Version: V10a 2020-12-02

User Stories for IEC/IEEE 60802 system definition

Contributors:
Guenter Steindl
Marius-Petru Stanica
Martin Ostertag
Rene Hummen
Stephan Kehrer
Mark Hantel

Scope:

User stories used as a format to explain expectations of a user. User is in this case used in a very broad scope.

Examples:

The entity relying on synchronization is as user.

The person plugging a device into a network is a user.

…

CONTENTS

User Stories

Terms

ES	End Station
B	Bridge
ME	TSN Domain Management Engine
Device	Bridge or End Station
Plug&Produce	The application of an End Station which is plugged to a TSN domain can without additional user or engineering tool action request, establish and use (real-time) streams for communication.
Power on	"After power on" is a state in which any application would be able to run.
Out of the box	A starting point for a device with no configuration or a vendor specified initial network configuration.
Topology discovery	Device discovery and discovery of physical device interconnections

Constraint

The following is an unsorted list of constraints. The order is incidental and not intended to convey meaning.

Application

No application actions are shown, only end stations and bridge related behavior

Communication before ME sends network configuration to its TSN domain devices

This behavior depends on the starting point:

A) Device is part of the TSN Domain, previous ME configuration is stored
All configured traffic classes are available and can be used

B) Device is part of a different TSN Domain
TSN Domain identifier check shows Domain Boundary,
Only limited traffic classes are available

C) Device is in "out of the box" state
TSN Domain identifier check shows Domain Boundary,
Only limited traffic classes are available

Network configuration – in "ME sends network configuration"

Parameter set used by the ME to configure the Ethernet portion of bridges, bridged end stations and end stations. This should be done using Netconf or SNMP together with the needed YANGs and MIBs. The network configuration deployed by a ME may be customized, using vendor specific management tools by the responsible person for the TSN Domain.

Security

Security, (e.g. authentication, encryption) is assumed to be part of the overall solution.

Commented [MS3]: Should this sentence end actually with ":" and not with "."? If so, maybe "history" is not the right term, perhaps "situation" is better?

Commented [MS4]: Perhaps it could be clearer if stated so: "Device has no specific TSN domain configuration (e.g. is brand new, non-configured/out-of-the-box, or was used somewhere else, in a completely different context).

Commented [MS5]: Is the definition intended to be about the "Ethernet portion of bridges" or rather about the "TSN aspects of bridges"? Should we also propose a monitoring concept for "policy abiding" in device configuration, e.g. if some engineer makes a change which would reflect in a disrespect of the policy? And other aspects of policy management, e.g. policy information model, policy enforcement methods, policy decision points, etc.

Synchronization

Synchronization is assumed to be part of the solution but ignored to simplify the user story discussions.

TSN Domain Identifier

The TSN Domain Identifier is expected to be unique in space and time to prevent accidental unintended combination of TSN domains (e.g. different feature sets and configurations may be active in these domains).

TSN Domain Management Engine

The following user stories assume at least three configuration models:

A) All devices are engineered offline
- Topology
- Bridges and End Stations
- Streams

Offline must be equal to Online.
Topology discovery and verification is completed
Bridge and End Station configuration is done
Streams are setup in Bridges and End Stations

B) Some devices are engineered offline and some are engineered online
- Partial topology
- Bridges and End Stations
- Configuration of a subset of streams (e.g. bound to the offline topology)

There is a portion of a TSN Domain that is offline engineered and a portion that is online engineered (e.g. could be added later). (e.g. Machine variants could define additional bridges and end stations.)
Topology discovery and validation (Offline must equal online)
Bridge and End Station configuration
Stream setup in Bridges and End Stations

C) Network and End Station only. Devices are Plug & Produce
- Bridges and End Stations

Topology discovery
Bridge and End Station configuration

Device pre-configuration situations

If a device supports plug & produce it must support the minimal set of features required to be managed by an ME.

60802 devices must implement features to enable the ME configuration step that allows plug & produce.

Device Options

1. The device is shipped out of the box with all plug & produce features enabled.

Commented [MS7]: Maybe “Bridges and end stations”?

Commented [MS11]: We should probably have a section here with examples. One example is the already provided text here. Another example would process plants, where devices (end stations and bridges may be added while the plant is running or in commissioning).

Commented [MS12]: Somewhat unclear what is meant by “network and end station only”. Are they both online configured? What is online in this case? Plug and produce? If yes, does it mean that the procedure of commissioning is intended to have a fully configured ME which is able to recognize and configure the end stations and bridges, one by one, thus slowly creating the network of the system in an “on-the-fly” concept? But shouldn’t the ME have already an offline version of all the configurations needed for bridges and end-stations? Then how is this C- different than A-?

Alternatively, if the ME must not have the offline configurations of all expected endstations and bridges, then how would it know what kind of configuration must it apply to them? Is this part of the network policy? Then we should state it.

Another question is how the ME would recognize which end station/bridge is being connected in a given moment and create configuration for it? Do we require a special identification concept/protocol?

Commented [MS13]: We should define the “plug and produce” concept. Do we define it in terms of flexibility of production in brown-field projects support or do we define it in terms of automatic network interface configuration of green-field projects? Or both?
Anyway, usually, the term refers to "simple solutions that are needed to enable near-immediate implementation – with no special tools or highly trained engineers or electricians required.”

2. The device comes out of the box unable to be used as plug & produce (e.g. LLDP is disabled) but is harmless to the network and needs to be configured in-place before being usable by the ME.

3. The out-of-the-box configuration is harmful (e.g. IP address reuse, loops with STP disabled) to the network and needs to be configured stand-alone before being introduced to the network.

Examples:

- A. End-station non-configured, out-of-the-box
  - The end station lacks whatever configuration (network-related or not)
  - It comes with a producer MAC address
  - Some examples of features that may be present which require configuration to allow an ME to configure:
    - The end-station has a default IP address
    - If there is a bridge embedded with the end station, then
      - No spanning tree activated
      - No DHCP activated
      - LLDP and SNMP/Netconf agents deactivated
      - No L3 routing active
    - Reachable over SSH over the default IP address, alternatively on another type of connection (e.g. USB, serial)

- B. Basic network configured end-station (OEM, system integrator, other)
  - The end station bears a requested MAC address/default MAC address
  - The end station bears a project-required fixed IP address / has its DHCP activated
  - The end station has its LLDP active, also SNMP/Netconf
  - If a bridge is present too
    - The bridge has the default STP activated

- C. Advanced network interface configured end station
  - The end station bears a requested MAC address/default MAC address
  - The end station bears a project-required fixed IP address / has its DHCP activated
  - The end station has its LLDP active, also SNMP/Netconf
  - The end station has a given TSN domain ID configured
  - The end station has its time synch protocol activated and pre-configured as required by a project
  - If a bridge is present too
    - The bridge has the default STP activated

US1: Simple TSN Domain Startup

Costumer creates a TSN Domain by configuring the ME with a TSN Domain identification and a network policy.

As a next step it creates a network out of four entities:

- 1 ME
- 1 Bridge
- 2 End Stations

as shown in Figure 1.

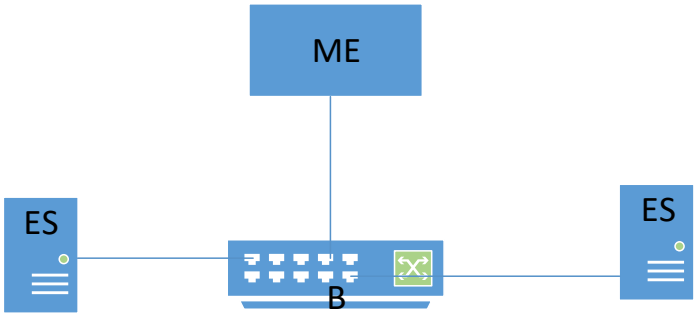


Figure 1: Simple TSN Domain

Expected behavior

The ME discovers the connected network. For any discovered device assigned to its TSN Domain the Ethernet portion will be configured according to the ME stored policy.

Thus, all devices of the TSN Domain will be configured according to the ME network policy.

Any unused port will be configured as “TSN Domain boundary”.

US2: Topology Updating / Topology change discovery

A ME needs an up-to-date topology including all bridges and end stations of a TSN Domain.

Expected Behavior

A ME can on a periodic basis walk its’ TSN domain to ensure the stored topology is still valid.

US2.1: Plugging an additional device

Customer plugs another bridge and/or end station.

When a user plugs a new device into a Time Sensitive Network domain, the user needs to be able to configure the device through the ME, and thus the ME needs to know it exists.

Commented [MS14]: Should we maybe describe how do we expect the behavior of the system, in this use case, for each of the three configuration models shown above? I can take over and do a first draft, if this is OK with the rest of the group.

Commented [RB15]: An ME (same for all following ones)

Commented [RB16]: Discovery can be made by both periodic scanning as well as annunciations of joining and leaving; this is not implicit, e.g. this line says only scanning; it would be nice to make it explicit as there are bandwidth and consistency implications. Joining and leaving are respectively in US2.1 and US2.2, I guess they should be here too.

Commented [MS17]: Should we maybe describe this use case in terms of the Device Options above? Same point further as the previous comment.

Commented [RB18]: There were discussions about the cell-phone model. In that model there is a registration step. Do we consider it here? Registrations could have a time window to avoid useless data in the ME.

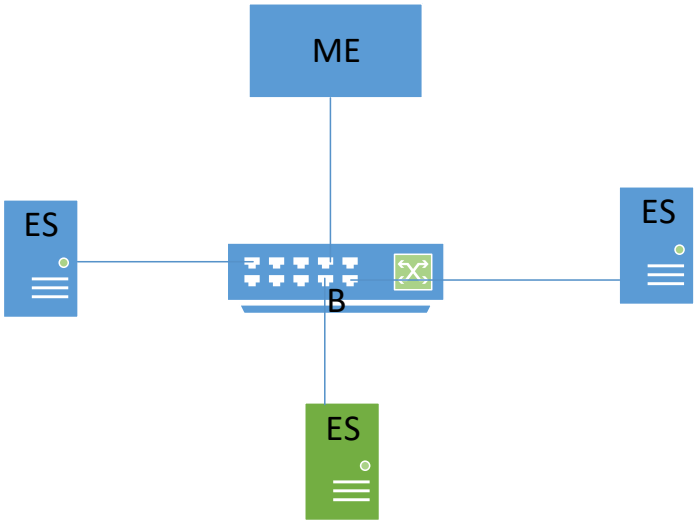


Figure 2: Simple TSN Domain – plug an ES

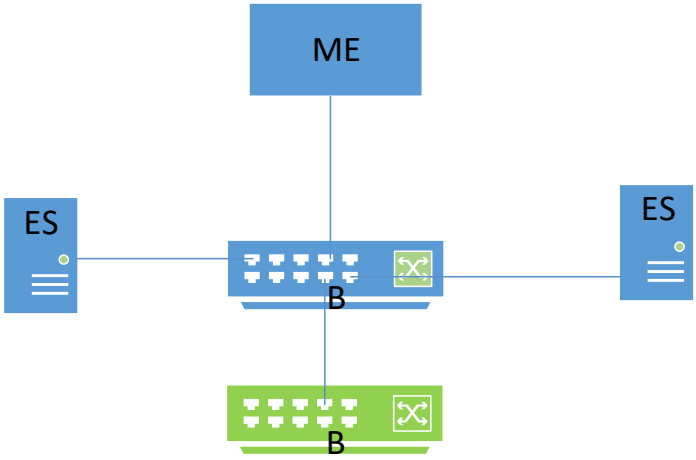


Figure 3: Simple TSN Domain – plug a Bridge

Expected behavior

ME discovers the additional connected device using Topology Discovery, checks (by using the TSN Domain identification) whether it belongs to this TSN Domain.

LLDP is implemented on every device that is 60802 conformant. The device plugged in will advertise its presence and identifiers to adjacent nodes. Identifiers must include MAC address, IP address, and TSN Domain Identifier.

If it belongs to the TSN Domain, then the ME configure the Ethernet portion according to the ME stored policy.

If not, no action.

Adjacent nodes can provide information about the new device to the ME. Adjacent nodes can store information about the new device in their memory to be read by a ME. Time constraints (60802 Use Case 20) may require the adjacent node to provide information to the ME.

US2.2: Removing a Device from The Topology

When a user removes a device from a TSN domain, the user needs to see this device removed from the ME.

Commented [RB19]: Soft or hard removal. below I agree with Marius about considering a state machine for the ME.

Commented [MS20]: In general, such activities happen either as a result of a maintenance activity, either as a human error, either as a device or entire process area failure. Now, I am wondering whether we should introduce a simple state machine for the ME. This would help, for example, avoiding unneeded alarms and events, in case of a maintenance.

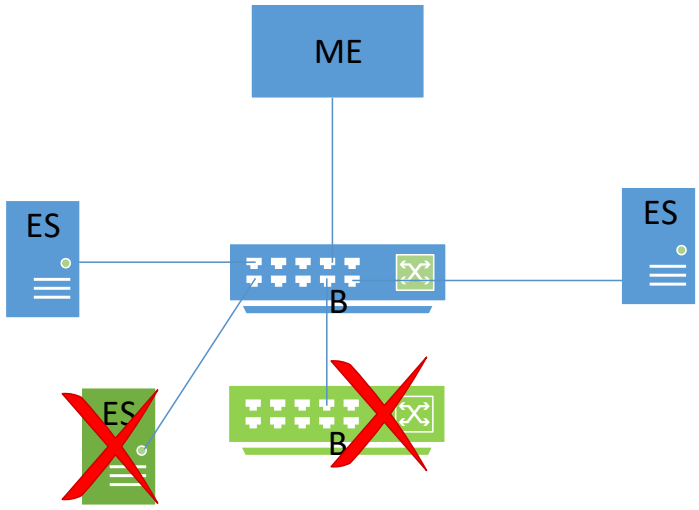


Figure 4: Simple TSN Domain – remove a device

Expected Behavior

The removed device will no longer be a part of the discovered topology.

Adjacent nodes can provide information about the removed device to the ME. Adjacent nodes can store information about the removed device in their memory to be read by a ME. Time constraints (60802 Use Case 20) may require the adjacent node to provide information to the ME.

US3: Combining two TSN Domains

A user introduces a new physical link that joins two TSN domains.

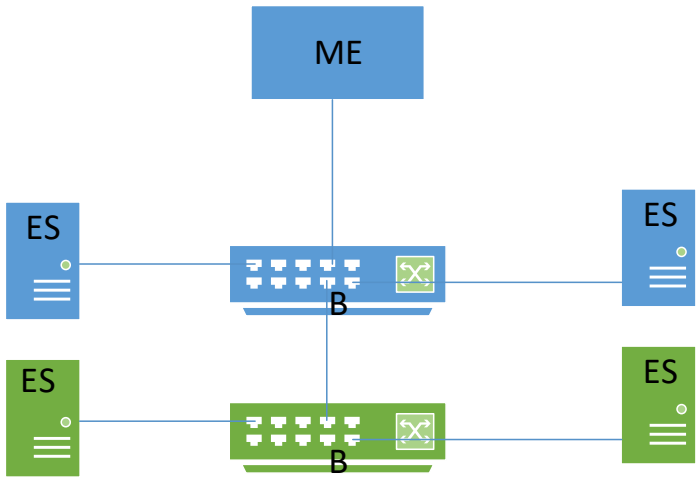


Figure 5: Simple TSN Domain – combine previously splitted TSN Domain

Expected Behavior

If each TSN Domain has the same identifier because they were previously combined or engineered to be combined by sharing compatible TSN mechanisms and identifier, the two domains will be joined into one. If each TSN domain has an independent ME, one ME will be selected. If the identifiers are different TSN Interdomain communications may be established by the Management Entities.

US4: Splitting a TSN Domain

A user removes a physical link that creates a single TSN domain.

Commented [MS21]: This use case, brings, in my view, yet again, the need for a state machine for the ME. Performing such an action, that is connecting a whole new network segment (or several, depending on the project) may create a large disturbance in a running system. If the ME though is put into a maintenance state, then it can activate a maintenance configuration of the system, which may allow such an use case, without any danger to the process(es) supported by the two TSN domains.

Commented [MS22]: This is a complex situation, which would need further analysis of a ME state machine. In my opinion, in order to be able to safely perform such an use case, both ME state machines must be in "compatible" states, in order to allow the joining of their domains. Only after this, a mechanism of election of the new ME can be activated. Such an election mechanism should be also detailed, in my view, there are also some options: either only one ME is chosen, either one ME becomes the redundant of the other one, thus a sort of configuration mirroring process between the two could be activated.

Furthermore, the newly elected overall ME (covering both TSN domains) must get information about the existing devices and streams of the newly connected TSN domain. Following options seem present:

- if the newly added TSN domain is simply a collection of TSN devices without any ME, then the devices must be discovered by the only existing ME (the one belonging to the initial TSN domain). So, use case 2.1 is repeated for each device of the new TSN domain.
- if the newly added TSN domain has a ME, then once the election process is done, a data exchange must be put into place between the two MEs so that the newly elected overall ME has the information about the devices in the newly added TSN domain
- Furthermore, in case one of the MEs becomes redundant of the other one, a mirroring mechanism has to be put into place.

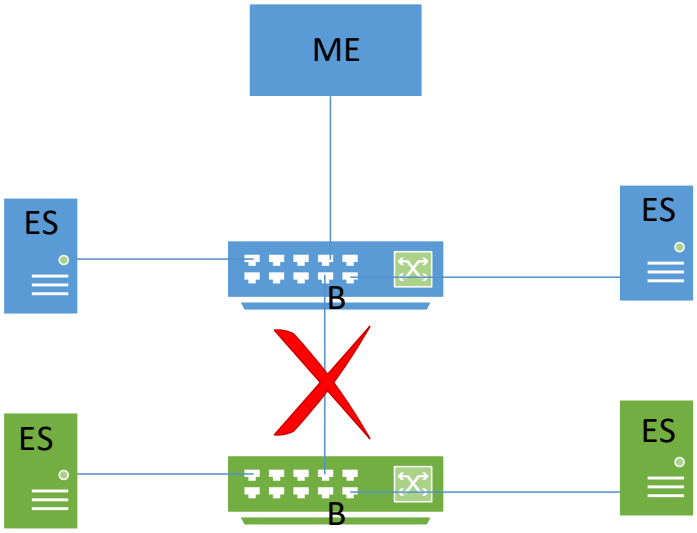


Figure 6: Simple TSN Domain –split TSN Domain

Expected Behavior

Each network will continue to operate as two separate TSN domains and are managed by separate Management Entities. If one of the resulting domains does not have a ME, no functions that require an ME will be available. Each will automatically maintain the same TSN Domain Identifier.

US5: Assigning TSN Domain Identifier

Whether a device belongs to a TSN Domain is identified by checking its TSN Domain Identifier. This identifier is expected to be available as LLDP TLV / MIB object defined in IEEE802.1Q scope.

Expected Behavior

When the user assigns the application driven identification, additionally the to be connected TSN Domain needs to be specified.

Thus, the ME can identify whether this device is in its responsibility or not.

US5.1: Auto assignment of TSN Domain Identifier

Additional to the TSN Domain Identifier, different means may be used to identify whether a connected device belongs to the TSN Domain or not.

Expected Behavior

If the A) configuration model of the ME is used, the device and its position in topology can be used to assign all addressing information to the device. This includes the TSN Domain Identifier.

Commented [MS23]: Yet another use case requiring the existence of a ME state machine, which can allow a functioning mode for the TSN domain specific to maintenance, so that the process is as little as possible influenced. Anyway, a similar discussion as in the previous comment should be done with regards to the presence or not, after the split, of a ME per each of the newly created TSN domains.

Commented [MS24]: Perhaps in a first phase. This depends on how we assign TSN domain identifiers, once the TSN domain is split and the ME state machines of each TSN domain (if present in both) out of maintenance state, a new TSN domain identifier may be assigned to any of the newly created TSN domains.

Commented [MS25]: It requires a maintenance issue for the 802.1Q, in my view, as I believe it may be necessary outside industrial automation, e.g. in automotive perhaps.

Commented [RB26]: While “application driven identification” is not defined, maybe this phrase could be re-written as “When the user assigns the identification to the application, it is also necessary to specify the TSN Domain to connect to.”

Commented [MS27]: At least for me, the message of this phrase is unclear.

Commented [MS28]: In this case, the text of the use case seems not to follow the title of the use case. Is it about the assignment of a TSN domain identifier or is it about identifying whether a device belongs to a TSN domain or not (based on an already existing TSN identifier, in the respective device)?

Commented [MS29]: The topic of identifying whether a device belongs to a TSN domain, before the respective device has received a TSN domain identifier seems relates directly, in my view, with the three device options we identified above and should be described following them.

US6: Media redundancy

Media redundancy is used to ensure availability of communication.

Simple topologies e.g. rings are often used to fulfill the requirements for media redundancy. More complex requirements lead to more complex topologies e.g. coupled rings allowing multiple concurrent faults.

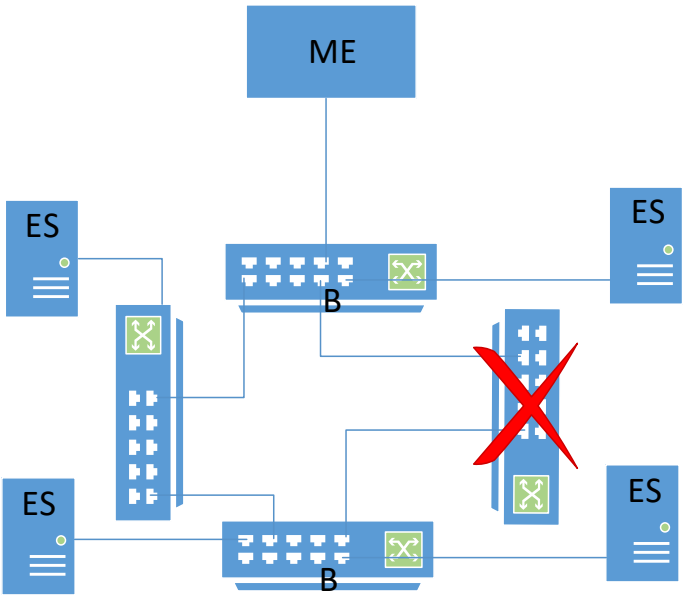


Figure 7: Simple ring topology – one fault

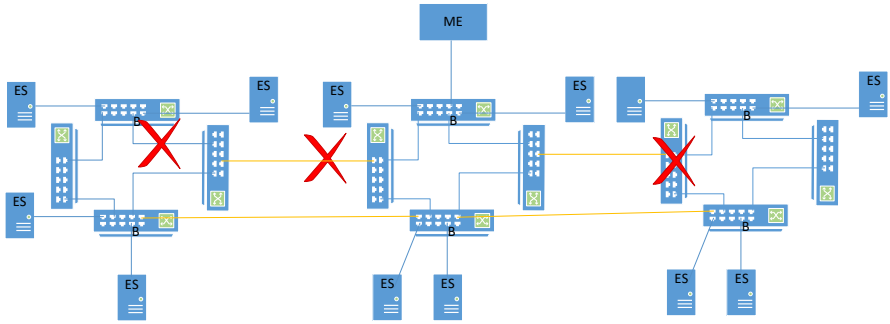


Figure 8: Coupled rings topology – multiple faults

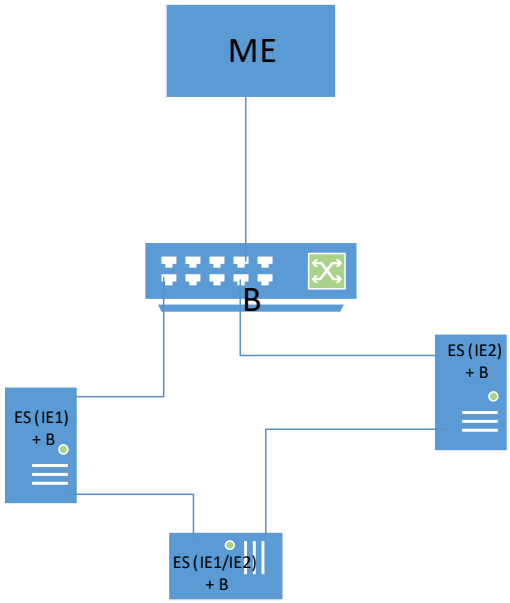


Figure 9: Ring composed of end stations and embedded bridges from various vendors and supporting various Industrial Ethernet (IEx) frameworks

Expected Behavior

One or more faults, e.g. wire break or disconnected bridge, do not stop the production until a repair happens.

Commented [MS30]: A common interoperable seamless media redundancy concept is needed not only for fault tolerance, in my view. Given the tradition of various existing industrial Ethernet frameworks, adherence to different media redundancy protocols was created. Thus, today, even without fault, the presence in the same ring of end stations with embedded bridges, produced by different industrial Ethernet protocols is not possible (even using a non-seamless media redundancy protocol). With .1 CB, this is issue is removed for future multivendor systems.

US6.1: Switch over redundancy

Simple topologies (e.g. rings) are often used to fulfill the requirements for resiliency.

These requirements apply to simple rings and to coupled rings.

Expected Behavior

As soon as a fault happened, the system redirects the traffic to follow the substitution path. Depending on the application requirements the time which is needed to redirect the traffic may or may lead to a communication disturbance.

Commented [RB31]: happens

Commented [RB32]: may or maty not lead

Spanning-tree protocols (e.g. RSTP, MSTP) can be used as a resiliency protocol.

If traffic engineered streams are used, the ME may need to “reroute” them according the new topology.

US6.2: Seamless redundancy

Traffic engineering two most disjunct paths from Talker to Listener allow seamless redundancy with zero switch over time.

Commented [RB33]: allows

These requirements apply to simple rings and to coupled rings.

Seamless redundancy based on FRER may be solved by end-station FRER or bridge FRER at the end-station ports; it needs to be solved by bridge FRER at the ring coupling ports.

Expected Behavior

Seamless redundancy shall be available for simple and coupled rings and supports one fault per ring.

Commented [RB34]: supports

US7: Application cycle

Industrial applications as defined in IEC 61131 rely on the cyclic execution of their program. This program may contain multiple tasks. These tasks are executed if their assigned events occur.

Commented [MS35]: There are also periodic tasks, which do not need an event to be executed. Mostly, this is the model of the IEC 61131. The IEC 61499 is mostly an event-based task execution specification. Perhaps the earlier reference to 61131 is then too detailed for this context.

Thus, an industrial application may have multiple application cycles as shown in Figure 10.

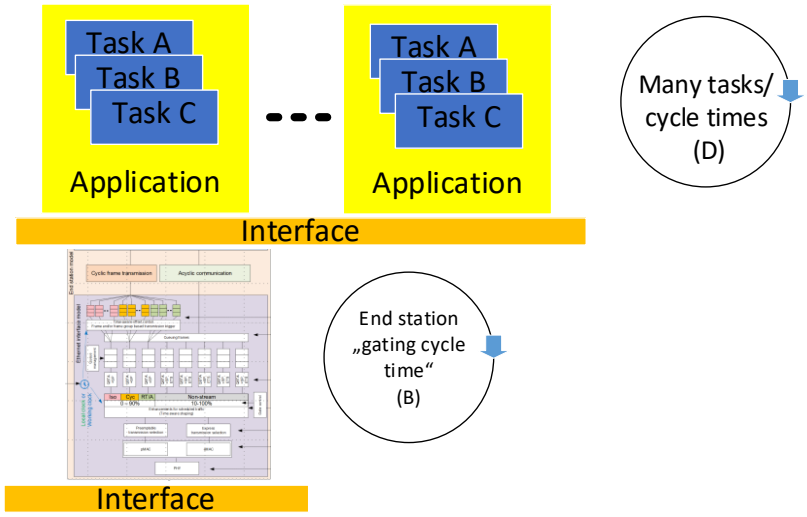


Figure 10: Whats an application?

These e.g. are

- Background tasks (none, one or more), which is executed when no other task is running
- Main task (none or one), which is started, often triggered by local time, periodically
- Global time tasks (none, one or more), bound to Global Time, which is started e.g. every noon or every Friday or …

- Process driven tasks (none, one or more), which is started e.g. if sensor value reaches a defined point or a process fault happens or …
- Control loop tasks (none, one or more), bound to Working Clock, which are started periodically

The term application cycle is mostly used to specify the runtime of the Main task; but sometimes used for any periodically execute task in application program of a PLC.

Special case: Control loop tasks

Control loops rely on synchronized applications behavior at the involved IA-devices and IA-controllers. That's implemented by using the same Working Clock, the same starting point relative to the Working Clock and the same duration of this control loop task at the involved IA-devices and IA-controllers. Multiple control loop task may be implemented and running in parallel at the involved automation devices.

This understanding shall be independent from the time when the device is powered up or connected to the TSN Domain.

Special case: Start of control loop tasks

A common understanding of the start of a control loop task is built out of the working clock in the following manner:

If you divide the integer working clock value by the integer "time for the control loop task", and the remainder is zero, the control loop task starts.

If a defined control loop task cycle is addressed, the resulting quotient may be used to create a common understanding of future or past application cycles. This may be used to check for alignment errors between the involved applications, e.g. to check whether the data objects conveyed are from the expected control loop task cycle.

➔ Application cycle seems to be no feasible term!

Expected Behavior

Start of application tasks is synchronized, if needed, independent from the time a device is connected to the network.

Base is the Working Clock.

US7.2: Application vs. stream

How to create a frame / stream between two devices?

Constraint: Various application run in parallel on an automation device. Data objects represents the exchanged information between local and remote application. For this purpose, they are assigned to Application Relations and located in the process image.

Commented [MS36]: I have some doubts about this description (e.g. process control does not always need a working clock, but it still needs synchronization in a control loop, based on a global time, what is a "starting point" in this context is it about the epoch start?). Can we find a specification which we can refer to instead?

Commented [MS37]: What is it mean by "this understanding"? The fact that the description above (namely the epoch start, perhaps? To be clarified) is generic and has nothing to do with the moment of time when the device is powered up or connected to the TSN Domain?

Commented [MS38]: Reference would be useful.

Commented [MS39]: Is "addressed" the right word here? The whole phrase needs to be re-focused, in my view. I am not fully following its sense. Perhaps start with the next phrase and then use this phrase as an additional clarification.

Commented [MS40]: The start of the application tasks is synchronized between the applications themselves only or to an external epoch start of a clock domain?

- Application engineering selects for each application

-- the data objects which needs to be exchanged

-- the App_Interval and App_Deadline requirements for each data object; in case of a control loop all assigned data objects (e.g. are grouped) do use the same value for App_Interval and App_Deadline.

-- …

This information is used to create a stream object assigned to the Application Relation created in the above shown step. These Application Relations are typically relying on bi-directional communication relations each represented by a stream object. The requirements for both directions shall be satisfied otherwise stream reservation for both is expected to fail.

This stream object contains e.g. frame size, MaxLatency (Deadline), Interval, Talker, Listener(s), … . The data objects with the strictest requirements defines the requirements for the stream object. MaxLatency is only included as Deadline in a stream request, if at least one data object is assigned to a control loop. Non control loop application may just require that latency of a stream is smaller than the App_Interval. In this case the MaxLatency is only included as Latency containing the value App_Interval of strictest data object requirement.

What's the difference between App_Deadline and MaxLatency (Deadline)?

In case of control loop, the App_Deadline needs to be later than the (assigned/used) MaxLatency (Deadline) relative to the Working Clock.

What's the difference between App_Interval and Interval?

In case of control loop,

the App_Interval and the App_Deadline are used to calculate the Interval and the assigned "end station gating cycle" (an App_Interval may contain multiple "end station gating cycle time") relative to the Working Clock.

In other cases,

the App_Interval is used to calculate the Interval and the assigned "end station gating cycle" (an App_Interval may contain multiple "end station gating cycle time") relative to the Working Clock.

Expected Behavior

Application tasks using data objects provided by a local interface e.g. a process image, for executing their program and as sink for the results of their calculation. These data objects are conveyed by the assigned stream objects.

Requirements for the update of these data objects are application task dependent and used to define the attributes for the stream objects. Timescale to align application and communication is the Working Clock.

US7.3: Network cycle

The term network cycle suffers a similar problem as the application cycle as shown in Figure 11.

Multiple automation systems may be located in one TSN Domain. These automation systems, which are built from several end stations are connected to bridges, may share a common understanding of a “end station gating cycle time” or each station may have its own “end station gating cycle time”.

Any additional connected automation systems may use a different understanding of a “end station gating cycle time”.

What’s in this case a network cycle?

Special case: Complete offline configuration

In this case, an engineering system may align all “end station gating cycle time” in a way that this setup may be seen as the common “network gating cycle time”.

This case is often bound to the use of Qbv in bridges.

US8: Application (de-)coupling

Expected Behavior

Moving from parallel to sequential control raised the problem of the data basis for the program execution together with its execution speed.

That lead to the implementation of a so-called Process Image, containing the to be exchanged data objects, which was in the beginning only a few bytes in size.

Basic timing model, which applies for each implemented task – fetching inputs, writing outputs, compute, repeat

Commented [RB41]: This seems to imply that indeed the network access is only once per cycle, as mentioned in the message comment. This is per application.

Increasing the available memory, the available compute, the amount of connected IOs, integrating motion and of remote IOs just extended this basic model. Always ensuring that the customer applications are running unchanged!

Nowadays, the process image containing the exchanged data objects has a size from less than 1k Bytes up to 1M Byte.

This amount of data, provided by up to 1024 communication partners, takes some time to be collected. Applications require update times of their portion of the process image from 25µs/31,25µs up to 1s.

Thus, the update of the Process Image and the Network Access interact to support the requirements for timeliness from the applications. Compute contains the background task, which creates the trigger for a snapshot of the process image from the intermediate process image.

Additionally, many time triggered tasks, either by working clock or global time are executed concurrently.

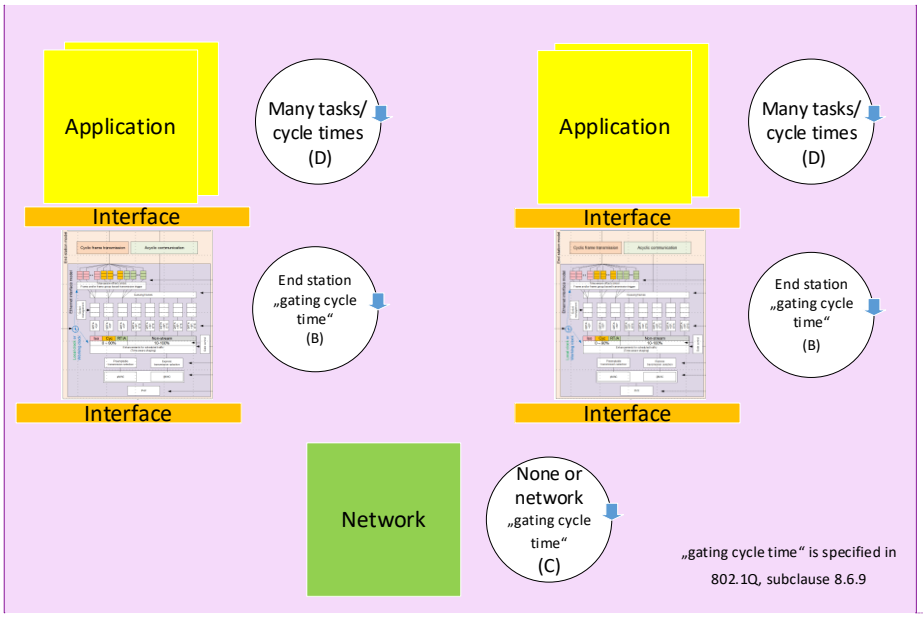


Figure 11: Principles

Figure 11 shows the connection between two end stations.

Principle use cases

Automation systems do support, depending on their needs, many different models. The following five are very often seen in the automation arena.

Case “unsynchronized”

All entities do have their own cycle.

Latency variation are the result of this “unsynchronized” cycles.

Case “synchronized application cycles”

All entities do have their own cycle, but the application cycles (A) and (D) are synchronized to each other.

Latency variation are the result of the “unsynchronized” cycles of this model.

Case “synchronized scheduling cycles”

All entities do have their own cycle, but the scheduling cycles (B) and (E) are synchronized to each other.

Latency variation are the result of the “unsynchronized” cycles of this model.

Commented [MS42]: Is it “end station interface” meant in the drawing or network interface of the end station?

Case "synchronized application and scheduling cycles"

All entities do have their own cycle, but the scheduling cycles (B) and (E), and the application cycles (A) and (D) are synchronized to each other.

Latency variation are the result of the "unsynchronized" cycles of this model.

Case "synchronized application, scheduling cycles and network cycle"

All entities do have their own cycle, but they are synchronized to each other.

Latency variation are the result of the "synchronized" cycles of this model.

Application

General

The following application models are based on a decoupling between communication and application based on a shared memory concept used as interface for the exchange of data.

The application space thus looks like Figure 12.

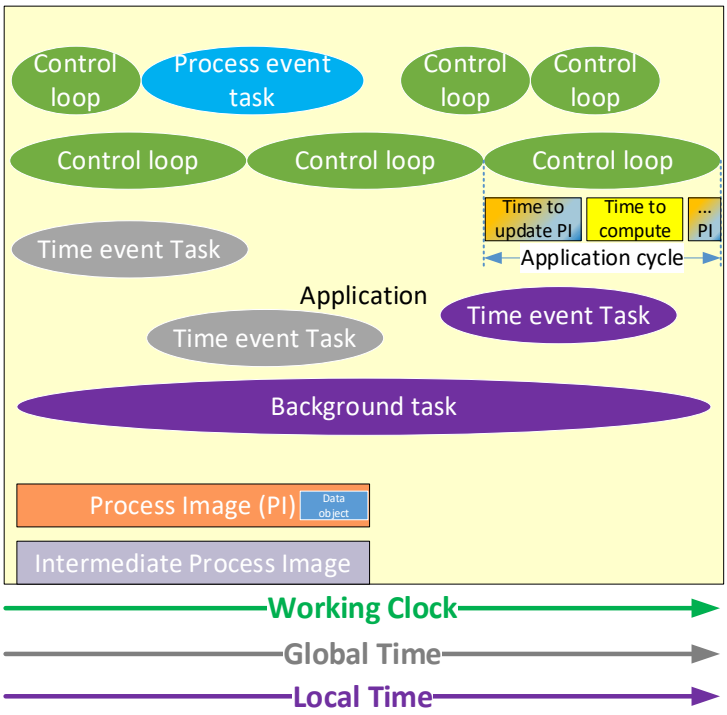


Figure 12: Application area

Behavior of Process Image

Application cycle

Most devices do have an application cycle which updates the Process Image bound to local understanding of time (Local time).

Updating the Process Image means, that the content of the intermediate Process Image is copied (actions for coherence and consistence are done) into Process Image (Inputs) and vice versa (Outputs).

Background task

Just working on the actual content of the Process Image.

Time event task

Time event tasks may either be triggered by Local Time (e.g. start every 10ms) or Global Time (e.g. at the top of the hour). They may create a local copy of the Process Image portion, if they need to ensure that they work on data from one Process Image update.

Process event task

Process event tasks are triggered by process events (e.g. diagnosis alarm). These are normally working on the actual content of the Process Image.

Control loop task

Control loop tasks are triggered by Working Clock (otherwise they are implemented as Time event tasks). They may create a local copy of the Process Image portion, if they need to ensure that they work on data from one Process Image update.

Communication

General

The following communication models are based on a decoupling between communication and application based on a shared memory concept used as interface for the exchange of data.

The communication space thus looks like Figure 13.

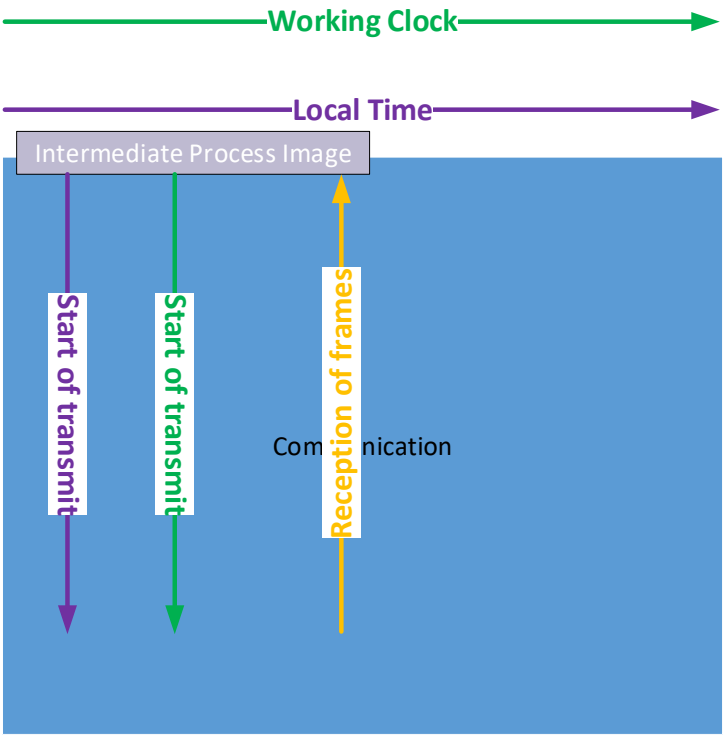


Figure 13: Communication area

Periodic transmission of frames is either driven by Local Time or Working Clock. Source and sink of data are the intermediate Process Image.

Transmit (Local Time triggered)
Periodic transmission is driven by Local Time. An alignment with the Application is not intended.

Transmit (Working Clock triggered)
Periodic transmission is driven by Working Clock. An alignment with the Application is possible based on events driven by Working Clock timescale.

Reception
Reception of frames is independent from the timescale used for transmitting.

Alignment between Application and Communication

Alignment between Application and Communication is done by relying on Working Clock as shown in Figure 14.

Deadline is used as “latest point in time” when the data for the control loop need to be available in intermediate Process Image.

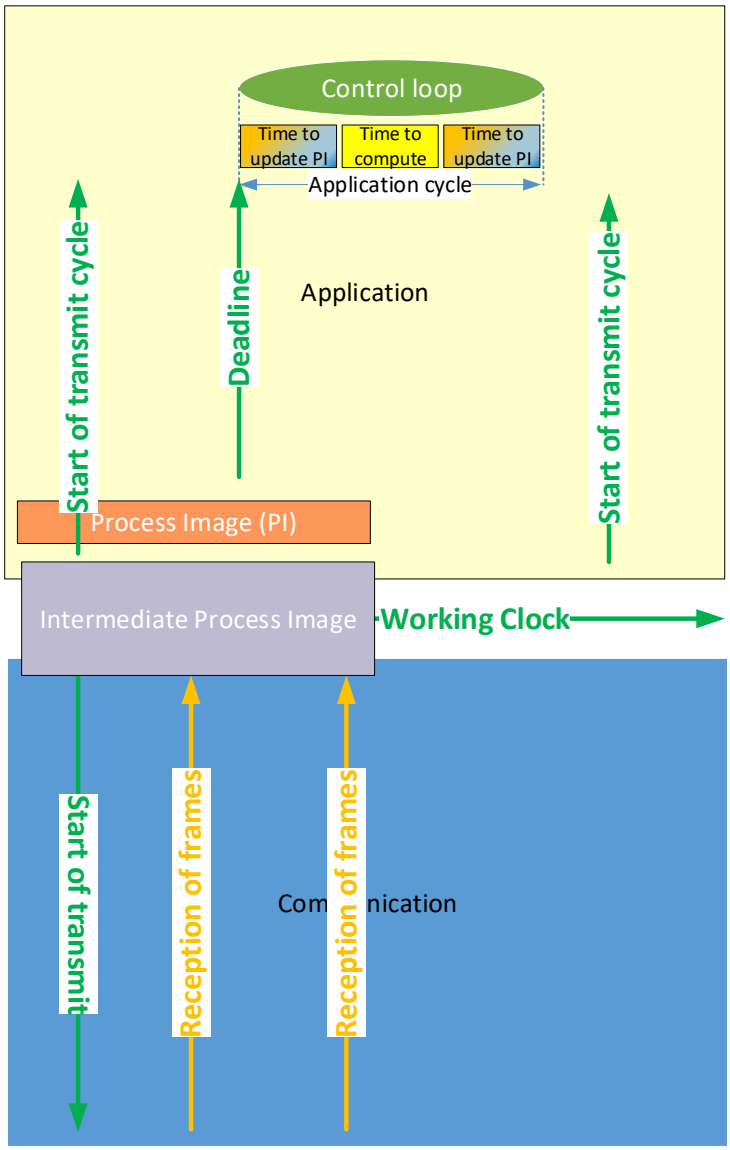


Figure 14: Alignment model

US9: Timescales and Clocks

Commented [SG(FTAE43]: Text changes and comments from Martin Ostertag are integrated.

Expected Behavior

GlobalTime and WorkingClock are needed in many automation devices.

Automation applications need to work independent from connected communication interfaces. Thus, use cases in which the synchronization is lost or reestablished shall be covered. This is often done by implementing two instances of a Clock for one PTP end instance. shows this clock usage model.

Deviation of the Application Clocks, at the GM and the PTP end instance, are important and thus, checked during certification, for the industrial use cases. The deviation of intermediate Clocks e.g. the PTP Clock in Figure 15, are only of interest as influence to the quality of the Application Clock and the Gate Control List Clock.

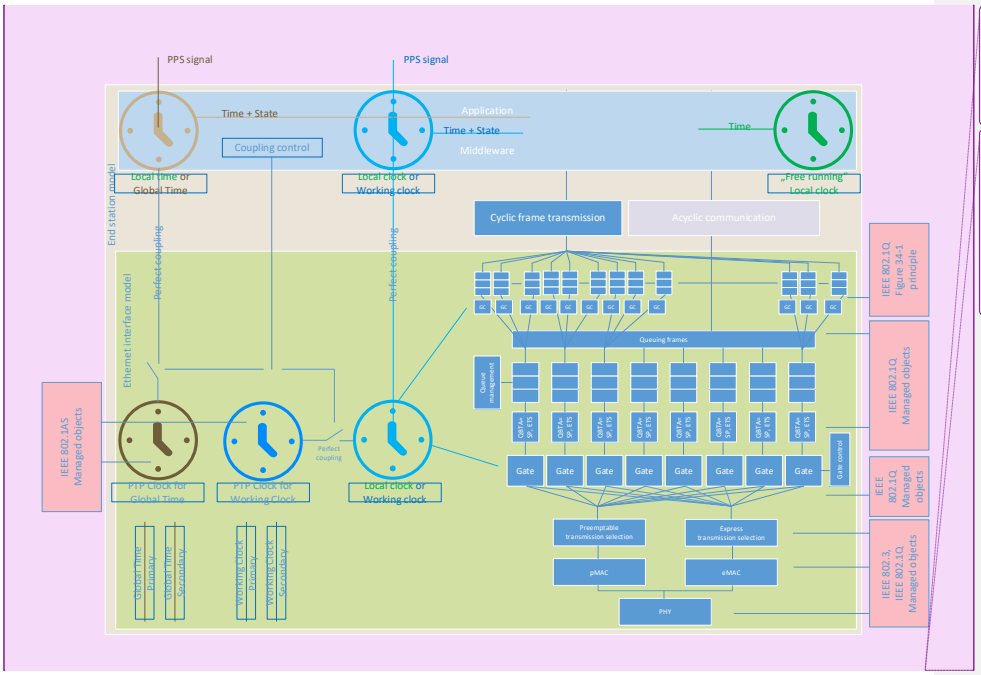


Figure 15: Clock usage model

Commented [MS44]: I would like to propose an update of this drawing, as this drawing raised many comments also in D1.2. It is useful, in my view, for the reader, to associate the clocks governing the timely behavior to their relevant sections in an end station.

Commented [MS45R44]: I would still like to add the proposal I have done for this drawing, in Figure 14a(next page). The proposal currently shown in Figure 14 seems to be only one way of using the clocks. Personally, I find it very complex, due to the mixing of the PTP clock at the application level. Perhaps needed in some situations, but in my view, definitely not needed for all end stations which will be TSN enabled (thus it could be optional).

Figure 16a: Alternative clock usage model

State information is provided together with the time information to the application.

US9.1: GlobalTime (timescale TAI/PTP)

Expected Behavior

GlobalTime is if supported by the device always available independent from an existing / connected GM.

- Monotonically increasing time, where monotonic means that a tick can mean +0, +1 or + >1. Both, how often your able to do “+0” or “+ >1” is limited by the allowed acceleration/deceleration per 1 ms.
- The maximum acceleration/deceleration is limited and is in the range of e.g. +/-1000 ppm (200 ppm to compensate the rate difference for the supported 100 pm oscillators and 800 ppm for offset correction) for the PTP end instance offset correction (that’s equal to 1 µs for an interval of 1 ms or a correction every 1000 ticks at 1GHz)
- (“value jump in case of set value” is allowed, but must be documented at every PTP end instance to comply with application needs or legal rules (e.g. Power Generation))
- The threshold/algorithm for the PTP end instance’s clock to declare inSync is within +/- 100 µs deviation to the GM
- 64/100 Hops must comply with the +/-100's deviation window to the GM
- GM and PTP relay and PTP end station timer have +/-50 ppm oscillators
- The frequency of these oscillators may change due to external influences e.g. df/dt of 3 ppm
- Connection to GPS or similar possible/supported

Any violation of these requirements makes GlobalTime either useless OR disrupts machines and equipment in the worst case.

US9.1: WorkingClock (timescale ARB)

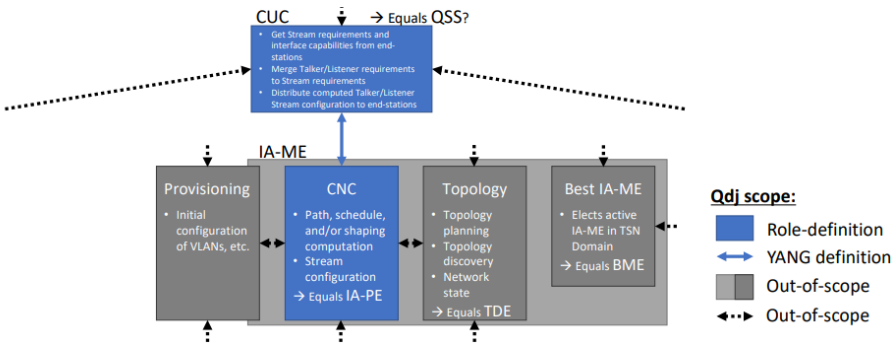
Expected Behavior

WorkingClock is if supported by the device always available independent from an existing / connected GM.

- Monotonically increasing time, where monotonic means that a tick can mean +0, +1 or + >1. Both, how often your able to do "+0" or "+ >1" is limited by the allowed acceleration/deceleration per 1 ms.
- The maximum acceleration/deceleration is limited and is in the range of e.g. +/-250 ppm (200ppm to compensate the rate difference for the supported 100pm oscillators and 50ppm for offset correction) for the PTP end instance offset correction
  (that's equal to 250 ns for an interval of 1 ms or a correction every 4000 ticks at 1GHz)
- ("value jump in case of set value" is always associated with SyncLoss to avoid destruction of equipment)
- The threshold/algorithm for the PTP end instance's clock to declare inSync is within +/- 1 µs deviation to the GM
- 64/100 Hops must comply with the +/-1 µs deviation window to the GM
- GM and PTP relay and PTP end station timer have +/-50 ppm oscillators
- The frequency of these oscillators may change due to external influences e.g. df/dt of 3 ppm

Any violation of these requirements makes WorkingClock either useless OR destroys machines and equipment in the worst case.

US10: Management



A ME in this context can be divided into two functions:

A provisioning function

A stream reservation function

Expected Behavior

US10.1: Provisioning a Time-Sensitive Network

A provisioning function should be composed of a topology discovery engine, and it needs to have the capability to select a policy or base network configuration necessary before sending stream reservations. Network State belongs in the provisioning function

Expected Behavior

US10.2: Rolling out Stream Reservations in a Time-Sensitive Network

A stream reservation function should be composed of a topology discovery engine, CNC which includes a path computation engine, and provides the interface to a CUC.

Expected Behavior

US10.3: Management Entity Redundancy

Redundancy needs to be defined for a management entity. A method for selecting the primary management entity needs to be defined, the “best management entity”.

Expected Behavior

If a primary management entity fails, a backup ME will take over.

US10.4: Management Entity Goes Away

Expected Behavior