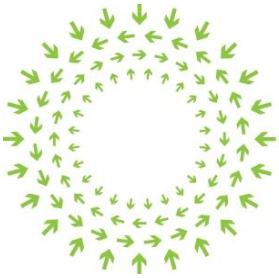

DATA CENTER PTP PROFILE

Nov 2022

Executive Summary

This document defines a PTP profile to support time-sensitive applications within a data center environment. The document is developed within the Open Compute Project (OCP) Timing Appliances Project community [1]. The PTP profile is based on IEEE Std 1588™-2019 [2]. When applicable, the profile also references and reuses information from other PTP profiles or other industry specifications. The document provides a set of requirements for implementing, deploying, and operating timing appliances within a data center. A timing appliance is an element that is PTP-aware such as a switch/router, time server, NIC card, software module, timing card, monitoring device, etc.

Table of Contents

1 Introduction

Time is a key element to get the highest efficiency in a distributed system. The performance of a distributed system depends in part on the level of synchronization between the elements. Several industries such as telecom, power, industrial, automotive, professional audio, and video have embraced the need for highly accurate and reliable distribution and synchronization of time across packet networks. Although the use case scenario for each of the industries is different, they all share one common thing and that is, time synchronization. Each use case scenario defines a set of requirements and configurations that are specified in a 'PTP profile'. This document defines a PTP profile to serve the needs of data center time-sensitive applications, data center network infrastructure and the use of synchronized clocks [3]. The profile specifies the set of PTP features and attribute values applicable to a PTP instance that operates in a single device (e.g., such as a switch, router, server) and within exactly one PTP domain. Additionally, this specification also addresses additional requirements and use cases that are outside the definition of a PTP profile.

2 Terminology

The IEEE 1588 committee is working on an amendment to recommend alternative terminology that is more inclusive than some of the terminology currently used in IEEE Std 1588. The amendment has not yet been approved by the IEEE Standards Association and has not yet been published as of October 2022.

This document uses the following translation of terms used by IEEE1588:

IEEE Std 1588-2019 terms	OCP DC PTP Profile terms
Master	TimeTransmitter (TT)
Slave	TimeReceiver (TR)
Grandmaster	Grandmaster (GM) or Open Time Server (OTS)

3 PTP Profile Definition

A PTP profile is "a document, or a portion of a document, specifying the set of PTP features and attribute values applicable to a PTP instance, and written by an organization following the specification of IEEE Std IEEE1588-2019. The profile allows organizations to specify selections of attribute values and optional features of PTP for the purpose of meeting requirements of a particular application. A PTP profile applicable to data center is defined in this document.

A PTP profile is a set of required options, prohibited options, and the ranges and defaults of configurable attributes. Some examples are:

- Path delay measurement option (delay request-response or peer delay)

- Range and default values of all configurable attributes and dataset members
- PTP Instance types
- Options required, permitted, prohibited
- Uncertainty specifications
- Transport mechanisms required, permitted, or prohibited
- If relevant, the value of the observation interval τ used for PTP Variance measurements.

4 Reference Model

The model referenced in this section is designated as Model 1. The model consists of three layers. The time reference layer consists primarily of sourcing a time reference (e.g, GNSS) and the PTP Open Time Server (GM) functionality [4]. The network fabric layer consists of a set of network elements that support PTP clocks such as the transparent clock (TC) or the boundary clock (BC). The server layer consists of a group of end-hosts that support PTP clocks such as the ordinary clock (OC), and where the time-sensitive applications typically reside.

In Model 1, the network fabric layer consists of a chain of TCs.

4.1 Model 1 – Transparent Clock Model

The high-level characteristics of Model 1 shown in Figure 1 are:

- GM (or Open Time Server) has a single network physical port and always distribute time towards the network fabric layer and server layer. The GM defined in this PTP profile (and the term GM used in this document) is a timeTransmitter-only[1] OC with a single PTP port according to 9.2.2.2 of IEEE Std 1588-2019.
- TC can have multiple network physical ports (e.g., 16, 48). The TC can have multiple capable PTP ports.
- OC has a single network physical port and always receives time from the network fabric layer and the GM. The OC defined in this PTP profile (and the term OC used in this document) is a timeReceiver-only[2] OC according to 9.2.2.1 of IEEE Std 1588-2019.
- All network elements that provide transport of the PTP messages between a GM and an OC are PTP-aware (i.e., TC-capable).
- Hardware timestamping (as close to the medium as possible) is available at each PTP port.
- In normal operating mode, an OC has connectivity to more than 1 GM.
- There are a number of GMs that are either active or standby.
- An OC communicates with a GM based on the unicast discovery and unicast negotiation protocol.
- Communication between PTP clocks is primarily based on IPv6.

[1] See Section 2 - Terminology

[2] See Section 2 - Terminology

- The PTP clock discovery and selection algorithm do not rely on multicast or broadcast communication.
- The <meanPathDelay> computation is based on the end-to-end delay mechanism.
- The number of TCs between GM and OC is constant. For example, if the number of TC = 5, then there will be 7 clocks in total (i.e., including 1 GM and 1 OC) with 6 links interconnecting the clocks.
- The forward path direction (GM to OC) and reverse path direction (OC to GM) might not be congruent. That is, PTP packets in the forward direction might traverse a different set of TCs from PTP packet in the reverse direction. However, the number of TCs in both directions in a non-congruent scenario is expected to always be the same and the effect on delay asymmetry is expected to be negligible.
- Delay asymmetry due to fiber links is assumed to be negligible, but likely not zero, in comparison to the time error requirements.

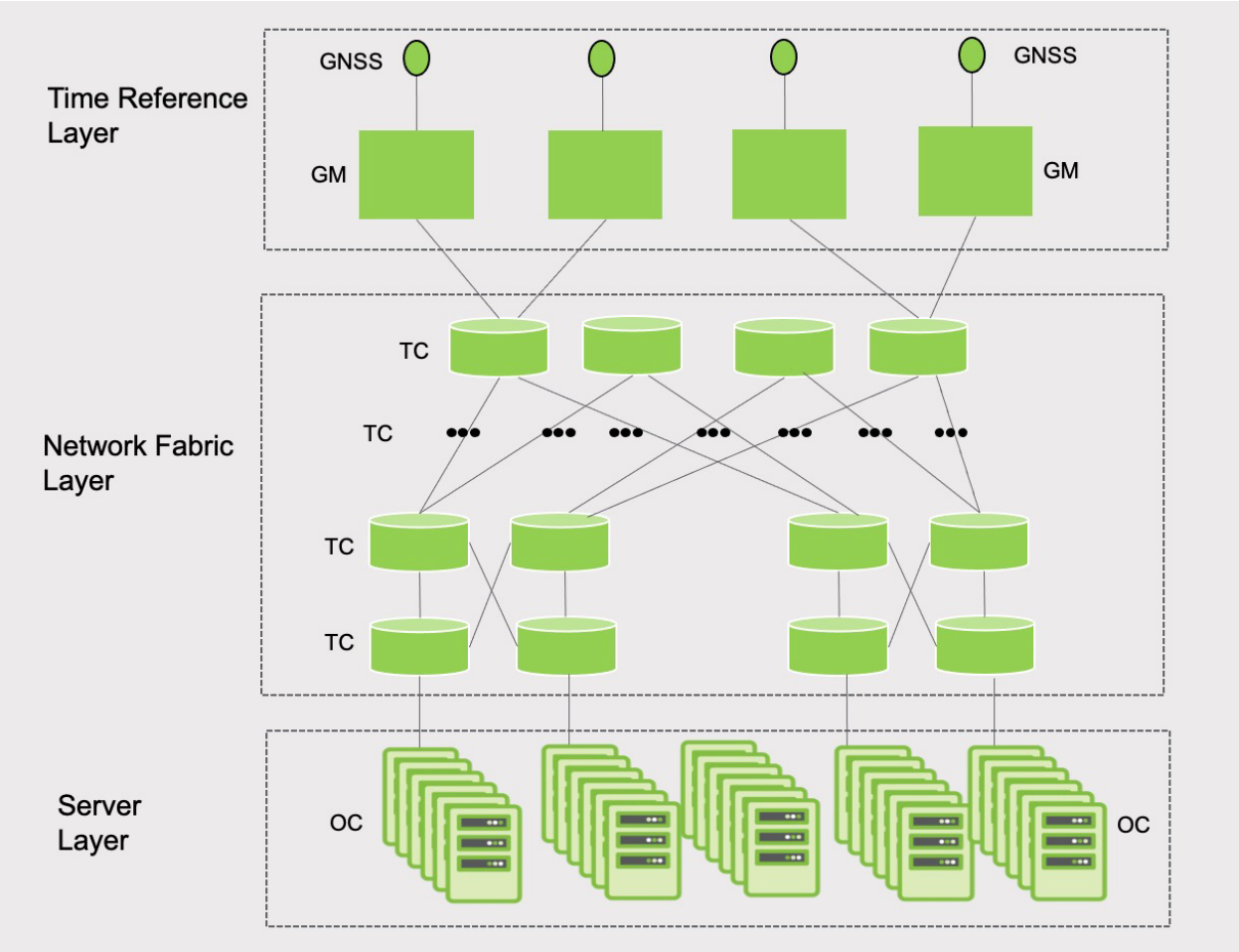

Figure 1. Model 1 – Chain of Transparent Clocks

5 Model 1 - Additional Requirements

- The higher layer applications require UTC traceability. The PTP protocol transports the PTP Timescale (i.e., TAI) plus all information to derive the UTC Timescale from the TAI timescale. It is up to the application to perform timescale conversion.
- The maximum time error between any two OCs must be within ±5 microseconds, i.e., $|T_{OC,j} - T_{OC,k}| \leq 5$ µs for $k \neq j$.
- The maximum time error between a GM and any OCs must be within ±2.5 microseconds. i.e., $|T_{GM} - T_{OC}| \leq$ 2.5 µs.
- The maximum time error between any two GMs must be within ±100 nanoseconds, i.e., $|T_{GM,j} - T_{GM,k}| \leq$ 100 ns for $k \neq j$.
- The maximum time error generated by a TC must be within ±100 nanoseconds, i.e., $|T_{TC,j}| \leq 200$ ns.
- In normal operating conditions, each OC has connectivity into multiple GMs. Under failure of a GM, an OC must be capable of having connectivity to at least another GM.
- The GMs in the network are either active or standby.

6 PTP Profile

The PTP profile is based on IEEE Std 1588-2019.

6.1 Profile Identifier

The information below identifies the profile. The profile is defined by OCP.

profileName: PTP profile for data center application (DC-PTP Profile 1)

profileNumber: 1

primaryVersion: 1

revisionNumber: 1

profileIdentifier: 7A-4D-2F-01-01-00

organizationName: Open Compute Project (OCP)

sourceIdentification: This profile is specified by OCP and can be downloaded from https://www.opencompute.org

6.2 Clock Types

The profile allows for the following clocks to be used. See clause 3 of IEEE Std 1588-2019 for the full definitions.

GM	The PTP clock that is the source of time for all clocks in the domain.

TC	A PTP clock that measures the time for a PTP event message to transit the PTP clock and provides this information to PTP clocks receiving this PTP event message. The PTP clock in this profile supports the delay request – response mechanism (i.e., end-to-end Transparent Clock).
OC	A PTP clock that has a single PTP port in its domain and maintains the timescale used in the domain.

Some additional requirements that pertain to the GM and that are outside the PTP Profile are defined in the OCP-TAP Open Time Server project [4].

6.3 Message Types

The profile allows for the following messages:

a) Announce
b) Sync
c) Follow_Up
d) Delay_Req
e) Delay_Resp
f) Signaling
g) Management

6.4 Transport mechanisms required, permitted, or prohibited

The transport mechanism UDP over IPv6 per Annex D of IEEE Std 1588-2019 must be supported.

The transport mechanism UDP over IPv4 per Annex C should be supported.

The UDP checksum must be computed when a PTP message is retransmitted (see 3.1.65 of IEEE Std 1588-2019) by a TC.

The UDP destination port numbers per Annex C.2 of IEEE Std 1588-2019 must be supported. The UDP source port number of a unicast PTP message can be any ephemeral port number and should be preserved throughout the lifetime of a PTP connection that has been established using the unicast negotiation mechanism.

6.5 Clock identity

The clockIdentity must be an EUI-64 as specified in 7.5.2.2 of IEEE Std 1588-2019. The EUI-64 must be globally unique. If the EUI-64 is formed from an existing EUI-48, it must be done by appending two octets after the final six octets of the EUI-48 such that the 64 bits of the clockIdentity are not the same as the bits of any EUI-64 that has previously been assigned or may be assigned in the future by an authorized assignee of the MA-L, MA-M, or MA-S from which the EUI-48 was assigned. This means that either the entity that forms the EUI-64 owns the MA-L, MA-M, or MA-S from which the EUI-48 was formed, or the owner of that MA-L, MA-M, or MA-S has given the entity that forms the EUI-64 the sole right to the clockIdentity being formed.

Note: When using the MAC address, the clock identity is created by appending two octets after the final six octets of the MAC address. Note that in IEEE Std 1588-2008 the clock identity was formed by adding the two octets 'FFFE' between the 3rd octet and 4th octet of the MAC address, however, that mapping has been deprecated by the IEEE.

6.6 Path delay Measurement Mechanism

The path delay measurement mechanism must be the delay request-response mechanism. The value of the data set member portDS.delayMechanism must be E2E.

6.7 Class of Service

PTP event messages should set the DSCP field of the IPv6 Traffic Class field to the highest class of service possible. This should minimize latency and delay variation as PTP packets traverse a set of transparent clocks. In Model 1, the GM and OC should set the traffic class value.

6.8 PTP Security

PTP security is out of scope given the network will be a single trusted domain managed by a single entity.

6.9 Profile Isolation and Domain Number

All PTP instances must communicate using a single domain number, and the domainNumber value must be 0.

The sdoId is a new parameter in IEEE Std 1588-2019. A recognized standards organization, industry trade association, regulatory or government organization, or other organization as described in 20.3.2 of IEEE Std 1588-2019, can obtain an sdoId from the IEEE Registration Authority (RA). The sdoId is used to ensure that a PTP profile is isolated from any other PTP profiles running on the same network that are developed by other organizations.

An organization can obtain only one sdoId. If the organization develops multiple PTP profiles and requires that they be isolated, the isolation is further done using domainNumber. If an organization does not obtain an sdoId, the PTP profile will use the sdoId 0x000.

This PTP profile does not require an sdoId since it is assumed it will be the only profile within the data center network. If the assumption is not correct, another profile running on the network will conflict with this profile if the sdoId and domainNumber of the other profile are both 0.

Note – The sdoId is backward compatible with IEEE Std 1588-2008. The first nibble of the sdoId, i.e., the majorSdoId, corresponds to the transportSpecific field of IEEE Std 1588-2008. The final 8 bits of the sdoId, i.e., the minorSdoId, was reserved in IEEE Std 1588-2008 and was specified as 0x00.

6.10 One-step and two-step operation

A GM defined in this profile must support one-step or two-step operation on transmit or can support both on transmit.

A TC defined in this profile must support one-step operation on transmit (i.e., egress) on all its ports, or must support two-step operation on transmit on all of its ports. A TC defined in this profile should support one-step operation on all its ports (see 6.11).

All PTP clocks must support both one-step and two-step operation on receive (i.e., ingress).

A PTP port can transmit a Sync message as one-step or two-step. If the transmission of the Sync message is one-step, the twoStepFlag of the PTP common header is set to FALSE, otherwise it is set to TRUE. For PTP messages other than Sync, the twoStepFlag must always be set to FALSE. All PTP Ports must be capable of receiving and processing one-step and two-step Sync messages.

Note: one-step operation reduces the number of PTP messages transmitted by a PTP port. This may be applicable when considering scalability of unicast communication that a GM can serve. A one-step operation might ease meeting the requirements regarding the transmission of Sync messages specified in 9.5.9 of IEEE Std 1588-2019.

Note: IEEE Std 1588-2019 allows one-step versus two-step operation to be on a PTP port basis. However, IEEE Std 1588-2019 does not describe this capability. This profile requires that all PTP ports on a per clock basis be the same.

6.11 End-to-End TC with two-step operation

This section applies to the scenario where two-step TC operation may be used.

If an end-to-end TC uses two-step operation, each Delay_Req and corresponding Delay_Resp message must traverse the same end-to-end TC. This is because the end-to-end TC timestamps the Delay_Req message on ingress and egress and computes the residence time of the Delay_Req message. However, in the two-step case the TC updates the residence time of the corresponding Delay_Resp message. This is described in detail in 10.2.2.2.2 and 10.2.2.2.3 of IEEE Std 1588-2019. The former subclause describes the one-step case and specifies that the “<residenceTime> of the Delay_Req message must be added to the correctionField of the Delay_Req message by the egress PTP Port of the TC prior to the retransmission of the Delay_Req message.” In this case, it is the Delay_Req message that is altered by the TC, and not the Delay_Resp message. However, the latter subclause describes the two-step case, and specifies that the “<residenceTime> must be added to the correctionField of the Delay_Resp message associated with the Delay_Req message prior to transmission of the Delay_Resp message on the egress PTP Port, which is the ingress PTP Port for the Delay_Req message.”

If all the TCs are two-step, the Delay_Req and Delay_Resp must traverse the same set of transparent clocks (links and network elements) between the GM and OC to meet the subclause requirements. This property might not always hold true when using for example packet spraying, load balancing and equal cost multipath techniques. This is particularly applicable to data center environments and a reason for recommending the use of one-step operation TCs as noted in section 6.10.

If all the TCs are one-step, the Delay_Req and Delay_Resp need not traverse the same set of TCs (links and network elements) between the GM and OC.

6.12 PTP message rate

Table 1 defines the range of message rates for Announce, Sync, Delay_Req, and Delay_Resp messages. A GM must support the full range. An OC should support the full range but can support a subset of the range. The message rate selected by an OC relates to the performance expected. A TC is agnostic to the PTP message rate.

Message	Upper end of logMessageInterval range	Mean rate corresponding to upper end of range (pps)	Lower end of logMessageInterval range	Mean rate corresponding to lower end of range (pps)	Default value of logMessageInterval
Announce	+4	0.0625 (1 per 16s)	-3	8	4
Sync	+3	0.125 (1 per 8 s)	-7	128	0
Delay_Req & Delay_Resp	0	1	-7	128	0

Table 1. Range of logMessageInterval for a PTP Port

6.13 PTP inter-message interval

The requirements for the actual inter-message intervals for unicast Announce, Sync, Delay_Req, and Delay_Resp messages are specified in 16.1 of IEEE Std 1588-2019. There are requirements for:

- (a) the arithmetic mean of the successive inter-message interval computed over a suitable number of successive intervals
- (b) the distribution of the inter-message intervals

For Announce and Sync messages, the arithmetic mean of the inter-message intervals must be within ±30% of the granted inter-message period. For Delay_Req and Delay_Resp messages, the arithmetic mean of the Delay_Req inter-message intervals must not be less than 90% of the granted inter-message interval for the Delay_Resp messages. The purpose of this requirement is to ensure that the GM port receives Delay_Req messages at rates that it can handle. If the mean inter-message interval of the Delay_Req messages is less than 90% of the granted inter-message interval for the Delay_Resp messages, the grantor port may ignore any Delay_Req messages more than the granted interval.

For the distribution of the inter-message intervals, at least 90% of the inter-message intervals must be within ±30% of the granted mean inter-message interval. This requirement applies to Announce, Sync, and Delay_Req.

Consider N successive inter-message intervals Δt_i , $i = 1, 2, \ldots, N$, where $\Delta t_i = (t_i - t_{i-1})$. The arithmetic mean of the inter-message intervals, Δt_{av}, is

$$\Delta t_{av} = \frac{1}{N}\sum_{i=1}^{N} \Delta t_j$$

For example, if the grantor port grants Sync or Announce messages with logMessageInterval equal to 0, the mean inter-message interval is 1 s. This means that (a) the average of the durations of a suitable number of successive inter-message intervals Δt_{av} must be between 0.7 s and 1.3 s, and (b) 90% of the actual inter-message intervals must have durations that are between 0.7 s and 1.3 s. In addition, if the GM port grants Delay_Resp messages with logMessageInterval equal to 0, then (a) the average of the durations of a suitable number of successive Delay_Req inter-message intervals must be greater than or equal to 0.9 s, and (b) 90% of the actual Delay_Req inter-message intervals must have durations that are between 0.7 s and 1.3 s.

In principle, the mean Sync rate, and the mean Delay_Req/Delay_Resp rate need not be the same. If the actual delay on the PTP communication path is changing sufficiently slowly (after the OC has processed any correction field), then infrequent delay measurements compared to the mean Sync interval might give acceptable performance. In this case, the mean Delay_Req/Delay_Resp rate can be chosen to be smaller than the mean Sync rate. The Sync rate that is chosen depends on the implementation of the OC filter and how much noise the oscillator at the OC generates. If the oscillator has a large noise generation, then the Sync rate would likely be larger. In this case, the OC would use new Sync information more frequently to correct for time error.

6.14 Best Clock Algorithm and Clock Attributes

This profile uses the alternate BMCA (A-BMCA) described in this subclause. The requirements for an A-BMCA are described in 9.3.1 of [2]. The A-BMCA differs from the default BMCA found in [2] in its use of the new localGmPriority member of the unicastDiscoveryPortDS, which is described below.

The clock attributes for the GM and OC are given in Table 2. The attributes clockClass, clockAccuracy, offsetScaledLogVariance are set in the defaultDS and represent properties of the local clock, which is either the internal oscillator or an external time source that provides time to the GM outside of PTP or that is integrated with the GM. For clock class, the GM datasheet should specify the maximum amount of time required to transition from clockClass 7 to clockClass 52.

The priority1 attribute is not used and must be set to 128. It is not used in this PTP profile because it has higher preference in the A-BMCA than all the other attributes. A misconfiguration could cause an OC to choose the wrong GM.

The priority2 attribute can be configured to force potential GMs to be active or standby GMs for specific OC groups, and to implement full or partial redundancy based on active-standdy GMs as shown in section 6.15.3 and Appendix I. If priority2 is not used i.e., default value of 128 in all potential GMs, then the selection of the actual GM by the A-BMCA is based on localGmPriority and clockIdentity (assuming other attributes are equal).

The attribute timeReceiverOnly is TRUE for an OC and FALSE for a GM. The attribute timeTransmitterOnly is FALSE for an OC and TRUE for a GM.

The attribute ptpTimescale is always TRUE because this PTP profile uses the PTP timescale. The other timePropertiesDS attributes have values in the GM based on whether the values are traceable to a primary reference or in the case of timeSource based on the actual source of time for the clock.

The synchronizationUncertain attribute is optional. It is carried as a flag in the Announce message. This is new in IEEE Std 1588-2019 and might not be supported if the PTP nodes are based on a previous version of the protocol. If it is not used, its value is FALSE. If it is used at an OC, it is set to TRUE if:

- The synchronizationUncertain flag in the Announce message received from the GM is set to TRUE, or
- The state of the PTP port of the OC is UNCALIBRATED

Otherwise synchronizationUncertain for the OC is set to FALSE. If the synchronizationUncertain attribute is used at a GM, it is set to TRUE if the GM time or frequency, or both, are not traceable to a primary reference, otherwise it is set to FALSE.

The A-BMCA differs from the default BMCA of [2] in its use of the new member localGmPriority of the unicastDiscoveryPortDS, i.e., the unicast timeTransmitter table (UMT). This member is an array with datatype UInteger8, i.e., they can have values in the range [0, 255], with a default value of 128. The A-BMCA works exactly similarly to the default BMCA but differs from the default BMCA dataset comparison algorithm in that the localGmPriority comparison is made after the comparison of priority2 values and before the comparison of GM Identity values.

When comparing the two datasets A and B, the localGmPriorities of A and B are obtained from the UMT. When an Announce message is received, the potential GM is identified by the source IP address (i.e., the portAddress) of the IP packet that contained the Announce message. The respective localGmPriority for this potential GM is obtained from the UMT by finding the array index of this IP address in the UMT portAddress array. The localGmPriority is the element of the localGmPriority array having the same index.

Each OC requests Announce messages from ALL the potential GMs in the UMT. The data set comparison algorithm of the A-BMCA compares localGmPriority values after the priority2 value comparison and before the tiebreaker GM Identity comparison, of the two potential GMs being compared. If the attributes priority1, GM class, GM accuracy, Gm offsetScaledLogVariance, and priority2 of A and B are the same, the decision of which of GM A and GM B is better will be made based on localGmPriority. The localGmPriority values of a GM can be different at different OCs, because each OC has its own UMT whose members can be different from the UMTs of other OCs. This means that one OC can prefer GM A to GM B while another OC can prefer GM B to GM A. This feature allows a network using this PTP Profile to be configured for the active-active GM case. Examples are given in Appendix II.

If all the potential GMs of the UMT have the same value of localGmPriority, the A-BMCA operates exactly the same as the default BMCA of [2].

The data set members listed in Table 2 are not applicable to TCs. TCs do not participate in the A-BMCA.

Data set	Member	Value	
		GM	OC
defaultDS	clockClass	6 (traceable to a primary reference time source) 7 (in holdover, and within holdover specifications) 52 (in holdover but out of holdover specifications, or in free-run)	255
defaultDS	clockAccuracy	0x22 (250 ns)	0xFE (unknown)
defaultDS	offsetScaledLogVariance	0x4E5D (PTPVAR = 1.144×10^{-15} s^2, or TDEV = 30 ns)	0xFFFF (maximum possible value, signifying unknown)
defaultDS	priority1	128 (not used in this profile)	128 (not used in this profile)
defaultDS	priority2	Configurable over [0, 255]. Default value is 128. For active-standby configuration.	Configurable over [0,255]. Default value is 128
defaultDS	timeReceiverOnly	FALSE	TRUE
portDS	timeTransmitterOnly	TRUE	FALSE
timePropertiesDS	currentUtcOffset	If known, the value traceable to a primary reference that provides UTC. Otherwise the value when the node was designed	currentUtcOffset of E_{best}, after running BMCA
timePropertiesDS	currentUtcOffsetValid	TRUE if the values of currentUtcOffset, leap59, and leap61 are based on values obtained from a primary reference providing UTC; otherwise set to FALSE	currentUtcOffsetValid of E_{best}, after running BMCA

timePropertiesDS	leap59	If known, to a value traceable to a primary reference; otherwise set to FALSE	leap59 of E_{best}, after running BMCA
timePropertiesDS	leap61	If known, to a value traceable to a primary reference; otherwise set to FALSE	Leap61 of E_{best}, after running BMCA
timePropertiesDS	timeTraceable	FALSE (not used in this profile)	FALSE (not used in this profile)
timePropertiesDS	frequencyTraceable	FALSE (not used in this profile)	FALSE (not used in this profile)
timePropertiesDS	timeSource	If known, to the appropriate value from Table 6/IEEE Std 1588-2019. Otherwise set to INTERNAL_OSCILLATOR	timeSource of E_{best}, after running BMCA
timePropertiesDS	ptpTimescale	TRUE	TRUE
currentDS	synchronizationUncertain	FALSE (default)	FALSE (default)
unicastDiscoveryPortDS (unicast timeTransmitter table)	localGmPriority	128 (not used in this profile)	Configurable over [0,255]. Default value is 128

Table 2. Data set members and values

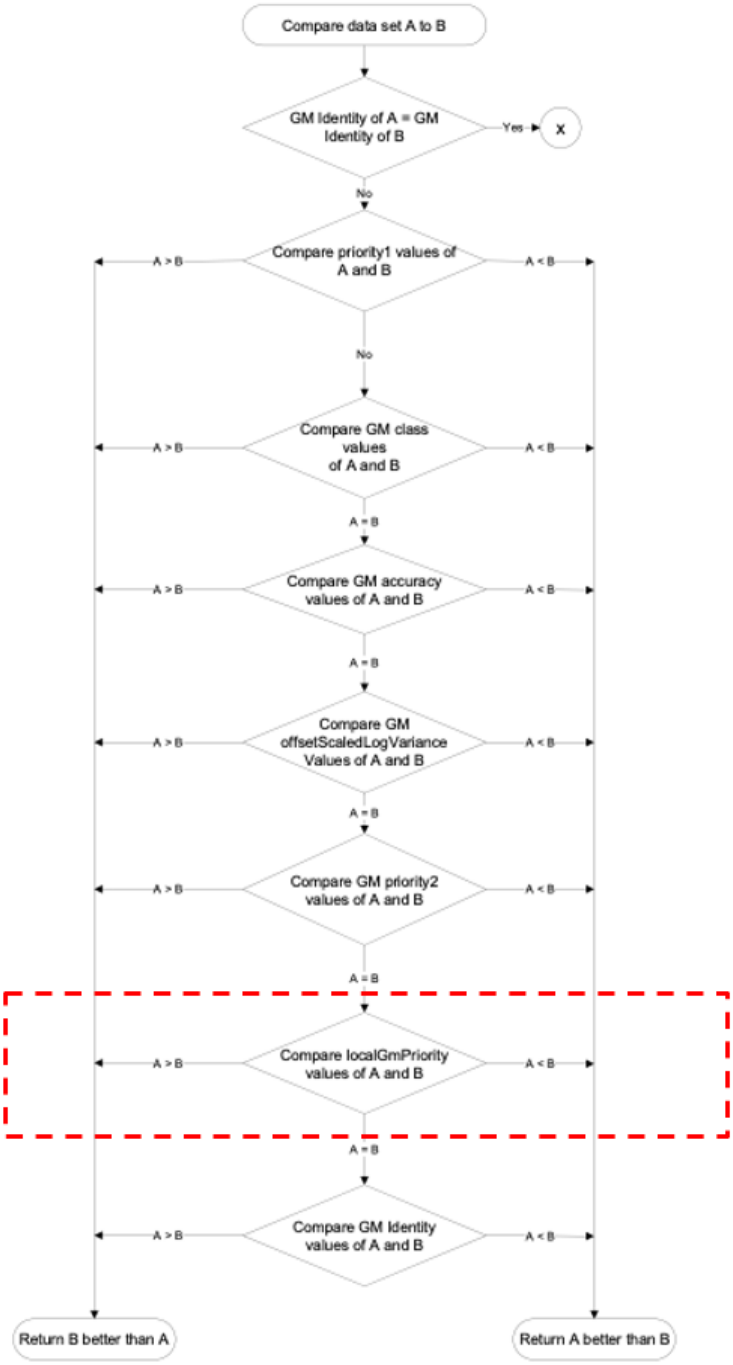

Figure 2. Dataset comparison algorithm for A-BMCA (the red, dashed box highlights the difference with the default BMCA algorithm of Figure 34 of [2])

Note: The BMCA defined in the previous document (i.e, primaryVersion: 1 revisionNumber: 0) was based on the default BMCA. In this document (i.e, primaryVersion: 1 revisionNumber: 1) the BMCA is based on an A-BMCA. The A-BMCA and the default BMCA can co-exist in a network in the sense that the protocol will function per IEEE Std 1588. A set of OCs can be synchronized by a GM per the default BMCA while another set of OCs can be synchronized by a GM per a-BMCA.

6.15 Unicast Communication

PTP communication in this profile is based on unicast. Most PTP profiles in the industry are based on multicast, except for two of the ITU-T telecom profiles that are based on unicast [5, 6].

Both unicast discovery (clause 17.4 of IEEE Std 1588-2019) and unicast negotiation (clause 16.1 of IEEE Std 1588-2019) must be supported. In Model 1, each OC first uses unicast discovery to determine the potential GMs, and then uses unicast negotiation to request Announce messages from the potential GMs. The OC then invokes the A-BMCA to determine which of the potential GMs becomes the actual GM, i.e., the active GM. Finally, the OC uses unicast negotiation to request Sync and Delay_Resp messages from the active GM and uses the Sync, Delay_Resp, and Delay_Req information to synchronize to the GM that was selected. The other potential GMs are available as backup if the active GM fails or can be selected as active GM for other OCs.

The A-BMCA can be used in two ways. In one of the ways, the elements of the localGmPriority array are all set to the same value, and the A-BMCA operates the same as the default BMCA of [2]. With this approach, full or partial redundancy is achieved by having some of the GMs active and some standby. This is described in 6.15.3, and examples of this are given in Appendix I. In the second way, the elements of the localGmPriority array are set to different values. With this approach, all the GMs are active and, when a GM fails, the OCs synchronized by that GM switch to other GMs. This is described in 6.15.4, and examples are given in Appendix II.

The unicast negotiation feature is permanently enabled. The unicastNegotiationPortDS.enable member (of the unicastNegotiationPortDS) must be TRUE for each PTP port (there is a unicastNegotiationPortDS for each PTP port). This dataset member applies to GM and OC and is not applicable to TC.

The unicastFlag of all PTP messages must be set to TRUE.

6.15.1 Unicast Discovery

Unicast discovery is specified in 17.4 of IEEE Std 1588-2019.

In Model 1 of this PTP profile, a table of potential GMs is configured in each OC. The table is sometimes referred to as the Unicast Table (UMT) and is defined in the unicastDiscoveryPortDS data set in clause 17.4.3 of IEEE Std 1588-2019. This data set contains the following members:

a) maxTableSize: the maximum number of potential GMs that can be in the table

b) logQueryInterval: the logarithm to base 2 of the mean time interval, in seconds, between successive requests that the OC makes to a potential GM for Announce messages (if a request is not granted), with a default value of 0,
c) actualTableSize: the number of potential GMs currently in the table; and
d) portAddress: an array containing the protocol addresses, i.e., IPv6 addresses of the potential GMs.
e) localGmPriority: an array containing the local priorities of the potential GMs. The data type is UInteger8. This array is a new member of the unicastDiscoveryPortDS used in the A-BMCA described in 6.14.

Each OC uses unicast negotiation to request Announce messages from each potential GM contained in the unicastDiscoveryPortDS. If a potential GM does not grant the request, the OC attempts again after a time interval corresponding to logQueryInterval. The received Announce messages cause a state decision event, which causes the A-BMCA to be invoked. This results in one of the potential GMs becoming the active GM for the OC. If that GM fails, the OC will stop receiving announce messages and the accounceReceiptTimeout will expire. This will invoke the A-BMCA. The A-BMCA will result in one of the other GMs (i.e., the best of the remaining potential GMs) becoming the active GM. If there are no GMs in the unicastDiscoveryPortDS or if none of the GMs in the unicastDiscoveryPortDS grants Announce messages to the OC, the OC will go into either free-run or holdover.

After the GM is selected, the OC uses unicast negotiation to request Sync and Delay_Resp messages from the GM. Upon being granted Sync messages, the OC receives the Sync messages from the GM. Upon being granted Delay_Resp messages, the OC sends Delay_Req messages to the GM and receives a Delay_Resp message in response to each Delay_Req message.

6.15.2 Unicast Negotiation

An OC requests Announce messages and then selects the best potential GM using the A-BMCA. The OC then requests Sync and Delay_Resp messages from that GM. After the OC is granted Sync messages, the GM sends Sync (and Follow_Up if the communication is two-step) messages to the OC. After the OC is granted Delay_Resp messages, the OC sends Delay_Req messages to the GM and the GM responds with Delay_Resp. The requesting of Announce, Sync and Delay_Resp messages is done using the unicast negotiation feature of IEEE Std 1588-2019. The unicast negotiation feature is performed using the following four TLVs:

- REQUEST_UNICAST_TRANSMISSION
- GRANT_UNICAST_TRANSMISSION
- CANCEL_UNICAST_TRANSMISSION
- ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION

Each TLV is attached in a Signaling message.

The sending, receiving, and processing of unicast negotiation TLVs by OCs and GMs must comply with the requirements of 16.1 of IEEE Std 1588-2019 and its subclauses. The following text in this section is a summary description of the unicast negotiation process.

TCs do not participate in the unicast negotiation process. However, they do forward the unicast Signaling messages that contain the unicast negotiation TLVs exchanged between the OCs and GMs.

The unicast negotiation process is illustrated in Figures 2, 3, 4 for requesting Announce, Sync, and Delay_Resp messages, respectively. An OC requests unicast Announce, Sync, or Delay_Resp from a GM by sending a REQUEST_UNICAST_TRANSMISSION TLV to the GM. This TLV contains the messageType field, which indicates the type of message (i.e., Announce, Sync, Delay_Resp), the logInterMessagePeriod field, which is the logarithm to base two of the desired mean interval, in seconds, between successive messages of this type, and the durationField, which is the number of seconds for which the GM should continue to transmit these messages. The GM responds with a GRANT_UNICAST_TRANSMISSION TLV to either grant or deny the request. This TLV contains the messageType field, which indicates message being granted, the logInterMessagePeriod field, which is the logarithm to base two of the granted mean interval, in seconds, between successive messages of this type, the durationField, which is the granted number of seconds for which the GM will continue to transmit these messages, and the R (Renewal Invited) flag, which is TRUE if the GM considers that the grant is likely to be renewed if the OC requests a new grant after the current grant expires and FALSE otherwise. A value of zero for the durationField indicates that the grant has been denied. The granted logInterMessagePeriod and durationField need not be the same as the requested logInterMessagePeriod and durationField, respectively.

An OC can request the logInterMessagePeriod to be any value in the range specified in Table 1. A GM can grant different message rates to different OCs.

The duration of the grant begins when the GRANT_UNICAST_TRANSMISSION_TLV is transmitted and ends after a time interval equal to the value of the durationField has expired. Typically, the OC requests that the grant be renewed by sending a new REQUEST_UNICAST_TRANSMISSION TLV before the grant expires (i.e., before the end of the duration) so that the service will be continuous.

After the GM has granted Announce or Sync messages to the OC, the GM sends Announce or Sync messages to the OC. After the GM has granted Delay_Resp messages, the OC then sends Delay_Req messages to the GM and the GM responds with Delay_Resp. The GM responds to all Delay_Req messages that arrive before the grant expires. However, the GM may respond to a Delay_Req message, i.e., by sending the corresponding Delay_Resp message, after the grant expires, as long as the Delay_Req arrives before the grant expires.

An OC can cancel the grant by sending the CANCEL_UNICAST_TRANSMISSION TLV to the GM. This TLV contains the messageType field, which indicates the type of message whose grant is being canceled, and the R (maintainRequest) flag set to FALSE. The GM responds by sending the ACKNOWLEDGE_ CANCEL_UNICAST_TRANSMISSION TLV to the OC.

If a GM cannot continue to provide the granted messages before the durationField has expired, it can inform the OC by sending the CANCEL_UNICAST_TRANSMISSION TLV to the OC with the G (maintainGrant) flag set to FALSE. The OC responds by sending the ACKNOWLEDGE_ CANCEL_UNICAST_TRANSMISSION TLV to the GM. The GM should (i.e., this is recommended but not required) continue to send the messages until it receives the ACKNOWLEDGE_ CANCEL_UNICAST_TRANSMISSION TLV or it has sent an implementation-specific number of CANCEL_UNICAST_TRANSMISSION TLVs to the OC.

A Signaling message can contain more than one TLV.

In this PTP profile, all requests are made by an OC and all grants are made by a GM. An OC cannot grant services and a GM cannot request services.

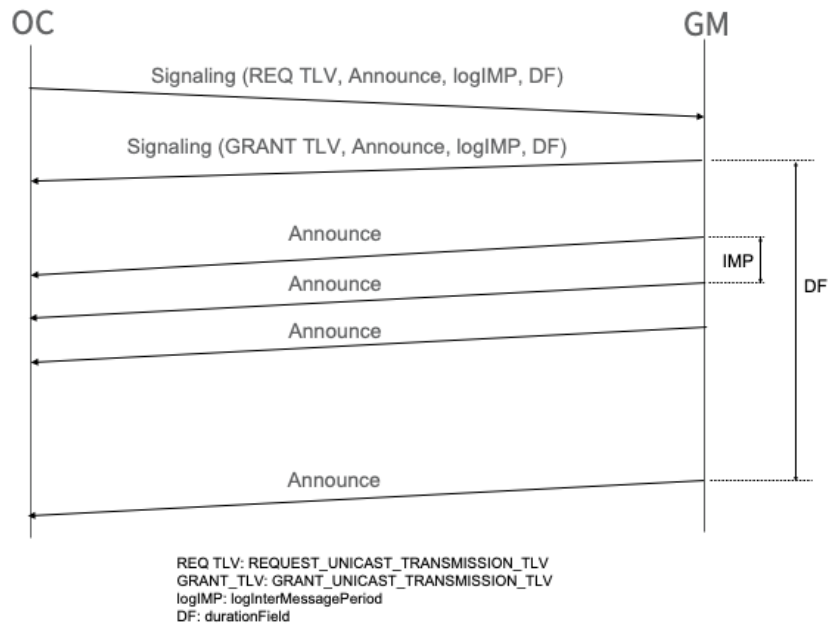

Figure 3. Unicast negotiation for Announce messages

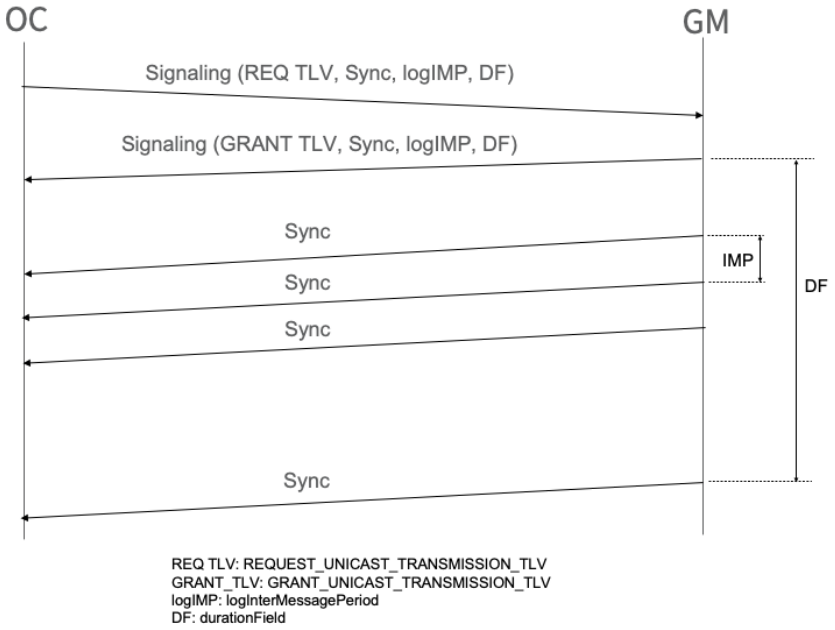

Figure 4. Unicast negotiation for Sync messages

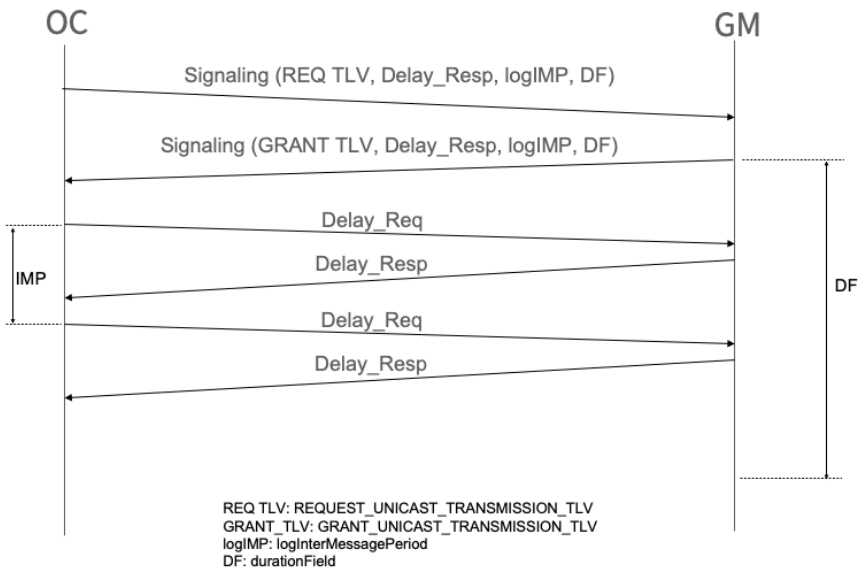

Figure 5. Unicast negotiation for Delay_Resp messages

6.15.3 Active-Standby Scenario

The A-BMCA can be used to provide full and partial redundancy of GMs during normal operation and failure scenarios, with some of the GMs being active and some being standby. In this scenario, all the localGmPriorities are set to the same value, and the A-BMCA operates in exactly the same manner at the default BMCA of [2]. Since all the localPriorities have the same value, the attributes of each potential GM are the same for all the OCs. This means that if one potential GM is deemed better than another potential GM by the dataset comparison algorithm by one OC, it is deemed better by all the OCs. As a result, a backup GM either is not used or only partially utilized when the primary GM (i.e., best GM, based on its attributes) fails.

Examples of the active-standby Scenario are given in Appendix I.

6.15.4 Active-Active Scenario

In the examples of 6.15.3, the backup GMs are in standby mode, i.e., they are not used during normal operation when there are no failures. A standby GM is only used when there is a failure. The examples in Appendix I illustrated that full redundancy, i.e., the ability to tolerate failure of all the GMs, requires an equal number of

standby GMs. In some applications, having a large number of GMs that are used only when there is a failure might be considered too costly. Unfortunately, as indicated in 6.15.3, the attributes of a potential GM are the same for all the OCs, i.e., it is not possible for different OCs to have different preferences among the GMs. This is because the dataset comparison algorithm of IEEE Std 1588 (see 9.3.4 of [2]) compares inherent attributes of the GMs. This means that if a GM that is intended to normally be active is preferred to a GM that is intended to normally be standby by one OC, all the other OCs have the same preference. To overcome this limitation, the A-BMCA described in 6.14 considers the new attribute, localGmPriority, in the dataset comparison algorithm, which allows different OCs to have different preferences among the potential GMs. Details are given in 6.14, and examples of the use of the A-BMCA in active-active scenarios are given in Appendix II.

6.16 PTP management messages

The profile uses the PTP management mechanism and PTP management messages (TLVs) defined in clause 15 of IEEE Std 1588-2019. The management messages are used by a PTP management node for the purpose of configuration and/or monitoring PTP Instances.

In this version of the profile, the following management TLVs must be supported:

- DEFAULT_DATA_SET (managementId 2000)
- CURRENT_DATA_SET (managementId 2001)
- PARENT_DATA_SET (managementId 2002)

The following additional TLV should be supported (Note: This TLV is an implementation specific TLV and is supported by the linuxptp implementation. The TLV contains a set of PTP message counters that can be used for monitoring):

- PORT_STATS_NP (managementId C005)

Additional PTP management TLVs might be defined for the purpose of calculating time error bounds. This is for further study.

6.17 Network Limits and Error Budget for Model 1

This section is an initial analysis. The network limit from Section 5, is:

- The maximum absolute time error of any OC, relative to TAI, must be $\max|TE_{OC}| \leq 2.5$ μs.
- The time accuracy difference between any two OCs must be within ±5 microseconds, i.e., $|T_{OC,j} - T_{OC,k}| \leq 5$ μs for $k \neq j$.

The following effects contribute to $\max|TE_{OC}|$:

a) Timestamp granularity. This is due to the clock frequency used for timestamping generation.

b) Timestamp generation. This is due to timestamping generation not being at the exact location where the timestamp is being taken, i.e., at the reference plane (see 7.3.4.2 of IEEE Std 1588-2019).
c) Combination of residence time and free-run accuracy of a TC. In this profile, the TCs are assumed to be free-running. They are not syntonized either at the physical layer or via PTP
d) Maximum number of TCs between a GM and an OC.
e) Noise generation due to OC oscillator characteristics.
f) PLL filter characteristics.
g) GM accuracy. This is the maximum time error of the GM relative to TAI when traceable. This analysis refers to ITU-T G.8272 PRTC-A specification [7].
h) Constant time error. This is due to link and node asymmetry after any compensation
i) Time error allowance produced by or within the application (i.e., any additional error between the PTP layer and the application/server).
j) Effect of a transient if an OC loses its active GM and switches to a backup GM.
k) Effect of long-term holdover of the GM (e.g, GNSS jamming, solar activity) or an OC if a backup GM is not available.

Table 3 contains initial assumptions for the effects given above. The value in the table, except for timestamp granularity, maximum residence time, number of TCs, and endpoint filter characteristics, refers to an absolute value.

Effect	Value
Timestamp granularity	8 ns
Timestamp generation	8 ns
Maximum residence time in a TC	0.1 ms
Free-run accuracy of TC oscillator	100 ppm
Maximum number of TCs	5
OC noise generation	100 ns (TBD)
OC endpoint filter characteristics	TBD
GM accuracy relative to TAI when traceable	100 ns
Constant time error	200 ns
Time error allowance for the application	200 ns (TBD)

Effect of a transient if an OC loses its reference to active GM and switches to a standby GM	1400 ns (TBD - see below)
Effect of long-term holdover of the GM with clockClass 7 on an OC if a backup GM with clockClass 6 is not available	1400 ns over time T specified by the vendor (TBD see below)

Table 3. Maximum absolute time error budget

The maximum error introduced by a TC due to free-run accuracy and residence time is $(0.1 \times 10^{-3}\ s)(10^{-4}) = 10^{-8}\ s$ = 10 ns. A TC will also introduce errors of 8 ns due to timestamp granularity and 8 ns due to timestamp generation. These errors will be added at both ingress and egress, for a total of 32 ns. The total error introduced by a TC in going from ingress to egress is therefore 42 ns.

The errors due to timestamp granularity and timestamp generation are also introduced at the GM egress and the OC ingress. These errors will add 16 ns, for a total of 32 ns.

The above errors contribute to total time error at the OC (to the end application). First, they accumulate as a Sync message traverses the network from the GM to the OC and contributes to the error in the recovered time at the OC. Second, they also accumulate as the Sync and Delay_Req message traverses the network from GM to OC and OC to GM and contribute to the error in the mean path delay at the OC. The total error that accumulates as either the Sync or Delay_Req message traverses the network, assuming there are 5 TCs in the path, is 5(42 ns) + 32 ns = 242 ns. The total error in synchronized time is therefore the sum of the error for Sync and the error in measured path delay, i.e., 242 ns (error in Sync) + 242 ns (error in meanPathdelay) = 484 ns (error in the time offset between the OC and GM). Finally, the 100 ns for the OC noise generation must be added to give 584 ns.

The error introduced by the GM, based on PRTC-A, is 100 ns.

The total allowance for constant time error due to link and node asymmetry is based on G.8271.1. G.8271.1 allows 800 ns for a network that consists of 20 hops with links that are likely much longer than those expected in a data center environment (i.e., the fiber length between nodes in a data center are within meters or tens of meters). Given that cTE is linearly additive and that the number of clocks consists of 5 TCs, 1 OC and 1GM, the total cTE is about ¼ the allocation found in G.8271.1. Therefore, the constant time error is 200 ns.

The total error at the input of the application is 584 ns + 100 ns + 200 ns + 200 ns = 1100 ns. This is well within the $\max|TE_{OC}| \leq 2.5\ \mu s$.

If the OC loses its connection to the network and enters holdover or the GM loses its connection to its time source (e.g., GNSS) and enters holdover with clockClass = 7, it can be assumed that the application already has already built-up an error of 1100 ns relative to TAI. In worst case, the application could drift another 1400 ns before it exceeds the 2.5 μs requirement. This means that the holdover requirement for the OC or the GM can be taken as 1400 ns over a time period T. This period T should be specified by the OC or GM datasheet. In addition, if the OC switches from one active GM to another active GM, any transient during this switch must be within 1400 ns.

7 References

[1] OCP Timing Appliances Project (TAP) Incubation Proposal, July 2020, https://www.opencompute.org/wiki/Time_Appliances_Project

[2] IEEE Std 1588-2019, IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, June 2020

[3] OCP Contribution, Practical Uses of Synchronized Clocks, Sept 2020, https://www.opencompute.org/wiki/Time_Appliances_Project

[4] OCP Open Time Server, https://www.opencompute.org/wiki/Time_Appliances_Project

[5] ITU-T G.8275.2, Precision time protocol telecom profile for phase/time synchronization with partial timing support from the network

[6] ITU-T G.8265.1, Precision time protocol telecom profile for frequency synchronization

[7] ITU-T G.8272, Timing characteristics of primary reference time clocks

8 Document Version

Version	Comments
0.1	Initial DC PTP profile document submitted to OCP
0.2	Converted v0.1 into OCP document template. Addressed comments received from various contributors
0.3	Added revision table. Updated OCP license section.
0.4	Added IEEE CID and updated profileIdentifier based on received CID (Company ID). Added PTP management messages. Added clarifying text on IPv6/IPv4/UDP.
1.0	Initial version submitted to OCP incubation committee and for review committee. Released on OCP-TAP wiki and github.
2.0	Minor editorial corrections. A-BMCA definition with localGmPriority attribute. UMT updates based on A-BMCA definition. Appendix with Active-Standby and Active-Active examples.

9 Contributors

EDITOR(S):

Michel Ouellette, Meta

CONTRIBUTOR(S):

Thomas Kernen, Nvidia

Greg Armstrong, Renesas

Alon Regev, Kingshuk Mandal, Keysight

Nazar Schynskyy, SiTime

Amit Oren, Broadcom

Oleg Obleukhov, Patrick Cullen, Alexander Bulimov, Meta

10 License

OCP encourages participants to share their proposals, specifications and designs with the community. This is to promote openness and encourage continuous and open feedback. It is important to remember that by providing feedback for any such documents, whether in written or verbal form, that the contributor or the contributor's organization grants OCP and its members irrevocable right to use this feedback for any purpose without any further obligation.

11 Appendix I. Examples of Active-Standby Configuration

In the examples of this appendix, the localGmPriority of each GM is set to the same value, i.e., 128. As a result, localGmPriority has no impact on the determination, by the dataset comparison algorithm, of which of two GMs is better. The A-BMCA operates in exactly the same manner at the default BMCA of [2].

Figure 6 shows an example which consists of 1000 OCs divided into 2 groups, each with 500 OCs. There are 4 potential GMs, designated 1 through 4, respectively. GM 1 and GM 2 are potential GMs for OC group 1 and their IPv6 address is entered into the unicastDiscoveryPortDS of each OC of group 1. GM 3 and GM 4 are potential GMs for OC group 2 and their IPv6 address is entered into the unicastDiscoveryPortDS of each OC of group 2. The attributes of the GMs are set such that GM 1 is better than GM 2 as determined by the A-BMCA and GM 3 is better than GM 4 as determined by the A-BMCA. Assuming the GMs all have the same clockClass, clockAccuracy, and offsetScaledLogVariance, this can be done by configuring the priority2 attribute such that priority2 for GM 1 and GM 3 is less than priority2 for GM 2 and GM 4, respectively. This assumes that priority1 is set to the same default value in all GMs to prevent it from accidentally overriding the effect of clockClass, clockAccuracy, and offsetScaledLogVariance. This is done in other PTP profiles such as ITU-T Rec. G.8275.2 [5], which is also based on unicast discovery and unicast negotiation. Alternatively, if clockClass, clockAccuracy, offsetScaledLogVariance, and prioirity2 are the same in each potential GM but the clockIdentities of GM 1 and GM 3 happen to be less than the clockIdentities of GM 2 and GM 4, respectively, GM 1 and GM 3 will also be chosen as the active GMs for groups 1 and 2, respectively. In addition, in this final case where the potential GMs have the same clock attributes, it might not matter which is active and which is standby. The A-BMCA will result in GM 1 and GM 3 being the active GMs for OC groups 1 and 2, respectively, and GM 2 and GM 4 being the standby GMs for OC groups 1 and 2, respectively. The use of clockIdentites is the tiebreaker.

Example 1 also shows that a standby GM is not utilized if the active GM of the respective OC group has not failed. In this example, failures of both active GMs can be tolerated. However, the two standby GMs are not utilized unless there are failures.

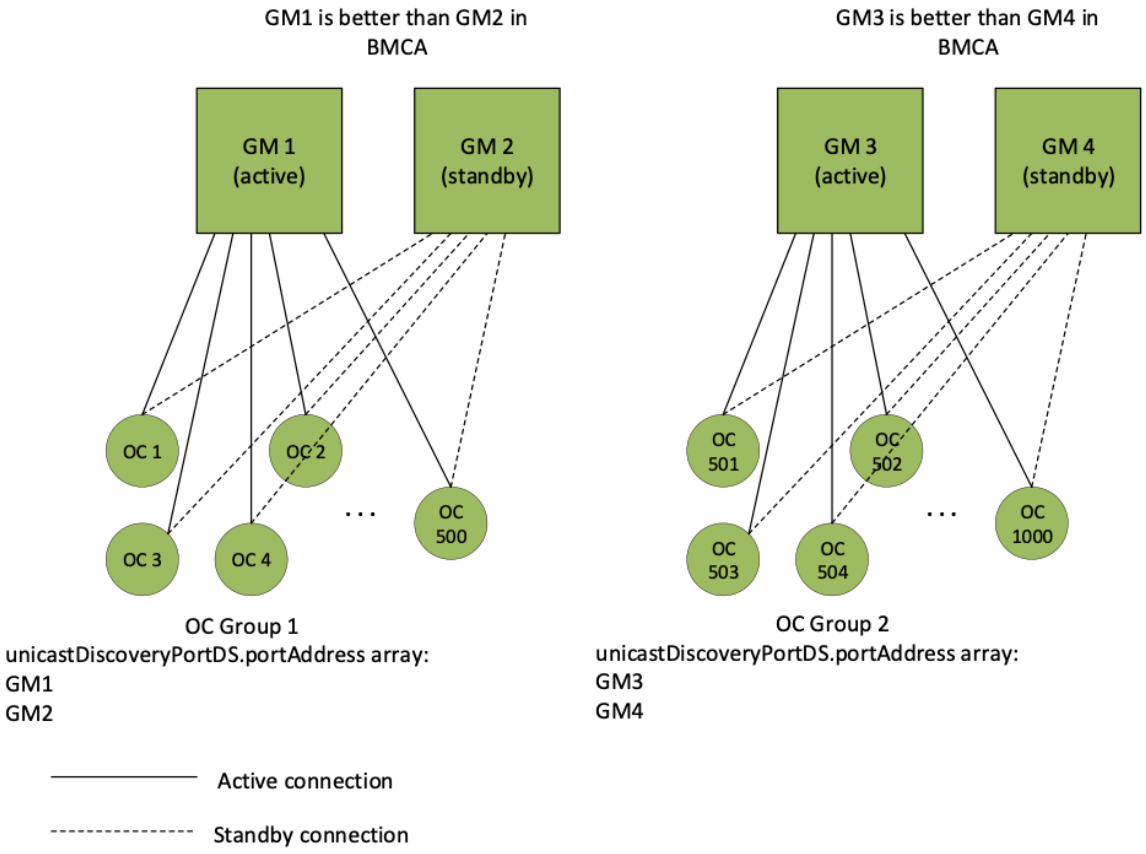

Figure 6. Example1 of Active-Standby GMs across two groups each with 500 OCs

Figure 7 shows an example which consists of 2 OC groups with 3 potential GMs, designated 1 through 3, respectively. GM 1 and GM 3 are potential GMs for OC group 1 and are entered into the unicastDiscoveryPortDS of each OC of group 1. GM 2 and GM 3 are potential GMs for OC group 2 and are entered into the unicastDiscoveryPortDS of each OC of group 2. GM3 is essentially a shared GM between the 2 OC groups. The attributes of the GMs are set such that GM 1 and GM 2 are each better than GM 3 as determined by the A-BMCA. As in the example above, this can be done by configuring the priority2 attributes such that priority 2 for GM 1 is less than priority 2 for GM 3, and priority2 for GM 2 is less than priority2 for GM 3. This will also occur if the clockIdentities of GM 1 and GM 2 are each less than the clockIdentity of GM 3 and all the other attributes of GMs 1, 2, and 3 are the same. The A-BMCA will result in GM 1 and GM 2 being the active GMs for OC groups 1 and 2, respectively. GM 3 will be the standby GM for both groups 1 and 2. If either GM 1 or GM 2 fails, GM 3 will become the GM for the group whose GM has failed. If both GM 1 and GM 2 fail, then either GM 3 will become the GM for both OC groups 1 and 2, and therefore must be able to handle the load of both groups or only a single failure (i.e., of a single GM) can be tolerated.

In Example 2, there is only a single standby GM, and therefore only a single GM is not utilized if there are no failures (unlike Example 1, where two GMs are not utilized if there are no failures). However, either the single

standby GM must handle a higher load if both active GMs fail, or else only a single active failure can be tolerated at any time.

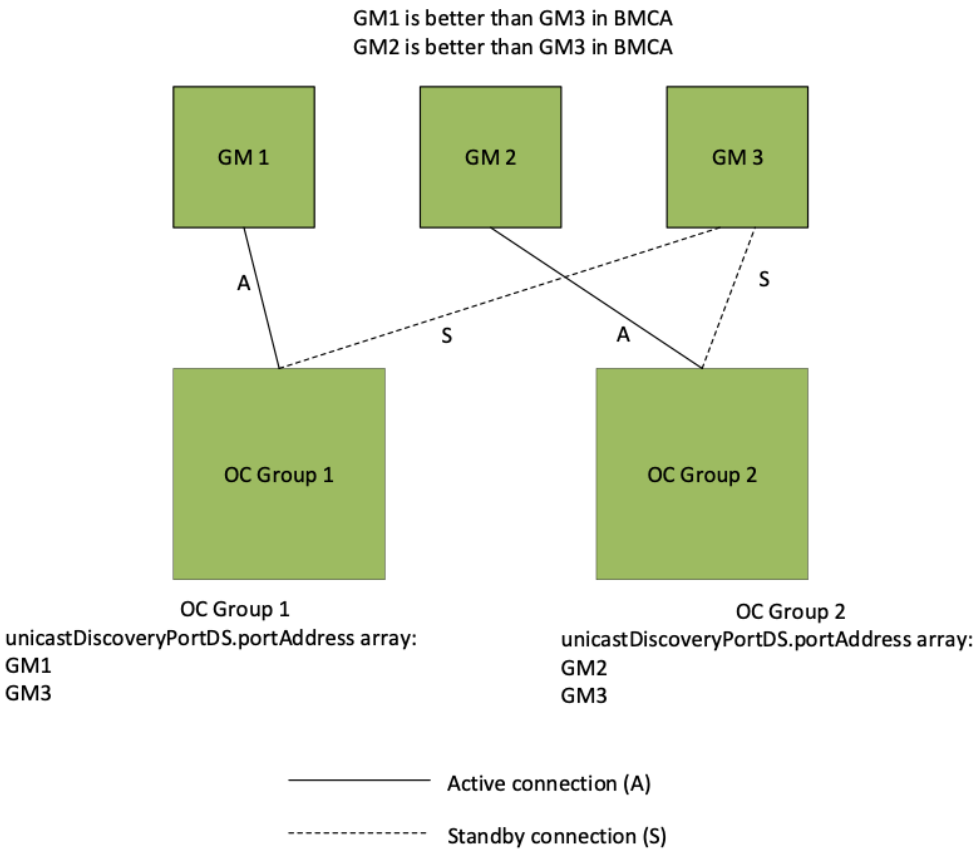

Figure 7. Example2 one active GM for each group and one standby GM for both groups

Example 1 illustrates the case of full redundancy where there is one standby GM for each active GM. Example 2 illustrates the case of partial redundancy where there are fewer standby GMs than active GMs. To balance the load among the active GMs, the OCs should be divided as evenly as possible among the active GMs. To balance the load among the standby GMs and also achieve maximum robustness to failures, the standby GMs should be assigned to equal numbers of OC groups. For example, if there are 60000 OCs, 12 potential active GMs, and 4 potential standby GMs, the OCs should be divided into groups of 5000 OCs each (i.e., 60000 OCs/12 GMs), and each of the 12 potential active GMs should be entered in the unicastDiscoveryPortDS of the OCs of exactly one group. Each potential standby GM should be entered in the unicastDiscoveryPortDS of the OCs of exactly 3 groups (and no group should have two standby GMs entered in the unicastDiscoveryPortDS of any of its OCs). With this approach, a standby GM serves as a backup for up to 3 OC groups. In this example, if a standby GM can handle the load of up to N groups ($N \leq 3$), then N active failures can be tolerated.

12 Appendix II. Examples of Active-Active Configuration

II.1 Introduction

When the A-BMCA of 6.14 is used, an OC requests Announce messages from the potential GMs in the UMT without regard to their localGmPriority values (i.e., potentially from all the potential GMs in the UMT). The OC can be granted Announce messages by some or all the GMs. In any case, the OC uses the received Announce messages in the A-BMCA. If all the GMs have the same values for priority1, clockClass, clockAccuracy, offsetScaledLogVariance, and priority2, the best GM will be chosen based on the localGmPriority values. Since the localGmPriority table values can be different in each OC, this enables different OCs to favor different GMs in different orders of priority, with the objective of achieving equal distribution of PTP connections amongst the set of active GMs. In addition, if a GM fails, the OCs will no longer receive Announce messages from that GM. In this case, the OCs for which this GM had the highest localGmPriority will now choose the GM with the second highest localGmPriority as the best GM.

In the general case of *N* GMs, the UMT of each OC has *N* GM entries and *N* values of localGmPriority. In this case, there can be up to *N* – 1 failures with synchronization still provided to the OCs (if all the GMs fail, synchronization is no longer provided by the GMs). It can be shown that equal distribution of OCs among GMs after each of the successive *N* – 1 failures can be obtained if the *N* values of localGmPriority are different and if each OC has a different permutation of the *N* different values of localGmPriority. Unfortunately, the number of permutations is very large even for a moderate number of GMs. To simplify the approach, the GMs and OCs can be divided into groups, with approximately the same numbers of GMs and OCs in each group. This results in a smaller number of GMs in each group and a smaller number of permutations of the values of localGmPriority for that group. The drawback is that, if there are one or more failures within a group, the resulting equal distribution of OCs among the GMs is only within that group and not across all the OCs. The result is that some of the GMs will be more heavily utilized than others.

II.2 Illustration

The approach is illustrated here for the case of a cluster of 4 GMs and 24 OC groups to achieve equal numbers of OCs synchronized by each GM after up to 3 failures (i.e., the total number of permutations of four objects is 24). Figure II-1 shows a cluster consisting of 4 GMs (GM 1, GM 2, GM 3, and GM 4) and 24 groups of OCs (Groups 1 through 24). The number of OCs in each of the 24 groups does not matter and is dependent on the total number of OCs within the cluster. The UMT localGmPriority values for the OCs in each group are shown in Table 4. Every possible permutation of the four GM priorities is contained in the 24 entries of Table 4. This is possible because for 4 GMs there are 4! = 24 permutations.

OC Group Number	Best GM (GM with priority 0)	2nd best GM (GM with priority 1)	3rd best GM (GM with priority 2)	4th best GM (GM with priority 3)
1	1	2	3	4
2	1	2	4	3
3	1	3	2	4
4	1	3	4	2
5	1	4	2	3
6	1	4	3	2
7	2	1	3	4
8	2	1	4	3
9	2	3	1	4
10	2	3	4	1
11	2	4	1	3
12	2	4	3	1
13	3	1	2	4
14	3	1	4	2
15	3	2	1	4
16	3	2	4	1
17	3	4	1	2
18	3	4	2	1
19	4	1	2	3
20	4	1	3	2
21	4	2	1	3
22	4	2	3	1
23	4	3	1	2
24	4	3	2	1

Table 4. UMT localGmPriority for OCs in each group, for the case of 4 GMs and 24 OC groups.

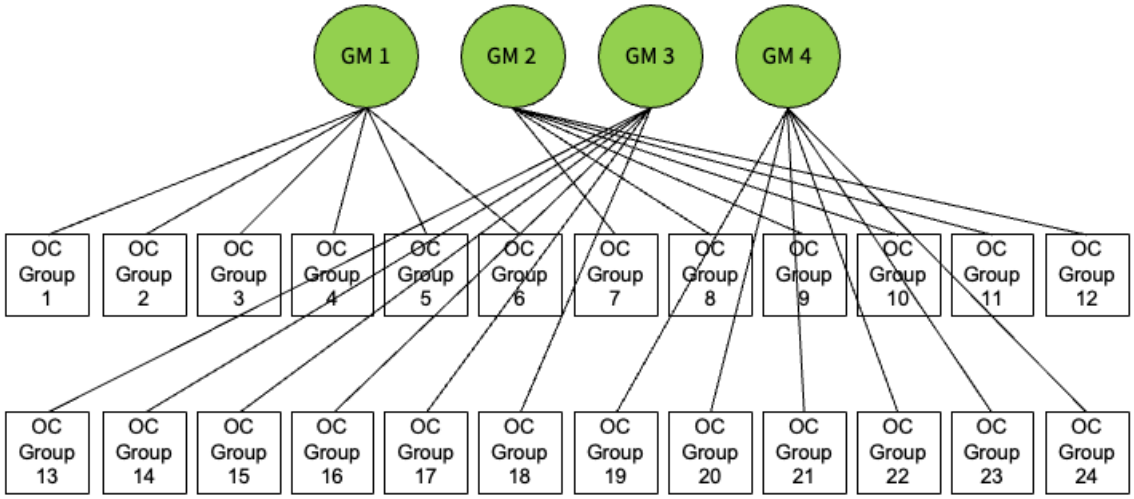

Figure 8. OC Groups 1 to 24 synchronized by the respective best priority GM, based on priorities given in Table 4.

Figure 8 also shows the best priority GM for each OC Group. Since every permutation of the four priorities is present in Table 4, each GM synchronizes 6 OC Groups. The result is a uniform distribution of the OC Groups among the GMs. GM 1 synchronizes OC Groups 1 to 6, GM 2 synchronizes OC Groups 7 to 12, GM 3 synchronizes OC Groups 13 to 18, and GM 4 synchronizes OC Groups 19 to 24.

Figure 9 shows the case where GM 1 has failed. Based on Table 4, OC Groups 1 to 6, which were previously synchronized by GM 1, switch to the second-best GMs in their UMTs. This is GM 2 for OC Groups 1 and 2, GM 3 for OC Groups 3 and 4, and GM 4 for OC Groups 5 and 6. After the failure, each remaining GM synchronizes 8 of the OC Groups (instead of 6 originally), and the distribution of OC Groups across the GMs is still uniform and the PTP connections are equally balanced across the remaining GMs. GM 2 synchronizes OC Groups 1, 2, and 7 to 12, GM 3 synchronizes OC Groups 3, 4, and 13 to 18, and GM 4 synchronizes OC Groups 5, 6, and 19 to 24.

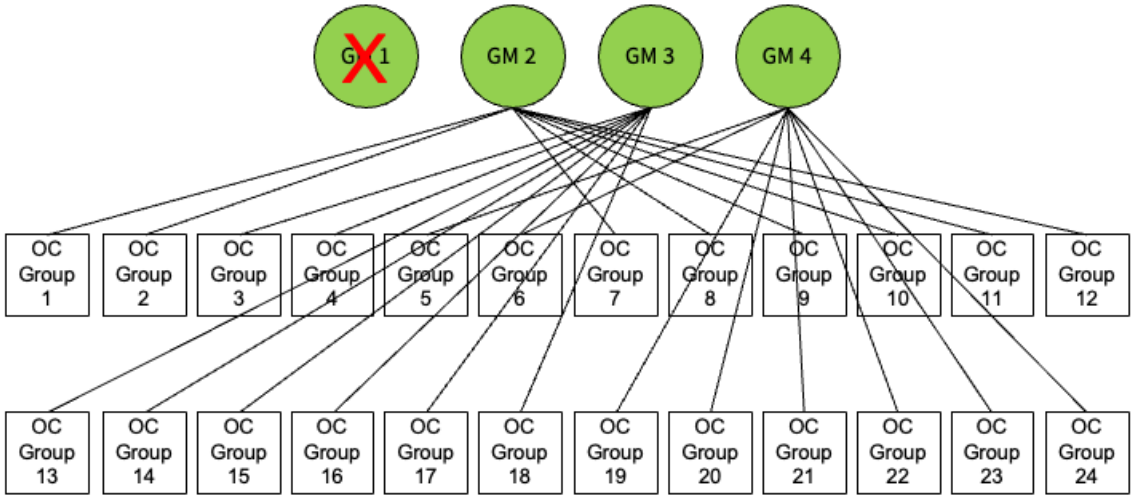

Figure 9. OC Groups 1 – 24 synchronized by the respective best priority GM, based on priorities given in Table 4, for the case where GM 1 has failed.

Figure 10 shows the case where GM 1 and GM 3 have failed. Assuming GM 1 has failed first, then based on Table 4, OC Groups 3, 13, 15, and 16, which were previously synchronized by GM 3 (see Figure 9), switch to GM 2. OC Groups 4, 14, 17, and 18, which were previously synchronized by GM 3 (see Figure 9) switch to GM 4. After the failure, each remaining GM synchronizes 12 of the OC Groups, and the distribution of OC Groups across the GMs is still uniform. GM 2 synchronizes OC Groups 1, 2, 3, 7 to 12, 13, 15, and 16, while GM 4 synchronizes OC Groups 4, 5, 6, 14, and 17 to 24. Note that the same final configuration is the result if GM 3 fails first, followed by GM 1.

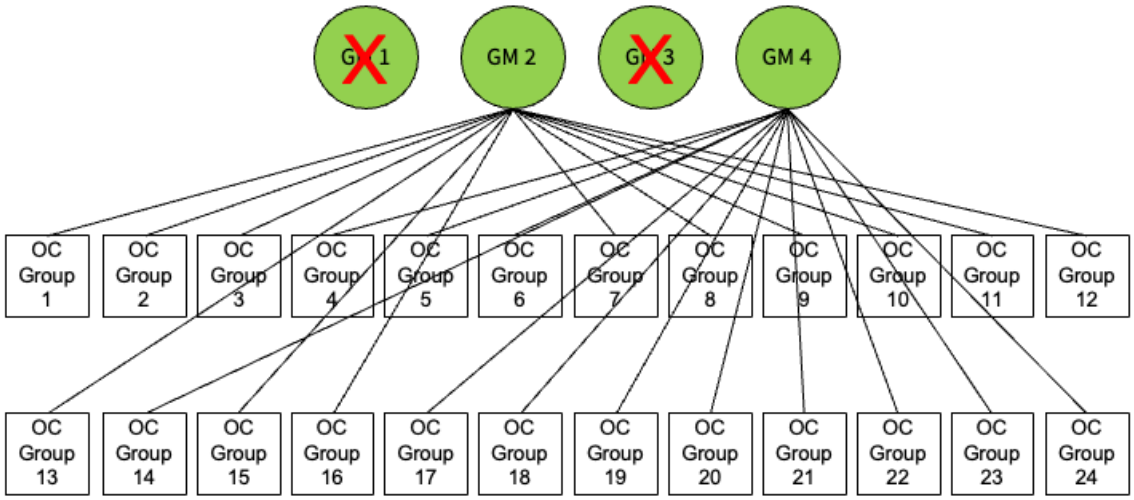

Figure 10. OC Groups 1 – 24 synchronized by the respective best priority GM, based on priorities given in Table 4, for the case where GM 1 and GM 2 have failed.

It is seen from Table 4 that if 3 of the 4 GMs fail, all the OC groups switch to the remaining GM that has not failed. Finally, it is seen that the reason the uniform distribution of OC Groups across the four GMs is obtained after a failure is that the UMTs of the 24 groups contain all possible permutation of the four GM priorities. To satisfy this condition, the number of OC groups must be N! if there are N GMs. Dividing the GMs into a set of independent clusters is necessary to minimize implementation complexity.