FTTM Change Summary for Comment Resolutions
Major categories of changes

- Highlight “integrity” (per proposed PAR for P802.1ASed)
- Uniquely identify FTTM operation with single time input
- Change FTTM inputs from “domain” to “time”
- Only have one default algorithm
- Use 802.1Q’s state-machine structure
- Make states atomic w/o infinite state looping
- Define all variables/parameters
Highlight Integrity

• Added subclauses on availability and on integrity and trust

19.2.1 Availability of time

The continuous availability of time is enhanced by redundancy. For gPTP, this redundancy can be implemented by using multiple time domains, multiple time distribution paths, and multiple gPTP instances in Bridges and end stations.

19.2.2 Trust and Integrity of time

For this standard, a trusted time is one that passes a specified criterion that identifies it as being within a safe bound of a non-faulty time and is, thus, safe to use. This establishment of trust gives integrity to the time.

For gPTP, trust, and hence integrity, can be established through the comparison of the times coming from independent time sources and the observation that they match within the specified criterion. See 19.2.7.4 for an example of such a criterion.
Highlight integrity, operation with single time

- Replaced “redundant” with “trust” in the algorithms
- Added the following to FTTM introduction (19.3)

The availability and integrity scenarios supported by the FTTM are listed below.

- Enhanced availability and integrity scenario:
  This scenario operates with multiple times and multiple domains.
  In this mode, the FTTM supports increased availability and integrity.

- Enhanced availability and limited integrity scenario:
  This scenario operates with multiple times and a single domain.
  In this mode, the FTTM supports increased availability and potential time distribution integrity, but not GM integrity.

- Regular availability and no integrity scenario:
  This scenario operates with a single time and, hence, a single domain.
  In this mode, the FTTM does not support increased availability or integrity.
Change FTTM inputs from “domain” to “time”

- Functional block diagram redrawn
- DDSA renamed to DTSA
- IDSA renamed to ITSA
Only have one default “algorithm”

• “Closest-pair algorithm” removed
• “Mid-value selection algorithm” retained and renamed to “Mid-value time-index selection process” (MVTISP)
  • DTSA and ITSA are “algorithms”
  • MVTISP is the “process” used by the default DTSA and the default ITSA
  • “time-index” clarifies that the index is selected and not the time
State-machines

- **Use 802.1Q’s state-machine structure**
  - New state-machine defined for default DTSA
  - New state-machine defined for default ITSA
    - Formerly called the default FTTM state-machine
  - Mid-value time-index selection process (MVTISP)
    - Has no states
    - Is run in the SELECT_TIME_INDEX state of the new default DTSA/ITSA state-machines
    - Is shown by pseudo-code

- **States are atomic without infinite looping**
  - Default DTSA/ITSA state machines use the ClockTargetEventCapture application interface
  - ClockTargetEventCapture.invoke events are used to gather new time information (via ClockTargetEventCapture.response) and start a new round of state transitions
Default DTSA state machine

BEGIN

INITIALIZE

Itsa_mode = FALSE
num_time_indexes = user assigned
med_time_index = NQ
med_next_time_index = NQ
prev_trust_status = NO_TRUST

For (x = 1; x <= num_time_indexes - 1, x++)
{
For (y = x+1; y <= num_time_indexes - 1, y++)
{
maxAs[x][y] = user assigned
hyst[x][y] = user assigned
prev_time_index_pair_status[x][y] = UNTRUSTED
}
}

ASSIGN LOOP VARIABLES

For (x = 1; x <= num_time_indexes - 1, x++)
{
isSynced_status[x] = isSynced[x]
gmPresent_status[x] = gmPresent[x]
ToD[x] = timeReceiverTimeCallback[x]
}

UCT

SELECT TIME_INDEX

Run the mid-value time-index selection process (MVTISP)

DTSA_OUTPUT

DTSA_OUTPUT = ClockTarget Interface corresponding to selected_time_index = MEDIAN_TIME_INDEX

UCT
Default ITSA state machine

BEGIN

INVOKE_CLOCKTARGET_IF

Simultaneously invoke gathering of all ClockTarget interface results by using:

ClockTargetEventCapture.invoke

reception of results from all ClockTarget interfaces via:

ClockTargetEventCapture.result

and optionally:

ClockTargetPhaseDiscontinuity.result

ASSIGN_LOOP_VARIABLES

For \( x = 1; x \leq num\_time\_indexes - 1, x++ \)

\{ 

isSynced_status[x] = isSynced[x]

gmPresent_status[x] = gmPresent[x]

ToD[x] = timeReceiverTimeCallback[x]

\}

INITIALIZE 1

Itsa_mode = TRUE

num_time_indexes = user assigned

rRlimit = user assigned

rRSDlimit = user assigned

prev_trust_status = NO_TRUST

allowed_TF = ANY_TF

med_time_index = NQ

med_next_time_index = NQ

num_time_indexes != 1

ONE_INDEX

ITSA_OUTPUT = ClockTarget Interface corresponding to time index = 1

INITIALIZE 2

For \( x = 1; x \leq num\_time\_indexes - 1, x++ \)

\{ 

For \( y = x+1; y \leq num\_time\_indexes - 1, y++ \)

maxAs[x][y] = user assigned

hyst[x][y] = user assigned

prev_time_index_pair_status[x][y] = UNTRUSTED

\}

SELECT_TIME_INDEX

NO_TRUST

prev_trust_status == NO_TRUST

ITSA_OUTPUT = ClockTarget Interface corresponding to selected_time_index = NQ

ELSE

\{ 

[prev_trust_status == FREQ_TRUST) &&

(|itsaRateRatio[med_time_index]| < |mean(itsaRateRatio[med_time_index])| + rRlimit) &&

(Std_Dev(itsaRateRatio[med_time_index]) < rRSDlimit)

\}

TIME_TRUST

ITSA_OUTPUT = ClockTarget Interface corresponding to selected_time_index = MED_TIME_INDEX

FREQ_TRUST

ITSA_OUTPUT = ClockTarget Interface corresponding to selected_time_index = MED_TIME_INDEX

prev_trust_status == TIME_TRUST

UCT

Run the mid-value time-index selection process (MVTISP)

B

MVTISP

A

B

A

B

A
Mid-value time-index selection process

```c
// Gather the current skew between the TIDs of all the time indexes.
// For x = 3, x <= num_time_indexes - 3, x++) {
// For y = x + 1, y <= num_time_indexes, y++) {
  TID_diff[x][y] = |TID[x] - TID[y]|
}

// Clear status before starting a new round of time index comparisons.
// Trust status - NO_Trust
// time_index_pair_status[x][y] = UNTRUSTED for all x and all y
// time_index_status[x] = UNTRUSTED for all x
// num_sorted = 1
// exclude_time_index[x] = FALSE for all x

// Find all trusted time indexes, considering hysteresis.
// For x = 3, x <= num_time_indexes - 3, x++) {
// For y = x + 1, y <= num_time_indexes, y++) {
  if (TID_diff[x][y] <= maxskew[x][y] &&
      prev_time_index_pair_status[x][y] == UNTRUSTED ||
      (TID_diff[x][y] <= maxskew[x][y] + hyst[x][y]) &&
      prev_time_index_pair_status[x][y] == TRUSTED) &&
      (isSortedstatus[x] && status[x] &&
      isSortedstatus[y] && status[y]))
  {
    // trust found for the pair
    trust_status = TIME_TRUST
    time_index_pair_status[x][y] = TRUSTED
    time_index_status[x] = TRUSTED
    prev_time_index_pair_status[x][y] = TRUSTED
    prev_time_index_status[x] = TRUSTED
  }
  else
  {
    // trust not found for the pair
    prev_time_index_pair_status[x][y] = UNTRUSTED
  }
}
```

Define all variables/parameters

• Management objects proposed in 14.23
• Define all variables for MVTISP
  • Mostly done (see 19.3.3.2.1)
    • Need to define object types for all the vectors and arrays
• Define all variables for default DTSA/ITSA state-machines
  • Still in progress...
Outlook

• Plan to finish the following within a few weeks:
  • Have all variables for default DTSA/ITSA described with appropriate object types in a couple of weeks
  • Do further updates based on comments received from this meeting or subsequent interactions

• Other:
  • Need help on YANG model
  • Do we need objects to connect FTTM inputs to DTSA/ITSA, DTSAs to ITSA, and ITSA to FTTM output?
    • Management objects for this are proposed in 14.23
Thank You