PDV)

But the receiver measures the arrival time using its local clock and this local clock has wander with respect to source clock

Two contributions

So the observed arrival times differ from the timestamps due to two effects:

- the difference in frequency between local and source clocks
- the packet propagation delay, which in turn is made up of
  - the minimal (electrical) propagation delay
  - the queuing delay (time the packet waits because other traffic in queue)

The job of a time distribution system is to

- lock the local clock frequency onto the source clock
- measure the propagation delay (ranging)

With Sync-E the first task is performed by the physical layer and so the first effect is minimized

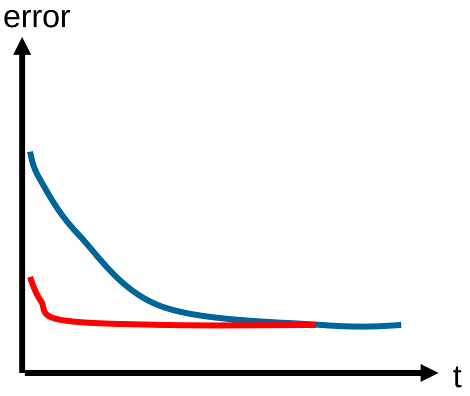
Convergence time

Assume that our implementation

- first stabilizes local frequency reference
- and only afterwards locks the time

Then the time to converge to desired time accuracy is the sum of

- the time to obtain frequency lock
- the time for ranging procedure to converge



If we eliminate the first term by using Sync-E

we certainly reduce the convergence time

Even if our implementation simultaneously

adapts frequency and time

its convergence time is longer

since early time adaptations are relatively meaningless

Steady state error

Once converged, standard time distribution systems

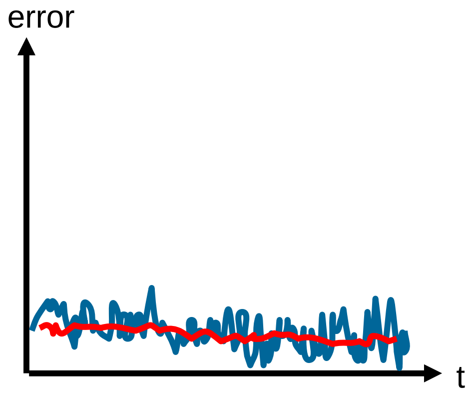
- continue updating the local frequency reference
- continue ranging to update local time

The time error is the sum of

- the frequency error (wander) contribution
- the ranging error (caused by PDV, asymmetry, etc.)

If we eliminate the first term by using Sync-E

we certainly reduce the steady-state error



Sync-E still leaves some residual frequency error

but physical layer frequency locking is

- more reliable
- more accurate

than higher layer frequency distribution

A subtler (but important) effect

There is yet another way Sync-E can help 1588 or NTP

Ranging techniques function by
- request and response exchange
- computing round-trip time using four timestamps
- estimating one-way time assuming symmetry
- minimize PDV effect by *minimum gating* (if possible)

Minimum gating
- assumes that there are *some* packets that traverse network with essentially no queuing delay
- identifies these packets by finding minimum round-trip delay

If the probability of minimal delay one-way traversal is $\boldsymbol{p}$
then the probability of round-trip minimal delay traversal is $\boldsymbol{p}^2$

This probability may be vanishingly small !

if P=1% update 100pps
then 1 sec until one way minima
but 100 sec until round-trip minimum

But we don't really need to have a single transaction
with minimal traversal time in both directions

Two separate minima

We can monitor the four timestamps

- and note the minimal difference independently in each direction
- the probability of these two events is ***p*** not ***p*²**

Were the slave's clock to be locked to the master clock

- the regular calculation could be used

But when the slave's clock drifts

- between the two minimal traversal events

the calculation acquires a corresponding error term $\int \Delta f\left(t\right)dt$

By using Sync-E we ensure that the slave clock is locked

- thus enabling use of two separate minimal traversal events
- and thus immensely improving the ranging performance

Using separate minimum method with IEEE 1588

If slave clock has offset T_0
  then slave time t
  corresponds to master time $t - T_0$
If slave also has frequency offset $\Delta f(t)$
  then slave time t
  corresponds to master time $t - T_0 - \int \Delta f(t)\, dt$

without Sync-E:

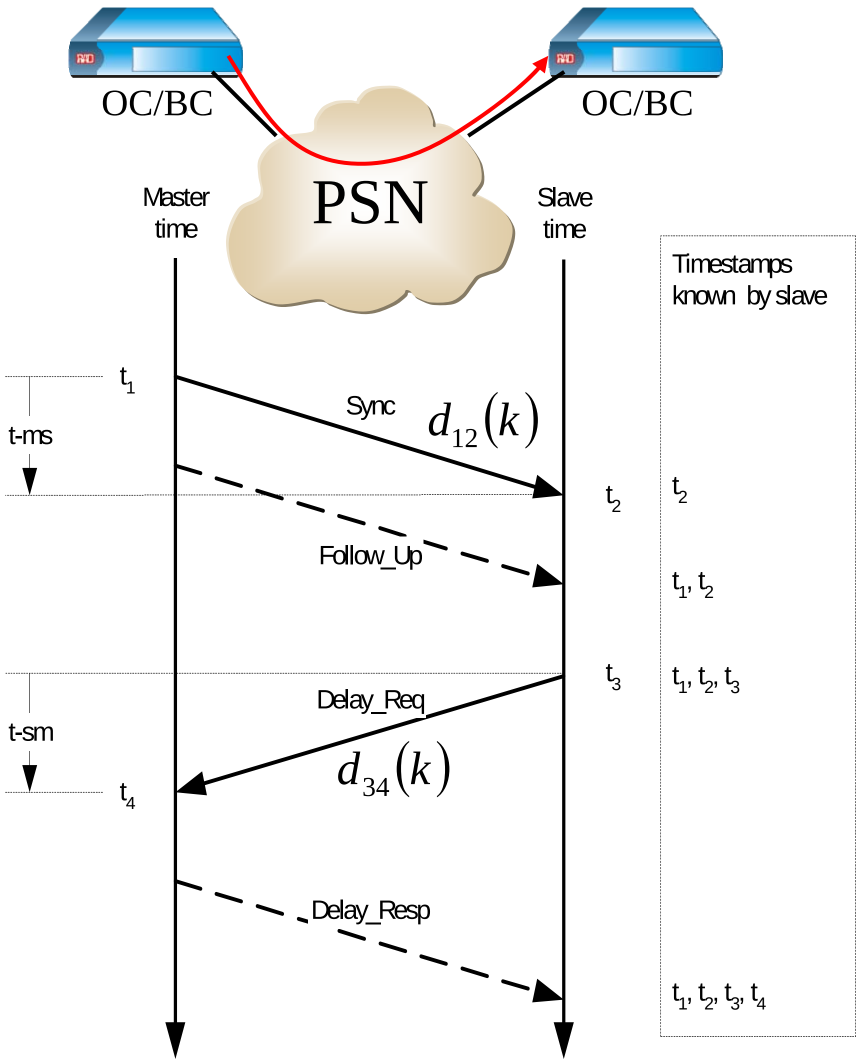
$$t_2(k)-t_1(k)=d_{12}(k)-T_0-\int_0^{t_2}\Delta f(t)dt$$

$$t_4(k)-t_3(k)=d_{34}(k)+T_0+\int_0^{t_4}\Delta f(t)dt$$

with Sync-E we can take $\Delta f(t)=0$:

$$t_2(k)-t_1(k)=d_{12}(k)-T_0$$

$$t_4(k)-t_3(k)=d_{34}(k)+T_0$$



Theoretic discussion assumptions

To be specific, we will assume:

Time update rate
- 100 PPS in the forward direction (for Adaptive Clock Recovery - ACR)
- 10 PPS in the backward direction (for TOD)

Adaptive clock recovery based solely on forward direction
Adaptive clock recovery bandwidth of 10 mHz
Adaptive clock recovery is based on an OCXO local reference

For white Gaussian network delay
    mean delay is assumed to be symmetric
For truncated distributions
    minimum delay is assumed to be symmetric

Additive Gaussian Packet Delay

When the PDV distribution is Gaussian

there is not an appreciable proportion of packets with *minimal delay*

so we rely on the entire set of delay values (no minimum gating)

Assuming symmetry, we average over k: $\frac{1}{2}\left\langle\left[t_4(k)-t_3(k)-\left(t_2(k)-t_1(k)\right)\right]\right\rangle =$

$$= \boxed{\frac{1}{2}\left\langle d_{34}(k)-d_{12}(k)\right\rangle} + T_0 + \boxed{\cancel{\frac{1}{2}\left\langle \int_0^{t_2(k)}\Delta f(t)dt + \int_0^{t_4(k)}\Delta f(t)dt \right\rangle}}$$

= 0 assuming mean symmetry

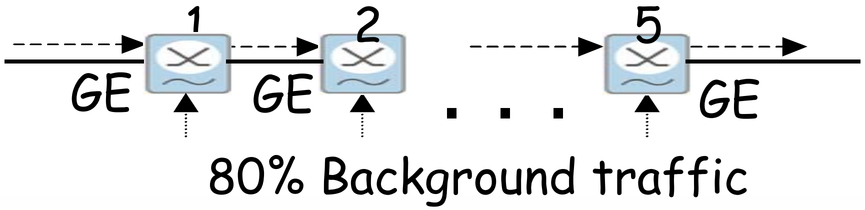
Note:

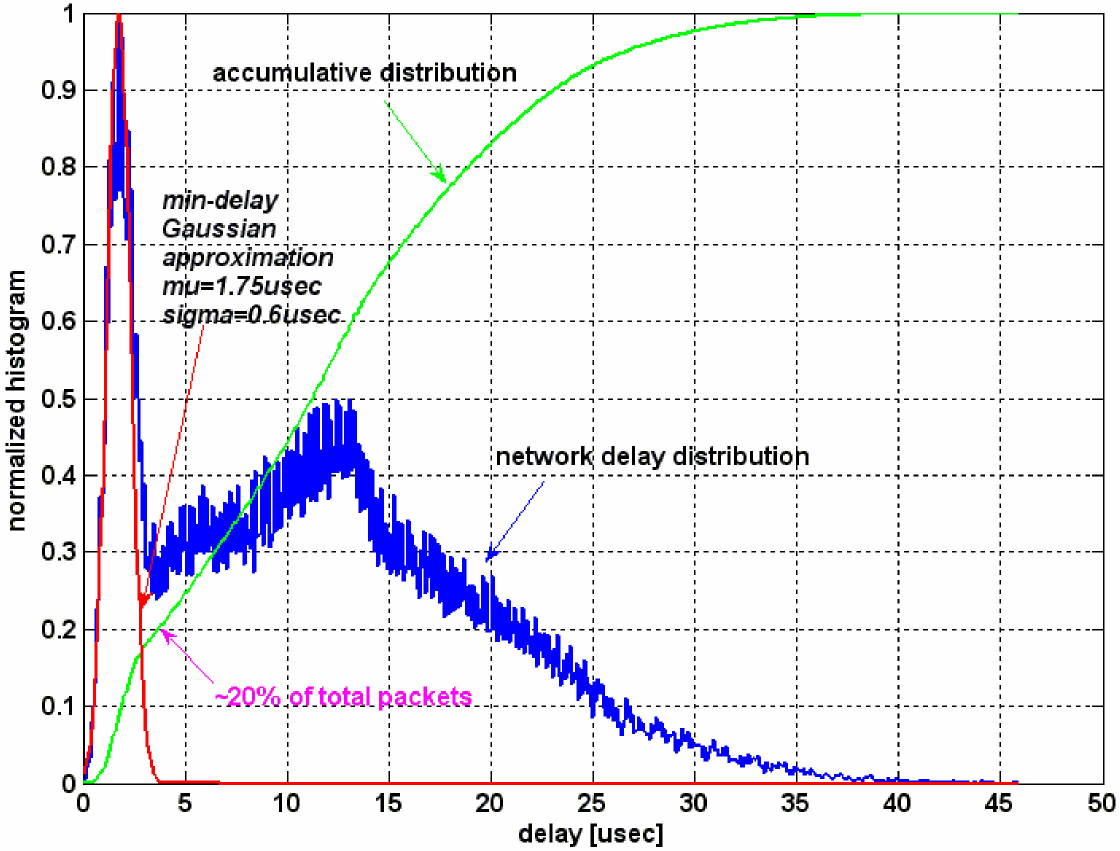
we need to average over a long time since the first term is zero only on *average*

when Δf≠0 the last expression contributes unavoidable error to the TOD estimation

Thus, having zero frequency error (Sync-E) makes a difference

‘Truncated’ Packet Delay
5 Cascaded Switches





20% of packets in each direction undergo *minimum delay* that is approximately Gaussian with 1.75 μsec mean and 0.6 μsec std-dev (total delay has 8 μsec std-dev)

$$\frac{\sigma_{\text{tot}}}{\sigma_{\text{min}}} = 13.33$$

Independently exploiting the *min* on both directions

Here we have a portion of the packets experiencing *min delay* relying on the approximately 20% of entire delay values set on each direction

Therefore, any time offset due to frequency error cannot be accurately measured anymore:

$$\min\{t_2(k_m)-t_1(k_m)\}=\min\{d_{12}(k_m)\}-T_0-\cancel{\int_0^{t_2(k_m)}\Delta f(t)dt}$$

$$\min\{t_4(k_n)-t_3(k_n)\}=\min\{d_{34}(k_n)\}+T_0+\cancel{\int_0^{t_4(k_n)}\Delta f(t)dt}$$

Again zero frequency error makes a difference!

↓

Using Sync-E

$$\frac{1}{2}\langle\min\{t_4(k_n)-t_3(k_n)\}-\min\{t_2(k_m)-t_1(k_m)\}\rangle=$$

$$=\boxed{\underbrace{\frac{1}{2}\langle\min\{d_{34}(k_n)\}-\min\{d_{12}(k_m)\}\rangle}_{\text{= 0 assuming min symmetry}}}+T_0+\boxed{\cancel{\frac{1}{2}\left\langle\int_0^{t_2(k_m)}\Delta f(t)dt+\int_0^{t_4(k_n)}\Delta f(t)dt\right\rangle}}$$

Steady state pk-to-pk TOD error performance (for *T* sec integration time)

For the Gaussian case (using ACR) TOD can be approximated by:

$$\mathrm{TOD}_{\mathrm{pk2pk\_err}} \approx 7 \cdot \left(\frac{\sigma_{\mathrm{delay}}}{\sqrt{20T}} + \boxed{\cancel{\sqrt{\frac{10\mathrm{mHz}}{50\mathrm{Hz}}\,\sigma^2_{\mathrm{delay}}}}} \right)$$

ranging error

ACR error

7 to convert std to pk2pk

10 pps so # of TOD transactions = 10T

For the 'truncated' case (using ACR) TOD is only approximately:

$$\mathrm{TOD}_{\mathrm{pk2pk\_err}} \approx 7 \cdot \left(\frac{\frac{\sigma_{\mathrm{delay}}}{13.33}}{\sqrt{4T}} + \boxed{\cancel{\sqrt{\frac{10\mathrm{mHz}}{50\mathrm{Hz}} \cdot \left(\frac{\sigma_{\mathrm{delay}}}{13.33}\right)^2}}} \right)$$

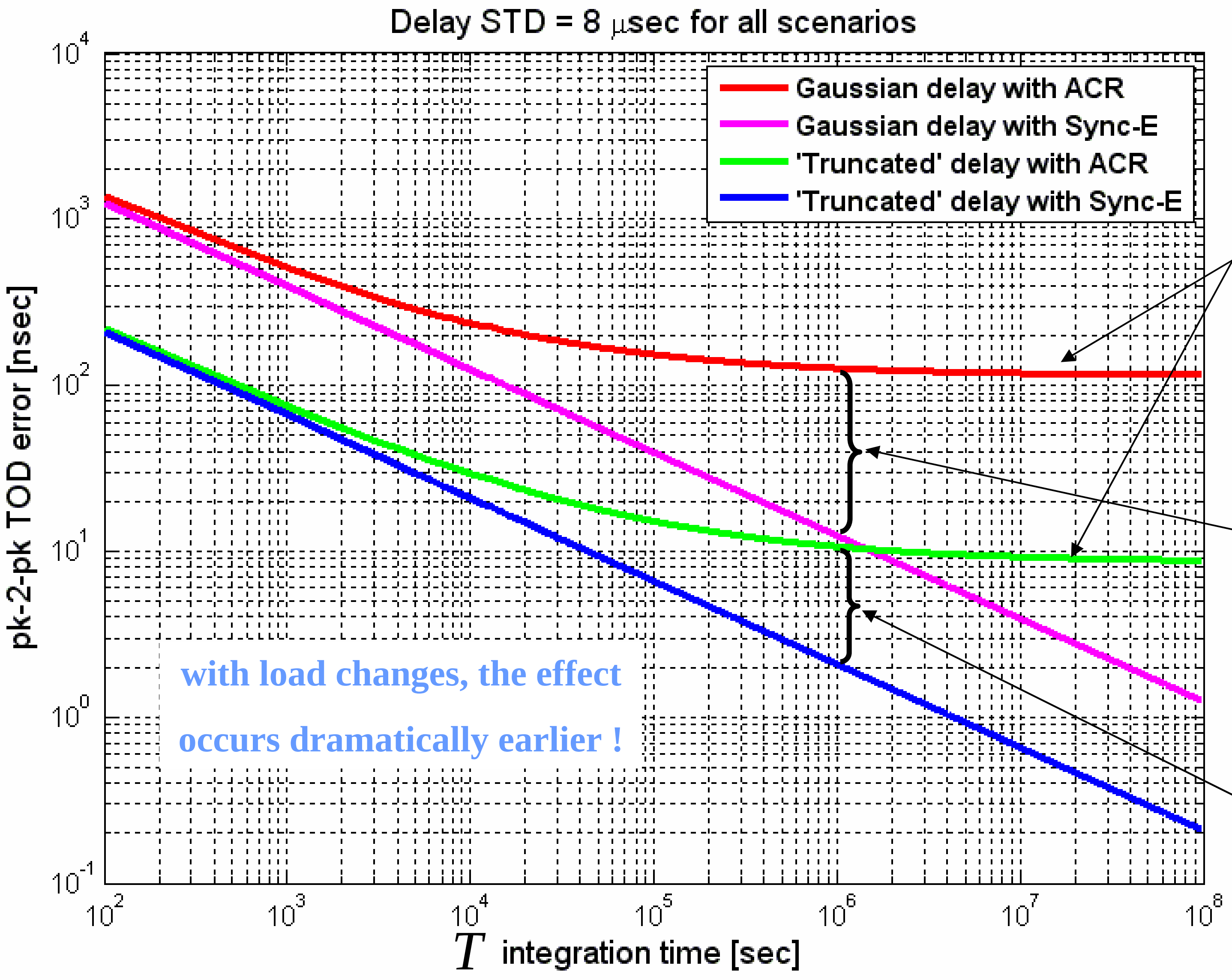
ACR's phase error contribution assuming linear PLL model and constant 80% load

input-output BW ratio

only 4T since one fifth have min delay

Now, applying Sync-E

Asymptotic behavior



with load changes, the effect occurs dramatically earlier !

lower bound due to ACR non-perfect frequency lock

> order of magnitude improvement

about an order of magnitude improvement

Simulations

We performed a range of simulations to test our proposal

For each simulation we ran two time recovery algorithms

- full 1588 algorithm - frequency and time recovery
- time-only 1588 algorithm - frequency taken from SyncE

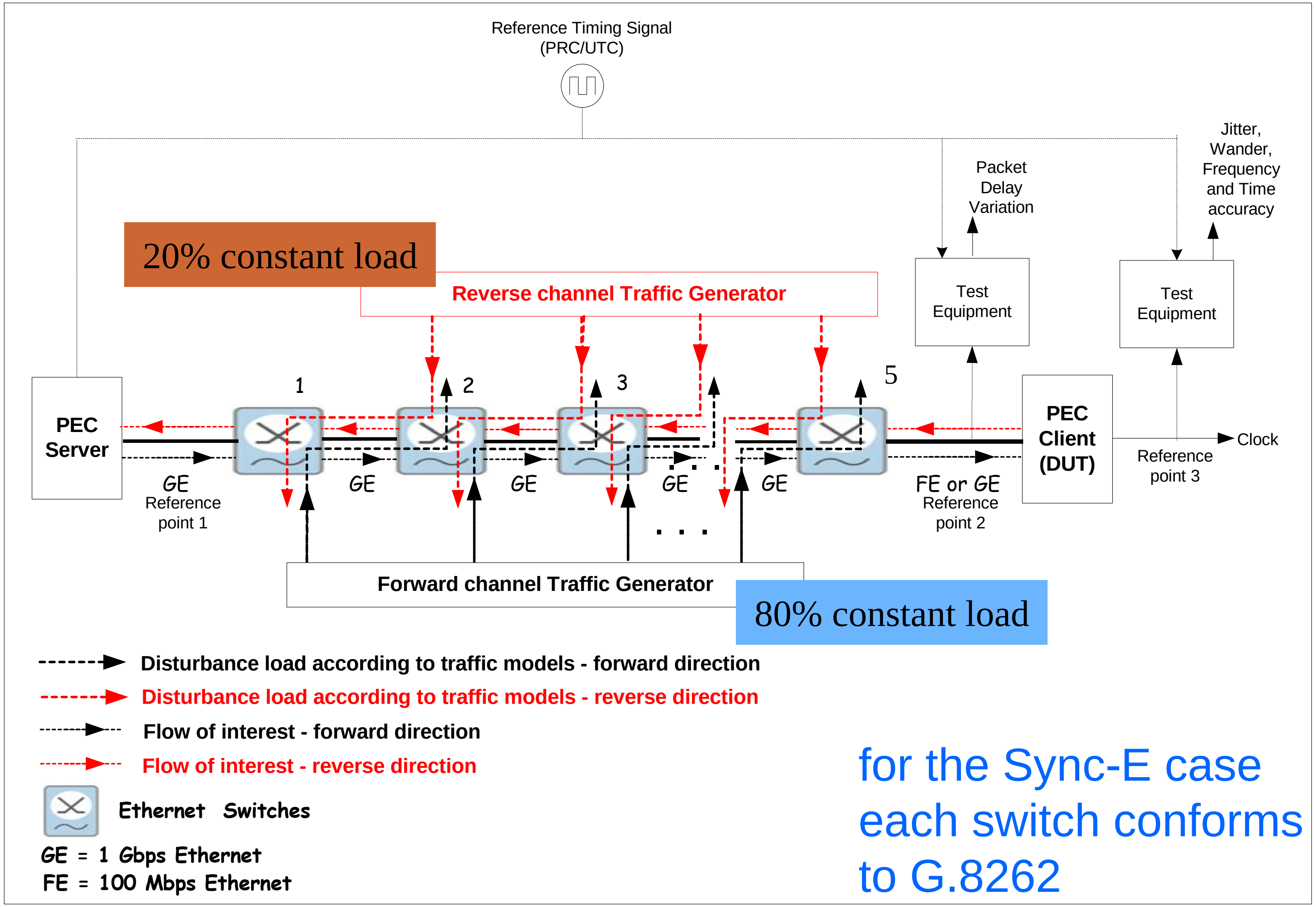
The update rates were taken to be:

- Master->Slave - 100pps
- Slave->Master - 10pps

We ran two network scenarios

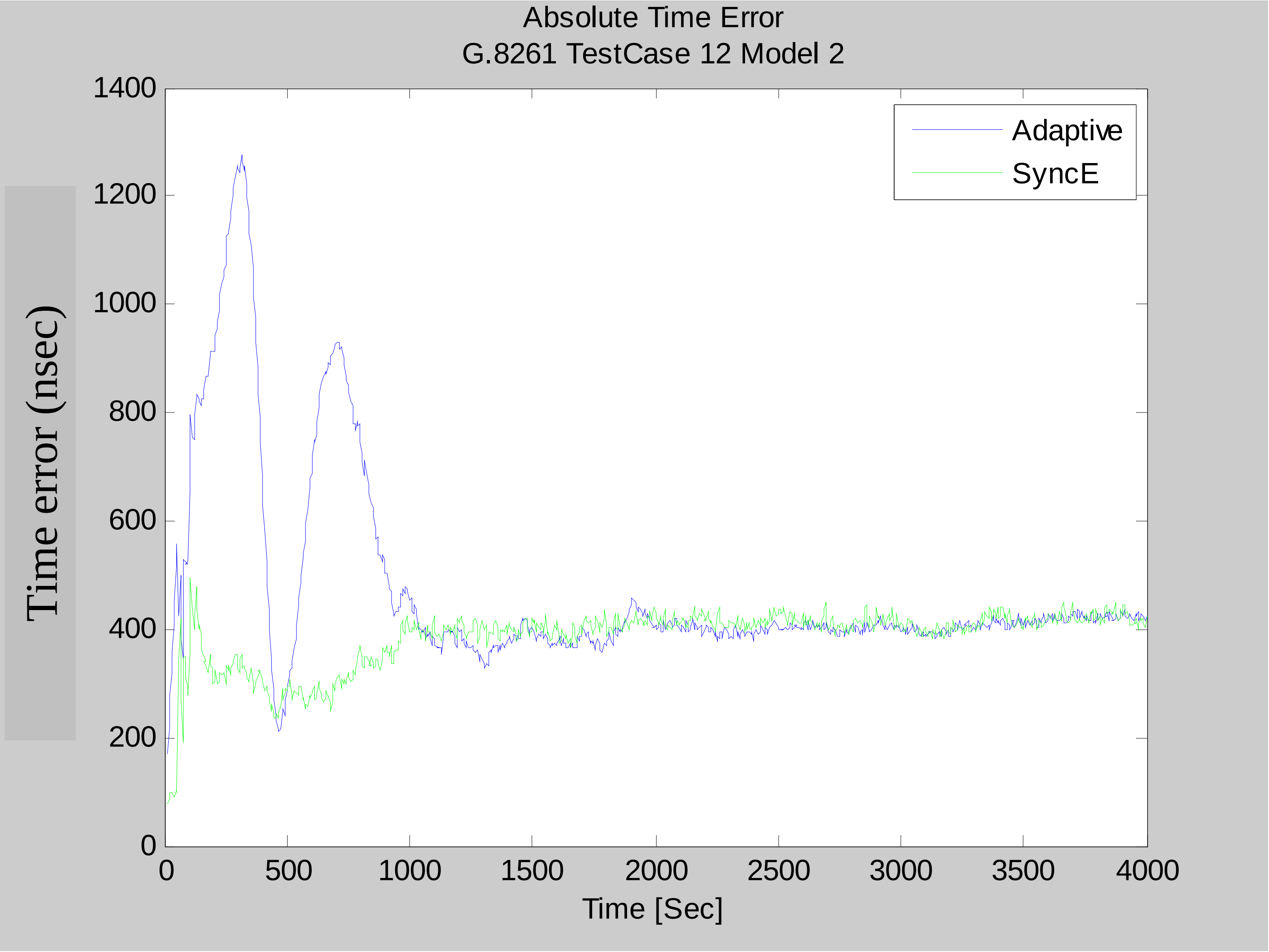
- G.8261 Test scenario for two-way protocols
- Gaussian PDV

Simulation setup (based upon G.8261 two-way testing setup)



for the Sync-E case each switch conforms to G.8262

G.8261 symmetric scenario



don't worry about
400 nsec offset

Gaussian PDV scenario