## Residential Ethernet:

# Time-of-day timer synchronization

Maintained by David V James



This is an RE slide set, with many slides created by DVJ.

RE stands for “Residential Ethernet”, and 802.1 study group.

Alternative names abound, which mean the same thing:
  Residential Bridges, Residential Bridging
  AV Bridges, Audio/Visual Bridges, etc.

Credit is due to many others, whose reviews/comments evolved this concept.

# 802.1 Residential Bridges Requirements

- Inexpensive (insignificant incremental bridge costs):
  - Standard bridge crystals are sufficient
  - Snapshot times are sent within the next frame
  - Same snapshot HW for 1588 + and 802.1-RB
- Efficient: 64-byte frames are preferred (must be less than 72 bytes)
- Accurate:
  - Not response-time dependent (periodic symmetric transmissions)
  - Limited only by HW snapshot accuracies
  - Endpoint accuracies are quickly attained (<2.5 seconds)
- Robust:
  - Concurrent grand-master selection and synchronization
  - No administration required (plug-and-play)
  - Minimal grand-master handover transients
  - Reasonable rogue-frame lifetimes (<2.5 seconds)
  - Grand-master precedence is higher-level selectable
  - Seconds never overflow and are 1588 compatible
- Sampling times:
  - 10 ms offset adjustments (sufficient firmware execution time)
  - 100 ms rate adjustments (rate difference errors are insignificant)

## 802.1 Residential Bridges Assumptions

- Bridges have 100 PPM quartz crystals
- Bridge cascade depth is limited to 7
- The range of 1588 seconds range is 80 bits
- Ethernet cable scoping is limited:
  - Full-duplex
  - PAUSE is never used
  - 100 Mbs and higher (not 10 Mbs)
- Bridges and endpoints have uPs
  - 10 ms is “doable” by the simplest of uPs
  - 10 ms is “tolerable” by the fastest of PCs

February 21 | IEEE 1588/802.1 Joint Meeting NIST in Gaithersburg, MD | 3

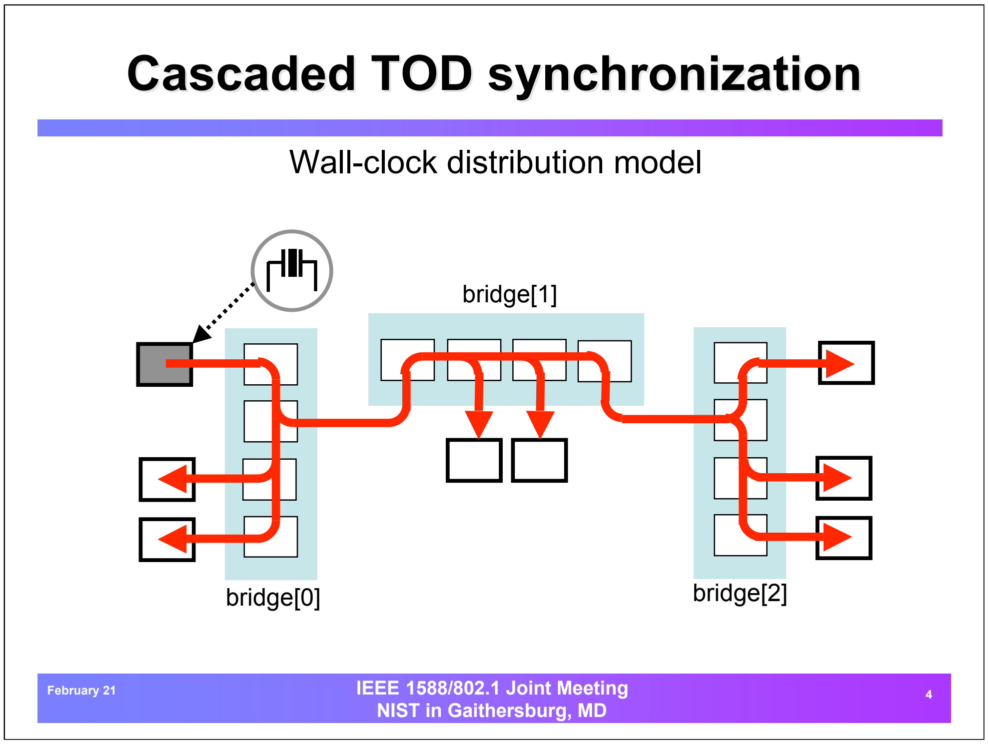
The main consideration is to be cheap:
 Easily integrated into consumer bridges.
 Easily integrated into consumer end-points.

Simplicity is valued, since fewer mistakes are likely to be made.
 Delayed snapshots avoid time-critical processing requirements
 Symmetric processing reduces process-sequence steps

Precision is valued, since audio-files are very sensitive
 ps jitter desired, although theoretical and filtered is probably OK
 Minimal-glitch handovers due to accurate/symmetric compensations

Robustness is also critical, with minimal-glitch handovers.

Is this realistic? Fortunately, we can benefit from existing IEEE 1588 knowledge.
 without its legacy constraints (e.g., broadcast CSMA/CD).




The clock-distribution scheme is that of IEEE 1588.

The grand clock master (grand master) is the station whose time-of-day clock is the reference.

From a logical perspective, the clock master broadcasts the current time-of-day to the attached stations.

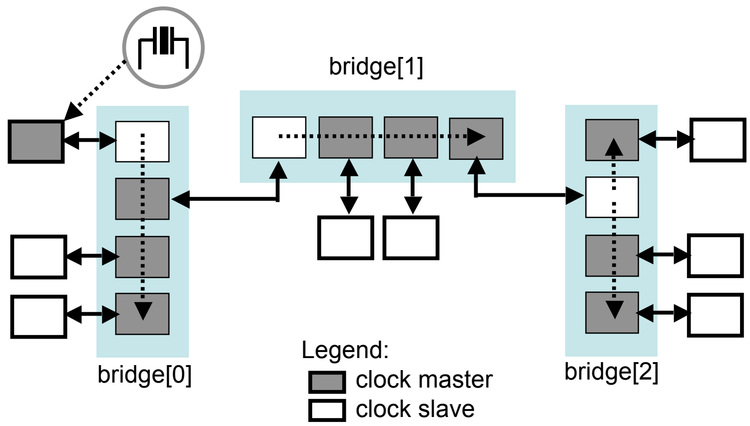
From a physical perspective, such a multicast time distribution would be inaccurate:

1) There may be source transmission delays.
2) There may be bridge forwarding delays.

Therefore, a more-precise synchronization protocols is used.

# Cascaded TOD synchronization

Cascaded adjacent-synchronization hierarchy



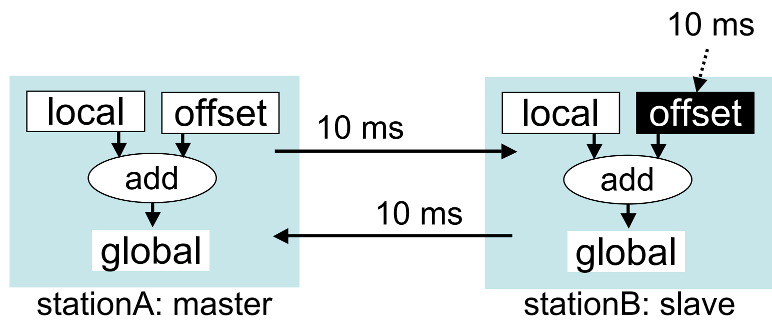



To avoid propagation-time inaccuracies:
- Synchronization is done on a point-to-point basis
- Internal bridge distribution (port-to-port) is “magical” and beyond our scope.

A key point: its no enough to standardize how one identifies slave errors:
- One must address how the slave eliminates errors.
- One must address how cascaded slaves compensate cumulative errors.

# Adjacent-station synchronization

Offset value adjustments



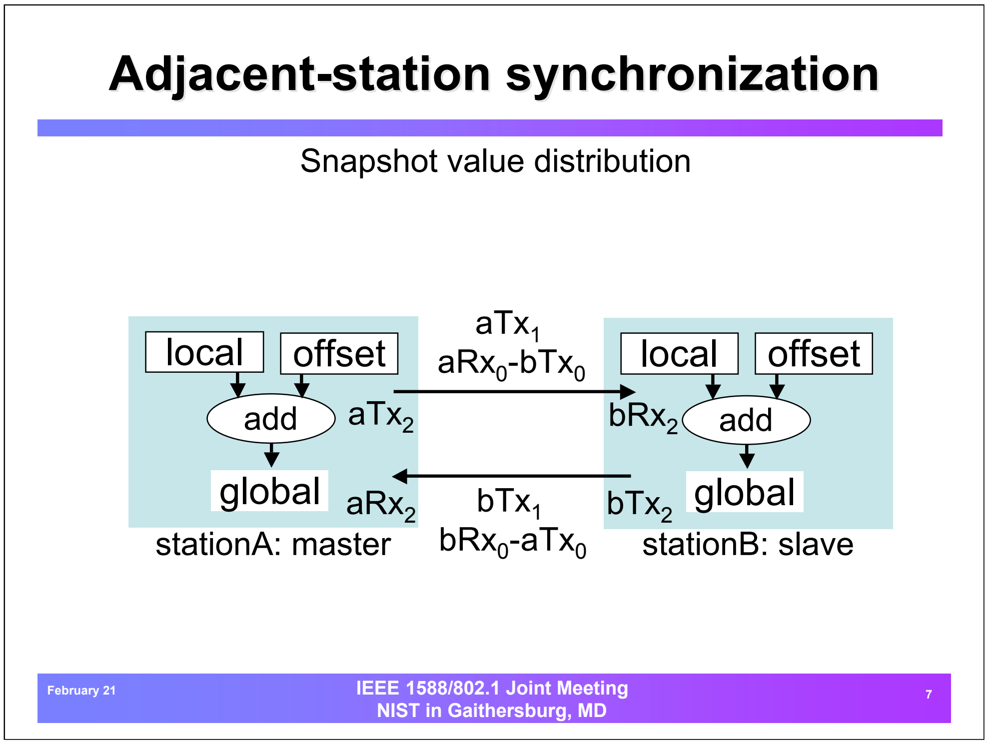



Initial studies indicate a faster update rate is practical.

While 1ms is practical, a slower 10ms allows the cheapest of microprocessors.

The primary purposes of these transfers are for:
- Grand-master selection (reduces rogue frame stabilization times)
- Offset adjustments, to force current time acceptance

(For precise synchronization, rate adjustments may also be required.)

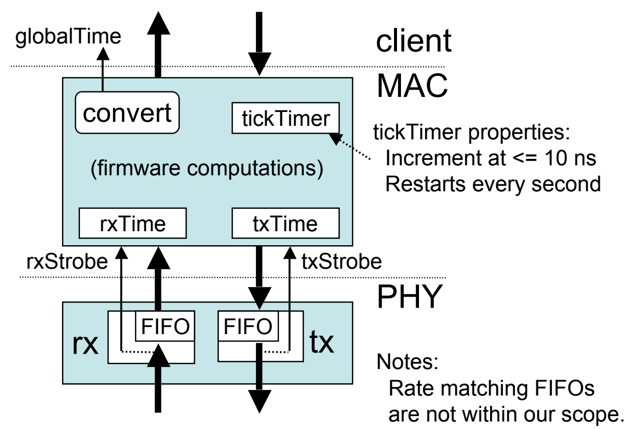



Time snapshots are best sent in the next “cycle”.
  Cheap: easily implemented in hardware (and possibly firmware).
  Precise: observations are more precise than predictions

In the case of stationB, the aTx and aRx values must be sent, since these were measured by stationA and are not known to stationB.

The value of bTx is (in concept) known to stationB and need not be transmitted. However, for simplicity, transmission of this value allows it to be more easily affiliated with the same-cycle indexed aRx value.

# Minimalist HW design model






What is the hardware design model?

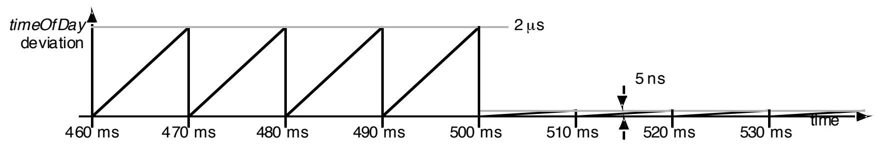
Simple hardware to snapshot the arrival/departure times
 The local time reference (a minimum frequency is sufficient) is OK.

External communications are through normalized time values.

Firmware performs the conversions, frame formatting, etc.

Several strategies for precise snapshots are possible:
 FIFOs add ambiguity (existing hardware)
 The MAC arranges for FIFOs to be nearly empty, at critical times
 The PHY signals the actual clockSync arrival/departure times

# Uncompromised precision






With a 200PPM clock deviation, offset adjustments have limitations.

The problem is the drifts, due to clock-frequency differences.

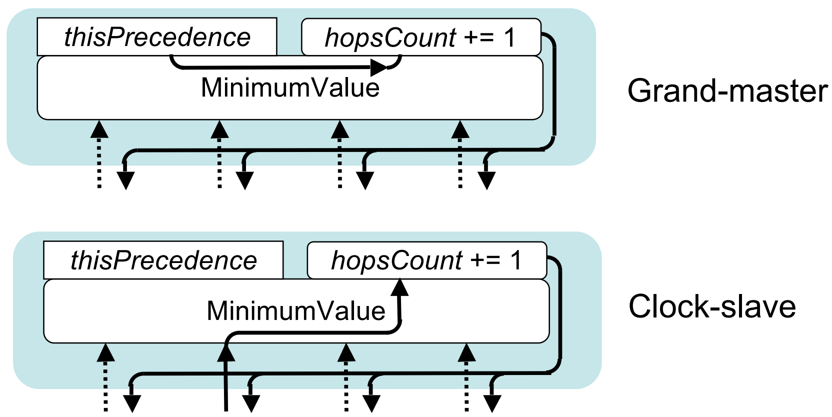
Thus, the most apparent solution is to have the “slave” match the rate of the master.

Frequency deviations are easily measured, from time snapshots measured over a larger time interval (100ms, perhaps).

Various frequency compensation schemes are possible; the above waveform illustrates one possible scheme.

## Grand-master selection protocol

Grand-master precedence comparisons

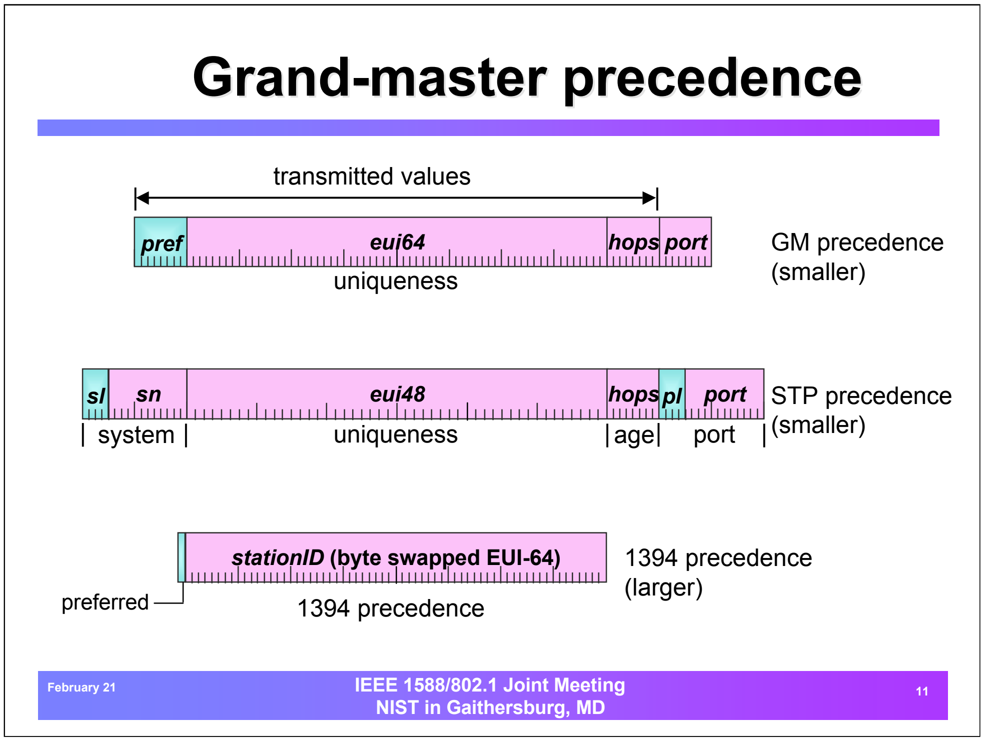



February 21 | IEEE 1588/802.1 Joint Meeting NIST in Gaithersburg, MD | 10

For the grand-master selection, spanning-tree protocol is popular in 802.1.

The “minimum” value is distributed throughout the network.

A hopCount value breaks ties, in favor of the shortest-span lengths.




What is the precedence value?
The precedence numbers must be unique, so that only one clock master will be selected. For this purpose, the station address is sufficient.

To communicate preferences, a *pref* (station priority) value is provided. This overrides the MAC address, allowing users to assert their preferences. This weighting can be accessed through the MIB.

For stations with equal *pref* values, the *eui64* becomes the tie breaker. This resolves grand-master in a unique (but somewhat arbitrary) manner.

For ports on a station, the *hops* value selects port with the shortest distance from the grand-master, measured in hops-through-bridges. In the case of a tie, the port number selects the preferred path.

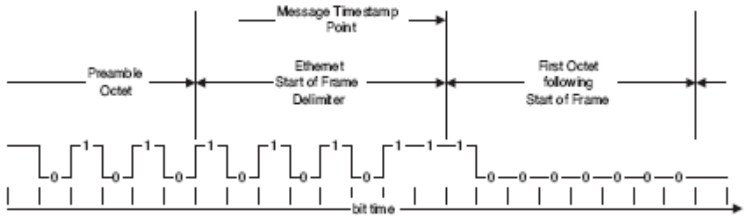
The lowest numerical value has the highest precedence.

Default weighting of *pref* is the mid-range numerical value.

The setting of other weights is a higher level protocols and is beyond the scope of this standard.

The 802.1 spanning-tree protocol assumes “smart” things set the precedence, “simple” things do the comparisons. This is similar, but simpler while remaining compatible with 1394 and networks with 64-bit MAC addresses.

# Timing specifics…




**(from IEEE 1588-2002, subclause D.1.1, page 127)**



What (exactly) is the frame arrival/departure time?

Depends on the physical layer details.

Some are already specified, in IEEE 1588.

Parallel-bit transmission schemes may need clarifications.

1G CAT-5
10G (in general?)

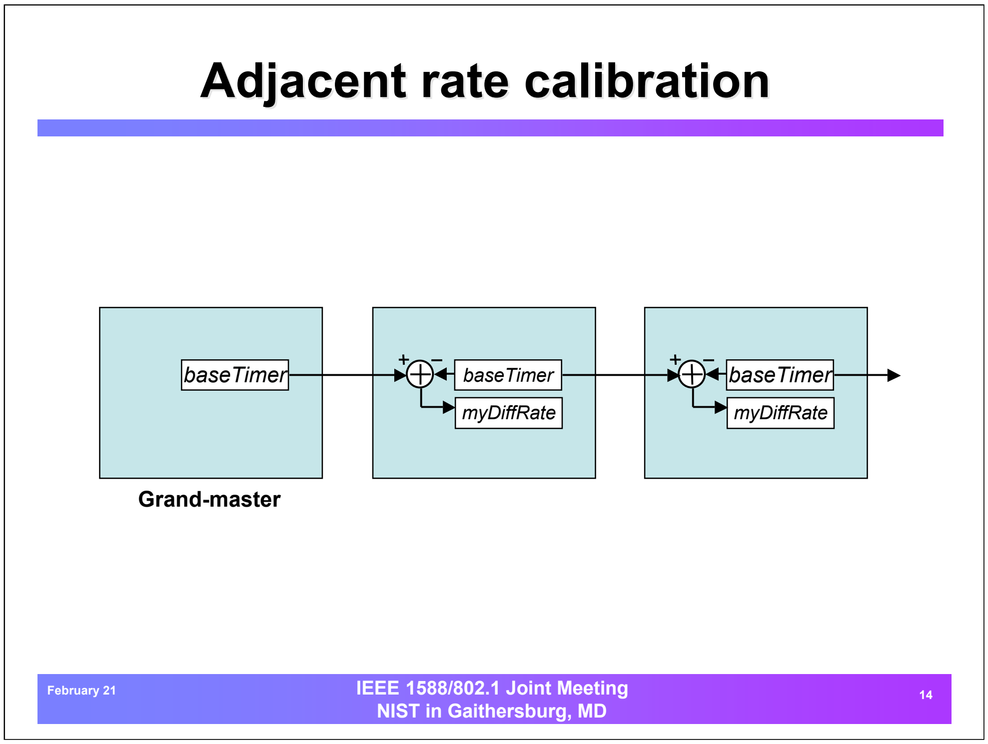
## Rate adjustments

Compute nearest neighbor errors
  – Based on adjacent baseTimer information
- Cumulative values are computed
  – Rate differences are added in a cascaded fashion
- The grand-master “timer” is assumed to be correct
- Rate changes after grand-master changes
  – Saving rate offsets complicates the protocols
  – Could degrade the new-grand-master accuracy



Rate synchronization involves:
100 ms to track temperature differences
longer than 10ms to reducing sampling inaccuracies

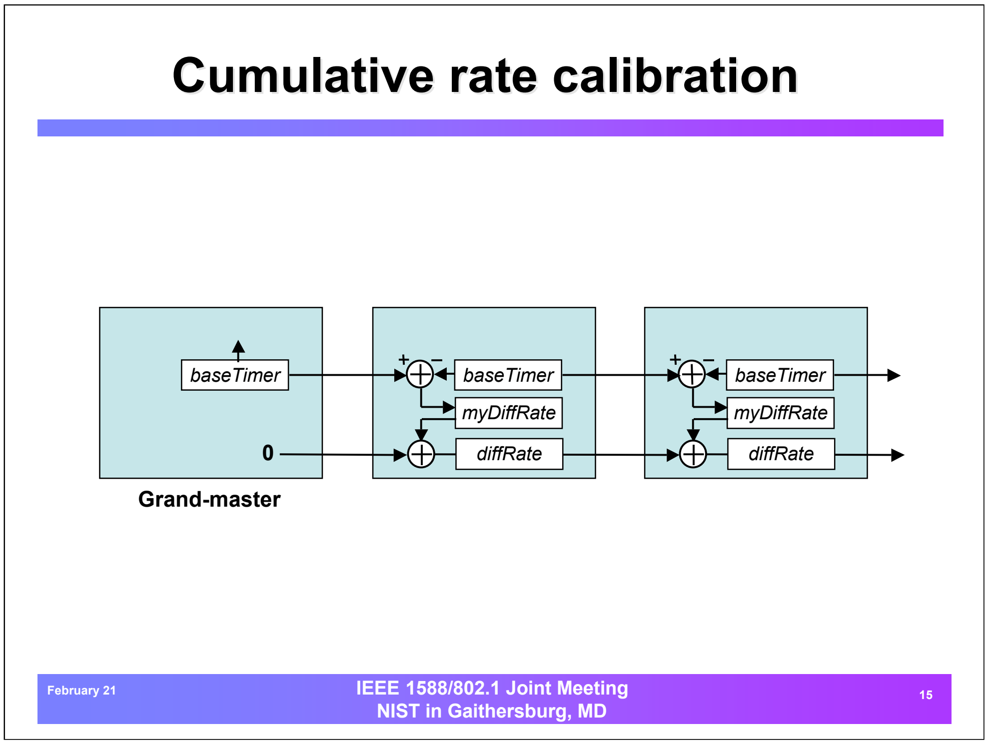
In general, timers are never reset or changed
scaling values can be multipliers
offsets values can be additions




Adjacent station offset computations involve periodic time forwarding.

The *baseTimer* value is a fixed-rate timer, with identical nominal rates.
In reality, may be any convenient HW timer, with firmware conversions.

Based on neighbor interchanges, the neighbor drifts are computed.

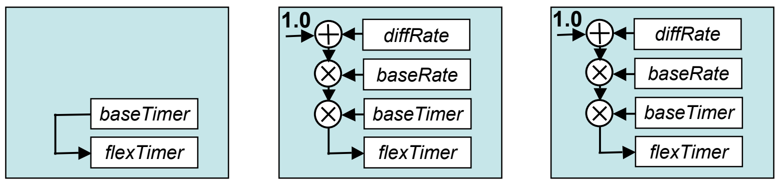



The adjacent-neighbor drifts are accumulated downstream.

Thereafter, each station's *diffRate* value accurately represents the drift from the grand-master, rather than one's adjacent neighbor.

While oftentimes illustrated separately, both are done concurrently:
  Compute the next value of *myDriftRate*
  Accumulate the cumulative *driftRate* values

# Rate-based adjustments




Grand-master

February 21

IEEE 1588/802.1 Joint Meeting
NIST in Gaithersburg, MD

16

The accumulated rate differences are applied to flexTimers.

In the grand-master, they are both the same (although actually in error).

For the other clock-slave stations, the flexTimer runs possibly faster or slower.

The implementation of these multipliers is conceptual only and only computed on demand (not every clock tick).

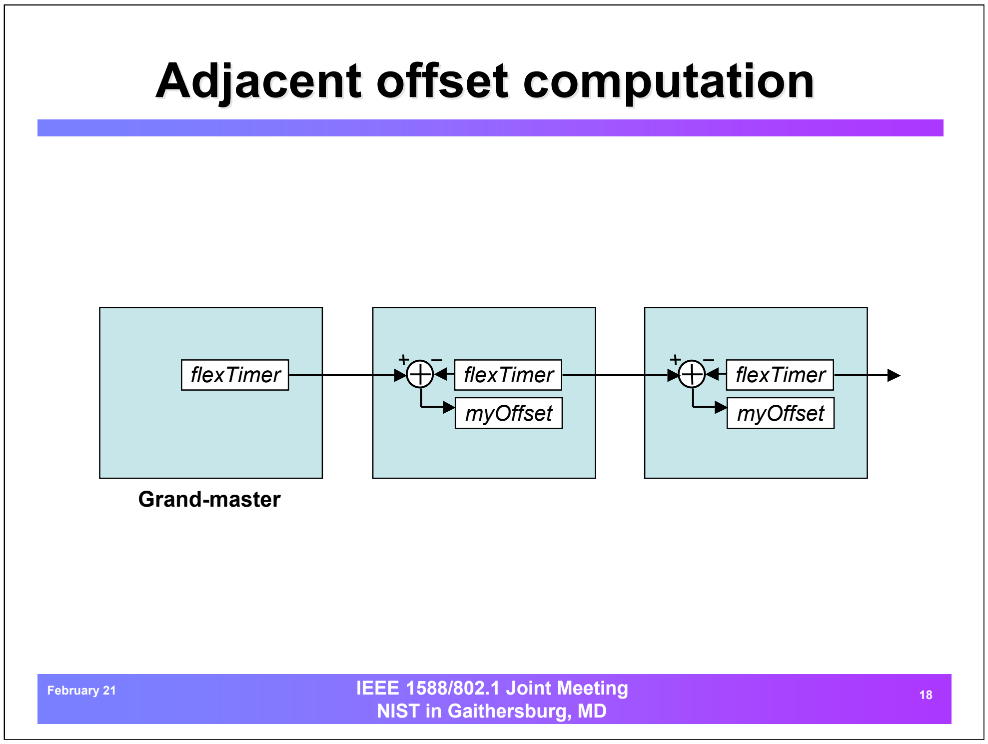
## Timer offset compensation

Compute nearest neighbor errors
  – Based on adjacent timer information
- Cumulative values are computed
  – Offsets are added in a cascaded fashion
- The grand-master “timer” can be set/adjusted
  – How-to details are beyond our scope
  – The actual value is accumulated, like other offsets
- Responsive to grand-master changes
  – Each receive port offset is constantly computed
  – Handover calibration is thus “precomputed”



Timer synchronization involves:
  offset-time adjustments, 10 ms for short-term coarse-grained tracking
  offset-rate adjustments, 100 ms for long-term fine-grained tracking

Again, timers are never reset or changed
  offset values are added, these can be changed

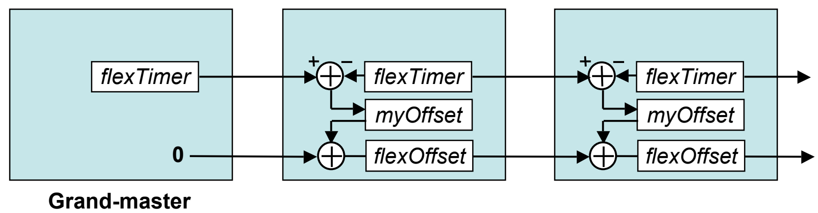



Adjacent station offset computations involve periodic duplex exchanges.

Based on neighbor interchanges, the neighbor differences are computed.

The *flexTimer* value is adjusted-rate timer, with identical nominal rates.
In reality, there will be minor inaccuracies.
Nearly insignificant, due to the longer sampling times.

# Cumulative offset computation

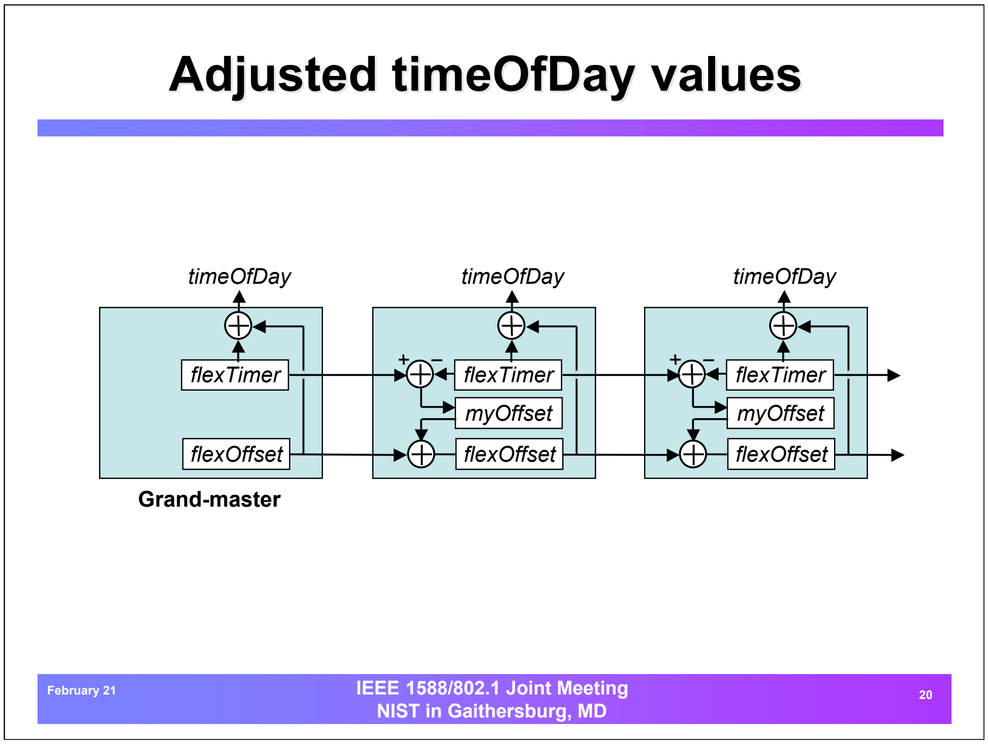





The adjacent-neighbor offsets are accumulated downstream.

Thereafter, each station's *flexOffset* value accurately represents the offset from the grand-master, rather than one's adjacent neighbor.

While oftentimes illustrated separately, both are done concurrently:
  Compute the next value of *myOffset*
  Accumulate the cumulative *flexOffset* values



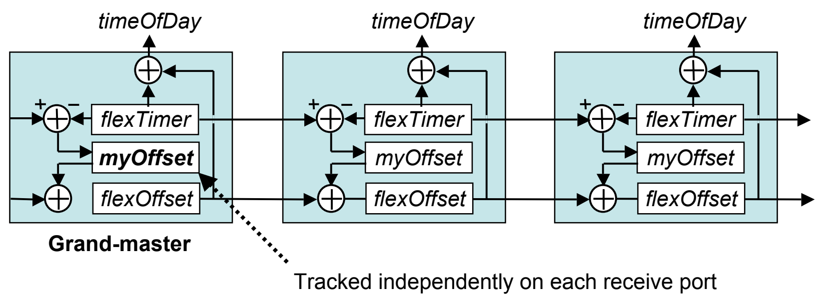

Cumulative offset values allow the slaves to track the grand-master.

However, the grand-master's time may be set incorrectly.

Changing the grand-master's *flexTimer* would have bad consequences, causing potentially large discontinuities in *myOffset* values.

Instead, a distinct *flexOffset* value is distributed.
While the time discontinuity is unavoidable, the computed myOffset values are undisturbed by the transient.

# Seamless grand-master handover




February 21 | IEEE 1588/802.1 Joint Meeting | 21
NIST in Gaithersburg, MD

What if the grand-master changes?

This could by:
  another station being added
  a grand-master preference being changed

Or, the path from the grand-master could be changed.

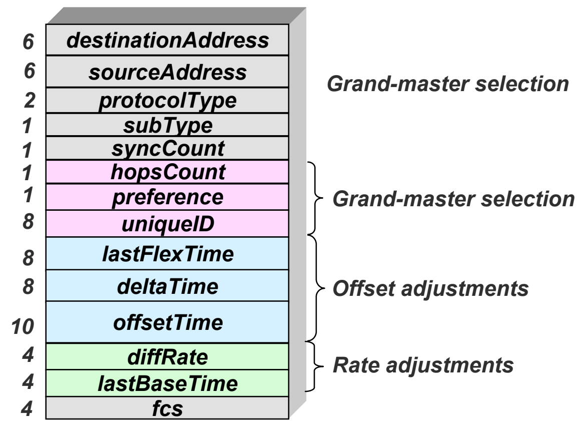
To reduce such transient effects, the grand-master monitors receive ports.

Its receive port independently/constantly computes *myOffset* values.

When a handover occurs, the nearly correct *myOffset* value is available, so *flexOffset* discontinuities are minimal.

This assumes the new grand-master maintained synchronization by:
  Participating (as a clock-slave station) in synchronization protocols.
  Maintaining synchronization through other means (e.g., radio station).
  Maintaining precise time through physical design (e.g., atomic clock).

# Frame format




February 21 — IEEE 1588/802.1 Joint Meeting, NIST in Gaithersburg, MD — 22

What is contained within each frame?

The standard header.

A syncCount to detect missing frames, thus avoiding last-cycle sampling errors.

Values for the grand-master selection.

Values for the offset adjustments

Values for the rate adjustments

And, this all fits in a minimum-length 64-byte frame!
(Thus, there is no advantage to having “smaller” frames.)

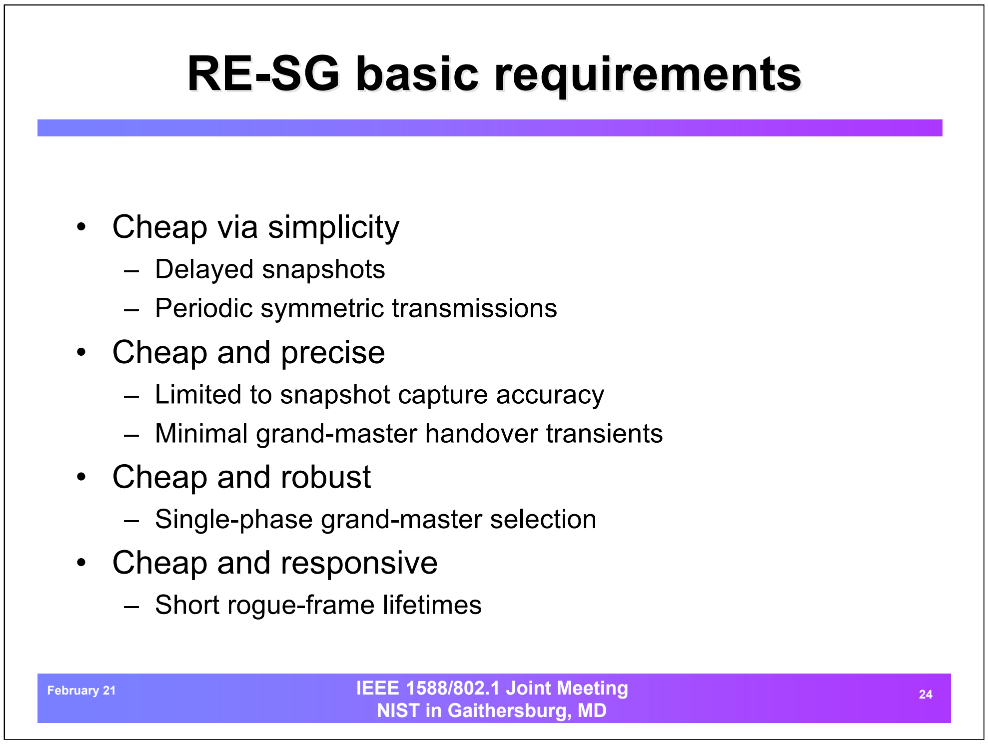
# Backup slides for

## Residential Ethernet:
## Time-of-day timer synchronization

Maintained by David V James



Backup slides, for pre-review and-or extended question responses.



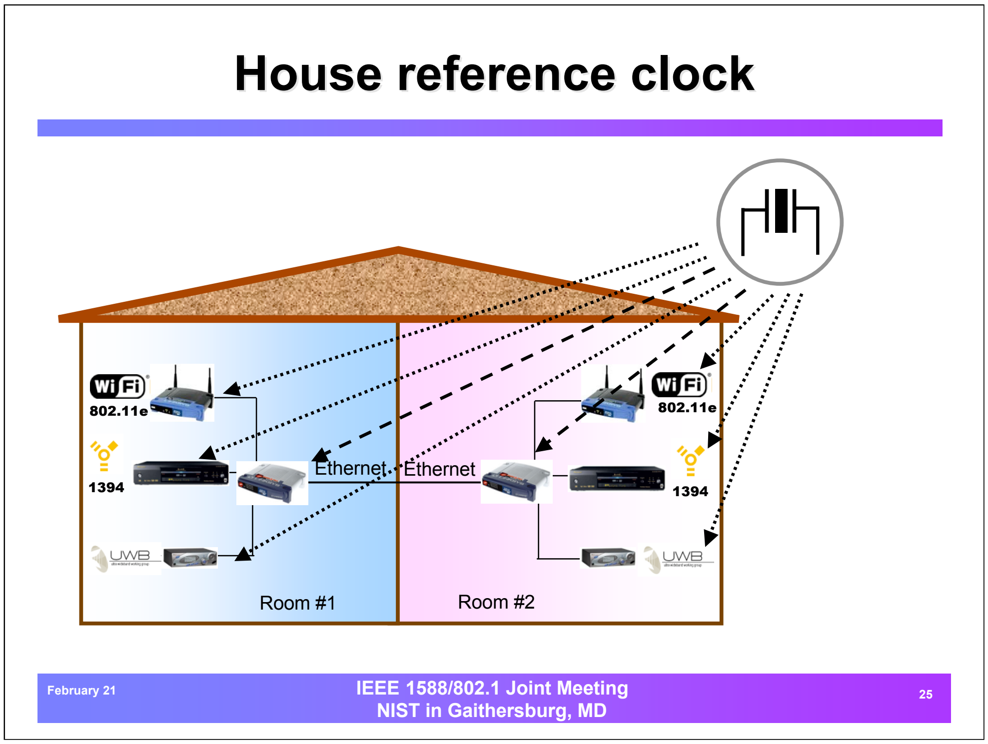

The main consideration is to be cheap:
  Easily integrated into consumer bridges.
  Easily integrated into consumer end-points.

Simplicity is valued, since fewer mistakes are likely to be made.
  Delayed snapshots avoid time-critical processing requirements
  Symmetric processing reduces process-sequence steps

Precision is valued, since audio-files are very sensitive
  ps jitter desired, although theoretical and filtered is probably OK
  Minimal-glitch handovers due to accurate/symmetric compensations

Robustness is also critical, with minimal-glitch handovers.

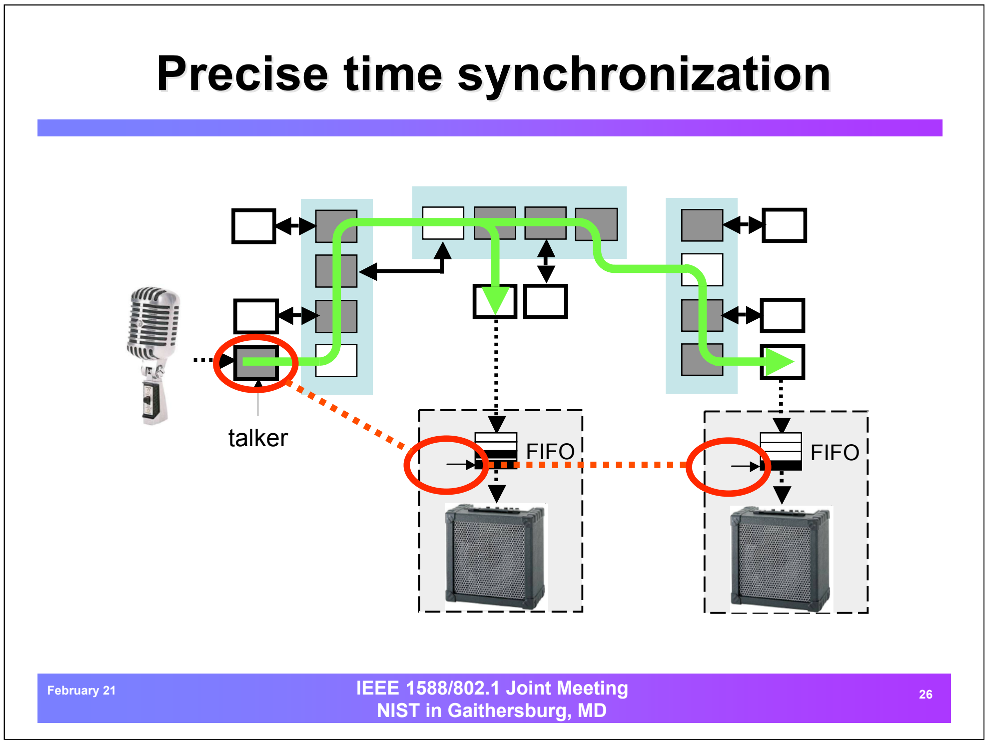
Is this realistic? Fortunately, we can benefit from existing IEEE 1588 knowledge.
  without its legacy constraints (e.g., broadcast CSMA/CD).




In support of synchronous transfers, all RE devices are assumed to have the same impression of time.

For this presentation, assume an 8kHz cycle time, although a decision on this value has not been finalized.

Requirement: 8kHz cycle frequencies are locked and the “same

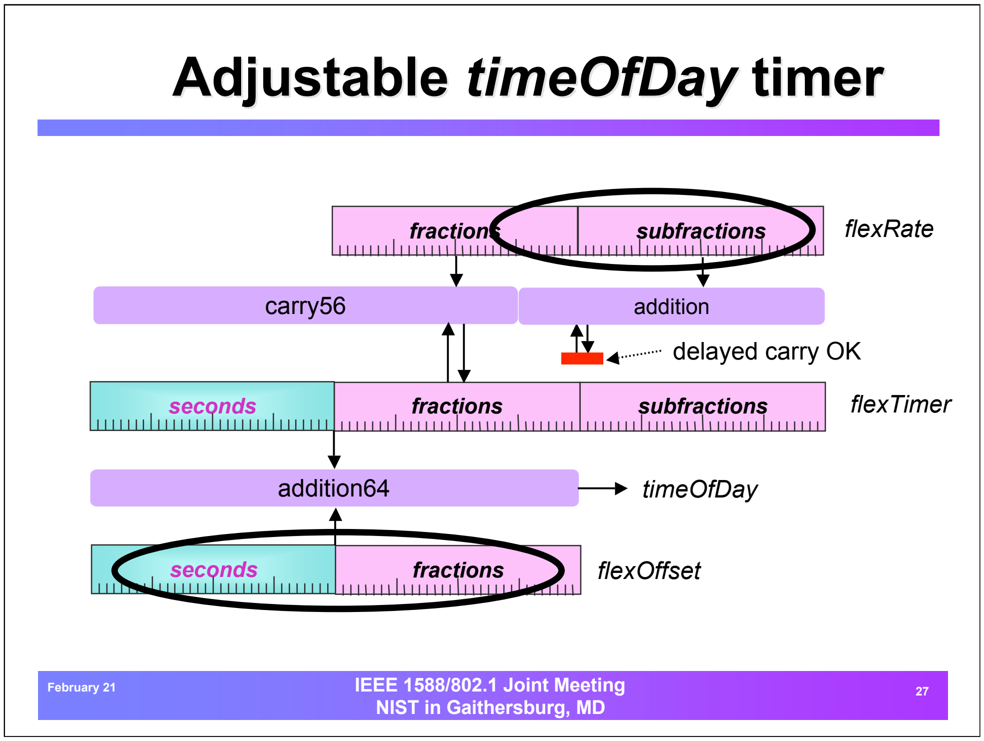



Data arrives early; data is gated at its presentation time.
(Each frame has a presentation time stamp.)

Bridge reclocking has a relatively modest clock-sync accuracy requirement, where microsecond deviations could be acceptable.

Source-data and presentation-data clocking requirements are more severe.

1) Frequency drift is unacceptable, since dropped/replicated values are audible.
2) Presentation time jitter is sub nanosecond, based on slew rates and D/A accuracies.




The basic model assumes rate-adjustable and offset-adjustable clocks.
Since bridge values are rarely needed:
  Any high-enough frequency clock is sufficient
  Value can be converted “on-demand”.

The rate can be adjusted, by updates to flexRate.

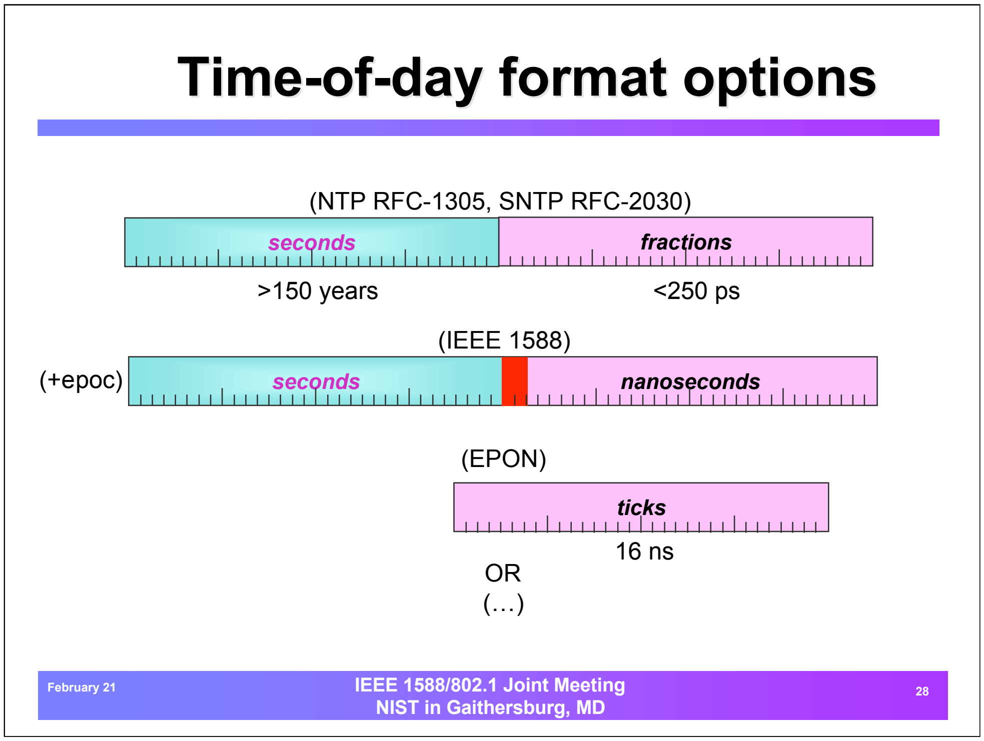
The offset can be adjusted, by changes to flexOffset.

How is a fractions-of-second timer conceptually implemented?
Clocks are typically multiples of something else!

The clock can tick at a natural rate, and the tick-size need-not be “one”.

For example, consider a clock that is updated with a 16ns clock.
  The update value is 68.719476736
  Since the LSB is insignificant, the carry can be delayed

The less significant bits provide precise rate adjustment, for precise tracking




What is synchronized?

Preferably a binary seconds and fractions-of-seconds value.

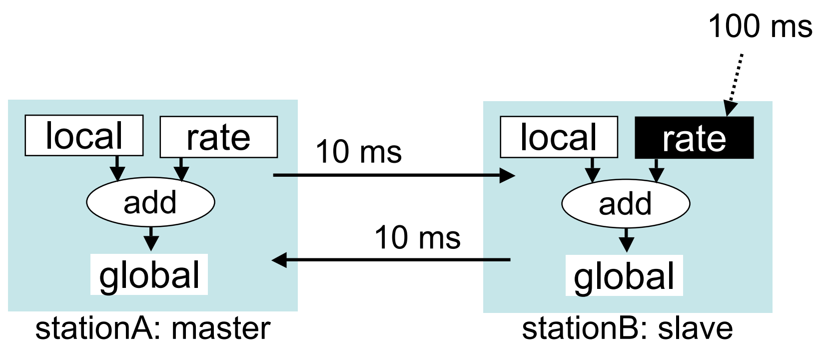
- Doesn't overflow within our lifetimes.
- Time resolution induced errors are insignificant.
- Easily added and subtracted.
- Readily converted to other formats…

Other formats are also possible:

- because we have 10 fingers
- a multiple of the cycle frequency
- ….

# Adjacent-station synchronization

Differential rate adjustments





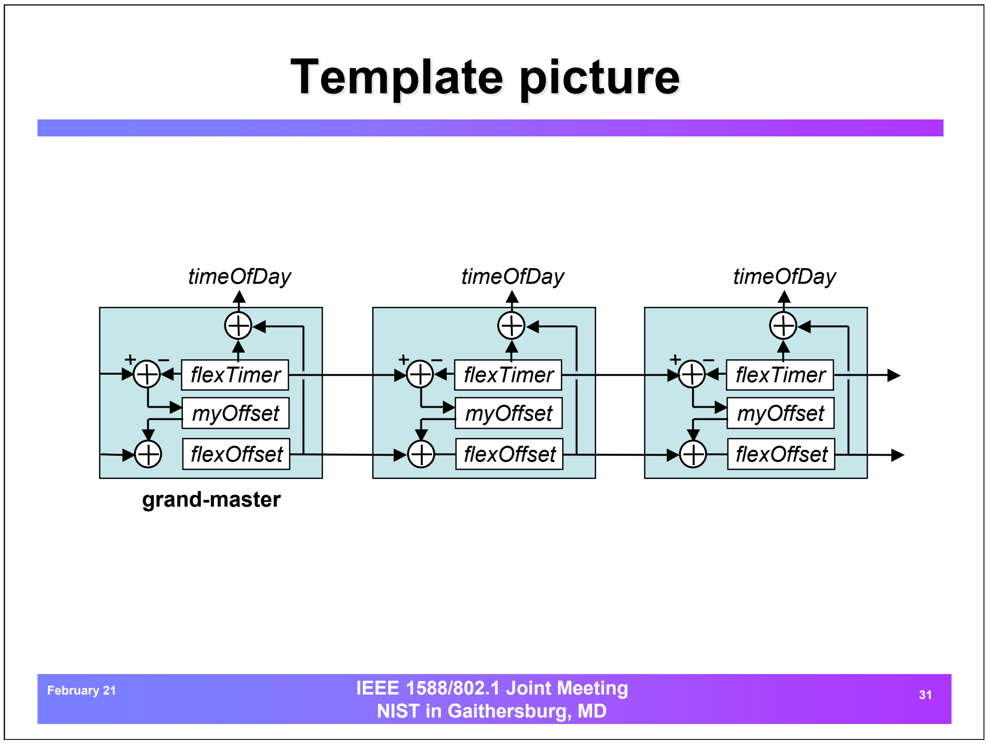

Adjacent-station synchronization is preferrably done less frequently, so that the time-snapshot errors are smaller.

For a 10x longer interval, the effective frequency error is nearly zero.

Yet, the sampling interval is small enough to compensate for modest time-varying temperature drifts.

# Basic requirements

- KISS (keep it simple, stupid)
  - Delayed snapshot processing
  - Periodic symmetric transmissions
  - Etc., etc.
- NTP (RFC-1305) and SNTP (RFC-2030)
  - Definition of the 64-bit time-of-day value
- For a detailed summary, see:
  - http://dvjames.com/esync
  - dvjTimeSync2005Dec12.pdf (or later revision)




What is the hardware design model?

Simple hardware to snapshot the arrival/departure times
  The local time reference (a minimum frequency is sufficient) is OK.

External communications are through normalized time values.

Firmware performs the conversions, frame formatting, etc.

Several strategies for precise snapshots are possible:
  FIFOs add ambiguity (existing hardware)
  The MAC arranges for FIFOs to be nearly empty, at critical times
  The PHY signals the actual clockSync arrival/departure times