SAE INTERNATIONAL

AEROSPACE STANDARD

AS6802™ REV. A

Issued 2011-11
Reaffirmed 2016-11
Revised 2023-02

Superseding AS6802

(R) Time-Triggered Ethernet

RATIONALE

This version includes improved normative descriptions based on program and user experiences collected over the last 10 years.

INTRODUCTION

Time-triggered Ethernet functionality described in this SAE Aerospace Standard (AS) is a Layer 2 quality-of-service (QoS) enhancement for Ethernet networks. It provides the capability for deterministic, synchronous, and congestion-free communication, unaffected by any asynchronous Ethernet traffic load.

This occurs via a fault-tolerant, self-stabilizing synchronization strategy, which helps to establish temporal partitioning and ensures isolation of the synchronous time-critical dataflows from other asynchronous Ethernet dataflows.

By implementing this standard in network devices (network switches and network interface cards), Ethernet becomes a deterministic network which can be shared by low-latency, low-jitter, and non-time-critical applications. This means that distributed applications with mixed time-criticality requirements (e.g., real-time command and control, audio, video, voice, data) can be integrated and coexist on one Ethernet network.

TABLE OF CONTENTS

TO PLACE A DOCUMENT ORDER: Tel: 877-606-7323 (inside USA and Canada)
Tel: +1 724-776-4970 (outside USA)
Fax: 724-776-0790
Email: CustomerService@sae.org
SAE WEB ADDRESS: http://www.sae.org

For more information on this standard, visit
https://www.sae.org/standards/content/AS6802A/

1. SCOPE

This SAE Aerospace Standard (AS) defines a fault-tolerant synchronization strategy for building and maintaining synchronized time in a distributed system of end systems and switches (we use the term “end system” for data terminal equipment (DTE), as specified in IEEE 802.3), which can be used to support communication among these components for traffic, which may have different levels of time criticality. In particular, the standard defines algorithms for clock synchronization, clique detection, startup, and restart. These algorithms have been designed to allow scalable fault-tolerance and provide self-stabilization mechanisms.

Time-triggered Ethernet supports the design of communication systems with mixed time criticality in which several applications of mixed time criticality share a single physical network. In particular, an Ethernet network can be used to transfer frames in a time-triggered mode (synchronous communication) and non-time-triggered modes (asynchronous communication as for example Ethernet frames transmitted according to the best-effort strategy). The time-triggered Ethernet synchronization strategy inherently compensates for latency and jitter resulting from this integration and ensures high-quality synchronization despite increased network latency and jitter. Synchronized time provides the foundation for partitioning and isolation of critical applications from the less critical or non-critical ones.

End systems exchange application data with each other by transmitting standard Ethernet frames. The points in time when end systems dispatch these frames can be coupled to the synchronized time. The transfer of these frames is then called time-triggered transfer, because the trigger for frame dispatch is derived from time. Time-triggered Ethernet formally defines the relationship between the synchronized time and the time-triggered transfer.

Time-triggered Ethernet covers only the network aspects for mixed time-criticality systems[1]. Time-triggered Ethernet does not address how to integrate mixed time-criticality applications within a single node. Hence, partitioning strategies for shared resources other than the network, e.g., memory partitioning, are not discussed in time-triggered Ethernet. Furthermore, the fault-tolerance strategies discussed in this standard also address only the networking aspects. Time-triggered Ethernet does not specify or recommend any complete system architecture for highly reliable systems.

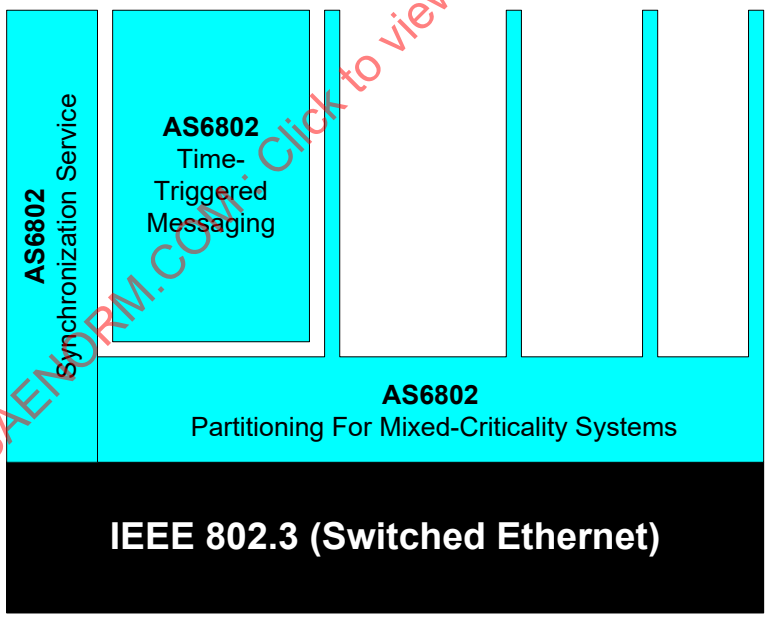


Figure 1 - Scope overview: This standard specifies a fault-tolerant synchronization protocol to be leveraged for time-triggered communication and partitioning

[1] A distributed system implementing this standard can use the same physical network for applications of mixed time criticality and unified Ethernet networking.

1.1 Purpose

This standard is a Layer 2 quality-of-service (QoS) enhancement that defines time-triggered services for Ethernet networks. Time-triggered Ethernet is designed for the development of highly dependable systems for applications in multiple industries, including integrated systems in aerospace, ground vehicles, and industrial process control. It provides the capability for deterministic, synchronous, and congestion-free (lossless) communication among distributed applications, unaffected by any asynchronous Ethernet traffic load. This standard is compatible with higher OSI layers (3 through 7) and is transparent to applications designed to use asynchronous Ethernet.

1.1.1 Efficient Access Control Management

Global time can provide a powerful fault isolation mechanism for devices with temporal faults because global time operates as a temporal firewall. In case of a failure, it is not possible for a faulty application to have access to the network at points in time other than those configured *a priori* and stored in locations not accessible to applications. Depending on the location of the failure, either an end system or a switch will block faulty transmission attempts. Failures of a switch can be masked by particular design choices, i.e., the so-called high-integrity designs, such as self-checking pairs. This fault masking transforms any failure of a time-triggered Ethernet switch into an inconsistent omission failure. This means that inconsistent omission failure is taken into account by the synchronization services described in this standard.

1.1.2 Unified Networking

The fraction of communication bandwidth assigned to time-triggered communication can be precisely located in the temporal domain. This temporal specification allows isolation of time-critical messages from messages that are not time-triggered. Bandwidth that is either not assigned to time-triggered communication or assigned but not used is free for communication that is not time-triggered. In this standard, two traffic classes, in addition to the time-triggered traffic class, are supported. These are named rate-constrained (compatible with the ARINC 664-p7 concept) and standard Ethernet (IEEE 802.3) traffic. However, a communication infrastructure that implements the time-triggered services specified in this standard document may use the non-time-triggered bandwidth for any protocol only as long as the impact of the non-time-triggered traffic on the time-triggered traffic is bounded to an application-specified degree.

1.1.3 Efficient Resource Use

The global time contributes to efficient resource use in several ways. For example, time-triggered communication allows for minimizing the memory buffers in network devices (e.g., switches) as the time-triggered communication schedule is free of conflicts. Therefore, the switches do not have to prepare for worst-case bursts of frames arriving from multiple ports to be delivered over the same destination physical link.

1.1.4 Precise Diagnosis

A global time-stamping service, such as can be provided by this standard, simplifies the process of reconstructing a chain of distributed events. At the same time, the synchronous capturing of sensor values makes it possible to build snapshots of the overall system status.

1.1.5 Temporal Composability

Using global time allows the specification of device interfaces not only in the value domain, but also in the temporal domain. This means that during the design process of devices, there is a predefined access pattern to the communication network. Because of this, devices can be developed in parallel. Upon integration of the individual devices, prior service stability guarantees that the individual devices operate as a coordinated whole.

1.1.6 Real-Time Capability

Time-triggered communication is highly suited for periodic command and control tasks or synchronous data delivery with constant latency and minimum jitter (in microseconds or sub-microseconds).

1.1.7 Scalable Network

Time-triggered Ethernet QoS enhancements described in this document are appropriate for a wide range of applications with scalable fault-tolerance requirements and design of N-redundant Ethernet networks.

1.1.8 Circuit Switching Emulation

Circuit-switching behavior with fixed latency and sub-microsecond jitter can be emulated in packet-switched Ethernet networks. Time-triggered Ethernet switching devices send packets according to a schedule that relies on global time.

1.2 Application

This standard has been designed to cover a broad spectrum of fault-tolerance and dependability requirements; e.g., single and dual fault-tolerance. At the same time, the synchronization strategy can adjust in scale for compatibility with cross-discipline applications (e.g., aerospace, automotive, medical, and industrial).

1.3 Interpretation

The following definitions shall apply when used in this document, unless otherwise stated:

MAY: An allowed action.

SHALL: A mandatory requirement.

SHOULD: A recommended action.

WILL: A declaration of intent.

1.4 Structure

Each section of this document consists of both informative and normative subsections. In general, the last subsection is the normative section and is always labeled “Normative Description” All other subsections are informative. Within the normative sections, paragraphs beginning with “NOTE:” identify informative text.

2. REFERENCES

2.1 Applicable Documents

The following publications form a part of this document to the extent specified herein. The latest issue of SAE publications shall apply. The applicable issue of other publications shall be the issue in effect on the date of the purchase order. In the event of conflict between the text of this document and references cited herein, the text of this document takes precedence. Nothing in this document, however, supersedes applicable laws and regulations unless a specific exemption has been obtained.

2.1.1 ARINC IA Publications

Available from SAE ITC, ARINC Industry Activities, 16701 Melford Blvd, Suite 120, Bowie, MD 20715, Tel: +1 240-334-2578, https://www.aviation-ia.com/.

ARINC 653P0 Avionics Application Software Standard Interface

ARINC 664P7-1 Aircraft Data Network, Part 7, Avionics Full-Duplex Switched Ethernet Network

2.1.2 IEEE Publications

Available from IEEE Operations Center, 445 and 501 Hoes Lane, Piscataway, NJ 08854-4141, Tel: 732-981-0060, www.ieee.org.

IEEE STD 802.3-2005 Ethernet Standard

2.2 Definitions

ACCEPTANCE WINDOW: The window (in time) around a scheduled receipt point. An integration frame received within this window is in schedule, while an integration frame received outside its respective acceptance window is out of schedule.

BABBLING IDIOT FAILURE: Babbling Idiot Failure describes a particular class of behavior of a component or device that exhibits a failure. A component/device that exhibits this failure mode will randomly produce arbitrary messages on its interfaces.

BEST-EFFORT (BE) MESSAGE: Ethernet frame that is sent as best-effort traffic. It is recommended to use the term “best-effort-transferred frame” instead.

BEST-EFFORT (BE) FRAME: Ethernet frame that is sent as best-effort traffic. It is recommended to use the term “best-effort-transferred frame” instead.

BEST-EFFORT (BE) TRAFFIC: Non-critical Ethernet traffic. BE traffic is typically the lowest priority.

BEST-EFFORT (BE) TRANSFER: An Ethernet frame that is communicated over a communication link without any bounded timing constraints or hard temporal guarantees.

CHANNEL: A redundancy concept implied by a fault-tolerant system; always defined within the scope of a cluster or multi-cluster. End systems may attach to one or more channels. A message traverses devices and communication links belonging exclusively to one channel. This implies that a channel does not share devices and communication links with other channels defined within this cluster or multi-cluster. Channels are named with letters (e.g., A, B, C). For more elaborate topologies (e.g., ring, partly redundant topologies), do not use the term “channel”; rather, use “logical link” or “frame route.”

CLIQUE: Cliques are formations of subsets of synchronization masters and/or compression masters, such that the components synchronize within the subset, but not over subset boundaries.

CLIQUE DETECTION: Clique detection refers to an algorithm or procedure that is capable of detecting cliques once they are formed.

CLIQUE RESOLUTION: Clique resolution refers to an algorithm or procedure that is capable of resolving a clique scenario once it is formed. Typically clique resolution involves the restart of a subset of the time-triggered Ethernet devices or a complete network restart.

CLUSTER: A cluster is a logical group of devices (i.e., end systems and switches) with the same synchronization domain and the same synchronization priority, including the interconnecting communication links. Any device may be part of several clusters; currently available devices, however, can only be part of one synchronization priority and one cluster.

CLUSTER CYCLE: One iteration of the cyclically executed schedule of TT-transferred frame. As there may be different message periods in the schedule, the cluster cycle consists of the least common multiple of these message periods.

COLD-START ACKNOWLEDGE FRAME: A PCF used to acknowledge a previously received cold-start frame. A cold-start frame and cold-start acknowledge frame constitute the fault-tolerant handshake.

COLD-START FRAME: A PCF; synchronization masters periodically dispatch cold-start frames, as long as the devices in the network are not synchronized or in case of clique formations. Upon synchronization, only faulty devices will dispatch cold-start frames. A cold-start frame and cold-start acknowledge frame constitute the fault-tolerant handshake.

COMMANDER/MONITOR (COM/MON): In this high-integrity self-checking pair, a commander device performs the real actions. At the same time, a monitor device observes all inputs received by the commander, performs the same computations as the commander does, and shuts down the commander if it tries to apply any action that the monitor would not. In a switch, the monitored action typically is the transmission of a frame with specific contents (high-integrity guarantees that an operation is either correct or stopped, but never faulty).

COMMERCIAL OFF-THE-SHELF (COTS): Sometimes referred to as “component off-the-shelf,” COTS denotes products that are ready-made and available for sale, lease, or license to the general public.

COMMUNICATION CHANNEL: See “channel.”

COMMUNICATION CONTROLLER: The electronic circuit responsible for transmitting and receiving frames.

COMMUNICATION LINK: A connection between two physical ports, usually realized as a copper or glass fiber cable. Communication links may be unidirectional or bidirectional. In time-triggered Ethernet context, only consider bidirectional communication links.

COMPONENT: Short form of network component.

COMPRESSION MASTER (CM): A compression master collects PCFs from the synchronization masters. During synchronized operation, the compression master calculates a fault-tolerant average (time), based on the dispatch points (in time) of the PCFs from the synchronization masters. The compression master then dispatches a PCF of its own, where the dispatch point of this new PCF is derived from the fault-tolerant average time. The synchronization masters and synchronization clients then use the dispatch point of the new PCF to resynchronize their local clocks.

During unsynchronized operation (e.g., startup, restart, or clique resolution) the compression master executes a startup protocol, which constitutes rules on whether or not the compression master regenerates cold-start or cold-start acknowledge frames. The difference between regenerate and relay is that a regenerated frame is modified from the original received frame (e.g., the CT ID changes), while a relay process does not modify the frame.

CONTROLLER: The abbreviated term for a communication controller.

CYCLIC REDUNDANCY CODE: A checksum transmitted together with each frame. It allows the frame to be checked for correctness at the receiver by means of cyclic redundancy.

DATAFLOW: This is a generic term to describe the movement of information (e.g., Ethernet frames) within a network. This term is used interchangeably with the term “flow.”

DEVICE: An active network component (i.e., a switch or an end system).

DISTRIBUTED COMPUTER NETWORK: A network where at least some of the end systems (i.e., computers) work together (i.e., cooperate in a distributed way). From an application’s point of view, the application is distributed across several end systems and its parts run concurrently on them.

DYNAMIC DELAY: As a frame is transmitted from a producer to a consumer device, this frame will be delayed in its flow through the network. There is a difference between static delays and dynamic delays: static delays can be calculated offline, but for dynamic delays only upper bounds can be calculated offline. Frame transmission conflicts at the outgoing queues of a device usually impose dynamic delays.

DYNAMIC TRANSMISSION DELAY: See “dynamic delay.”

END SYSTEM: End system is the data terminal equipment (DTE), as specified in IEEE 802.3

ETHERNET FRAME: A frame according to the standard IEEE 802.3. An Ethernet frame contains a header comprising MAC destination address, MAC source address, and a 2-byte EtherType field. It also contains a 4-byte trailer comprising a CRC checksum.

ETHERNET LINK: A communication link over which frames are transmitted, according to the IEEE 802.3 standard.

ETHERNET NETWORK: A network where communication takes place, according to the IEEE 802.3 standard.

ETHERNET PORT: A physical port that is able to exchange Ethernet frames.

ETHERTYPE FIELD: The EtherType indicates either the length of the payload (values between 0 and 1500) or the subprotocol (i.e., the protocol used to process the payload (values of 1536 decimal (0x0600 hex) and greater)).

FAULT-TOLERANT HANDSHAKE: The sequence of cold-start and cold-start acknowledge frames. After a fault-tolerant handshake, the devices will enter a state in the protocol state machine that indicates synchronized operation.

FLOW: This is a generic term to describe the movement of information (e.g., Ethernet frames) within a network. This term is used interchangeably with the term “dataflow.”

FRAME: A complete unit of transmission, a data packet of fixed or variable length, encoded for digital transmission over a communication link. Each frame consists of a header, usually containing source and destination information and other transmission-relevant information; payload (application data); and an optional trailer, usually containing some kind of checksum.

FRAME ROUTE: Physical route of a frame from one end system (i.e., sender) to a number of other end systems (i.e., receivers). This includes information about which switches are “visited” by the frame in which order, and which ports and communication links it uses.

FRAME TYPE: The switch and end systems handle frames of the same frame type identically; however, different frame types may differ in their size, purpose, payload layout, and routing information, as well as in other aspects.

GUARDED COMMANDS: Term used to describe an algorithm in the form of a state-transition graph. If an algorithm is in a given state in the state machine, the guard describes a set of conditions to be met to trigger a set of commands. If these conditions are satisfied, then a set of commands associated with the guard can be executed.

HIGH INTEGRITY: High integrity refers to a particular design of a component. A component that is designed to be high integrity is also called a high-integrity component. In time-triggered Ethernet, a high-integrity component is formed by a commander/monitor pair. When a component is designed as high-integrity component, then it is industrial practice that the failure modes of this high-integrity component can be restricted. In the case of time-triggered Ethernet, the failure mode of a high-integrity component is inconsistent-omission faulty.

HOPS: The number of communication links between the producing end system of a frame and the receiving end system, minus one. Consequently, in a simple cluster, the maximum number of hops is one.

HOST: The computational unit within a node that executes the operating system and application software, usually comprising the CPU, RAM, and some kind of non-volatile memory (e.g., flash, ROM, EEPROM).

HOST PORT: An interface used to transport Ethernet frames to and from the host CPU.

INPUT FRAME: A frame received on any one of the Ethernet ports.

INTEGRATION CYCLE: The integration cycle is the nominal period of integration frames. From a clock synchronization perspective, the integration cycle is the best-case resynchronization interval. As time-triggered Ethernet permits continued synchronized operation, even in scenarios where a configurable number (e.g., y) of consecutive integration frames is lost, the worst-case resynchronization interval is given by y × integration cycle.

INTEGRATION FRAME: An integration frame is a PCF with the type field set to 0x2 hex. During synchronized operations, integration frames are sent in order to keep the local clocks of the different devices synchronized to each other.

LOGICAL LINK: A logical, unidirectional connection from one source end system to one or more other end system destinations. A logical link may be used, as required (desired connectivity), and does not need to specify the exact route.

LONGEST MESSAGE PERIOD: Maximum of all message periods in the cluster.

MAC ACCEPTANCE TABLE (MAT): A filtering mechanism to specify authorized management systems in the network.

MESSAGE: This term is used in two ways:

a. Generally speaking and in abstract context, it is used to denote a transmitted data entity. In a specific context this document uses the specific term (e.g., frame, signal, etc.).

b. Specifically, it is used to denote an element of data communication. As such, a message is defined by a data type and (optionally) an initialization value. In a TT context, messages have state semantics (i.e., each message contains information about the current state of a system parameter, like current temperature or current pressure). State messages are strictly periodic (i.e., they have an update period: the message period); non-blocking; and have only one sender/producer, but potentially many receivers/consumers.

MESSAGE PERIOD: Time between two consecutive transmissions of the same message.

MULTI-CLUSTER: A multi-cluster consists of clusters with different synchronization priorities, but the same synchronization domain.

MULTICAST FRAME: An Ethernet frame transmitted to multiple destinations (i.e., to a group of switch ports, with the ports being donated by the multicast group). The Ethernet multicast destination addresses have the least significant bit of the highest order byte set to 1.

MULTICAST GROUP: The receivers of a multicast frame.

NETWORK: A system of interconnected electronic components or circuits, generally termed network components. A network may consist of one or more clusters or multi-clusters.

NETWORK COMPONENT: Anything that is part of a network (i.e., active components, such as switches and end systems, as well as passive components such as communication links or physical ports).

NODE:

a. The entire logical/functional part of an end system (i.e., the computer, usually comprising the host, end systems).

b. Other I/O electronics.

A node is also referred to as smallest replaceable unit (SRU) or electrical control unit (ECU).

NODE CONFIGURATION: Describes the hardware-specific aspects of a device.

OBSERVATION WINDOW (OW): The compression master collects PCFs asynchronously and uses them for calculation of a fault-tolerant average (i.e., time). Depending on the number of PCFs received, the collection phase has a duration ranging from the length of one observation window to a configurable number, *f*, of observation windows. The maximum duration, *max_observation_window*, is given as: $max\_observation\_window = (f + 1) \times observation\_window$.

OFFLINE: Offline refers to the design phase of a time-triggered Ethernet network. For example the communication schedule for time-triggered traffic is produced typically in the design phase of the time-triggered Ethernet network and stored locally in the time-triggered Ethernet devices. Therefore, when the time-triggered Ethernet network enters operation the schedule is already established and does not have to be produced then.

Offline also refers to re-configuration of a time-triggered Ethernet network during operation. For example a spacecraft application such as a satellite in orbit which receives an updated communication schedule. A general characteristic of the offline phase is that decisions are pre-planned rather than dynamically derived.

OUTPUT FRAME: A frame sent on any one of the Ethernet ports or forwarded to the built-in management module of a switch.

PAYLOAD: The actual (user relevant) data within a frame.

PERMANENCE: Associated with a single frame, permanence refers to a point in time, the permanence point in time, from which a consumer can use this frame with a guarantee of no receipt of frames sent prior to this first frame.

PHYSICAL PORT: A physical communication end point. Devices have physical ports for interconnection to other devices. A communication link always attaches to a physical port.

PI: The precision in the time-triggered Ethernet network. The precision is the maximum difference between two local clocks in two correct devices in the time-triggered Ethernet network, after establishing a synchronized global time.

PORT: An interface for frame input and output. A port may be a physical port, a host port, or a management port.

POSITIVE ACKNOWLEDGMENT (PACK): Sometimes denoted as ACK.

PROTOCOL: A protocol is a set of guidelines or rules, especially a set of rules governing communication within and between computing entities.

PROTOCOL CONTROL FRAME (PCF): A dedicated Ethernet frame that carries time-triggered Ethernet protocol control information, primarily to synchronize the local clocks of the individual devices. A PCF is either a cold-start frame (CS), a cold-start acknowledge frame (CA), or an integration frame (IN).

RATE-CONSTRAINED (RC) MESSAGE: Ethernet frame that is sent as rate-constrained traffic. It is recommended to use the term “rate-constrained-transferred frame” instead.

RATE-CONSTRAINED (RC) FRAME: Ethernet frame that is sent as rate-constrained traffic. It is recommended to use the term “rate-constrained transferred frame” instead.

RATE-CONSTRAINED (RC) TRAFFIC: Time-triggered Ethernet traffic used for applications with less stringent determinism and real-time requirements than strictly TT applications. RC traffic guarantees predefined bandwidth for each application, as well as defined limits for delays and temporal deviations. Safety-critical automotive and aerospace applications use RC traffic to meet requirements for highly-reliable communication and moderate temporal quality requirements (e.g., multimedia systems).

RATE-CONSTRAINED (RC) TRANSFER: An Ethernet frame that is communicated over a communication link, according to the rate-constrained communication paradigm.

REDUNDANT (SET OF) CHANNEL(S): A channel B is redundant to channel A, if it connects to exactly the same end systems as channel A. The topology of channel B does not need to match that of channel A.

ROUTING (OF A FRAME): The action of adding the frame to the queue, corresponding to the respective frame type of traffic, of the associated port or ports. The switch decides to which outgoing port to forward incoming frames. This decision derives from routing tables, which specify the CT ID traffic class, necessary checks, ports, etc.

SCHEDULE: A description of the temporal communication behavior, including TT traffic and RC traffic. Schedules apply to specific synchronization domains.

SCHEDULER: A tool that is capable of producing a schedule.

SHORTEST MESSAGE PERIOD: Minimum message period in the cluster.

SIMPLE CLUSTER: A cluster consisting of exactly one switch per channel. A cluster where the maximum number of hops is one, when no defined channels exist.

STANDARD ETHERNET NETWORK: An Ethernet network with only standard traffic (i.e., traffic according to the IEEE 802.3 standard).

STANDARD ETHERNET TRAFFIC: Traffic that occurs in non-time-triggered Ethernet networks. It can be classified as BE traffic.

STANDARD INTEGRITY: Components that are not designed as high-integrity components (see “high integrity”) are standard-integrity components. It is industrial practice that an arbitrary failure mode has to be assumed for standard-integrity components.

STATIC DELAY: As a frame is transmitted from a producer to a consumer device, it has a delayed flow through the network. Static delays can be calculated offline. Dynamic delays (only upper bounds) can be calculated offline, but usually have frame transmission conflict delays imposed at the outgoing queues of a device.

STATIC TRANSMISSION DELAY: See “static delay.”

STORE-AND-FORWARD: The principle according to which a frame is processed within the switch. The switch completely receives a frame and stores it in the switch local memory, which may be a queue. At a later point in time, depending on the traffic class, the frame is sent to the output ports, as defined in the routing information (see schedule stored in the switch).

SUBSCHEDULE: Implementation detail for representing the schedule in an end system.

SWITCH: A device with many ports, which forwards frames from any port to any port(s), depending on its configuration.

SYNCHRONIZATION CLIENT (SC): A device that is not configured as synchronization master or compression master. A synchronization client will not generate (or regenerate) PCFs, but consumes them for synchronization purposes.

SYNCHRONIZATION DOMAIN: All devices configured to belong to the same synchronization domain may eventually synchronize to each other. Devices configured to different synchronization domains will never synchronize to each other. Also, a schedule will always be associated with a single synchronization domain, whereas different synchronization domains require separate schedules.

SYNCHRONIZATION MASTER (SM): A device which transmits its local time in PCFs to the compression masters.

SYNCHRONIZATION PRIORITY: A synchronization priority is associated with a single cluster. While each single cluster can establish and maintain synchronization on its own, the synchronization priority can automatically synchronize different clusters to each other. The multi-cluster synchronization is then priority based, where the cluster with the highest priority provides the timebase and all other clusters use this timebase for synchronization. No two clusters within the same synchronization domain may have the same synchronization priority assignment.

SYNCHRONIZATION PROTOCOL: A set of rules that is executed in the devices for the purpose of closely synchronizing device-local clocks to each other. Typically, a synchronization protocol distinguishes three phases of execution:

a. Acquiring the relative differences of the device-local clocks.

b. Calculating a correction term, based on the measured relative differences.

c. Applying the correction term to the device-local clocks.

In addition to these three phases, a synchronization protocol may define a dedicated startup protocol for establishing initial synchronization in the system. Time-triggered Ethernet defines such a startup algorithm.

SYNCHRONIZATION RESET: The event of losing synchronization with the network. This is the case if the synchronization state machine exits the *SM_SYNC* or *SM_STABLE* state for any state other than these two.

TIME-TRIGGERED (TT) MESSAGE: Ethernet frame that is sent as time-triggered traffic. It is recommended to use the term “time-triggered-transferred frame” instead.

TIME-TRIGGERED (TT) FRAME: Ethernet frame that is sent as time-triggered traffic. It is recommended to use the term “time-triggered-transferred frame” instead.

TIME-TRIGGERED (TT) TRAFFIC: Time-triggered Ethernet traffic used for applications with stringent determinism and real-time requirements. TT traffic guarantees predefined bandwidth and latency for each application; used for safety-critical automotive and aerospace applications that depend on highly reliable communication and have high temporal quality requirements (e.g., closed loop control systems).

TIME-TRIGGERED TRANSFER: An Ethernet frame that is communicated over a communication link, where the start of frame transfer is triggered by the synchronized global time.

TRAFFIC: Data transferred over a communication media or the transferred data itself. There are different kinds of traffic (i.e., traffic classes).

TRAFFIC CLASS: Traffic classes differ by the quality-of-service (QoS) they provide. In time-triggered Ethernet, traffic has three classes: time-triggered (TT), rate-constrained (RC), and best-effort (BE).

TRANSMISSION JITTER: The maximum variation in the transmission latency.

TRANSMISSION LATENCY: The duration starting with the sending trigger of a frame until the reception of the frame at a receiver.

TRANSMISSION RATE: Associated with RC traffic, the transmission rate of an RC frame is the inverse of the minimum duration between two successive transmissions of the RC frame.

TRANSPARENT SYNCHRONIZATION PROTOCOL: A synchronization protocol that can integrate on top of a legacy protocol and coexist with legacy dataflow.

UNICAST FRAME: An Ethernet frame transmitted to a single destination (i.e., to a single switch port). The Ethernet unicast destination addresses have the least significant bit of the highest order byte set to 0.

VIRTUAL LINK (VL): A logical link with specific properties, as defined in the ARINC 664 standard. Each virtual link is associated with a dedicated maximum bandwidth specified by the minimum frame interval, called the bandwidth allocation gap, and the maximum frame length.

WIRE SPEED: Usually refers to networking equipment with the capability to process data at aggregated wire speed (i.e., the bandwidth of the outgoing traffic equals the bandwidth of the incoming traffic).

2.3 Abbreviations

BE	Best-effort (prefix; see best-effort transfer)
CA	Coldstart acknowledge frame
COM/MON	Commander/monitor
COTS	Commercial (or component) off-the-shelf
CPU	Central processing unit
CRC	Cyclic redundancy
CS	Coldstart frame
DA	Ethernet (MAC) destination address
ECU	Electronic control unit
ES	End system
ETH	Ethernet (as a prefix)
FCS	Frame check sequence
ID	Identifier (mostly in combination with other acronyms)
IN	Integration frame
IP	Internet protocol

MAC	Media access control
OW	Observation window
PCF	Protocol control frame
RC	Rate-constrained
Rx	Receiver (prefix, e.g., to denote the direction of traffic)
SA	Ethernet (MAC) source address
TCP	Transmission control protocol
TT	Time-triggered (prefix, e.g., to denote time-triggered traffic)
Tx	Transmitter (prefix, e.g., to denote the direction of traffic)
UDP	User datagram protocol
VL ID	Virtual link identifier
VL	Virtual link

3. TIME-TRIGGERED ETHERNET OVERVIEW

Time-triggered Ethernet adds synchronization and time-deterministic data transfer characteristics to those Ethernet operations that use active star (e.g., hub or switch) topologies, while retaining full compatibility with the requirements of IEEE 802.3. Ethernet inherently uses an event-triggered transfer principle. Using this principle, an end system can access the network at arbitrary points in time. Service to the end systems is on a first come, first serve basis. An immediate drawback of this event-triggered principle is the cumulative transmission delay and jitter, which occurs when several end systems need to communicate over the same shared medium. In contrast, the time-triggered transfer principle uses a network-wide synchronized time base to coordinate between end systems, which lessens the transmission delay and jitter. Adding the time-triggered transfer principle to standard Ethernet makes it possible to reduce the delay and jitter to levels that meet the requirements for embedded real-time systems, including safety-critical control systems. Depending on the synchronization quality required, the implementation of these time-triggered capabilities can be in hardware (for high-precision synchronization) or in the form of software drivers for standard Ethernet chipsets (for less precise synchronization).

This standard defines the synchronization services that establish and maintain a global time, realized by the close synchronization of the local clocks of the end systems and switches. The global time forms the basis for implementing the beneficial time-triggered system properties (e.g., temporal partitioning, precise diagnosis, efficient resource use, or composability).

Time-triggered Ethernet provides characteristics and services in the following three categories:

a. Support for traffic with differing timing requirements: Time-triggered Ethernet enables time-triggered and event-triggered communication, as well as integrated time-triggered/event-triggered communication on the same physical network. Time-triggered Ethernet limits latency and jitter for time-triggered traffic, limits latency for rate-constrained (RC) traffic, while simultaneously supporting the best-effort (BE) traffic service of IEEE 802.3 Ethernet. This allows application of Ethernet as unified networking infrastructure.

b. Transparent Synchronization: Time-triggered Ethernet network uses protocol control frames (PCFs) to establish and maintain synchronization. The PCF traffic has highest priority; but, otherwise, it is similar to rate-constrained traffic. Time-triggered Ethernet compensates transmission jitter introduced during traffic propagation and operations in the switches. This compensation establishes a well-defined interface for fault-tolerant clock synchronization algorithms.

c. Scalable fault tolerance: Time-triggered Ethernet provides fault-tolerance mechanisms scalable to various application requirements. The time-triggered Ethernet design is scalable to cover two different failure hypotheses: a single-failure hypothesis and a dual-failure hypothesis.

3.1 Support for Traffic with Differing Timing Requirements

This standard specifies synchronization services for packet-switched Ethernet networks. An Ethernet-based system consisting of devices that implement these services is able to communicate synchronously with fixed latency and µs jitter. The core of these services is the synchronization of the local clocks in end systems and switches.

Time-triggered Ethernet specifies time-triggered services that are added to the standards for Ethernet established in IEEE STD 802.3-2005. Figure 2 shows a parallel view of the time-triggered services and the common Open Systems Interconnection (OSI) model layers. A communication controller that implements the time-triggered services can synchronize its local clock with the local clocks of other communication controllers and switches in the system. A system designer can define an offline schedule of frame transmission with respect to the synchronized time, and the time-triggered Ethernet devices can then dispatch frames according to this schedule. Such a transfer of a frame according to a synchronized time is called a “time-triggered transfer.” Offline scheduling tools can be used to guarantee that time-triggered transfers are conflict-free (i.e., no two time-triggered frames compete for transmission). Time-triggered communication is not priority-based and is driven only by time progression and frame schedule. This prevents network traffic congestion for time-triggered frames and enables communication with fixed latency in Ethernet networks, unaffected by any asynchronous Ethernet traffic load. As a result, circuit-switched communication can be emulated on top of a packet-switched network standard such as IEEE 802.3 Ethernet.

In addition to the time-triggered (TT) transfer, the synchronized global time is used to specify temporal characteristics for intervals in which non-time-triggered communication may occur. This allows time-triggered Ethernet device implementations to support communication among applications with different real-time requirements over a single physical network. For example, in a particular time-triggered Ethernet realization, there may be three different traffic classes (see Figure 2): time-triggered, rate-constrained, and best-effort.

Tight latency, jitter, and determinism requirements can necessitate the use of TT transfers. All TT transfers are dispatched at predefined times. In cases where an end system decides not to use its assigned time slot (e.g., if there is no new data to send), the switch recognizes the inactivity of the sender and frees the bandwidth for the other traffic classes. TT transfers best suit communications distributed in real-time systems.

Rate-constrained (RC) traffic establishes periodic communication with maximum bandwidth use to ensure bounded latency in complex networks. Successive transfers of RC frames belonging to the same rate-constrained dataflow are guaranteed to be offset by a minimum duration as specified offline. This is in contrast to TT communication, which in addition to the minimum duration also specifies a maximum duration for this interval. The RC communications paradigm is specified in ARINC 664 standard part 7. A system designer may decide to use RC transfers when determinism and real-time operating requirements are less strict than those that drive the use of TT communication. RC transfers guarantee sufficient bandwidth allocation for each transmission, with defined limits for delays and temporal deviations. In contrast to TT transfers, RC transfers are not dispatched with respect to a system-wide synchronized time base. Hence, different communication controllers may dispatch RC-transferred frames at the same point in time. Consequently, the RC-transferred frames may queue up in the network switches, leading to increased transmission jitter and requiring increased buffer space. Given the known transmission rate of the RC transfers and the network switch controls, it is possible to avoid frame loss by calculating the transmission latency offline.

Best-effort (BE) transfers implement the classic Ethernet approach. There is no guarantee if and when these frames will be transmitted, what delays may occur, or if BE-transferred frames will arrive at the recipient location. BE transfers use the remaining bandwidth of the network and have lower priority than TT and RC transfers.

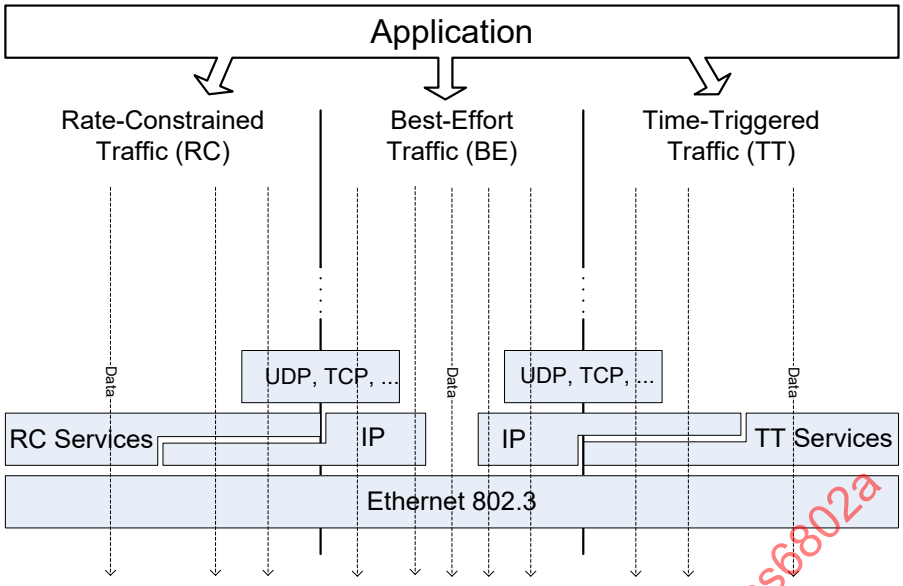


Figure 2 - Interaction of standards and Ethernet protocol layers: applications can use different protocol services and QOS enhancements for communication

As depicted in Figure 2, frames from higher-layer protocols, such as IP or UDP, can easily become TT transfers without modification of the frames' content. TT services are concerned with configured frame transmission timing, not with the content of frames.

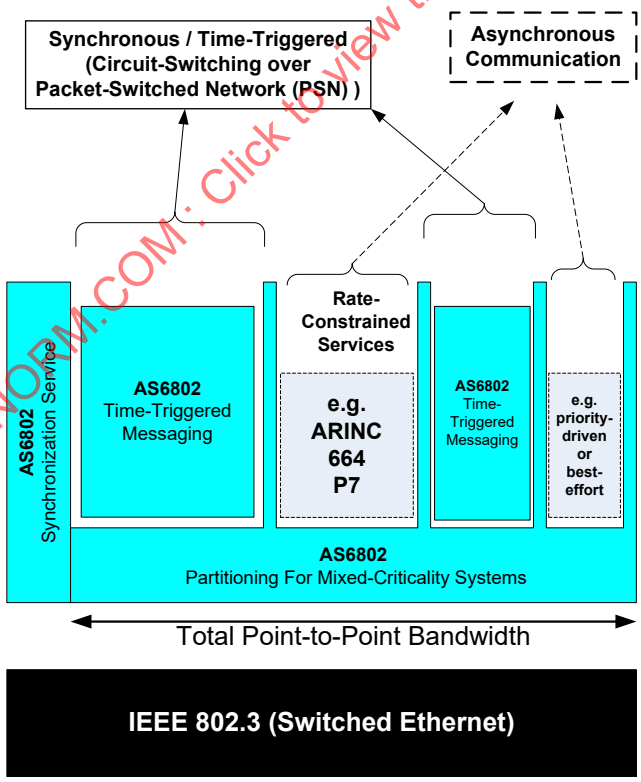


Figure 3 - Example of robust bandwidth partitioning through time-triggered Ethernet services

As depicted in Figure 3, the available bandwidth is shared by different QoS services due to robust bandwidth partitioning. Other QoS protocols can be added to the switch functionality and executed in a portion of bandwidth not used by TT-transferred frames. Time-triggered Ethernet switch implementations contain in a minimum configuration the support for synchronous TT.

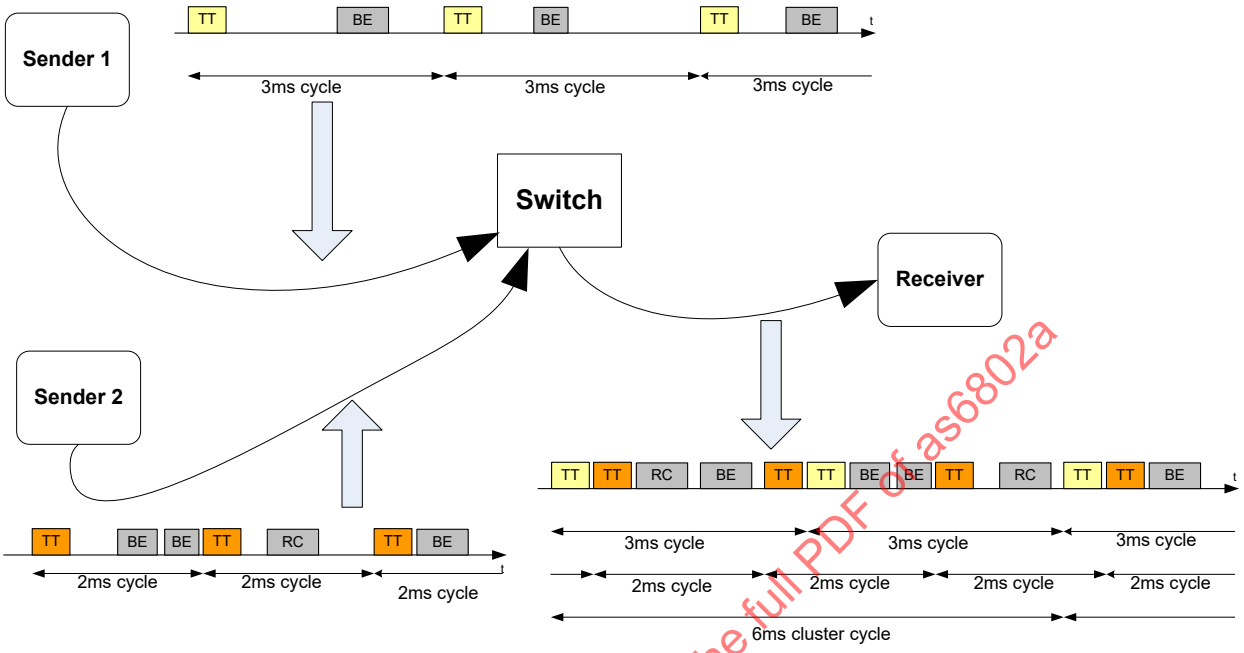


Figure 4 - Example time-triggered Ethernet network

Figure 4 depicts an example of a time-triggered Ethernet network implementation. This implementation consists of two end systems that send frames, a switch that integrates the frames from the two senders, and a receiver that collects the integrated dataflow from the switch. As depicted, Sender 1 sends a frame transferred as TT with a period of 3 ms, as well as frames transferred as BE. Sender 2 transmits a frame transferred as TT with a period of 2 ms, as well as frames transferred as BE, and frames transferred as RC. The resulting integrated dataflow appears in Figure 4, on the right-hand side of the image.

The transmission of a frame from a sender to a receiver typically requires several frame transfers in a switch-based network. For example, given a network consists of end systems all connected to a single switch, each frame transmission requires two frame transfers, a first frame transfer from the sender to the switch and a second frame transfer from the switch to the receiver. In a multi-hop network, the number of frame transfers increases with every hop, i.e., an additional frame transfer is required for each multi-hop connection. The individual frame transfers that make up the frame transmission can be a combination of rate-constrained and time-triggered transfers. For example, a frame is transferred as TT from the sender to a first switch and as RC from the first switch to the second switch and as RC from the second switch to a receiver.

See Appendix B and the following section for specifics regarding the binding of particular frames to traffic classes (e.g., in an ARINC 664-p7, compatible system, the binding to the traffic class is based on the Ethernet MAC destination address).

3.1.1 Time-Triggered Dataflow Specifics

Figure 5 depicts the flow of a TT-transferred frame f_i from a sender to a receiver via an intermediate switch. In order to uniquely locate the frame transfer in the network, the figure shows the frame transfer with the respective dataflow links ([A, D] or [D, E]). In general, we use $[v_x, v_y]$ to denote a link and $f_i^{[v_x, v_y]}$ to denote the frame f_i transferred on this link.

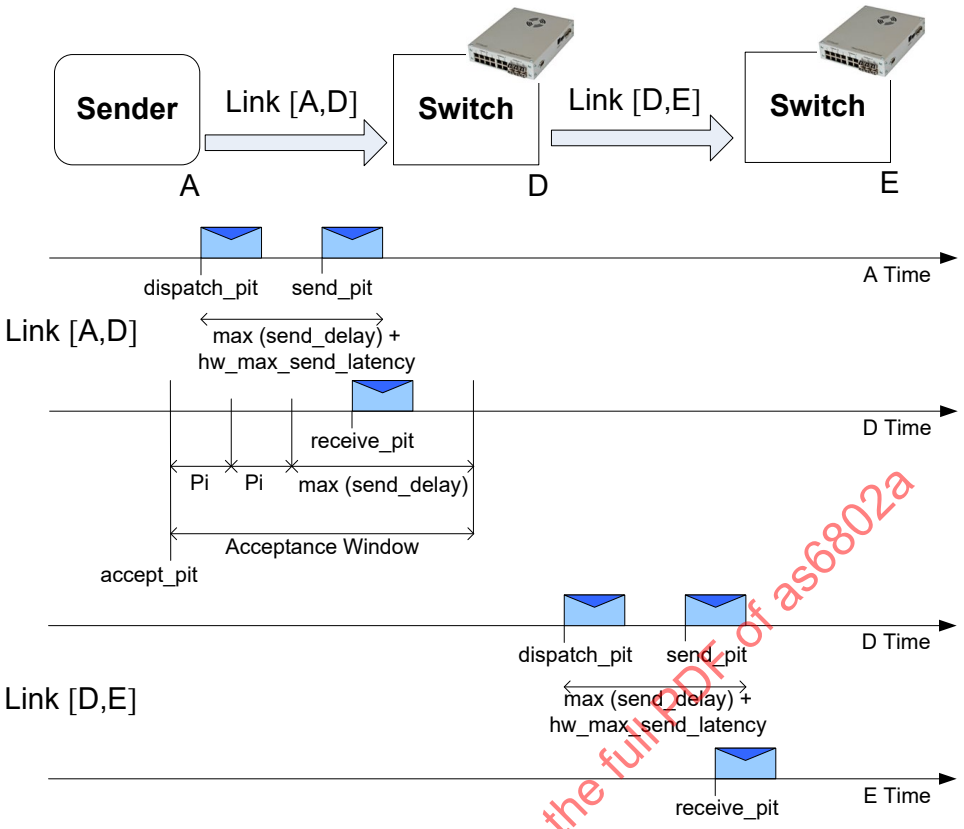


Figure 5 - Relevant points in time in the flow of a frame

This flow identifies the following relevant points in time and how they are related:

a. Dispatch point in time (*dispatch_pit*): The dispatch point in time of a frame f_i transferred as TT is the statically defined periodic interrupt that triggers the transfer of a frame. It is generated for each TT transfer by the scheduler (typically an offline configuration tool).

b. Send point in time (*send_pit*): The send point in time of a frame f_i transferred as TT is the instant when the leading edge of the first bit of the first symbol after the Start of Frame (SOF) delimiter is transmitted on the communication link.

$$\text{send\_pit} \in [\text{dispatch\_pit}, \text{dispatch\_pit} + \max(\text{send\_delay}) + \text{hw\_max\_send\_latency}] \quad \text{(Eq. 1)}$$

where *send_delay* is the delay resulting from a busy communication link when there is already a frame in transit when *dispatch_pit* is triggered. The *max(send_delay)* function describes the offline calculable worst-case delay. *hw_max_send_latency* describes the send latency imposed by the hardware other than the latency resulting from queuing effects.

c. Receive point in time (*receive_pit*): The receive point in time of a frame f_i transferred as TT is the instant of receipt of the first bit of the frame. The temporal difference between *send_pit* and *receive_pit* is the communication link latency *link_latency*.

$$\text{receive\_pit} = \text{send\_pit} + \text{link\_latency} \quad \text{(Eq. 2)}$$

d. Start acceptance window point in time (*accept_pit*): The switches can check the temporal correctness of a frame transferred as TT. A frame transferred as TT will qualify as temporally correct, if the first bit of the frame is received within an acceptance window starting at *accept_pit* and ending at $(\text{accept}_{\text{pit}} + 2 \times \text{Pi} + \max(\text{send}_{\text{delay}}) + \text{hw\_max\_send\_latency})$, where *Pi* is the maximum temporal difference of any two correctly synchronized local clocks in the network (i.e., system timebase precision).

$$\text{accept\_pit} = \text{dispatch\_pit} + \text{link\_latency} - \text{Pi} \quad \text{(Eq. 3)}$$

The dispatch point in time of a frame f_i transferred as TT on a communication link [v_x, v_y], $f_i^{[v_x, v_y]}$.*dispatch_pit*, is identified by the period of *x*, *period*_x and an offset within the period (0 in this example). As depicted in Figure 6 periods can be harmonic or non-harmonic.

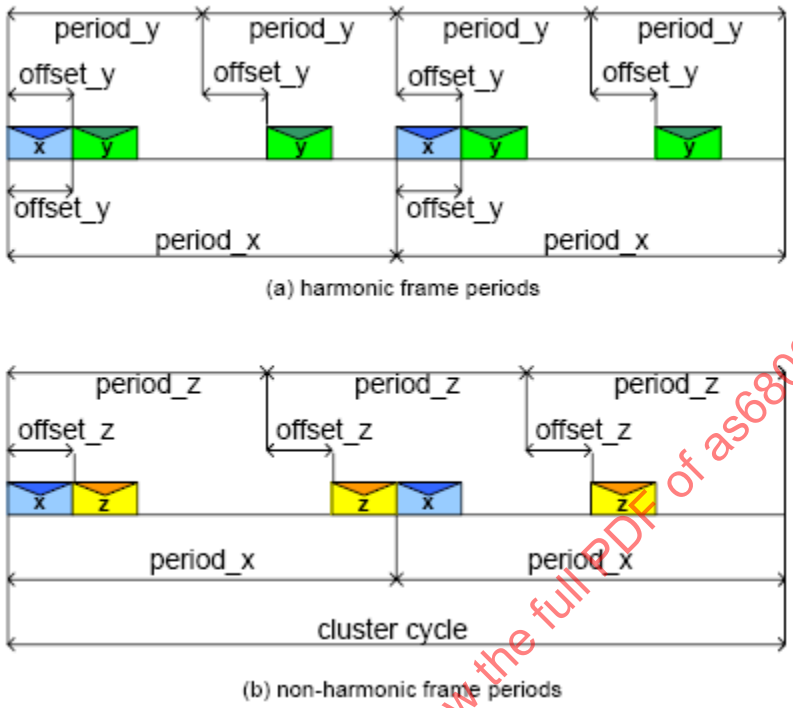


Figure 6 - Characterization of a frame

$f_i^{[v_x, v_y]}$ represents frame f_i transferred as TT on a communication link [v_x, v_y]. It is completely temporally specified by the following equation:

$$f_i^{[v_x, v_y]} = \{f_i.\text{period}, f_i^{[v_x, v_y]}.\text{offset}, f_i.\text{length}\} \quad \text{(Eq. 4)}$$

The frame period of a frame transferred as TT and frame length parameters are offline configured in the system. It is the task of the scheduler to assign values to F^L.*offset* for all frames *F* on all links *L* in the network.

A frame transferred as TT will be identifiable by a bit pattern inside the Ethernet frame.

NOTE: Typically some of the fields in the header of an Ethernet frame would be used as the bit pattern to identify a frame as TT-transferred frame. For example in a realization with ARINC 664-p7, the traffic class is encoded in the Ethernet MAC Destination address. Another implementation may use the EtherType field of the Ethernet header as bit pattern. IEEE802 Ethernet standard specifies the EtherType field as part of the header of an Ethernet frame.

When the EtherType field is used to encode the traffic class, EtherType has the value 0x88d7 hex.

NOTE: EtherType 0x88d7 hex prevents use of higher layer protocols such as UDP and TCP.

3.1.2 Rate-Constrained Dataflow Specifics

A frame transferred as RC is specified by its maximum rate and its length.

$$f_i = \{f_i.\text{rate}, f_i.\text{length}\} \quad \text{(Eq. 5)}$$

A non-faulty sender will respect the rate of frame transmission for a frame transferred as RC. A faulty sender may aim to send frames more frequently than allowed. Hence, switches for rate-constrained traffic implement a traffic policing function known as leaky bucket or token bucket algorithm (see Figure 7).

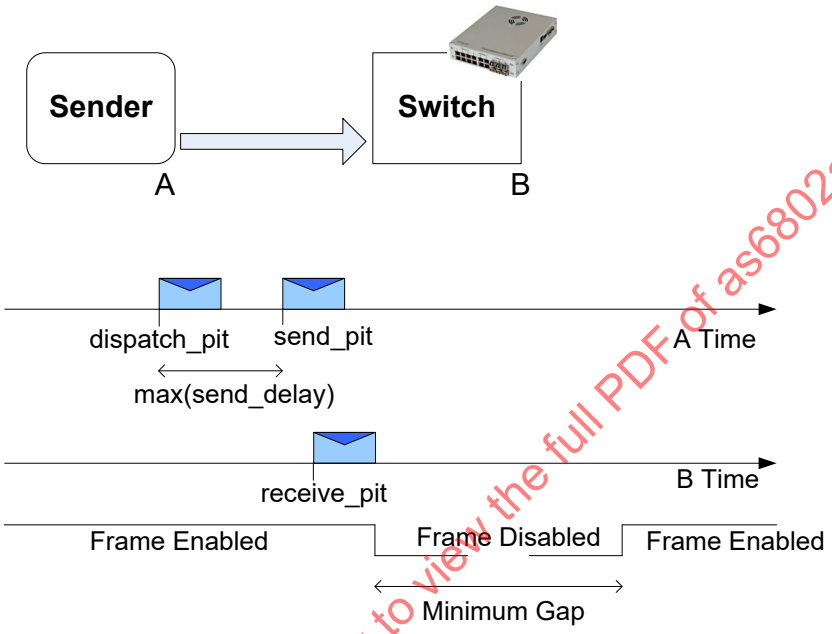


Figure 7 - Leaky bucket algorithm

The leaky bucket algorithm measures the time between two frame receptions. If this time is shorter than a specified minimum gap, the frame is dropped.

3.2 Transparent Synchronization

Time-triggered Ethernet is a transparent synchronization protocol, which means that it is able to coexist with other traffic on the same physical communication network. Examples of traffic that would coexist with the synchronization protocol include standard Ethernet or rate-constrained services such as ARINC 664-p7.

In time-triggered Ethernet, the periodic exchange of synchronization messages (i.e., PCFs) establishes and maintains the global synchronization. For fault-tolerance reasons, a multitude of devices can be configured to generate PCFs. The devices generating the PCFs may distribute these PCFs with a large number of intermediate devices between them (e.g., switches).

For example, in an Ethernet network that consists of ten switches that are connected in sequence, the components that generate the PCFs may be located ten hops away from each other. In standard Ethernet networks, the transmission latency and transmission jitter are directly proportional to the number of hops between any two senders. Therefore, the receive order of these PCFs is not necessarily the dispatch order of these messages. For instance, if end system A shares the same switch as end system B, it will receive generated PCFs from end system B prior to those sent from end system C (located three hops away), although end system C sends the PCFs earlier than end system B. Close proximity of end systems, however, does not guarantee earliest PCF receipt. This is due, in part, to the unpredictable switch buffer allocation at runtime.

Time-triggered Ethernet defines basic functionality that allows for transparent integration of TT services on top of message-based communication infrastructures, such as a standard Ethernet. Time-triggered Ethernet defines a novel application of the transparent clock mechanism, which enables the concept of the permanence point in time, making it possible to reestablish the dispatch order and relative timing of messages as follows:

a. Application of transparent clock mechanism: All devices in the distributed computer network that impose a delay on the transmission, reception, or relay of a PCF add this delay to a dedicated field in the PCFs.

b. Calculation of the permanence point in time: The application of the transparent clock mechanism makes it possible to reestablish PCF order and relative timing with high accuracy.

    1. Calculate the worst case delay offline.

    2. Upon reception of the PCF, delay each PCF for worst case delay minus dynamic delay, where the dynamic delay is added to the PCF as it flows through the communication channel. The point in time at the worst case delay minus dynamic delay, after the reception point in time, is the permanence point in time.

The message dispatch order is extremely important for fault-tolerant algorithms, particularly fault-tolerant synchronization algorithms. Any fault-masking synchronization protocol that ensures synchronization of local clocks in a distributed computer network requires a reestablished PCF dispatch order.

In addition to reestablishment of the dispatch order of PCFs, the permanence point in time is also used for remote clock reading during synchronized operation, as depicted in Figure 8. In this example, the clock of the generator of a PCF is slower than the clock of the consumer of the PCF. At the dispatch point in time of the PCF, the local clock in the consumer is already 5 µs ahead of the generator. This example assumes the following static configuration:

a. A precision (worst-case offset between any two correct local clocks in the system) of 10 µs.

b. A scheduled point in time, given by the worst-case transmission delay (*max_transmission_delay*), of 50 µs.

c. A scheduled window, which is statically defined as a ± precision window around the scheduled point in time.

Next, instead of directly calculating the clock difference, the following dynamic computations will be executed upon receipt of the PCF:

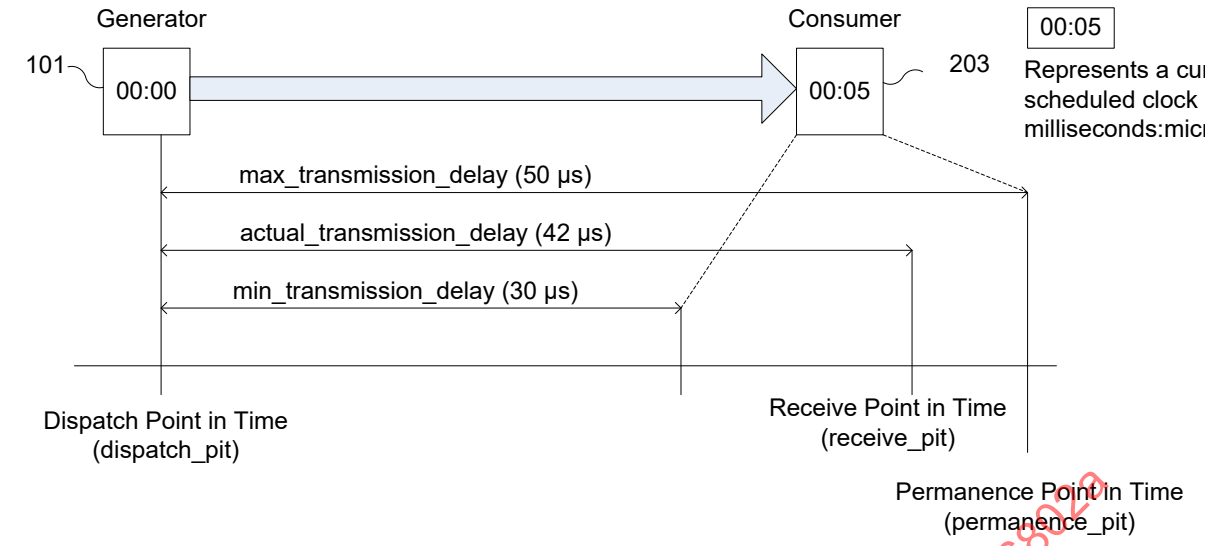
a. The consumer calculates the permanence delay (8 µs).

b. At 8 µs after the reception of the PCF, the permanence point in time of the respective message is generated.

c. The difference between this permanence point in time and the scheduled point in time equals the clock difference value.

The benefit of this approach is the independent size of the scheduled window from the transmission delays in the network, which only depends on the precision in the system. Also, in the case of multiple PCF generators, the permanence function allows for clock difference values, which measure in increasing order, simplifying further fault-tolerant calculations.

The permanence function constitutes an algorithmic interface to the clock synchronization algorithm. The single network-dependent configuration parameter is the *max_transmission_delay*, which defines the position of the receive window in time.



Example

Static Configuration	Precision	10µs
	Scheduled Receive Point in Time (scheduled_receive_pit)	00:00 + max_transmission_delay (50 µs) = 00:50
	Scheduled Receive Window	00:50 (+/-) 10µs = [00:40 ; 00:60]
Dynamic Computation	Permanence Delay	max_transmission_delay (50 µs) – actual_transmission_delay (42 µs) = 8µs
	Permanence Point in Time (permanence_pit)	receive_pit (00:05 + 42 µs) + 8 µs = 00:55
	Clock Difference	**scheduled_receive_pit (00:50) – permanence_pit (00:55) = -5µs**

Figure 8 - Transparent clock: remote clock time reading

3.3 Scalable Fault Tolerance

Time-triggered Ethernet specifies a synchronization protocol, which uses active redundancy to tolerate a broad class of failure modes, discussed in the following subsections. The failure behavior at one level of a system becomes a failure mode at the next higher level of a hierarchical system.

3.3.1 Failure Modes

This section describes well-known failure modes. With respect to the communication network, the discussed failure modes (see Figure 9) relate to the behavior of the devices (e.g., end systems and switches) and their network interfaces.

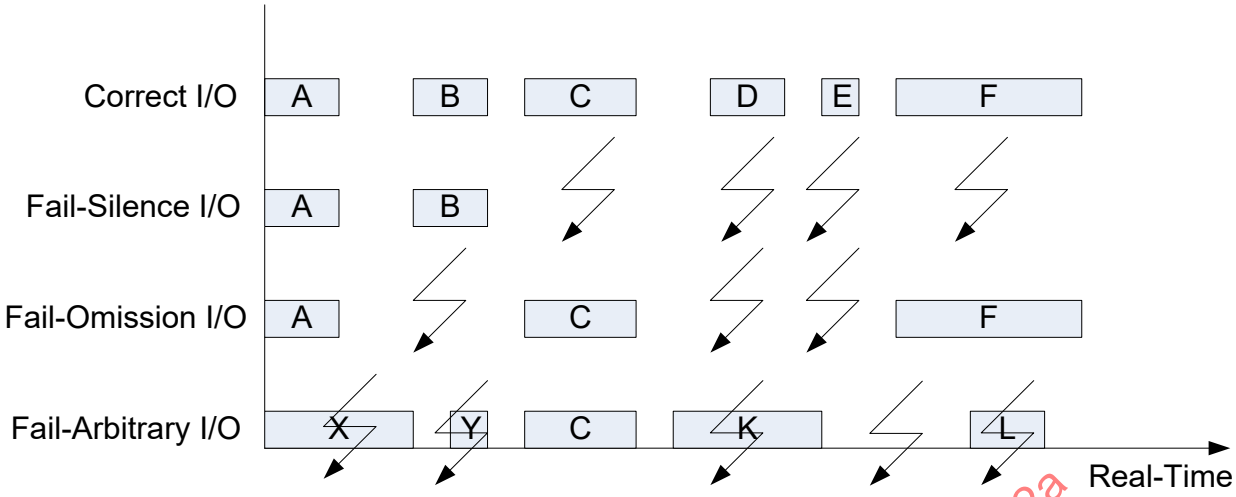


Figure 9 - Failure modes overview

a. Fail-silence failure mode: A device fails silently, when it simply stops producing output.

b. Fail-omission failure mode: In contrast to the fail-silence failure mode, a device that fails in fail-omission failure mode will fail to send and/or receive an arbitrary number of frames.

c. Fail-inconsistent failure mode: This failure mode impacts transmissions from a single sender to multiple receivers. Fail-inconsistent means the sender fails in such a way that different receivers perceive the message differently. For example, one receiver may qualify the message as correct while another one qualifies the message as incorrect

d. Fail-inconsistent-omission failure mode: This mode is a combination of the fail-inconsistent and fail-omission failure modes. A faulty device can arbitrarily classify a subset of received messages as correct, while classifying the remaining messages (potentially correct) as incorrect. The faulty device can also arbitrarily select message communication port destinations.

e. Fail-arbitrary failure mode: A device that fails arbitrarily can generate messages with random contents at arbitrary points in time. Furthermore, the fail-arbitrary device may send its messages on an arbitrary selection of outputs.

3.3.2 Failure Hypothesis

Time-triggered Ethernet has been designed to support single-failure and dual-failure hypotheses.

a. Single-failure hypothesis: Under the single-failure hypothesis, time-triggered Ethernet is intended to tolerate either the fail-arbitrary failure of an end system or the fail-inconsistent-omission failure of a switch. The switches in time-triggered Ethernet network can be configured to execute a central bus guardian function. The central bus guardian function ensures that even if a set of end systems becomes arbitrarily faulty, it masks the system-wide impact of these faulty end systems by transforming the fail-arbitrary failure mode into an inconsistent-omission failure mode. The arbitrarily faulty failure mode also includes babbling-idiot behavior. Time-triggered Ethernet switches establish fault-containment boundaries.

   In order to satisfy the single-failure hypothesis, the network needs to contains at least four end systems acting as synchronization masters and at least two switches operating on independent channels and acting as compression masters.

b. Dual-failure hypothesis: Under the dual-failure hypothesis, time-triggered Ethernet networks are intended to tolerate two fail-inconsistent-omission faulty devices. These devices may be two end systems, two switches, or an end system and a switch. The last failure scenario (i.e., end system and switch failure) means that time-triggered Ethernet network tolerates an inconsistent communication path between end systems. This failure mode is one of the most difficult to overcome.

   In order to satisfy the dual-failure hypothesis, the network needs to contain at least five end systems acting as synchronization masters and at least three switches operating on independent channels and acting as compression masters.

   In both the single-failure and the dual-failure case any number of end systems and/or switches may be configured as synchronization clients. Note, that all switches in the network, no matter whether they are configured as synchronization client or compression master, must adhere to the fail-inconsistent-omission failure mode. Note further, that synchronization masters need access to all independent channels while a compression master needs access to a single independent channel only. Therefore, synchronization master functionality is usually implemented in end systems while compression master functionality is usually implemented in switches. While this is not compulsory, this standard uses this association. Also, all switches being part of the same independent channel are perceived as a single fault containment unit (i.e., any number of switches of the same independent channel may become faulty, which will account for a single failure). Communication links are perceived as part of the transmitting device and need to adhere to the same failure mode as the respective device (e.g., a link connecting a switch and an end system in case of the single-failure hypothesis will be tolerated to be fail-arbitrary faulty with respect to frames sent by the end system to the switch; it will be tolerated to be fail-inconsistent-omission faulty with respect to frames send by the switch to the end system).

c. Arbitrary disturbances: Time-triggered Ethernet networks are intended to tolerate transient disturbances, even in the presence of permanent failures. Under both single- and dual-failure hypothesis, time-triggered Ethernet provides self-stabilization properties. Self-stabilization means that synchronization can reestablish itself, even after a transient upset in a multitude of devices in the distributed computer network. Time-triggered Ethernet networks stabilize from an arbitrary to a synchronized system state. This is of particular importance in synchronous systems to solve the “clique problem.” Cliques are formations of subsets of time-triggered Ethernet devices, such that the devices are synchronized within the subset, but not over subset boundaries. Each such subset is called a clique.

The single-failure hypothesis, dual-failure hypothesis, and tolerance against arbitrary disturbances define the basic fault-tolerance concept in a time-triggered Ethernet network. In addition to this basic fault-tolerance concept, the system-of-system features as discussed in the next section and in Section 10 can contribute to an even higher level of fault tolerance. This system-of-systems approach supports system-level fault-tolerance: the overall failure or power-down of a time-triggered Ethernet-based system can be mitigated by other redundant time-triggered Ethernet-based systems. When different time-triggered Ethernet-based systems are up and running, time-triggered Ethernet also specifies an algorithm for system-of-system synchronization.

The design of fault-tolerant, real-time distributed algorithms is notoriously difficult and prone to error. The combination of faults, interleaving concurrent events, and variations in real-time durations leads to a case or system state explosion that adversely challenges designers. Consequently, the development of time-triggered Ethernet fault-tolerant algorithms included formal methods for proving the correctness of the algorithms.

3.4 System-of-Systems Support

Time-triggered Ethernet provides native support to system-of-systems communication by implementing synchronization priorities and synchronization domains.

3.4.1 Synchronization Domains

Synchronization domains within a time-triggered Ethernet network specify independent time-triggered Ethernet systems with respect to their synchronization. Hence, two components belonging to different synchronization domains in one time-triggered Ethernet network will never synchronize their local clocks to each other. Communication between two components of different synchronization domains is only possible with non-time-triggered traffic classes; e.g., rate-constrained or best-effort traffic. Still, a frame that is communicated within a synchronization domain as time-triggered may be forwarded as non-time-triggered when it crosses a synchronization domain boundary.

3.4.2 Synchronization Priorities

There may be several synchronization priorities within a synchronization domain, and each time-triggered Ethernet device belongs to exactly one synchronization priority (see Section 10).

3.5 Normative Description

3.5.1 Dataflow Requirements

A time-triggered Ethernet device shall provide the following traffic classes, at a minimum:

a. Protocol control frames.

b. Time-triggered traffic.

A time-triggered Ethernet device should assign the traffic class priorities in the following order (from high to low):

a. Protocol control frames.

b. Time-triggered traffic.

c. Rate-constrained traffic, if implemented.

d. Best-effort traffic, if implemented.

NOTE: More priorities can be implemented as necessary. The use of traffic classes with priorities higher than that of the protocol control frame can have a significant impact on the delay of the protocol control frames in the system. See Section 11 for calculation of the bounds.

The worst-case queueing delay as introduced by dataflow of higher priority than PCFs shall be bounded.

NOTE: These bounds cover the worst-case scenario. Further guidance for calculating these bounds can be found in Section 11 (shuffle delay, maximum shuffle delay).

A time-triggered Ethernet device shall implement a non-preemptive dataflow integration algorithm.

NOTE: A time-triggered Ethernet device may also implement a preemptive dataflow integration algorithm. A preemptive dataflow integration algorithm has side effects to be considered. For example, when a frame with high priority is to be transmitted on a given communication link, a preemptive dataflow algorithm will truncate the ongoing transmission of a frame with low priority. As a result, the receiver has to anticipate the reception of truncated frames. It is possible that the truncation of a frame results in a valid Ethernet frame whose CRC matches its contents.

A time-triggered Ethernet device that implements both preemptive and non-preemptive integration algorithms shall provide a configurable parameter, specifying which algorithm the component will use during operation.

Mechanisms to ensure partitioned access to shared resources (e.g., frame memory) shall be implemented.

NOTE: Guidelines for implementation of these partitioning mechanisms are out of scope of this document and are defined for example in ARINC 653.

3.5.2 Failure Hypothesis Requirements

In a time-triggered Ethernet cluster configured to tolerate a single faulty component, either a synchronization master or a compression master, documentation shall be provided that argues that the failure behavior of all compression masters and all switches is fail-silence or fail-inconsistent-omission.

NOTE: One way to achieve an inconsistent-omission failure mode is a high-integrity design; see Appendix A.

In a time-triggered Ethernet cluster configured to tolerate a single faulty component, either a synchronization master or a compression master, documentation shall be provided that the failure behavior of all synchronization masters is fail-silence, fail-inconsistent-omission, or fail-arbitrary.

In a time-triggered Ethernet cluster configured to tolerate multiple faulty components, documentation shall be provided that the failure behavior of any compression masters, synchronization masters, and switches is fail-silence or fail-inconsistent-omission.

NOTE: One way to satisfy an inconsistent-omission failure mode is to use a high-integrity design; see Appendix A.

NOTE: The inconsistent-omission failure behavior is NOT satisfied by *a posteriori* notification of a faulty transmission. All Ethernet frames that are output from a high-integrity time-triggered Ethernet device are either correct or detectably faulty within the receiving time-triggered Ethernet device.

In a time-triggered Ethernet cluster configured to tolerate a single faulty component, either a synchronization master or a compression master, the synchronization masters shall use the threshold values provided in Table 6, the compression masters shall use the protocol state machine of 9.5 and the threshold values provided in Table 8, and the synchronization clients shall use the threshold values provided in Table 10.

In a time-triggered Ethernet cluster configured to tolerate two faulty components, either two synchronization masters or two compression masters or one synchronization master and one compression master, the synchronization masters shall use the threshold values provided in Table 5, the compression masters shall use the protocol state machine of 9.4 and the threshold values provided in Table 7, and the synchronization clients shall use the threshold values provided in Table 9.

A time-triggered Ethernet device that is able to relay time-triggered traffic shall be configurable to accept a time-triggered-transferred frame only when the transfer is received within a defined acceptance window.

NOTE: This mechanism is depicted in Figure 5. The acceptance window is opened and closed at a point in time derived from the synchronized timebase. The length of this acceptance window is configurable.

NOTE: In order to prevent a faulty time-triggered Ethernet device from sourcing more TT-transferred frames or sourcing a TT-transferred frame at a faulty point in time, the time-triggered Ethernet device relaying TT-transferred frames should always execute this check.

NOTE: TT-transferred frames that are checked, and which fail this check, will not be relayed.

There shall be at least one configurable parameter for the acceptance window length.

NOTE: If more than one acceptance window length parameter is realized, the configuration of which acceptance window to apply to which TT-transferred frame may be derived from the synchronized timebase, a specified combination of bits in the Ethernet frame, or both.

A time-triggered Ethernet device that is able to relay RC-transferred frames shall be configurable to apply the leaky bucket algorithm as defined in 3.1.2, or a compatible algorithm, on a per frame basis.

NOTE: This leaky bucket algorithm is compatible with the algorithm specified in the ARINC 664-p7 standard.

A time-triggered Ethernet device that is able to relay frames shall be configurable to check the identity of the frames.

A time-triggered Ethernet device shall drop an Ethernet frame if the identity check is enabled and the frame has the wrong identity.

NOTE: This mechanism is intended to prevent masquerading failures in the network. For example, protocol control frames sent by a component other than a synchronization master or a compression master are dropped.

NOTE: The bits from the received frame used in this comparison depend on the underlying protocol of time-triggered Ethernet; e.g., in case of a realization of time-triggered Ethernet in conjunction with the ARINC 664-p7 standard, the Ethernet MAC destination address is used.

4. SYNCHRONIZATION PROTOCOL CONTROL FLOW

4.1 Supported Topologies

Time-triggered Ethernet allows for the synchronization of local clocks in a distributed computer network. This includes computer networks that exchange information via messages on communication links between network devices. In standard switched Ethernet, end systems connect to switches via bidirectional communication links. An end system will communicate with a second end system or a group of end systems by sending a message to the switch, which relays the message to the receiving end system or end systems. Also, switches can connect to each other via bidirectional communication links (i.e., a multihop architecture).

See Figure 10 (left image) for an example of multihop topology. The end systems are denoted numerically as 101 to 106; switches are 201 to 203. Communication links and switches form a communication channel between the end systems. The left side of Figure 10 depicts a single communication channel formed by the switches 201 to 203, with communication link 110. The right side of Figure 10 shows an example network with two redundant communication channels.

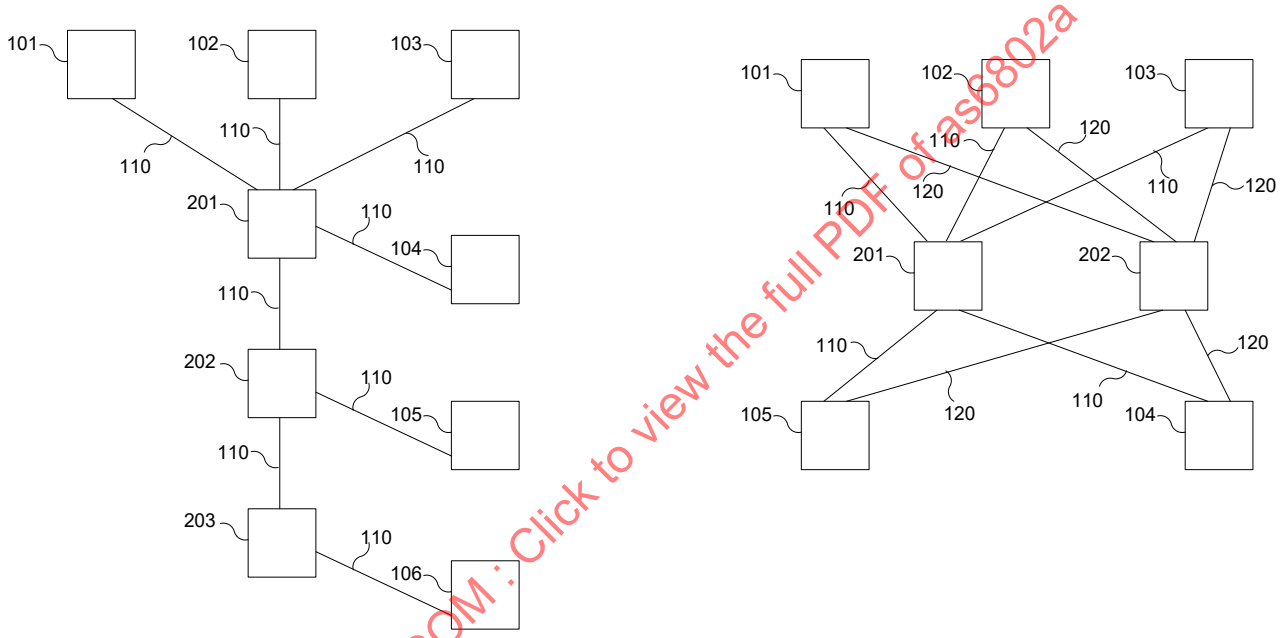


Figure 10 - Time-triggered Ethernet: example networks

End systems can connect directly to each other via bidirectional communication links, making a clear differentiation between end systems and switches difficult in certain configurations. To avoid potential confusion, the term “device” is used to refer to a physical device that acts as either an end system or switch. The functions performed by a device determine whether it is identified as an end system, a switch, or both.

4.2 Fault-Tolerant Synchronization Approach

Time-triggered Ethernet specifies a two-step synchronization approach, as depicted in Figure 11. In the first step, synchronization masters send PCFs to the compression masters. The compression masters then calculate an average value from the relative arrival times of these PCFs, sending out a new PCF in response. This new PCF is also sent to synchronization clients.

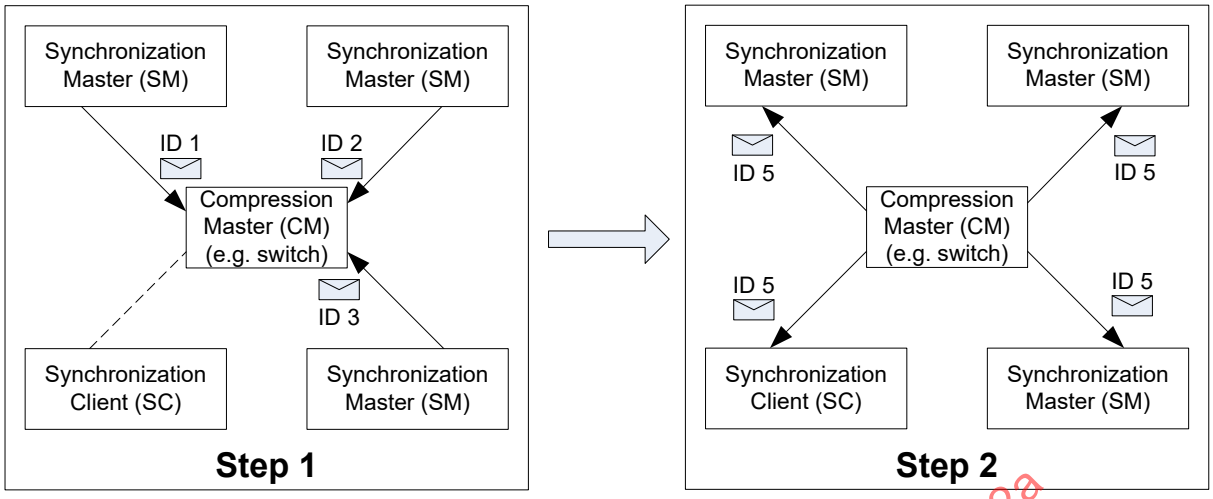


Figure 11 - Time-triggered Ethernet two-step synchronization approach during synchronized operation

The system architecture timing performance requirements determine which devices are configured as synchronization master, synchronization client, and compression master. These requirements are outside the scope of this standard. However, for simplicity, the examples in this time-triggered Ethernet standard configure end systems as synchronization masters and switches as compression masters. Note that this is not a time-triggered Ethernet restriction; system configurations with end systems configured as compression masters and switches as synchronization masters are possible. Switches and end systems not configured either as synchronization master or compression master will be configured as synchronization client.

In the example time-triggered Ethernet network shown in Figure 10 (left image), end systems 101 to 106 are configured as synchronization masters, while only switch 203 is a compression master. All other switches (i.e., 201 and 202) are synchronization clients. These synchronization clients transparently forward PCFs, using the PCFs from switch 203 (i.e., the compression master) for synchronization. See Section 6 for a detailed discussion of the compression master function.

For an example diagram of the PCF's flow during synchronized operation, see Figure 12. The end systems configured as synchronization master dispatch PCFs 301 to 304. The compression masters create a new PCF 380 and send this message back, received by the end systems and other intermediate switches (not depicted). The PCFs sent and compressed during the synchronized operation are known as integration frames. The synchronization masters dispatch these integration frames with a configurable period, called the integration cycle duration. An overall cluster cycle consists of a configurable number of integration cycles. The integration cycles number from 0 to *max_integration_cycle-1*. This allows end systems and switches to integrate not only at the beginning of a cluster cycle, but also at well-defined points during the cluster cycle. The relation of integration cycle to cluster cycle appears in Figure 12 (n represents the *max_integration_cycle-1*).

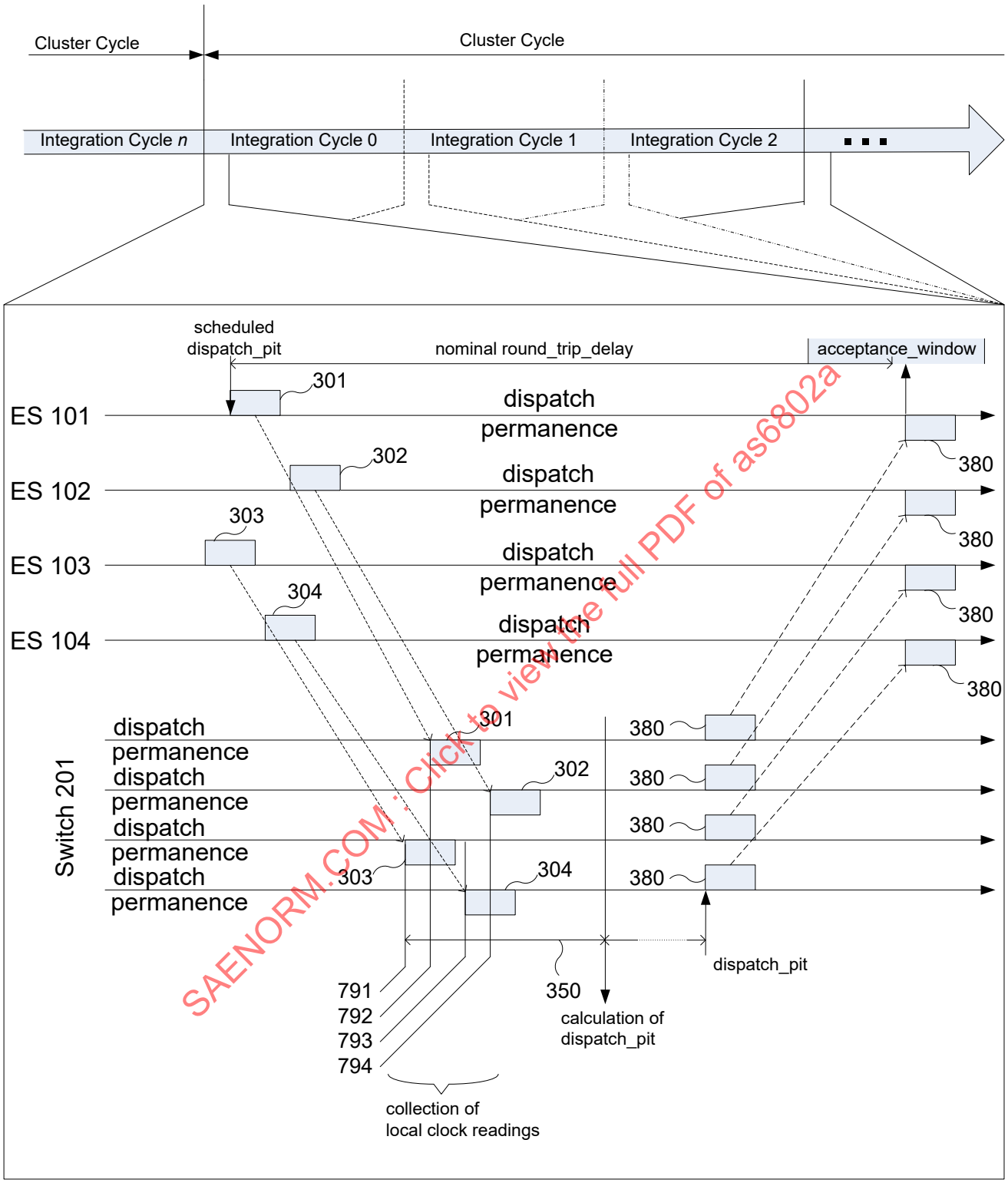


Figure 12 - Example of cluster cycle and integration cycles

During unsynchronized operation (e.g., a coldstart or restart of the time-triggered Ethernet network), the synchronization masters transmit dedicated PCFs that are coldstart and coldstart acknowledge frames. The compression masters collect the coldstart and coldstart acknowledge frames. Based on this input and their current state in the compression master protocol state machine, the compression masters decide whether to send coldstart or coldstart acknowledge frames back to the synchronization masters and synchronization clients.

A compression master will relay the PCFs back to the synchronization masters and clients, either uncompressed or compressed, as described in Table 1.

Table 1 - Compression of PCF frames

	Coldstart Frame	Coldstart Acknowledge Frame	Integration Frame
Single-Failure Hypothesis	Uncompressed	Compressed	Compressed
Dual-Failure Hypothesis	Uncompressed	Uncompressed	Compressed

4.2.1 Scenario 1 - Uncompressed PCF Flow

In the scenario depicted in Figure 13, five synchronization masters (SMs) send their PCFs to the compression masters (CM). Switches that are not compression masters are configured as synchronization clients (SCs) and forward the received PCFs from the synchronization masters towards the compression master. The compression master changes the PCF identifiers (ID), sending them back to the synchronization masters and synchronization clients.

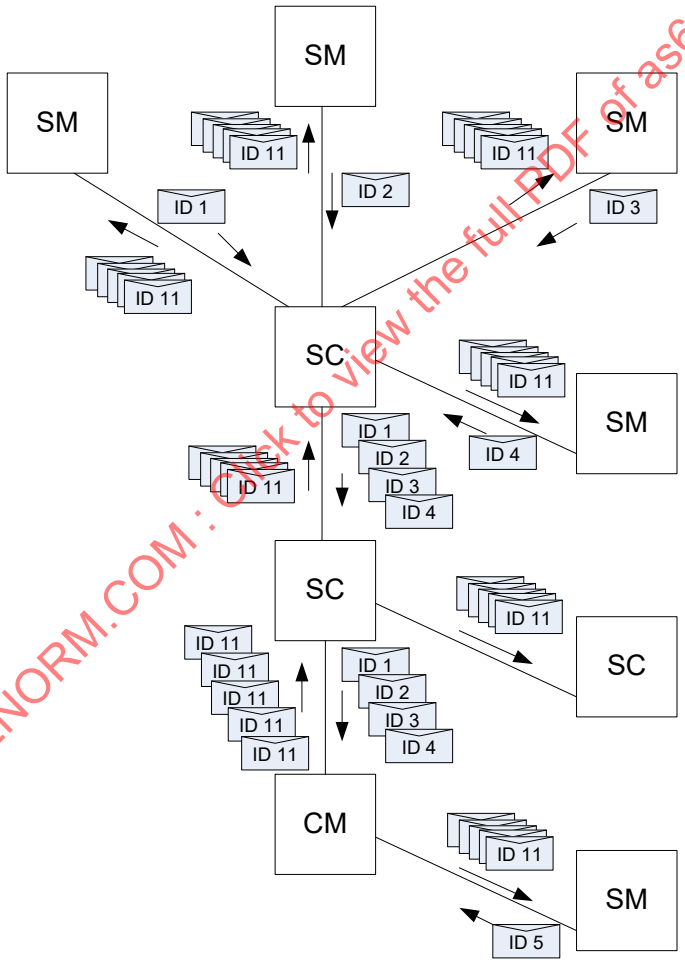


Figure 13 - Uncompressed PCF routing

4.2.2 Scenario 2 - Compressed PCF Flow

In the scenario depicted in Figure 14, five synchronization masters send their PCFs to the compression masters. Switches that are not compression masters are configured as synchronization clients and forward the received PCFs from the synchronization masters towards the compression master. The compression master creates a new PCF from the PCFs it received from the synchronization masters. It uses a previously configured PCF identifier (ID) for the new PCF, and sends it back to the synchronization masters and synchronization clients.

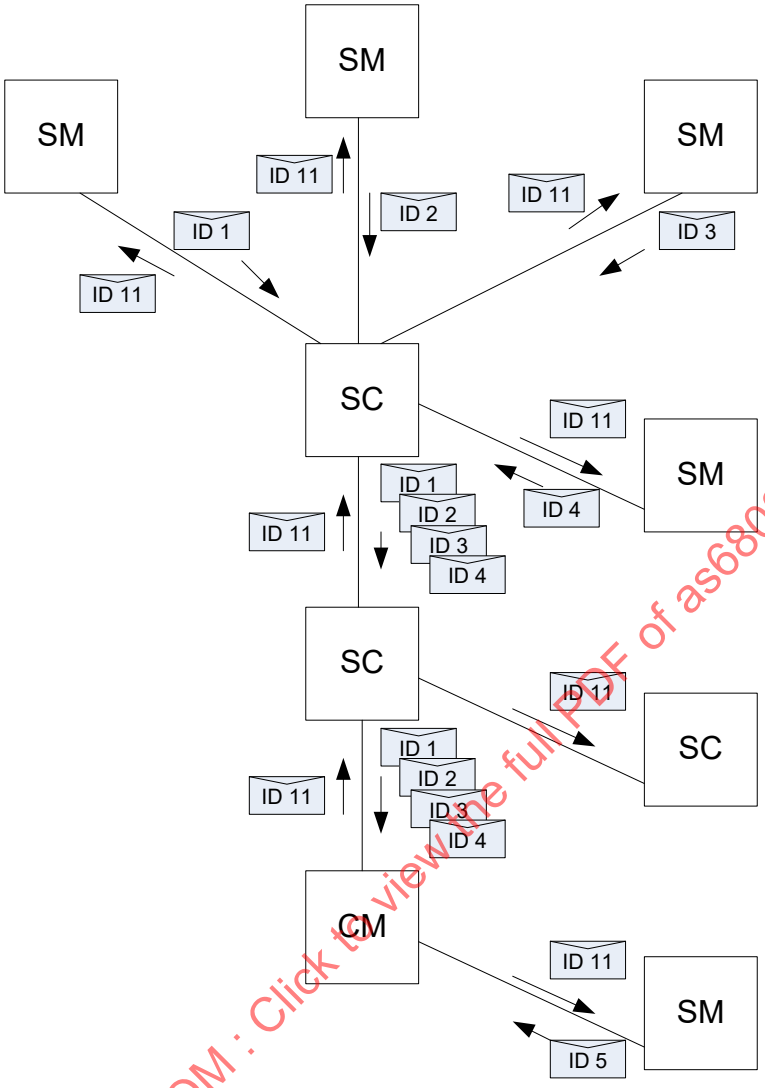


Figure 14 - Compressed PCF routing

4.3 Protocol Control Flow in a Simple Cluster

For a simple protocol control flow, consider the dataflow of PCFs from a synchronization master to one compression master (see Figure 11, Step 1) and from the compression master to a receiving synchronization master or synchronization client (see Figure 11, Step 2). For simplicity of this discussion, we assume that the synchronization master is implemented in end systems, the compression master is implemented in the switch. Synchronization clients may be implemented in both end systems and switches. The real-time axis in Figure 15 depicts events that occur asynchronously to the synchronized local clock in a device. Local clock represents events that are triggered via the synchronized local clock of a device. Characteristic points in time (PIT) appear in this dataflow.

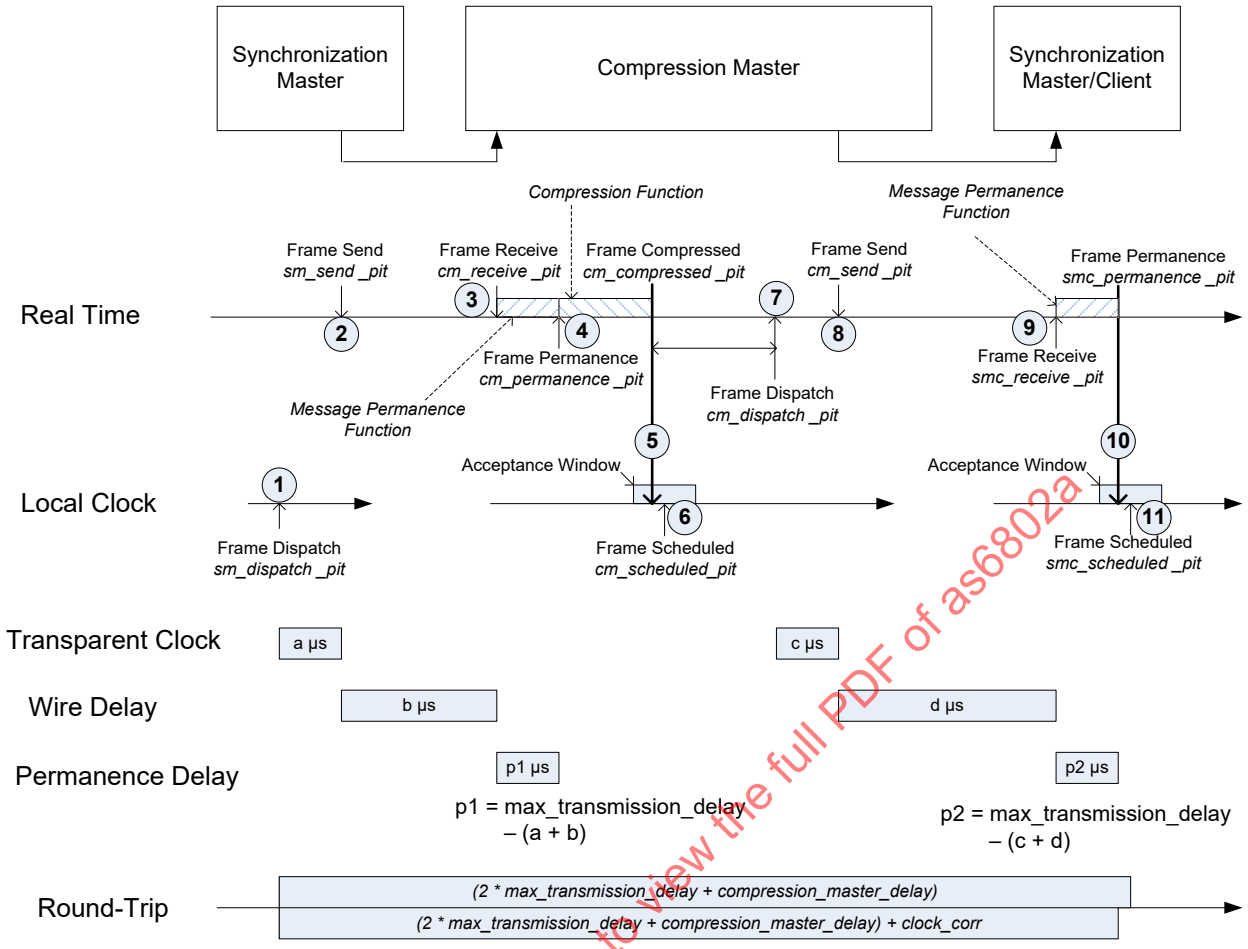


Figure 15 - Protocol control flow: detailed timing

The following list acts as a legend to Figure 15 and contains points in time (PITs) 1 through 11:

PIT 1 Synchronization master dispatch point in time (*sm_dispatch_pit*): The point in time when the internal schedule of the synchronization master triggers the dispatch of a PCF.

PIT 2 Synchronization master send point in time (*sm_send_pit*): The point in time when the leading edge of the first bit of the first symbol, after the start of frame (SOF) delimiter, transmits on the communication link. The synchronization master will add the temporal difference between *sm_dispatch_pit* and *sm_send_pit* to the PCF upon transmission.

PIT 3 Compression master receive point in time (*cm_receive_pit*): The point in time when the leading edge of the first bit of the first symbol after the SOF delimiter is received in a compression master.

PIT 4 Compression master permanence point in time (*cm_permanence_pit*): The point in time when a received PCF becomes permanent. This point in time derives from the maximum possible delay of a PCF minus the current delay of the PCF carried in the Transparent Clock field in the PCF.

PIT 5 Compression master compressed point in time (*cm_compressed_pit*): The point in time when the compression of the PCFs completes.

PIT 6 Compression master scheduled point in time (*cm_scheduled_pit*): The scheduled point in time of PCFs. During synchronized operation, an acceptance window around this point in time classifies whether the *cm_compressed_pit* is in schedule or out of schedule.

PIT 7 Compression master dispatch point in time (*cm_dispatch_pit*): The point in time when the compression master dispatches a frame. This point in time is a configurable offset from the *cm_compressed_pit*. There are two different modes in which a compression master operates: (1) operating with high-integrity synchronization masters, and (2) operating with standard-integrity synchronization masters.

The compression master sets the configuration option (i.e., whether a compression master is operating with high-integrity or standard-integrity synchronization masters); a compression master is not intended to operate with high-integrity and standard-integrity synchronization masters at the same point in time.

a. In configurations where the compression master operates with high-integrity synchronization masters, all compressed integration frames will be sent to the end systems. As a result, integration frames that are faulty or part of a clique formation will not be blocked by the compression master, but will also be relayed for evaluation in the clique detection services of the synchronization masters and synchronization clients.

b. In configurations with standard integrity synchronization masters, the compression masters will only forward compressed integration frames synchronized to the compression master's local clocks. The intended use case protects the system from a faulty synchronization master that sends arbitrary frames at arbitrary points in time.

PIT 8 Compression master send point in time (*cm_send_pit*): The point in time when the compression master transmits the leading edge of the first bit of the first symbol after the SOF delimiter on the communication link.

PIT 9 Synchronization master/synchronization client receive point in time (*smc_receive_pit*): The point in time when a synchronization master or synchronization client receives the leading edge of the first bit of the first symbol after the SOF delimiter.

PIT 10 Synchronization master/synchronization client permanence point in time (*smc_permanence_pit*): The point in time when a received PCF becomes permanent. This point in time derives from the maximum possible delay of a PCF minus the current delay of the PCF, carried in the Transparent Clock field in the PCF.

PIT 11 Synchronization master/synchronization client scheduled point in time (*smc_scheduled_pit*): The scheduled point in time of PCFs. During synchronized operation, an acceptance window around this point in time spans to classify whether the *smc_permanence_pit* is in schedule or out of schedule.

NOTE: Frame transmission is at TP2 and frame reception is at TP3 (or equivalent) as defined in the IEEE 802.3 standard. Further information can be found in the IEEE 802.3 standard.

4.4 Protocol Control Flow in Cascaded Clusters

As defined in Section 1, cascaded cluster configurations are network structures where a communication channel consists of multiple switches, as depicted in the example topology in Figure 10 (left image).

For simplicity, we assume that synchronization masters are implemented in end systems (end systems 101 to 105 in Figure 10 (left image)), while the compression master is implemented in a switch (switch 203 in Figure 10 (left image)). This is the only switch configured as compression master for this sample configuration. All other switches represent synchronization clients (switches 201 and 202). Likewise, all end systems not configured as synchronization masters are configured as synchronization clients (end system 106). The real-time axis in Figure 16 depicts events that occur asynchronously to the synchronized local clock in a device. Local clock represents events that are triggered by the synchronized local clock of a device.

Figure 16 depicts a protocol control flow in cascaded cluster configurations.

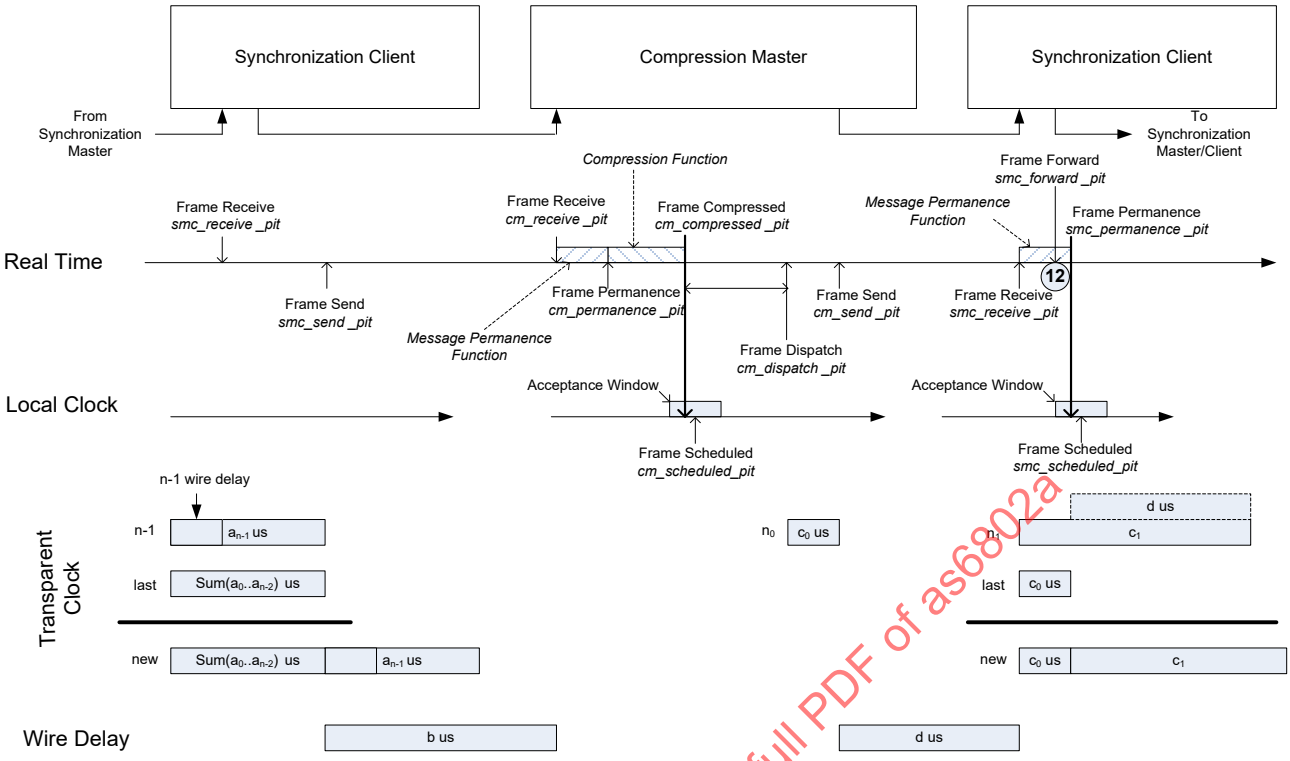


Figure 16 - Protocol control flow in multi-hop: detailed timing

Figure 11 shows the protocol control flow from the synchronization masters to the compression masters (see Figure 11, Step 1) and the protocol control flow from the compression masters to the synchronization masters and synchronization clients (see Figure 11, Step 2).

When a synchronization client receives a PCF from a synchronization master, it forwards this PCF transparently towards the compression master and adds the transmission delay to the PCF upon relay.

Once a PCF arrives at the compression master, the compression master will either compress (in case of an integration frame) or relay (in case of a coldstart frame) the PCF. The configuration of coldstart acknowledge frames allows, but does not necessitate, compression.

When a PCF from the compression master is received in a synchronization master or synchronization client, it is used for synchronization, and transparently forwarded to other attached synchronization clients and/or synchronization masters.

PIT 12 Synchronization master/synchronization client forward point in time (*smc_forward_pit*): The point in time when the first symbol after the SOF delimiter is transmitted from a synchronization master or synchronization client onto the communication link. This point in time relates to a forwarding action of a PCF from a compression master to synchronization masters and synchronization clients (as in contrast to the *smc_send_pit*—PIT 2—which denotes the send point in time of a PCF from a synchronization master or synchronization client to the compression master).

4.5 Protocol Control Flow in Cascaded Clusters with Multiple Compression Masters

Multiple devices within a communication channel can be configured as compression masters. In the case when a compression master receives a PCF from a synchronization master, it will not only use it, as described in 4.2, but also transparently forward the PCF to the other compression masters, if there are other compression masters attached that did not yet receive the PCF from the synchronization master. Compression masters will not use PCFs from other compression masters, rather they transparently forward these frames towards synchronization masters and synchronization clients. Refer back to the example topology from Figure 10 (left image), assuming all switches (201 to 203) are configured as compression masters and all end systems (101 to 106) are configured as synchronization masters. When switch 201 receives a PCF from end system 101, it will use it for the compression function and transparently forward this frame also to switch 202. Likewise, when switch 202 receives this PCF from switch 201, it will use it for compression and transparently forward this PCF to switch 203. Switch 203 will only use this PCF for the compression and will not forward it, as it is the last switch and all other compression masters have already received the PCF.

4.6 Normative Description

A protocol control frame shall be a standard Ethernet frame, whose Ethernet type field is set to 0x891d hex.

A time-triggered Ethernet device shall support protocol control frames being minimum-sized Ethernet frames (with 46 bytes payload).

Time-triggered Ethernet devices may support protocol control frames to have configurable size.

NOTE: Within a synchronization domain all time-triggered Ethernet devices must be configured to use the same size.

A time-triggered Ethernet device shall discard an ingress protocol control frame if it does not have the expected size.

NOTE: Figure 17 shows the layout of the Ethernet payload of a PCF. The payload of a time-triggered Ethernet PCF has a size of 28 bytes.

NOTE: Figure 17 depicts the frame layout as presented to the media independent interface (MII, GMII).

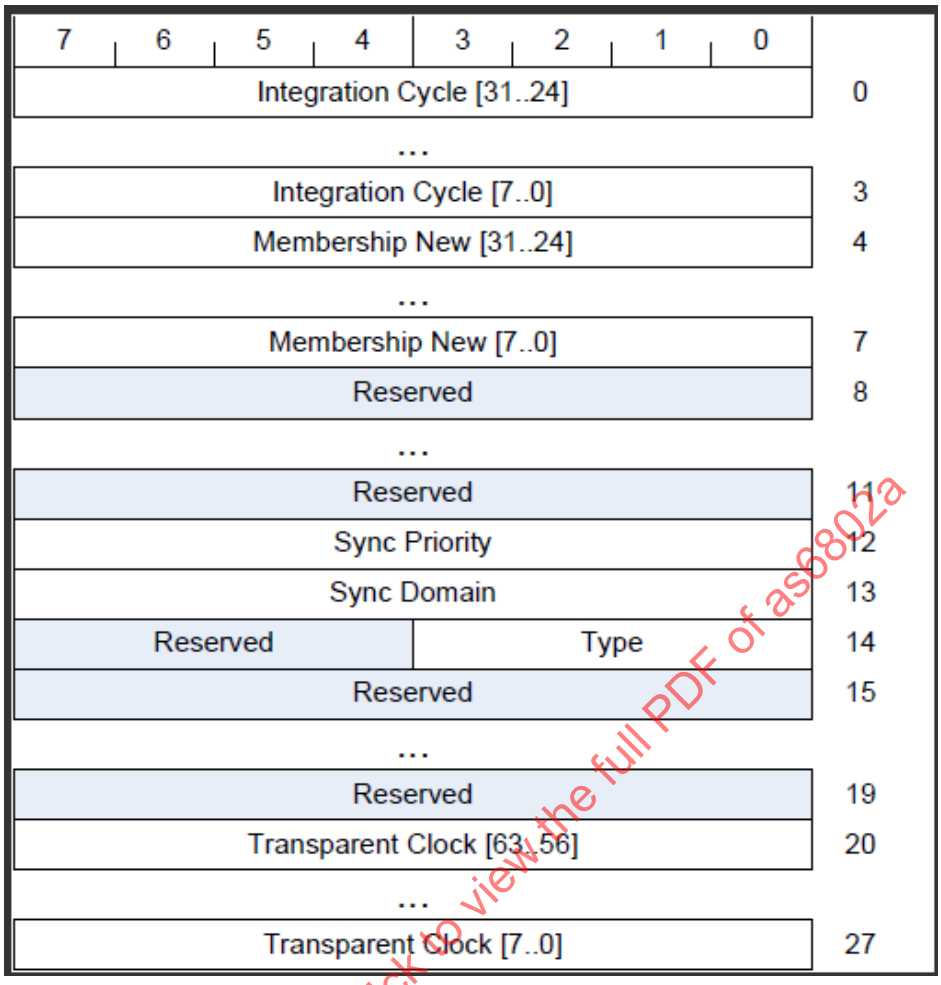


Figure 17 - PCF format

The payload of a PCF shall carry the fields detailed in Table 2 (note that the prefix “pcf” references the respective field in a PCF).

Table 2 - PCF payload

Name	Length	Description
pcf_integration_cycle	32 bits	Integration cycle in which the PCF was sent.
pcf_membership_new	32 bits	Bit-vector with a static configured one-to-one relation from a bit to a synchronization master in the system.
pcf_sync_priority	8 bits	Static configured value in each synchronization master, synchronization client, and compression master. Lowest priority = 0, highest priority = 255.
pcf_sync_domain	8 bits	Static configured value in each synchronization master, synchronization client, and compression master.
pcf_type	4 bits	Frame type of a PCF.
pcf_transparent_clock	64 bits	Stores the accumulated delay of a Protocol Control Frame from the generator of the PCF up to the consumer of the PCF. Time is represented in units of 2^{-16} ns.

Each synchronization master and compression master in the network shall dispatch only PCFs with their pre-defined synchronization priority, sent in the *pcf_sync_priority* field.

The compression master shall only accept PCFs with offline-configured synchronization priority.

Synchronization masters and synchronization clients shall be configurable to also accept PCFs with a priority other than their own.

NOTE: The intended use-case is that in case of compression master powerdowns or failures, the synchronization masters will still have synchronization capability with the remaining network compression masters. The synchronization priority scheme ensures that there are no cyclic dependencies that could potentially lead to oscillating synchronization effects (see Section 10 for priority field details).

The synchronization domain, represented by *pcf_sync_domain*, shall identify sets of devices that can synchronize with each other.

NOTE: Devices with different synchronization domain identifiers cannot synchronize to each other.

Each time-triggered Ethernet device in the system shall update the *pcf_transparent_clock* field to be the accumulated relay latencies of the frame through the network in units of 2^{-16} ns.

NOTE: For example, 10000 hex represents 1 ns and 28000 hex represents 2.5 ns. As for the endianness, the field formats include the most significant octet nearest to the beginning of the protocol data unit followed in order by octets of decreasing significance. In the case where one word holds several fields, the fields are sent in order from left to right.

NOTE: The requirement of 2^{-16} ns only specifies the coding resolution of the time representation. A system may very well only update this field on the order of nanoseconds or microseconds.

A PCF shall be either a coldstart frame (0x04 hex), a coldstart acknowledgement frame (0x08 hex), or an integration frame (0x02 hex), as indicated by the *pcf_type* field.

NOTE: If the *pcf_type* field contains any other value, neither the synchronization algorithm nor the compression algorithm will use the frame.

All AS6802 implementations shall specify the binding of the PCF payload to the addressing scheme of the underlying protocol.

NOTE: The address binding of the PCF to the underlying protocol is specified in the appendices to this document.

5. MESSAGE PERMANENCE FUNCTION

The message permanence function defines how the *smc_permanence_pit* is calculated from the *smc_receive_pit* of a PCF in a synchronization master and synchronization client. Analogously, the message permanence function defines how the *cm_permanence_pit* is calculated from the *cm_receive_pit* of a PCF in a compression master. As the message permanence function is the same for both steps in the synchronization process, this standard uses neutral descriptions of the points in time (e.g., *receive_pit* instead of *smc_receive_pit* or *cm_receive_pit*). Figure 18 shows an example time-triggered Ethernet network. It consists of six end systems (Figure 18, 101 through 106) and three switches (Figure 18, 201 through 203), where end systems connect to switches via bidirectional communication links (Figure 18, 110). Likewise, switches connect to each other via bidirectional communication links (Figure 18, 110). All bidirectional communication links (Figure 18, 110) are standard Ethernet connections. All end systems are configured as synchronization masters; only switch 203 is configured as a compression master. Therefore, switches 201 and 202 are configured as synchronization clients.

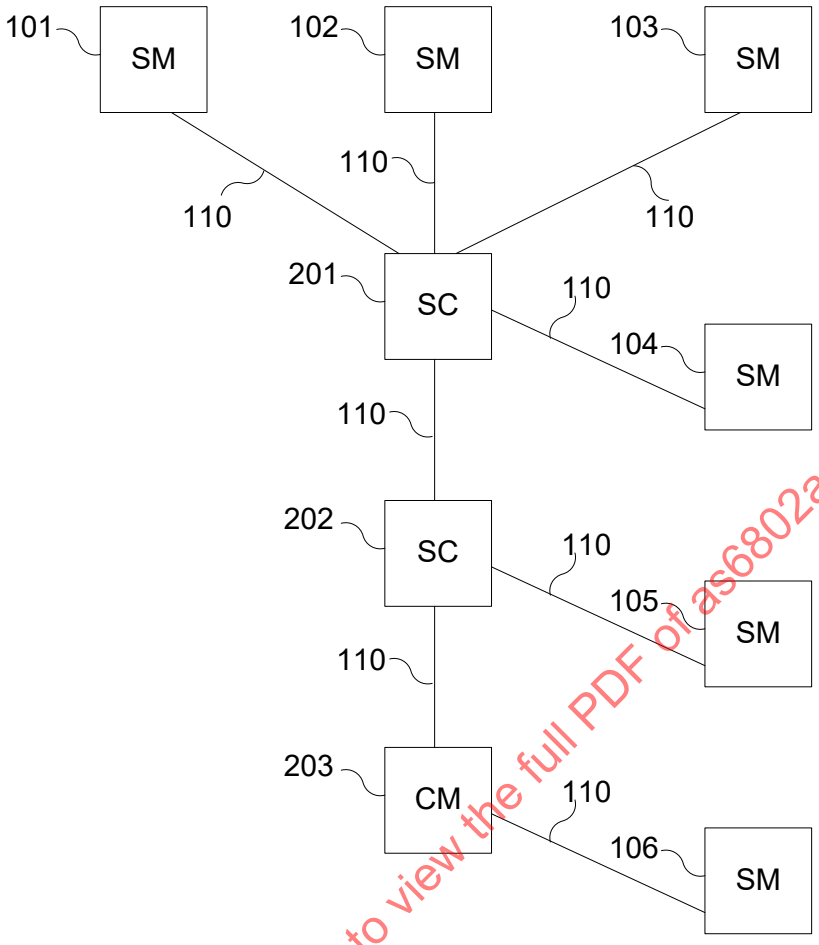


Figure 18 - Time-triggered Ethernet example network

5.1 Transparent Clock Calculation

The following parameters in the distributed computer network may impose dynamic transmission delays on communicated frames, particularly PCFs:

a. Dynamic send delay (*dynamic_send_delay*): The dispatch device (i.e., synchronization master or compression master) of the PCF may delay the transmission of the PCF to the network. The scheduled dispatch point in time of a PCF may occur when there is already another data frame in transmission progress, blocking the communication link 110 to the network. This delays the PCF transmission.

b. Dynamic relay delay (*dynamic_relay_delay*): Relay devices (e.g., synchronization client or compression master) between the dispatching device and the consuming device can introduce transmission delay and transmission jitter. For example, a PCF that is dispatched from end system 101 and consumed by switch 203, will be relayed by intermediate switches 201 and 202. Other devices are also attached to the intermediate switches 201 and 202, but have sending behavior not necessarily fully specified for several reasons. One potential reason is that the attached devices do not take part in the synchronization at all, but still use the same physical communication links 110 and switches 201 to 203 for data frame exchange (e.g., RC or BE traffic). In this case, it may happen that the outgoing link from switch 201 to switch 202 is already busy when a PCF arrives at switch 201. This PCF will be delayed until the communication link is idle again.

c. Dynamic receive delay (*dynamic_receive_delay*): The consuming device (e.g., synchronization master, synchronization client, or compression master) of the PCF may delay the reception of the PCF. Hardware restrictions may necessitate that the reception of PCFs occur sequentially with other internal actions of the consuming device. Compensation for this additional delay involves adding the delay to the PCF again, immediately before handing the PCF to the message permanence function. The *dynamic_receive_delay* is the dynamic part of the temporal distance between the *receive_pit* and the start of the permanence function upon the received PCF.

In addition to the dynamic transmission delays, static transmission delays also affect the PCFs. These static delays are analogously *static_send_delay*, *static_relay_delay*, and *static_receive_delay*, with respect to the devices that impose the static delay.

Each device also maintains a parameter (*wire_delay*) for each communication link it connects to. The *wire_delay* compensates for static wire delays. The *wire_delay* specifies the wire delay between two directly connected devices. The overall wire delay is calculated from the sum of the individual wire delays (*wire_delay*) as the PCF flows from the producer of the PCF to the consumer of a PCF. This enables maximum scalability and requires, in case of changes in wire length, only a device-local reconfiguration activity.

The *pcf_transparent_clock* of a PCF, on its flow from the dispatcher to the consumer via relay devices, is calculated as follows, where the devices are indexed in increasing order with the sequence of PCF flow, from 0 (dispatcher) to n (consumer):

a. Dispatcher device 0:

$$\begin{aligned} \text{pcf\_transparent\_clock}_0 &= \text{send\_pit}_0 - \text{dispatch\_pit}_0 + \text{static\_send\_delay}_0 \\ &= \text{dynamic\_send\_delay}_0 + \text{static\_send\_delay}_0 \end{aligned} \quad \text{(Eq. 6)}$$

b. Relay device i, 0 < i < n:

$$\begin{aligned} \text{pcf\_transparent\_clock}_i &= \text{pcf\_transparent\_clock}_{i-1} + \text{dynamic\_relay\_delay}_i \\ &+ \text{static\_relay\_delay}_i + \text{wire\_delay}_i \end{aligned} \quad \text{(Eq. 7)}$$

c. Consumer device n:

$$\begin{aligned} \text{pcf\_transparent\_clock}_n &= \text{pcf\_transparent\_clock}_{n-1} + \text{dynamic\_receive\_delay}_n \\ &+ \text{static\_receive\_delay}_n + \text{wire\_delay}_n \end{aligned} \quad \text{(Eq. 8)}$$

The *pcf_transparent_clock* field must be explicitly modified by all relaying devices. However, a final consuming device may either explicitly or implicitly modify the *pcf_transparent_clock* because the resulting value is used for internal calculation only (i.e., not forwarded by final device). For further discussion, the term "*pcf_transparent_clock*$_n$" represents the actual transmission duration of a PCF, starting with the dispatch event of a PCF in the dispatcher device and ending with the start of the message permanence function in the consuming device (i.e., the *cm_receive_pit* or *smc_receive_pit*). The *max_transmission_delay* is, therefore, the maximum possible *pcf_transparent_clock*$_n$ in the system, which requires determination offline.

$$\text{max\_transmission\_delay} = \max(\text{pcf\_transparent\_clock}_n) \quad \text{(Eq. 9)}$$

NOTE: The *max_transmission_delay* calculation in the presence of multiple priority PCFs also takes into account the delays imposed by PCFs of equal or higher priority, depending on whether the PCFs of different priorities cross the same device. The *max_transmission_delay* is a single per-synchronization-domain parameter. This means that any two devices that belong to the same synchronization domain will use the same *max_transmission_delay* value.

Figure 19 depicts a sequence of PCF transmissions where the send order of PCFs 301 and 305 is equal to the receive order of the PCFs 301 and 305 in switch 203. The figure presents the progress in real time from left to right.

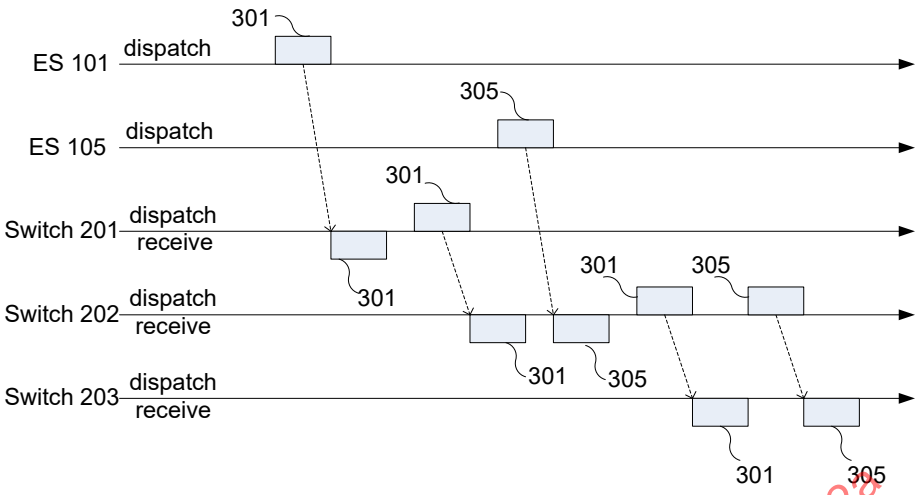


Figure 19 - Dataflow example: equal send and receive orders

The sequence starts with end system 101 dispatching PCF 301 to switch 201.

Switch 201 forwards PCF 301 to switch 202.

Switch 202 receives PCF 301.

End system 105 dispatches PCF 305 to switch 202.

Switch 202 receives PCF 305.

Switch 202 forwards PCF 301 to switch 203.

Switch 202 forwards PCF 305 to switch 203.

Switch 203 receives PCF 301 before PCF 305.

5.2 Permanence Delay Calculation

Figure 20 depicts a sequence of PCF transmissions where the dispatch order of PCFs 302 and 306 is different from the receive order of PCFs 302 and 306 in switch 203. This figure also shows how the permanence function reestablishes temporal order in the consumer. The figure presents the progress in real-time from left to right:

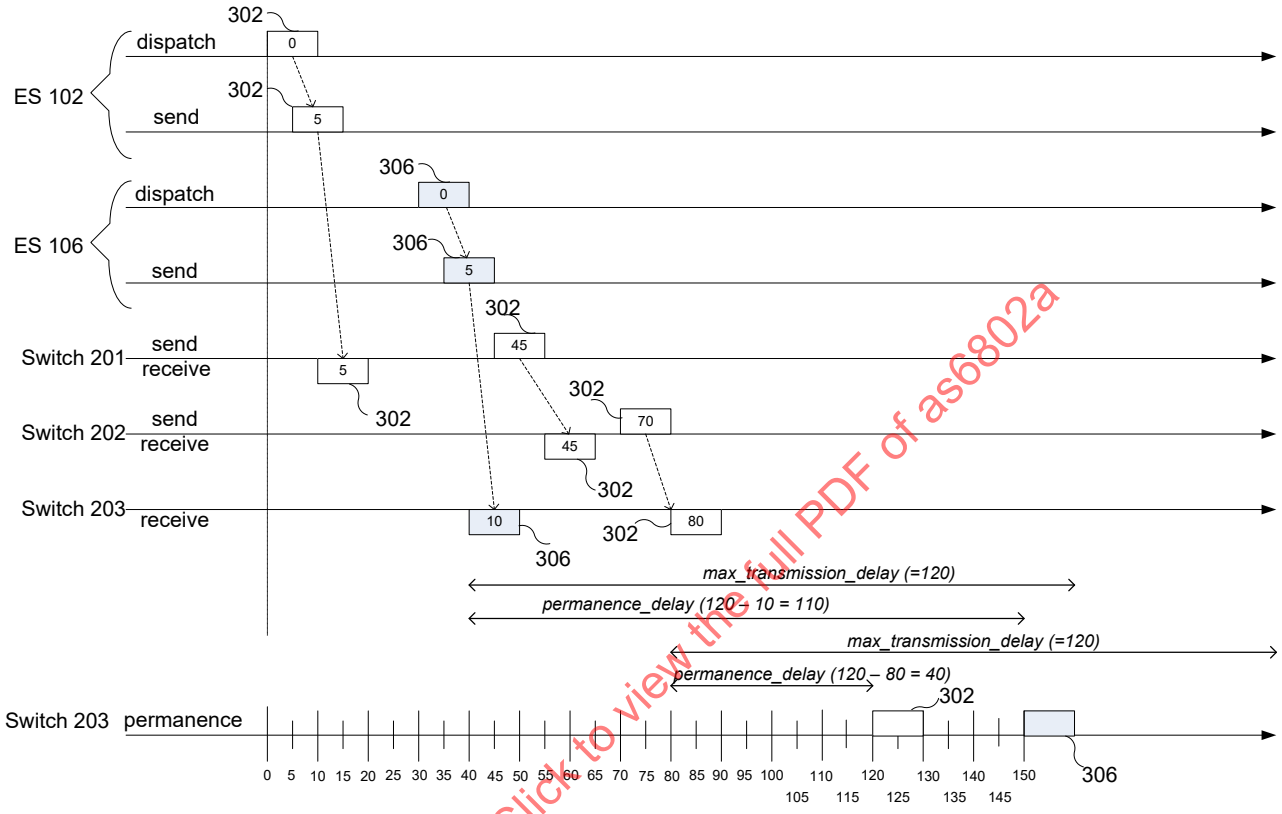


Figure 20 - Dataflow example: different send and receive orders

t = 0 The scenario starts with ES 102 dispatching PCF 302 (at *sm_dispatch_pit*).

t = 5 As there may be other frames already in transit when the dispatch point in time occurs, the actual transmission of PCF 302 on the communication medium may be delayed, in this case, for five time units. The sender writes this delay into PCF 302 and sends it to Switch 201 (at *sm_send_pit*).

t = 10 Switch 201 receives PCF 302 (at *smc_receive_pit*).

t = 30 ES 106 dispatches PCF 306 (at *sm_dispatch_pit*).

t = 35 Similarly to PCF 302, PCF 306 is delayed five time units prior to being sent to Switch 203 (at *sm_dispatch_pit*).

t = 40 Switch 203 receives PCF 306 (*cm_receive_pit*) and the permanence function for PCF 306 starts. According to Equation 10, PCF 306 will be delayed for (120 - 10) = 110 time units.

t = 45 Switch 201 updates the PCF transparent clock field and relays PCF 302 (*smc_send_pit*).

t = 55 Switch 202 receives PCF 302 (*smc_receive_pit*).

t = 70 Switch 202 updates the PCF transparent clock field and relays PCF 302 (*smc_send_pit*).

t = 80 Switch 203 receives PCF 302 (*cm_receive_pit*) and the permanence function for PCF 302 begins. According Equation 10, PCF 302 will be delayed for (120 - 80) = 40 time units.

t = 120 PCF 302 becomes permanent (*cm_permanence_pit*).

t = 150 PCF 306 becomes permanent (*cm_permanence_pit*).

The sequence depicted in Figure 20 shows that switch 203 receives PCF 306 significantly before PCF 302. In order to reestablish the temporal dispatch order of PCF 302 and 306 in switch 203, PCFs 302 and 306 are declared as permanent, after delay upon the reception in switch 203 for a duration that is equal to the permanence delay (*permanence_delay*):

$$\text{permanence\_delay} = \text{max\_transmission\_delay} - \text{pcf\_transparent\_clock}_n \quad \text{(Eq. 10)}$$

and consequently:

$$\text{permanence\_pit} = \text{receive\_pit} + \text{permanence\_delay} \quad \text{(Eq. 11)}$$

The *pcf_transparent_clock* contains the accumulated delay that was written in PCFs 302 and 306 while PCFs 302 and 306 were being transmitted from device to device.

Permanence is associated with a single PCF. The permanence point in time of a PCF is the point in time from which the receiver is guaranteed to have received all prior dispatched PCFs, which have not been dropped.

As a result of the permanence function, PCF 302 and PCF 306 will be delayed and the order and relative timing of PCFs 302 and 306 in switch 203, after permanence function application, is the same order and relative timing as the send order of the PCFs 302 and 306.

This reestablished send order is exact, in that the relative offsets between any two permanent PCFs is equal to the relative offsets between the PCFs sent from the original senders. This is despite the drift of the local oscillators during the permanence function and digitalization errors stemming from crossing clock boundaries.

5.3 Normative Description

Each time-triggered Ethernet device that relays a PCF shall add the delay it imposes on the PCF, plus a configurable wire delay (as defined in Equations 6 to 8) in the PCF field *pcf_transparent_clock*.

When a time-triggered Ethernet device receives a PCF and uses it in the synchronization process, it shall delay the PCF for a duration as described in Equation 10.

NOTE: Intermediate switches located between synchronization masters and the respective compression masters may operate as synchronization clients, forwarding those PCFs sent from the synchronization master to the compression master without making them permanent. However, they will use the PCFs returned from the compression master, making them permanent for their own use and in a concurrent process, forward them towards other synchronization clients and synchronization masters.

Devices that can be configured as time-triggered Ethernet synchronization masters or synchronization clients shall have the ability to execute multiple permanence functions in parallel.

NOTE: An upper bound to the number of permanence functions that may need to be executed in parallel in a synchronization master or client is defined by the number of synchronization masters times the number of compression masters.

NOTE: A time-triggered Ethernet synchronization master makes all PCFs permanent before usage.

Devices that can be configured as compression masters shall have the ability to execute the permanence function for all PCFs from non-faulty synchronization masters in parallel.

NOTE: Devices that can be configured as compression masters have the ability to execute multiple permanence functions in parallel. One correct synchronization master will contribute to at most one permanence function in a compression master at any point in time. Therefore an upper bound to the number of permanence functions that may need to be executed in parallel in a compression master is defined by the number of synchronization masters. To stay within this upper bound, the compression master may drop PCFs received from incorrect synchronization masters, e.g., accepting only one PCF per synchronization master within a given timeframe.

NOTE: A time-triggered Ethernet compression master makes all PCFs permanent before usage.

The wire delay shall be configurable per port in a time-triggered Ethernet device.

6. COMPRESSION FUNCTION

The goal of the compression function is to collect and summarize the content of multiple PCFs into fewer PCFs to reduce the network load and simplify the processing of the resulting information.

The compression function calculates the *cm_compressed_pit* from the *cm_permanence_pits* of PCFs (see Table 1 for the types of PCFs that are compressed). The compression function runs unsynchronized to the synchronized local time in the compression master. Hence, a compression function begins upon the reception of a PCF, rather than upon the synchronized local clock reaching a particular point in time. An outcome of the compression function is the *compressed_pit* value. The point in time when the compression master sends the PCF back to the synchronization masters and clients is based on this *compressed_pit*.

6.1 Compression Function Discussion

Figure 21 gives an overview of the compression function. This function is executed to collect and compress PCFs 301 to 304, which are dispatched from synchronization masters 101 to 104 at simultaneously scheduled dispatch points in time (*sm_dispatch_pit*). Due to differences in the different synchronization masters' local oscillators, the actual dispatch points in time (*sm_dispatch_pit*) differ slightly.

The compression function operates on the permanence points in time (*cm_permanence_pit*). It collects the associated permanence points in time from PCFs 301 to 304, calculating a fault-tolerant average from the relative differences of the permanence points in time (791 to 794) from the earliest permanence point in time (791).

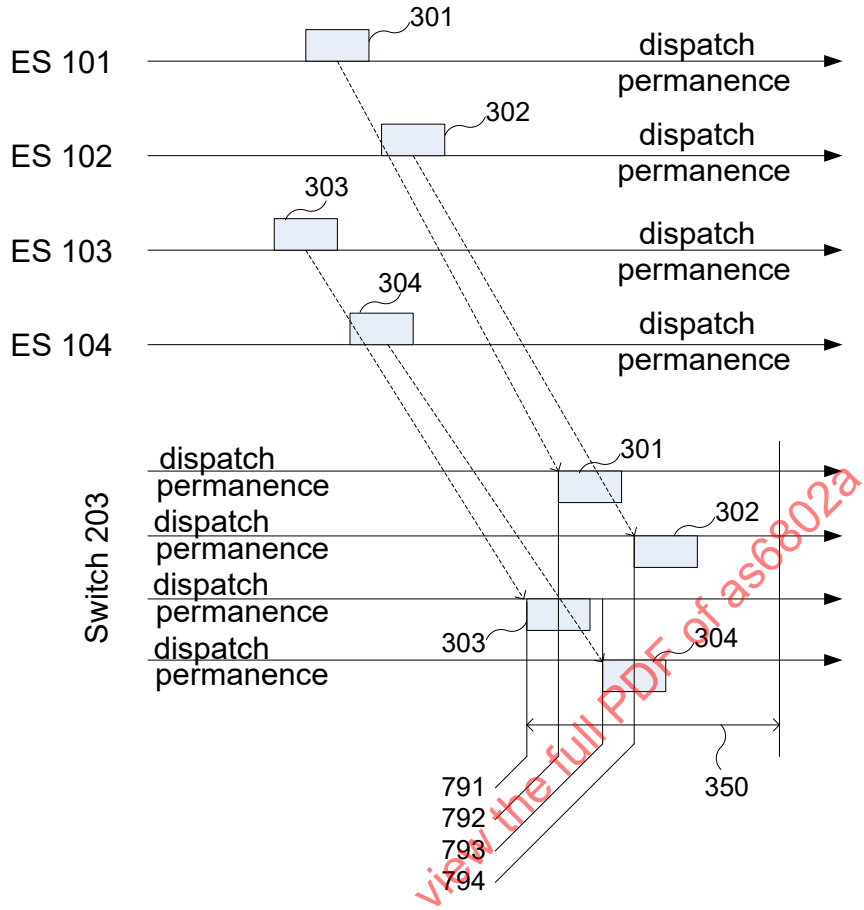


Figure 21 - Overview of the compression function

The message permanence function (discussed in Section 5) ensures compensation for dynamic delays introduced on the transmission paths, such that the *cm_permanence_pits* (791 to 794) reflect the relative differences from the individual actual *sm_dispatch_pits*. The compression function can then operate on the *cm_permanence_pits*.

The compression function is done in a fault-tolerant manner. The *max_observation_window* (350) is a function of the number of PCFs received and their *cm_permanence_pits* (791 to 794). Figure 22 depicts the compression function in detail.

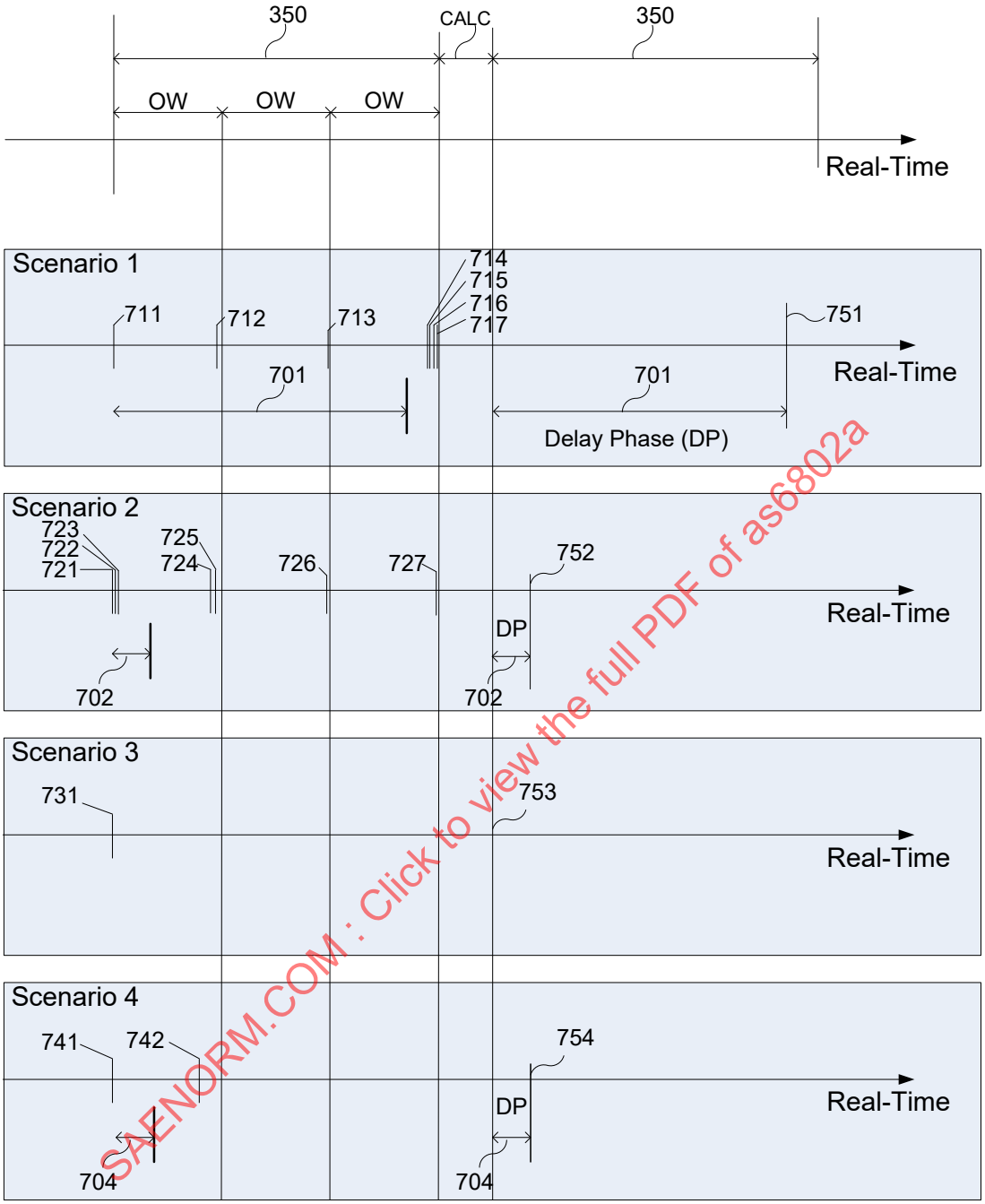


Figure 22 - Synchronization compression function: detailed description

The compression function consists of three phases: the collection phase, the calculation phase, and the delay phase. The length of the collection phase varies with the presence of failures or cliques. When the collection phase starts, it collects for the length of an observation window (OW) with *observation_window* length. Depending on the number of PCFs that become permanent during this window, additional windows will open and the collection phase will continue or not. In the calculation phase, the compression function calculates a fault-tolerant delay. Following the calculation phase, the compression function enters a delay phase which lasts a length of time equal to the fault-tolerant delay (delay phase). At the end of the delay phase, the compression function signals the *cm_compressed_pit* and delivers the compressed PCF.

Figure 22 depicts a realization with a *max_observation_window* (350) of three times an *observation_window* (shown as OW). The *observation_window* defines the granularity of the measurement for the observation duration of the collection phase in the compression function and is defined in Equation 38. Hence, in the fault-free case, all synchronization masters would send their PCFs, which would result in permanence points in time that are, at most, an *observation_window* apart from each other. Due to potentially faulty synchronization masters, the compression master cannot simply collect for a single *observation_window*, but has to prolong its collection phase, as described in 6.2.

The configuration of *max_observation_window* = 3 × *observation_window* is able to tolerate up to two faulty synchronization masters. In general, the maximum observation window follows the formula below:

$$\text{max\_observation\_window} = (f + 1) \times \text{observation\_window} \quad \text{(Eq. 12)}$$

where *f* = the number of faulty synchronization masters that have to be tolerated.

This is to mitigate the possibility that a faulty synchronization master causes the compression master to collect only a subset of the permanence points in time from correct synchronization masters (see Scenario 1 in 6.1.1).

6.1.1 Discussion Scenarios

Scenario 1: This is a collection scenario with two faulty synchronization masters that have fast clocks. The fastest faulty synchronization master results in the permanence point in time 711, kicking off the collection process in the compression master. Just before the end of the first *observation_window*, the second faulty synchronization master produces the permanence point in time 712, which leads the compression master to continue the collection process for another *observation_window*. Towards the end of the second *observation_window*, the first correct synchronization master produces a permanence point in time 713. This again causes the compression master to continue collecting for a final *observation_window*. By definition, all correct synchronization masters will now produce their permanence points in time (714 to 717) in this third *observation_window*, as the maximum deviation of any two correct clocks in the system will not exceed the size of one *observation_window*. At the end of the third *observation_window*, the compression master will calculate the fault-tolerant average/midpoint (701) from the received permanence points in time. After the calculation, the compression master waits for a duration that is equal to the outcome of the fault-tolerant average/midpoint (701) and the compressed point in time (751) is reached when this delay phase expires.

NOTE: This scenario also shows why it is not possible to collect for a single, predetermined interval. For any such given interval, the faulty synchronization master(s) can start the collection process such that it will finish with only a perceived subset of correct permanence points in time. For example, if the algorithm were constructed such that it always counted for two observation windows, then the calculation of the fault-tolerant average/midpoint would only recognize the permanence point of 713.

Scenario 2: This scenario is a collection that includes two faulty synchronization masters with slow clocks. The correct synchronization masters with fast clocks will produce the permanence points in time (721 to 723), which start the collection process in the compression master. At the end of the first *observation_window*, all correct synchronization masters have produced their permanence points in time. However, the first permanence point in time may have been sent from a faulty source. Hence, the compression master continues the collection phase for a second *observation_window*, observing another permanence point in time (726). This causes the compression master to continue collection for another *observation_window*, perceiving yet another permanence point in time (727). The calculation and production of the compressed point in time occurs similarly to Scenario 1.

Scenario 3: This scenario perceives a single permanence point in time. In this case, the compression master stops the collection of permanence points in time after the expiration of the first *observation_window*. To respect the nominal latency through the compression master process, the compressed point in time signals after the CALC at 753. Scenario 3 is an example of a single faulty synchronization master, which is far out of synchronization with the correct synchronization masters.

Scenario 4: This scenario is similar to Scenario 3, except that perception of the two synchronization masters occurs during the first *observation_window*, which causes the compression master to collect for two *observation_windows*. However, no additional value is received during the second *obervation_window*, so the collection phase stops. The compressed point in time (754) calculates analogously to Scenario 1.

6.2 Normative Description

The compression function shall consist of three phases: the collection phase, the calculation phase, and the delay phase.

The compression function shall execute the three phases in sequence.

6.2.1 Collection Phase

A new compression function shall begin to collect PCFs when a PCF with *pcf_type* = IN becomes permanent and no compression function is collecting PCFs with *pcf_type* = IN with matching *pcf_integration_cycle*.

A new compression function shall begin to collect PCFs when a PCF with *pcf_type* = CA becomes permanent and no compression function is collecting PCFs with *pcf_type* = CA and when operating with standard integrity synchronization masters (see Table 1).

A collecting compression function shall collect PCFs with *pcf_type* = IN and matching *pcf_integration_cycle*. A collecting compression function shall collect PCFs with *pcf_type* = CA.

NOTE: This also means that a potentially faulty synchronization master, which sends a PCF with a faulty *pcf_integration_cycle* relatively early to the correct synchronization masters, will cause the compression master to start a collection process with a faulty integration cycle. The compression master executes multiple instances of the compression function in parallel in the presence of faulty synchronization masters. The starting points of the compression function (i.e., 711 in Scenario 1, 721 in Scenario 2, 731 in Scenario 3, and 741 in Scenario 4) appear in 6.1.1.

The compression master shall accept a synchronization master's PCF only when the same synchronization master did not contribute to any other active compression function.

NOTE: One synchronization master can contribute to at most one compression function in a compression master at any point in time. A compression master will execute at most as many parallel compression functions as there are synchronization masters from which it collects PCFs. The compression master may, for example, use a leaky bucket algorithm to rate-limit the access to the compression function per synchronization master. An upper bound for the number of active compression functions is provided by the maximum number of synchronization masters as limited by the number of membership bits in *pcf_membership_new* (32 bits). An implementation may decide to support a smaller number of synchronization masters.

After the first observation window, a compression function shall stop collecting PCFs (at $cm\_permanence\_pit_1$+ *observation_window*), if the number of collected permanent PCFs is one.

NOTE: When a new compression function begins, the collection will be active for an *observation_window* with an offline specified duration. At the end of the first *observation_window*, the compression function evaluates the number of collected PCFs. If this number is greater than one, then a second *observation_window* starts, continuing collection (e.g., Scenarios 1, 2, and 4). If the number of collected PCFs is one, then the collection stops (e.g., Scenario 3).

After a specific second or higher observation window n > 1, a compression function shall stop collecting PCFs (at *cm_permanence_pit1* + (n * *observation_window*)), if the number of collected PCFs is equal to the number of collected PCFs at the end of the previous observation window. This means that during the last *observation_window* the compression function did not collect any more PCFs.

NOTE: At the end of the second *observation_window*, the compression function checks the number of permanent PCFs. If during the second *observation_window,* at least one more PCF has been collected, then the third *observation_window* will begin and the collection will continue (e.g., Scenarios 1 and 2). If during the second *observation_window* no more PCF is collected, then the collection function stops collecting (e.g., Scenario 4).

A compression function shall stop collecting no later than $cm\_permanence\_pit_1$ + *((f+1)* × *observation_window*), where *f* is a configurable parameter that specifies the number of tolerated faulty synchronization masters.

NOTE: For the described two-fault-tolerance case, the collection completes no later than at the end of the third *observation_window*.

6.2.2 Calculation Phase

The inputs used in the fault-tolerant average calculations are the relative distances of the *cm_permanence_pits* (for i ≥ 1):

$$\text{input}_i = \text{cm\_permanence\_pit}_i - \text{cm\_permanence\_pit}_1 \quad \text{(Eq. 13)}$$

The compression function shall use the following fault-tolerant average to calculate the compression correction:

a. One input: *Compression_correction* = input1

b. Two inputs: *Compression_correction* = (input1+input2)/2

c. Three inputs: *Compression_correction* = input2

d. Four inputs: *Compression_correction* = (input2+input3)/2

e. Five inputs*: Compression_correction* = (input2+input4)/2

f. More than five inputs: *Compression_correction* = arithmetic mean of kth largest and kth smallest input (where k is a configurable parameter, e.g., set to three for two-fault tolerance).

NOTE: When the collection finishes, the compression function calculates a fault-tolerant average value, *compression_correction* (Scenario 1: 701, Scenario 2: 702, Scenario 3: 0, Scenario 4: 704), from the relative distances of the individual permanence points in time (Scenario 1: 712 to 717, Scenario 2: 722 to 727, Scenario 3: none, Scenario 4: 742) from the PCFs to the permanence point in time of the first PCF (Scenario 1: 711, Scenario 2: 721, Scenario 3: 731, Scenario 4: 741). The overhead of the calculation phase, *calculation_overhead* (depicted by CALC in Figure 22), of this fault-tolerant average can be calculated offline and accounted for as a static delay dependent on the frame type.

The *compression_function_delay* shall be calculated as follows:

$$\text{compression\_function\_delay} = \max \text{observation window} + \text{calculation\_overhead} + \text{compression\_correction} \quad \text{(Eq. 14)}$$

NOTE: Default configuration for *calculation_overhead* is as follows:

Table 3 - Calculation overhead per frame type

	Coldstart Frame	Coldstart Acknowledge Frame	Integration Frame
Single-Failure Hypothesis	*acceptance_window*	*acceptance_window*	0
Dual-Failure Hypothesis	*acceptance_window*	*acceptance_window*	*acceptance_window*

The *cm_compressed_pit* shall be calculated as follows:

$$cm\_compressed\_pit = \begin{cases} cm\_permanence\_pit_1 + compression\_function\_delay, & for\ compressed\ frames\ per\ Table\ 1 \\ cm\_permanence\_pit + max\_observation\_window + calculation\_overhead, & for\ uncompressed\ frames\ per\ Table\ 1 \end{cases} \quad \text{(Eq. 15)}$$

NOTE: PCFs, irrespective of whether they are compressed or not, have a *cm_compressed_pit* defined. This is the point in time (see 9.1) when the startup and restart service (Section 9) will process the frame.

NOTE: The $cm\_permanence\_pit_1$ refers to the *cm_permanence_pit* of the PCF, which started a given compression function (711, 721, 731, and 741). See Figure 22 for a depiction of the *cm_compressed_pits* as 751 to 754.

6.2.3 Delay Phase

The duration of the delay phase equals the *compression_correction* and immediately follows the calculation phase. The *cm_compressed_pit* marks the end of the delay phase.

The startup and restart service (Section 9) will process the PCFs when *cm_compressed_pit* is reached.

NOTE: Figure 22 depicts the calculation phase that is executed after the maximum number of observation windows have passed. In this case, the result of the fault-tolerant midpoint calculation *compression_correction* as introduced in Equation 14 gives the actual length of the delay phase. It is also a valid implementation option to calculate the fault-tolerant midpoint immediately after the collection of values stops, which can occur at any observation window boundary. In this case, the delay phase is prolonged for the missing observation windows to the maximum number of observation windows.

NOTE: The overall delay through the compression master function is a key term for the flow of protocol control frames in the system.

6.2.4 PCF Field Assignments

In each compressed PCF that the compression function generates, the compression function shall set a bit in *pcf_membership_new* to one, if the corresponding synchronization master sent a PCF that became permanent during the collection phase of the compression function.

6.2.5 Bounded Influence Requirements

The compression master shall count a received PCF from a synchronization master as only one input to the calculation of the compression function.

NOTE: A faulty synchronization master will not cause the compression master to use its faulty PCF as more than one input when calculating the *compression_correction.*

The compression master shall be able to identify the synchronization master for a received PCF.

NOTE: A faulty synchronization master cannot masquerade as another synchronization master. In particular it cannot send a PCF causing the compression master to set a wrong bit in the *pcf_membership_new* of the compressed PCF.

6.2.6 Compression Function Parameter Ranges

The compression function shall tolerate up to two faulty synchronization masters.

7. CLOCK SYNCHRONIZATION SERVICE

A clock synchronization service ensures that the local clocks of the devices in the communication infrastructure remain synchronized with each other, once aligned. See Sections 5 and 6, respectively, for details regarding the message permanence function and the compression function. The message permanence function calculates the *permanence_pits* from the *receive_pits* of PCFs. The compression function calculates the *cm_compressed_pit* from the *cm_permanence_pits* of PCFs. This section discusses how the results of the message permanence function and the compression function modify the synchronized global time in devices in the same cluster (e.g., synchronization master, synchronization client, or compression master) in order to maintain synchronization.

NOTE: In time-triggered Ethernet, the following two synchronized variables in each component represent global time: *local_integration_cycle* and *local_clock*. The variable *local_integration_cycle* cyclically counts from 0 to *max_integration_cycle* and reflects the current integration cycle. The variable *local_clock* is the microtiming within an integration cycle. It is possible to identify the current point in the cluster cycle by the evaluation of the current values of *local_integration_cycle* and *local_clock*. This means that the epoch of the global synchronized time is one cluster cycle: once a cluster cycle finishes, the global synchronized time will wrap around and start over with 0. In a case that requires a longer epoch covering multiple cluster cycles, a simple application (or middleware) process allows for easy realization.

NOTE: Synchronization masters/clients, as well as compression masters, will only accept PCFs with a *pcf_sync_domain* field set to a preconfigured value in the respective component. Furthermore, the synchronization masters and clients maintain a local variable, *local_current_sync_priority*, that defines the current synchronization priority. Synchronization masters and clients will only accept PCFs that have their *pcf_sync_priority* field set to the current value of *local_current_sync_priority*. See Section 10 for the method used to modify this local variable. Compression masters, on the other hand, will only accept PCFs that have their *pcf_sync_priority* field set to a preconfigured value.

Figure 23 reviews the cluster cycle/integration cycle example from Section 4 (see Figure 12).

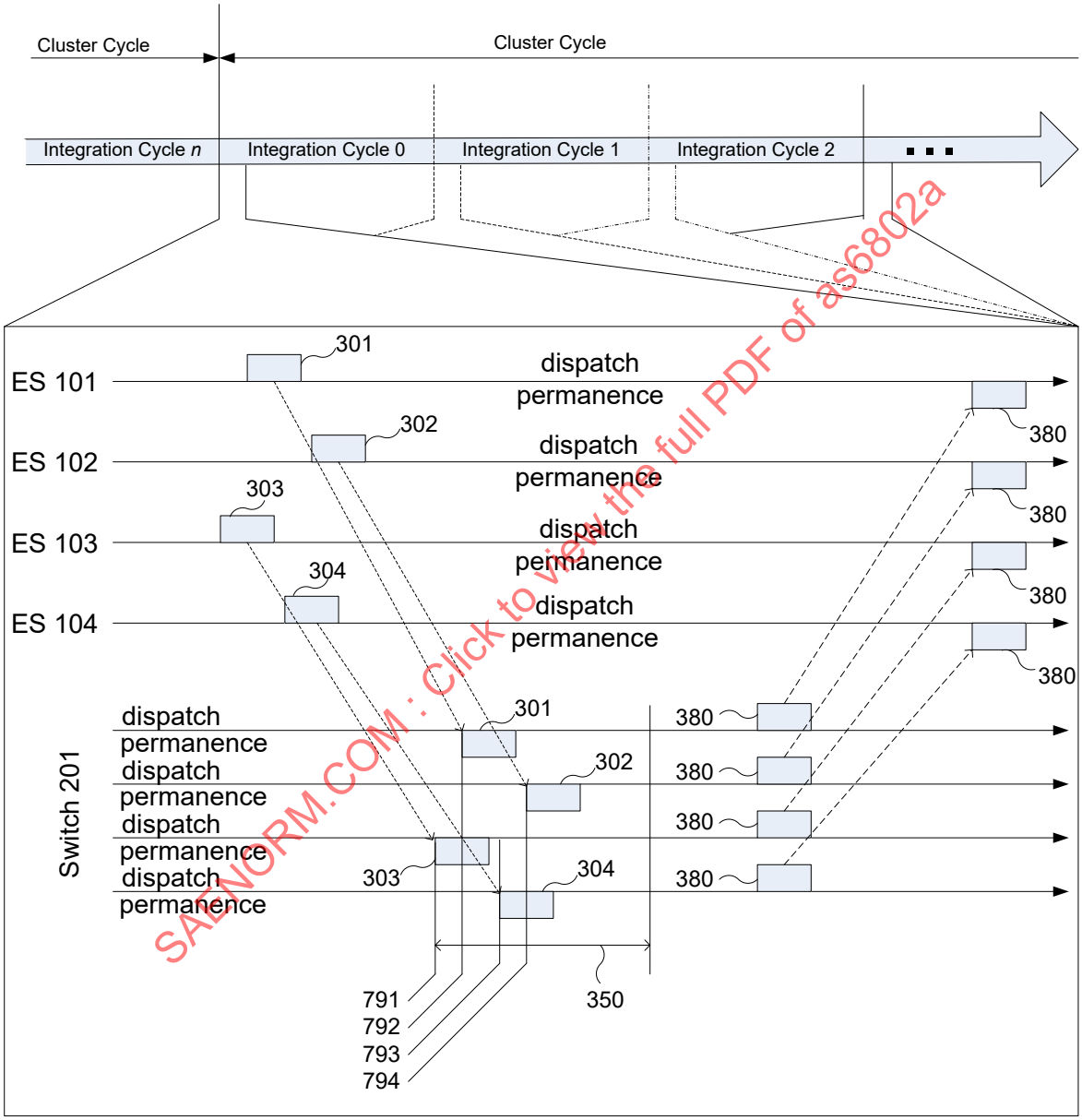


Figure 23 - Example of cluster cycle and integration cycles

7.1 Clock Synchronization in Synchronization Master/Client

During synchronized operation, the synchronized *local_clock* indicates when the *sm_dispatch_pits* and the *smc_scheduled_pits* are reached. See Figure 24 for an overview of the clock synchronization service in synchronization master and synchronization client.

NOTE: During synchronized operation, all PCFs that are exchanged are of type INT. In the following paragraph, these are referred to simply as PCFs if not explicitly stated otherwise.

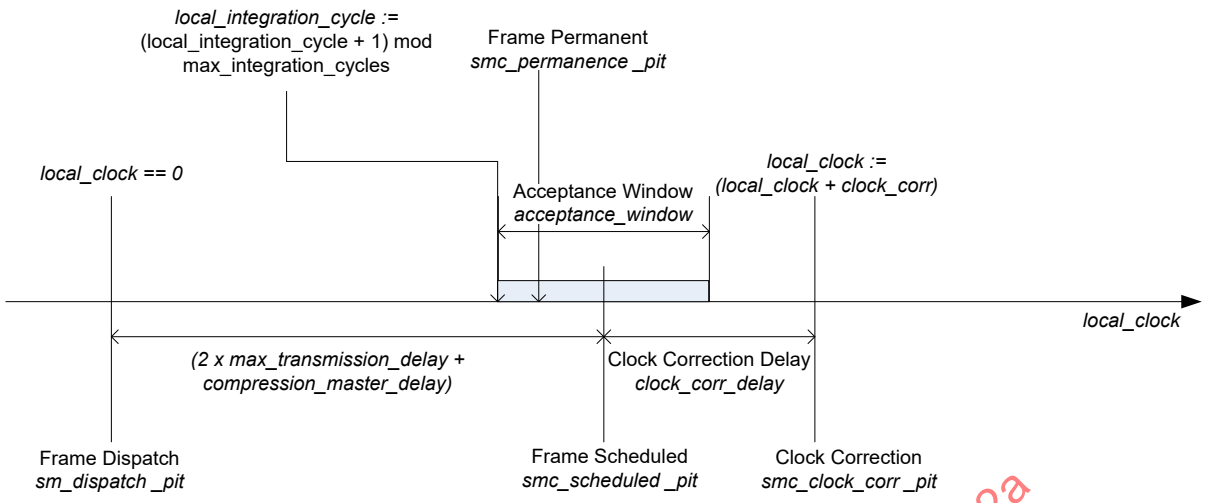


Figure 24 - local_clock in synchronization master and synchronization client

The description in Figure 24 starts with synchronization masters dispatching a PCF, followed by the synchronization masters and synchronization clients receiving PCFs back from the compression masters.

The variable *local_clock* acts as a local counter that cyclically runs from 0 to *integration_cycle_duration*. Synchronization masters and synchronization clients maintain a variable *local_integration_cycle*, which keeps track of the current integration cycle. The variable *local_integration_cycle* initializes to a configurable value upon entering synchronized operation, enabling a device to integrate at multiple points during a cluster cycle, rather than only at the start. During synchronized operation, the *local_integration_cycle* is updated in each integration cycle before an acceptance window starts.

For the *sm_dispatch_pit* the following holds:

$$(\text{local\_clock} == 0) \Rightarrow \text{sm\_dispatch\_pit} \qquad \text{(Eq. 16)}$$

Meaning that each time *local_clock* reaches 0, a synchronization master will dispatch a PCF with:

a. *pcf_integration_cycle* = (*local_integration_cycle*+1) modulo *max_integration_cycle*

b. *pcf_membership_new*[x] = 1 (if x is the index of the synchronization master in the *pcf_membership_new* vector), and *pcf_membership_new*[y] = 0 (for all other entries y ≠ x)

c. *pcf_sync_domain* = *sync_domain*

d. *pcf_sync_priority* = *sync_priority*

e. *pcf_transparent_clock* = 0

f. *pcf_type* = 0x2 hex (indicating an integration frame)

The parameters *sync_domain* and *sync_priority* are device-specific and statically configured parameters in the synchronization master, synchronization client, or compression master.

The synchronization clients will not dispatch PCFs. However, the *smc_scheduled_pit* is calculated offline with respect to the *sm_dispatch_pit*. The PCF flows to the compression masters (Figure 11, Step 1). The compression masters then either compress the PCF or transmit the PCF without compression back to the synchronization masters and synchronization clients (Figure 11, Step 2). The nominal temporal distance between the *sm_dispatch_pit* and the *smc_scheduled_pit* (assuming no clock drift, errors introduced when crossing clock boundaries, digitalization effects, etc., yielding *compression_correction* = 0) can be calculated offline as follows:

$$\begin{aligned}\text{smc\_scheduled\_pit} = \text{sm\_dispatch\_pit}_1 + 2\text{x max\_transmission\_delay} \\ + \text{max\_observation\_window} + \text{calculation overhead} + \text{dispatch\_delay}\end{aligned} \quad \text{(Eq. 17)}$$

The factor 2 for the *max_transmission_delay* accounts for the sending path from the synchronization master to the compression master (Figure 11, Step 1) and the sending path from the compression master back to the synchronization master/clients (Figure 11, Step 2), i.e., the round-trip delay. The parameters *max_observation_window and calculation_overhead* are the terms from the *compression_function_delay* as defined in Equation 14 (the third parameter, *compression_correction*, is set to 0) and *dispatch_delay* is an offset that is configurable per frame type.

NOTE: The nominal round-trip delay is equal for all PCF types.

NOTE: The message permanence function already accounts for the individual delay parameters and compressed and uncompressed PCFs experience the same nominal delay (see Equation 14).

Each synchronization master and synchronization client will open an acceptance window (*acceptance_window*) around the *smc_scheduled_pit* and monitor the *smc_permanence_pits* from PCFs via the compression master with (*local_integration_cycle* equals *pcf_integration_cycle*) (Figure 11, Step 2). Consider such PCFs as *received in-schedule*. PCFs received with *smc_permanence_pit* that lie outside the respective acceptance window or with *pcf_integration_cycle* not equal to *local_integration_cycle* are *received out-of-schedule*.

The monitoring occurs per communication link that connects a synchronization master and synchronization client. At the end of the acceptance window, the synchronization master and synchronization client select the PCF with the highest number of bits set in *pcf_membership_new* for each communication link. In case of multiple PCFs with the same maximum *pcf_membership_new*, the selected PCF is the one with the latest *smc_permanence_pit* within the acceptance window or the frame closest to the end of the acceptance window.

$$\text{smc\_best\_pcf}_{\text{channel}} = \max_{\text{smc\_permanence\_pit}}[\max_{\text{pcf\_membership\_new}}(\text{PCF}^{\text{in-schedule channel}})] \quad \text{(Eq. 18)}$$

For the calculation of the clock correction term (*clock_corr*), only those PCFs from the set of *smc_best_pcf_channels* are considered that have a number of bits set in their *pcf_membership_new* that are in the interval [*max(pcf_membership_new), max(pcf_membership_new) − MembershipAcceptanceRange],* where:

- *max(pcf_membership_new)* is seen with respect to the set of *smc_best_pcf_channels*, and
- *MembershipAcceptanceRange* is a configurable parameter typically set to the number of faulty synchronization masters to be tolerated.

The clock correction value to be applied, *clock_corr*, is then calculated by a configurable function which is either the median of the remaining values or the average of the extremes.

$$\text{clock\_corr} = \text{median/average}_{\text{channels}}(\text{smc\_scheduled\_pit} - \text{smc\_best\_pcf}_{\text{channel}}.\text{permanence}) \quad \text{(Eq. 19)}$$

The prefiltering of PCFs removes obviously faulty values from the median function, which may stem from a faulty compression master that did not receive all PCFs from non-faulty compression masters. Another failure case is when one or more faulty synchronization masters only send their PCFs to a subset of the compression masters.

The clock correction value (*clock_corr*) does not immediately apply to correct the local clock (*local_clock*) at the end of the *acceptance_window*. Instead, this correction is delayed for a configurable parameter (*clock_corr_delay*), where the following relation is satisfied:

$$\text{clock\_corr\_delay} > \text{acceptance\_window} \quad \text{(Eq. 20)}$$

This delayed clock correction ensures that the corrected local clock value will not fall within the *acceptance_window* (as the maximum negative *clock_corr* is -*acceptance_window*/2). The local clock requires correction within one integration cycle (i.e., before *local_clock* = 0 again). This means that clock correction terms are not added up over a sequence of integration cycles.

7.2 Clock Synchronization in Compression Master

Similar to the synchronization masters and synchronization clients, the compression masters will use the synchronized local clock (*local_clock*) during synchronized operation to indicate reaching the *cm_scheduled_pits*. The compression master, however, uses the *cm_compressed_pit* as a reference point to correct *local_clock*.

An overview of the clock synchronization service in the compression master appears in Figure 25.

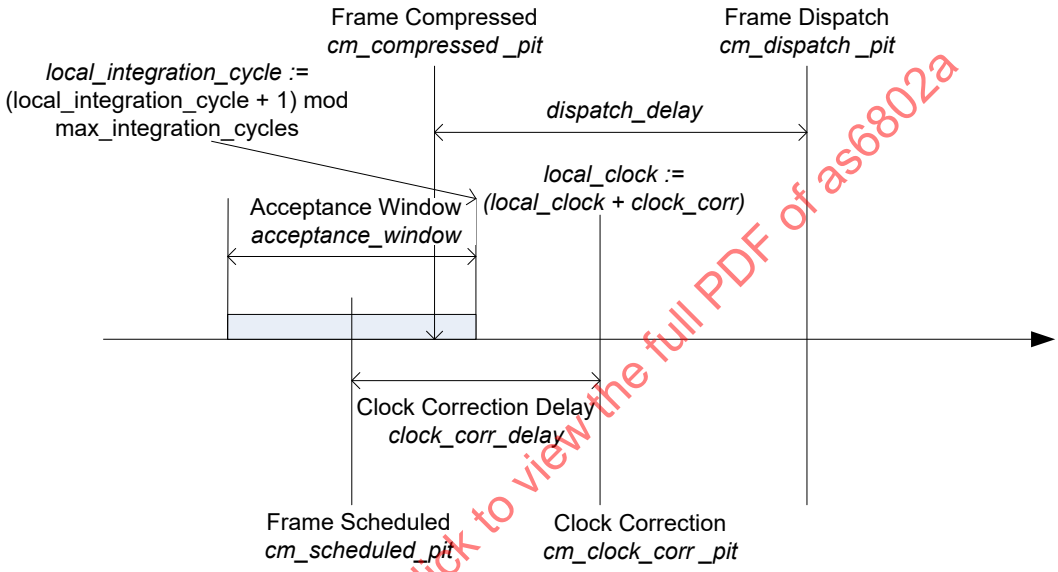


Figure 25 - local_clock in compression master

It is possible to interpret the *local_clock* as a local counter that cyclically runs from 0 to *integration_cycle_duration*, as previously noted. The parameter *cm_scheduled_pit* is a configurable parameter. In order to keep the local clocks of the synchronization masters, synchronization clients, and compression masters synchronized, configure them as follows:

$$\text{cm\_scheduled\_pit} = \text{sm\_dispatch\_pit}_1 + \text{max\_transmission\_delay} + \text{max\_observation\_window} + \text{calculation\_overhead} \quad \text{(Eq. 21)}$$

Compression masters maintain a device-local variable *local_integration_cycle* that keeps track of the current integration cycle. During synchronized operation, the *local_integration_cycle* updates once in each integration cycle as depicted in Figure 25.

Similar to synchronization masters and synchronization clients, each compression master will open an acceptance window (*acceptance_window*) around the *cm_scheduled_pit* and monitor the *cm_compressed_pits* from PCFs via the compression function (*local_integration_cycle* equals *pcf_integration_cycle*). These PCFs are known as received in-schedule. PCFs with *cm_compressed_pit* that lie outside the respective acceptance window or with different integration cycle than expected are considered *received out-of-schedule*.

In certain failure and clique scenarios (see Section 8 for discussion of cliques), more than one *cm_compressed_pit* of respective PCFs with the same *pcf_integration_cycle* may be generated within the same acceptance window in the compression master. The compression master will then use the PCF with the most bits set in *pcf_membership_new*, and the latest PCF in case of a tie in *pcf_membership_new*.

$$\text{cm\_best\_pcf} = \max_{\text{cm\_compressed\_pit}}[\max_{\text{pcf\_membership\_new}}(\text{Protocol control frame})] \quad \text{(Eq. 22)}$$

Likewise, the clock correction value in the compression master is calculated as follows:

$$\text{clock\_corr} = (\text{cm\_scheduled\_pit} - \text{cm\_best\_pcf.cm\_compressed\_pit}) \quad \text{(Eq. 23)}$$

NOTE: The compression function already performs the fault-tolerant average calculation.

Similar to the synchronization masters and synchronization clients, the clock correction value (*clock_corr*) is not immediately applied to correct the local clock (*local_clock*) at the end of the acceptance window. This correction instead is delayed for a configurable amount of time using *clock_corr_delay*. This correction instead is delayed for a configurable amount of time using *clock_corr_delay*, such that the following relationship is satisfied:

$$\text{clock\_corr\_delay} > \text{acceptance\_window} \quad \text{(Eq. 24)}$$

This delayed clock correction ensures that the corrected local clock value will not fall within the *acceptance_window* (as the maximum negative *clock_corr* will be *-acceptance_window*/2). The local clock requires correction within one integration cycle (i.e., before *local_clock* = 0 again). This means that clock correction terms may not add up over a sequence of integration cycles.

7.3 Compressed PCF Dispatch

The *cm_dispatch_pit* of a PCF is an offset that is configurable, per frame type (*dispatch_delay*), after the *cm_compressed_pit*. The *dispatch_delay* ensures that dispatch of the compressed PCF will not happen prior to the end of the acceptance window.

$$\text{cm\_dispatch\_pit} = \text{cm\_compressed\_pit} + \text{dispatch\_delay} \quad \text{(Eq. 25)}$$

NOTE: Equation 26 was removed in this revision, but all other existing equation references are unchanged compared to old revisions.

NOTE: Default configuration for *dispatch_delay* is as follows.

Table 4 - Dispatch delay per frame type

	Coldstart Frame	Coldstart Acknowledge Frame	Integration Frame
Single-Failure Hypothesis	0	0	*acceptance_window*
Dual-Failure Hypothesis	0	0	0

The *cm_dispatch_pit* is the start of Step 2 in the synchronization process (i.e., PCF flow back from the compression master to the synchronization masters and synchronization clients).

7.4 Normative Description

7.4.1 Synchronization Master/Client

When the local clock in a synchronization master reaches *local_clock* = 0, then the synchronization master shall dispatch an integration frame with:

a. *pcf_integration_cycle* = *(local_integration_cycle+1) modulo max_integration_cycle*

b. *pcf_membership_new[x]* = 1 (if x is the index of the synchronization master in the *pcf_membership_new* vector), and *pcf_membership_new[y]* = 0 (for all other entries y ≠ x)

c. *pcf_sync_domain* = *sync_domain*

d. *pcf_sync_priority* = *sync_priority*

e. *pcf_transparent_clock* = 0

NOTE: By the time the PCF is actually sent, the value will be set to the total delay between *dispatch_pit* and *send_pit*.

f. *pcf_type* = 0x2 hex (which indicates an integration frame)

Each synchronization master/client shall open an acceptance window around its scheduled point in time (*smc_scheduled_pit*), where the size of the acceptance window (*acceptance_window*) is twice the precision in the system and the *smc_scheduled_pit* is the center of the acceptance window.

Out of the integration frames received during the acceptance window and at the end of the acceptance window, each synchronization master/client shall select a set of integration frames using the selection function in Equation 18 where the selected integration frames are further called the best *in-schedule* received integration frames.

After the selection of the best in-schedule received integration frames, the synchronization master/client shall calculate the clock correction term (*clock_corr*), as specified in Equation 19.

The synchronization master/client shall wait for a duration of *clock_corr_delay* before it modifies its local clock (*local_clock*) using the value of the calculated clock correction term (*clock_corr*).

7.4.2 Compression Master

Each compression master shall open an acceptance window around its scheduled point in time (*cm_scheduled_pit*), where the size of the acceptance window (*acceptance_window*) is twice the precision in the system and the *cm_scheduled_pit* is the center of the acceptance window.

At the end of the acceptance window, each compression master shall select the best in-schedule received integration frame using the selection function given in Equation 22.

After the selection of the best in schedule received integration frame, a compression master shall apply Equation 23 to calculate the clock correction value (*clock_corr*).

Before the compression master modifies its local clock (*local_clock*) using the value of the calculated clock correction term (*clock_corr*), it shall wait for a duration of *clock_corr_delay*.

A compression master configured to operate with high-integrity synchronization masters shall dispatch the compressed integration frame when it reaches *cm_dispatch_pit*.

NOTE: The *cm_dispatch_pit* is calculated using Equation 25.

A compression master that is configured to operate with standard-integrity synchronization masters shall dispatch the compressed integration frame when the *cm_dispatch_pit* is reached and the compression master has used the compressed integration frame for synchronization itself.

NOTE: The *compression master state machine* discussed in 9.4 and 9.5 determines whether or not the compression master uses the integration frame.

8. CLIQUE DETECTION AND RESOLUTION SERVICES

A clique detection service detects clique scenarios. These are unintended scenarios where disjoined subsets of devices within a synchronization domain align within a subset, but not over subset boundaries. A clique resolution service is a method that reestablishes synchronization on a network level, upon clique detection.

There are three clique detection services in time-triggered Ethernet: synchronous clique detection, asynchronous clique detection, and relative clique detection. These clique detection algorithms operate on data structures called “membership” which are periodically updated through the exchange of the PCFs (*pcf_membership_new*).

Synchronous clique detection runs locally on all components in the system, checking for sufficient synchronization masters aligned with the respective component. This occurs via evaluation of the integration frame (a PCF with *pcf_type* set to IN) produced in Step 2 of the clock synchronization service (i.e., if the number of bits set in the *pcf_membership_new* field falls below a configurable threshold, then the synchronous clique detection returns successful).

The asynchronous clique detection also runs locally on all system components. It keeps track of all operating synchronization masters that are not aligned with respective components. The asynchronous clique detection locally stores the membership bits from those out of schedule received IN frames, stemming from synchronization masters that are not synchronized to the respective component. Once per resynchronization interval, the component checks whether the number of bits set in the *pcf_membership_new* fields of unsynchronized IN frames is equal to or above a configurable threshold. If so, the asynchronous clique detection service returns successful.

The relative clique detection is executed in the synchronization masters. It checks, at the beginning of each integration cycle, whether the number of synchronized synchronization masters is equal to or below the number of unsynchronized synchronization masters (if any). If this is the case, the relative clique detection returns successful.

The clique resolution service is performed by restarting the respective component that detected the clique situation. Depending on the number of components that detected the clique situation, some or all components will execute restart.

During a restart, an individual component can reintegrate to the set of components that remain in operation. Only the affected component will detect cliques, as all received PCFs will be out-of-schedule. See Section 9 for information regarding the time-triggered Ethernet integration and startup/restart protocol.

8.1 Synchronous Clique Detection Function

See Figure 26 for a depiction of the synchronous clique detection function.

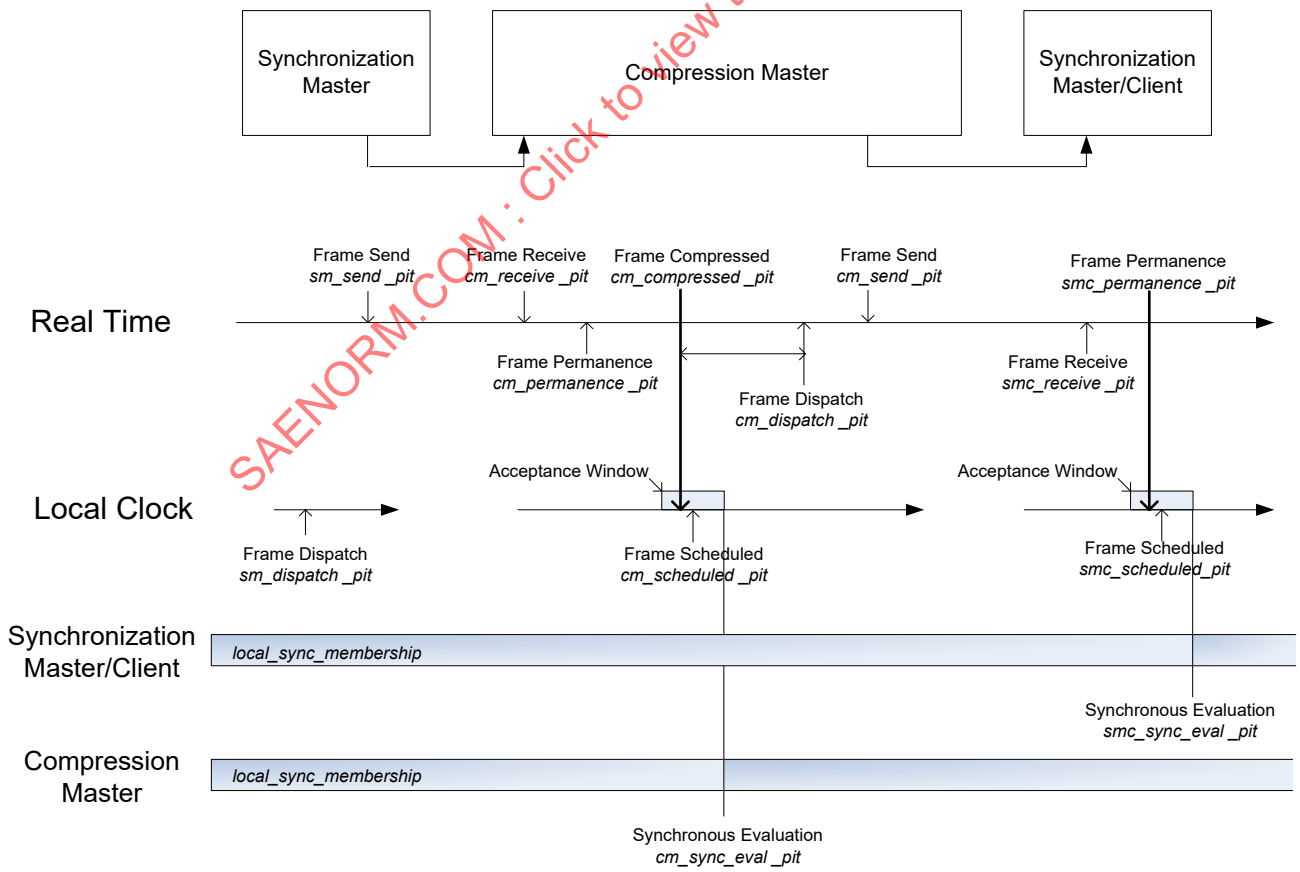


Figure 26 - Synchronous clique detection function

The synchronous clique detection function is executed in all devices (i.e., synchronization masters, synchronization clients, and compression masters).

The synchronous clique detection function uses the device-local variable *local_sync_membership* to keep track of the number of synchronization masters currently synchronized with the respective device. The *local_sync_membership* is a bit-vector with the same assignment of bits to synchronization masters as in *pcf_membership_new*.

The synchronous clique detection function will be executed upon reaching the *smc_sync_eval_pit* (in the synchronization masters and synchronization clients) or *cm_sync_eval_pit* (in the compression masters). This is immediately after the acceptance window around *smc_scheduled_pit* and *cm_scheduled_pit*, respectively.

Upon reaching the *smc_sync_eval_pit* or *cm_sync_eval_pit*, each device will update *local_sync_membership*. The device will set *local_sync_membership* (i.e., it will select the *pcf_membership_new* with the highest number of bits set from the set of in-schedule PCFs received during the acceptance window).

The synchronous clique detection function tests *local_sync_membership* against a state-specific threshold. If the number of bits set in *local_sync_membership* is less than the respective threshold, then the test is successful (i.e., cliques are detected).

8.2 Asynchronous Clique Detection Function

The asynchronous clique detection function appears in Figure 27.

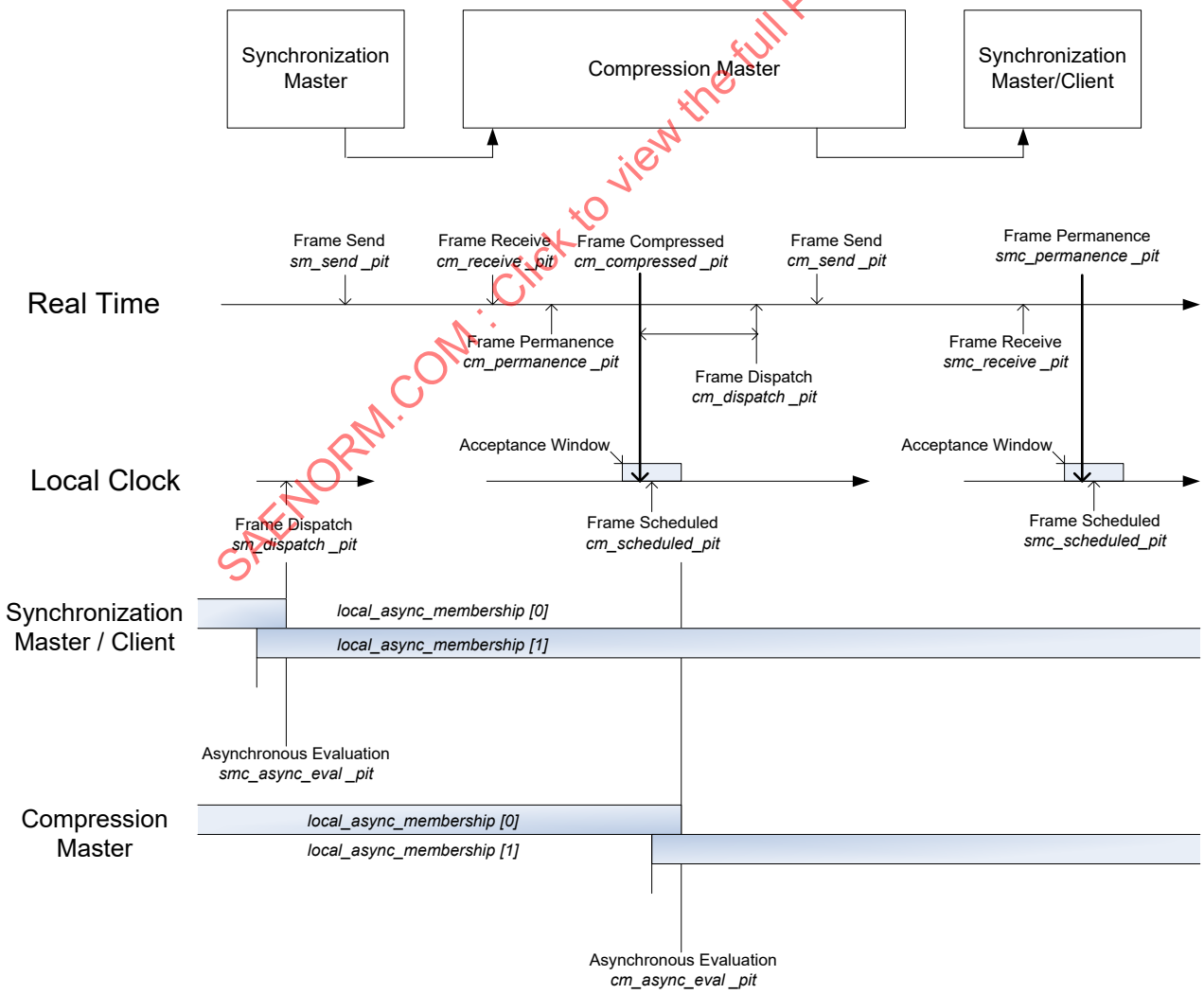


Figure 27 - Asynchronous clique detection function

The asynchronous clique detection function is executed in all devices (i.e., synchronization masters, synchronization clients, and compression masters). The asynchronous clique detection function uses two local variables (i.e., *local_async_membership[0]* and *local_async_membership[1]*) to keep track of the number of operational synchronization masters currently not synchronized with the respective device (excludes powered-off synchronization masters) and a third local variable *ae* (as an index) to switch between them. The parameter *local_async_membership* defines two bit-vectors with the same assignment of bit to synchronization master as in *pcf_membership_new*.

The asynchronous clique detection function monitors the communication links for out-of-schedule received PCFs.

NOTE: Out-of-schedule PCFs can be PCFs with *permanence_pits* outside of any acceptance window, as well as PCFs with *permanence_pits* inside an acceptance window, but with the wrong *pcf_integration_cycle* value.

The equation for the monitoring interval in synchronization masters and synchronization clients is as follows:

$$\left[\text{sm\_dispatch\_pit}_n - \frac{\text{acceptance\_window}}{2},\ \text{sm\_dispatch\_pit}_{n+1}\right] \quad \text{(Eq. 27)}$$

where n is an integration cycle.

Likewise, the equation for the monitoring interval in a compression master is as follows:

$$\left[\text{cm\_scheduled\_pit}_n, \text{cm\_scheduled\_pit}_{n+1} + \frac{\text{acceptance\_window}}{2}\right] \quad \text{(Eq. 28)}$$

The monitoring intervals are longer than the *integration_cycle_duration*, requiring two variables (*local_async_membership[0]* and *local_async_membership[1]*). The monitoring intervals in a device overlap, as depicted in Figure 27. The asynchronous clique detection function will update the respective *local_async_membership* upon receipt of an out-of-schedule PCF. The update occurs by adding the bits set in *pcf_membership_new* in out-of-schedule PCFs to the respective *local_async_membership*. The *local_async_membership* variable contains out-of-schedule received PCFs. In the overlapping phase, both *local_async_membership* variables are updated. Outside of the overlapping phase, *ae* determines which of the *local_async_membership* variables to update.

When a device reaches the *smc_async_eval_pit* (respectively *cm_async_eval_pit*), it checks the asynchronous clique detection function. For synchronization masters and synchronization clients the *smc_async_eval_pit* is immediately before the *sm_dispatch_pit*, which means that the clique detection is executed before the frame dispatch decision. For the compression masters, the *cm_async_eval_pit* is at *cm_scheduled_pit* + (*acceptance_window*/2). In the case of the single-failure hypothesis, the compression master executes the asynchronous clique detection before it dispatches a compressed IN PCF. In case that the asynchronous clique detection returns successfully, the compression master does not dispatch a compressed IN PCF.

The device tests *local_async_membership[0] and local_async_membership[1]* against a state-specific threshold. The asynchronous clique detection function will return successful when the number of bits set in the respective *local_async_membership* variable is at least as high as the state-specific threshold.

Otherwise, the asynchronous clique detection function will return unsuccessful and the value of *ae* will toggle. Successful implies the detection of cliques.

8.3 Relative Clique Detection Function

The relative clique detection function executes in the synchronization masters. The relative clique detection function is always running and returns a clique detected at *smc_async_eval_pit* when the following relation is satisfied:

$$\text{local\_sync\_membership} \leq \text{local\_async\_membership}[] \quad \text{(Eq. 29)}$$

provided any one of the *local_async_membership* bit vectors has at least one bit set.

8.4 Normative Description

A synchronization master shall implement the synchronous clique detection function, the asynchronous clique detection function, and the relative clique detection function.

A synchronization client shall implement the synchronous clique detection function and the asynchronous clique detection function.

A compression master shall implement the synchronous clique detection function and the asynchronous clique detection function.

The synchronous clique detection function shall indicate that multiple cliques have formed, when following conditions are met:

a. When the local clock has reached *smc_sync_eval_pit* in a synchronization master/client.

b. When the local clock has reached *cm_sync_eval_pit* in a compression master.

c. When fewer bits are set in the *local_sync_membership* bit-vector than defined in the state-specific synchronous clique detection threshold.

The asynchronous clique detection function shall return indication of detection of a clique, as follows:

a. When the local clock has reached *smc_async_eval_pit* in a synchronization master/client.

b. When the local clock has reached *cm_async_eval_pit* in a compression master.

c. When as many or more bits are set in any one of the *local_async_membership* bit vectors than specified in the state-specific asynchronous clique detection threshold.

The relative clique detection function shall indicate detection of a clique, as follows:

a. When the local clock has reached *smc_async_eval_pit* in a synchronization master.

b. When any one of the *local_async_membership* bit vectors has at least one bit set.

c. When as many or more bits are set in any one of the *local_async_membership* bit vectors than in the *local_sync_membership* vector.

9. STARTUP AND RESTART SERVICE

A startup service defines algorithms for initially synchronizing the devices in the communication infrastructure. This service includes procedures for coldstarts and integration/reintegration.

This section discusses the time-triggered Ethernet protocol state machines, which provide the framework that coordinates the previously discussed functions and services (see Sections 5 to 8). There are different protocol state machines for the synchronization master, synchronization client, and compression master. Also, this section presents two state machines for the compression masters. This is a result of different failure hypotheses:

a. Time-triggered Ethernet tolerates multiple faulty devices at the same point in time. If the failure hypothesis requires the tolerance of the concurrent failure of a synchronization master and a compression master, then the synchronization master devices have to be reconfigured as high integrity. Consequently, the compression master executes the state machine for high-integrity synchronization masters (see Figure 30).

b. If the synchronization master is a standard-integrity device, the Time-Triggered Ethernet network does not tolerate the concurrent failure of a synchronization master and a compression master. In this case, the compression master executes the state machine for standard-integrity synchronization masters (see Figure 31).

This section discusses system configurations in which each device executes exactly one of these protocol state machines. Section 10 covers advanced synchronization/network topologies and configurations.

After power on, the synchronization masters will send coldstart frames to the compression masters. The compression masters will relay the coldstart frames back to the synchronization masters. Synchronization masters will react to a coldstart frame reception by sending a coldstart acknowledge frame. This sequence of coldstart and coldstart acknowledge frames is known as the fault-tolerant handshake.

NOTE: In a high integrity configuration, a synchronization master will only accept coldstart frames that originate from a different synchronization master. Hence, a synchronization master will not acknowledge its own coldstart frame. In a standard-integrity configuration, the synchronization masters will acknowledge all coldstart frames, regardless of whether they have been sent by themselves.

Both coldstart and coldstart acknowledge frames provide coordination action that aims at initially synchronizing the local clock (*local_clock*) in the devices. After a fault-tolerant handshake, the devices will enter a state in the protocol state machine that indicates synchronized operation.

9.1 Description of the Protocol State Machine Formalism

As in previous sections, the time-triggered Ethernet synchronization protocol uses the following variables:

a. *local_timer*: An unsynchronized timer used to measure timeouts (e.g., the duration for which a synchronization master tries to integrate before coldstart).

b. *local_clock*: The synchronized timer used to measure the current point in time relative to the current integration cycle.

c. *local_integration_cycle*: A synchronized counter that cyclically counts the integration cycles.

d. *local_sync_membership*: A membership vector with a one-to-one mapping of bits to synchronization masters, used in the synchronous and relative clique detection functions.

e. *local_async_membership[]*: A membership vector with a one-to-one mapping of bits to synchronization masters, used in the asynchronous and relative clique detection functions.

All state machines start in an *_INTEGRATE* state at the beginning of state machine execution. The state transition from the current state to the next state depends on successful transition conditions (i.e., guards) of the transition. The next state becomes the current state, after successful transition. As the state machines process events in real time, mutual guards can be enabled at the same point in time. For this case, the guards of the transition have the following decreasing priority:

1. Permanent/compressed coldstart frame.

2. Permanent/compressed coldstart acknowledge frame.

3. Permanent/compressed integration frame.

4. Unsynchronized timeout, *local_timer* = = 0.

5. Local clock reaching predefined condition time point.

NOTE: Permanent/compressed refers to the *smc_permanence_pit* (see Section 5) of a PCF in synchronization masters and synchronization clients, and to the *cm_compressed_pit* (see Section 6) in compression masters.

In the case where two or more integration frames become permanent/compressed, the priority of the integration frames decreases first with the number of bits set in *pcf_membership_new* and decreases secondly with decreasing *pcf_integration_cycle*.

Each guard of a transition will only react to a single event, with the singular exception that it will monitor out-of-schedule integration frames for the asynchronous clique detection service. If multiple events occur at the same point in time (e.g., one input (channel) of a synchronization master receives a coldstart acknowledge frame simultaneously to a different input (channel) receiving an integration frame), the guard with the highest priority event takes precedence (e.g., the guard reacting to coldstart acknowledge frames). When multiple guards are activated at the same point in time, only the guard with the highest priority, according to the list above, will be executed and the state machine will drop all other guards. This implies that events will not queue up during execution of the protocol state machine.

During unsynchronized states, the permanence/compression of an integration frame may immediately enable a transition guard. During synchronized states, the permanence/compression of an integration frame does not enable a guard immediately. The integration frame reception will lead to either an update of the synchronous clique detection or the asynchronous clique detection data structure (i.e., *local_sync_membership* or *local_async_membership[]*). The state machine evaluates the *local_sync_membership* and *local_async_membership[]* when *local_clock* reaches a particular point in time. In the case where the permanence/compression of an integration frame occurs concurrently with the evaluation of either one of the clique detection functions, the respective clique data structure will be updated before evaluation of the transitions in the protocol state machines.

9.2 Synchronization Master Protocol State Machine

Figure 28 depicts the protocol state machine executed in a synchronization master. The synchronization master differentiates between unsynchronized states and synchronized states. Unsynchronized states include *SM_INTEGRATE* state, *SM_UNSYNC* state, *SM_FLOOD* state, and *SM_WAIT_4_CYCLE_START_CS* state. Synchronized states include *SM_TENTATIVE_SYNC* state, *SM_SYNC* state, and *SM_STABLE* state.

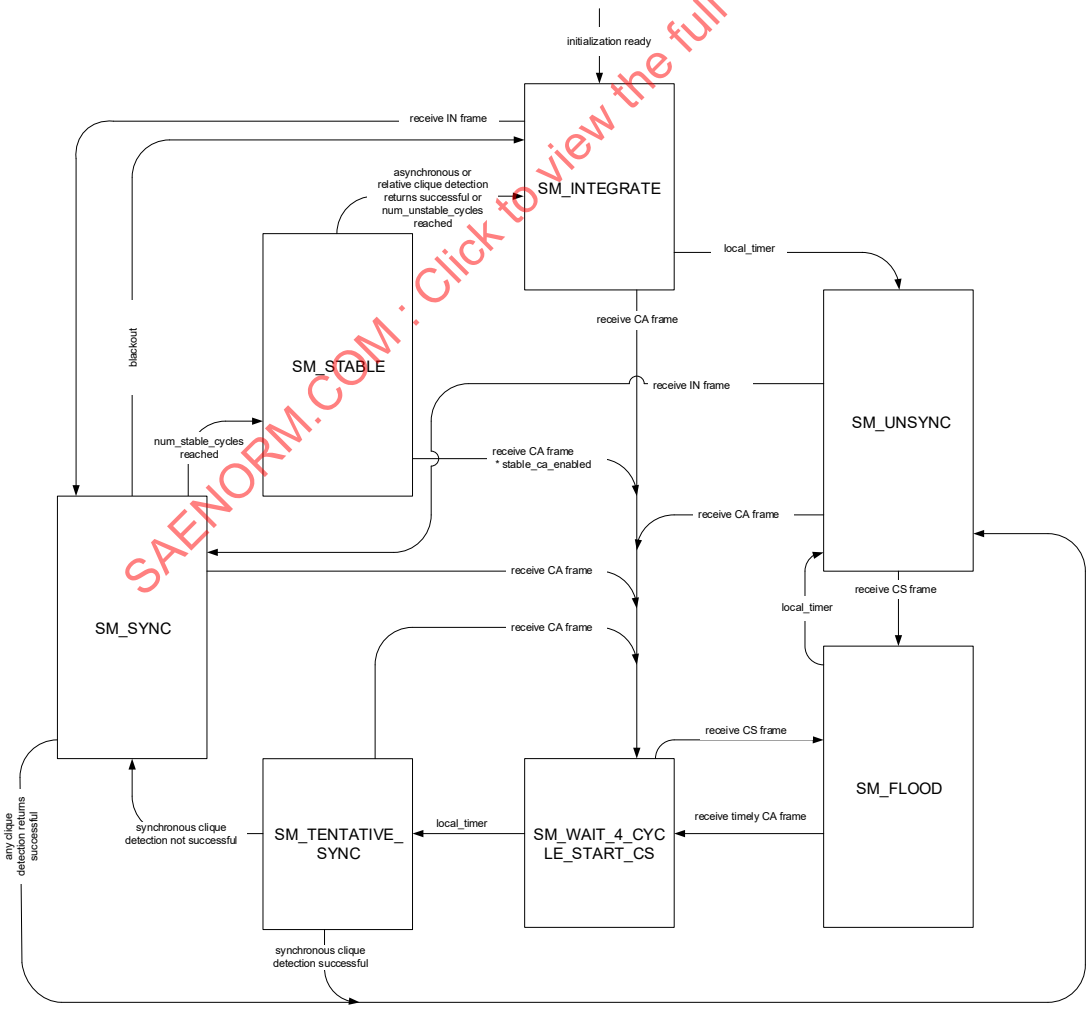


Figure 28 - Synchronization master protocol state machine

9.2.1 *SM_INTEGRATE* State

9.2.1.1 Description

The *SM_INTEGRATE* state is the first state entered after successful power-on of a time-triggered Ethernet device (*TTE_1*, say) with synchronization master functionality. The primary purpose of this state is to identify whether there are already synchronized time-triggered Ethernet devices running. If there is a set of time-triggered Ethernet devices communicating, *TTE_1* will receive PCFs from this set of devices. Depending on the number of membership bits set in the *pcf_membership_new* field, *TTE_1*, may decide to transit to a synchronized state. In this case, *TTE_1* has integrated to the running synchronized global time.

Conversely, if *TTE_1* detects no set or an insufficient set of time-triggered Ethernet devices, it will aim to startup the system itself. An insufficient number of devices operational is perceived by *TTE_1* by not receiving a protocol control frame with a sufficiently high number of membership bits set in the *pcf_membership_new* field for a sufficiently long duration after entering *SM_INTEGRATE* state.

NOTE: It is assumed that when the state machine is started, the underlying Ethernet functionality is already operational. Hence, it will not be the case that the state machine enters *SM_INTEGRATE* state and receives a timeout indicating that there is no communication on the network because the underlying Ethernet functionality of the component is not yet ready to receive Ethernet frames from the network.

9.2.1.2 Transition Summary

Table X.1 - SM_INTEGRATE state transition summary

	SM_INTEGRATE								
	Guard			Command					
	RX PCF	local_timer	pcf_membership_new	next state	TX_PCF	local_timer	local_clock	local_integration_cycle	local_sync_membership
1		TO		SM_UNSYNC	CS	sm_coldstart_timeout			
2	CA			SM_WAIT_4_CYCLE_START_CS		ca_offset			
3	IN		w(.) >= sm_integrate_to_sync_threshold	SM_SYNC		OFF	smc_scheduled_pit	pcf_integration_cycle	pcf_membership_new
	Setting local_timer to a new value ca_offset will delete any pending timeout condition TO (in case reception of a CA PCF happens at the same time as the timeout condition TO). Setting local_timer to OFF will de-activate the timer and delete any pending timeout condition TO (in case reception of the IN PCF happens at the same time as the timeout condition TO).								

9.2.2 *SM_UNSYNC* State

9.2.2.1 Description

In the *SM_UNSYNC* state, the synchronization master, say *TTE_1*, periodically transmits coldstart (CS) frames. The CS frame is a signal to the time-triggered Ethernet network that *TTE_1* is a coldstart master and is available to start a new synchronized global time.

When the synchronization master *TTE_1* receives a CS frame it transits to *SM_FLOOD* state, in which it acknowledges the received CS frame. This transition to *SM_FLOOD* will be taken if one of the following conditions are met:

1. The SM is configured for a high-integrity time-triggered Ethernet configuration and the received CS frame stems from a synchronization master other than *TTE_1*

2. The SM is configured for a standard-integrity time-triggered Ethernet in which case all CS frames are accepted (in contrast to the item above, the SM will also accept CS frames sent by itself).

9.2.2.2 Transition Summary

Table X.2 - SM_UNSYNC state transition summary

	SM_UNSYNC										
	Guard				Command						
	RX PCF	local_timer	high_integrity	pcf_membership_new	next state	TX_PCF	local_timer	local_clock	local_integration_cycle	local_sync_membership	flood_aux_state
1		TO				CS	sm_coldstart_timeout				
2	CS		TRUE	.[membership_position] = 0	SM_FLOOD		cs_offset				WAIT_AFTER_CS_RX
3	CS		FALSE		SM_FLOOD		cs_offset				WAIT_AFTER_CS_RX
4	CA				SM_WAIT_4_CYCLE_START_CS		ca_offset				
5	IN			w(.) >= sm_unsync_to_sync_threshold	SM_SYNC		OFF	smc_scheduled_pit	pcf_integration_cycle	pcf_membership_new	

9.2.3 *SM_FLOOD* State

9.2.3.1 Description

In the *SM_FLOOD* state a previously received CS frame will be acknowledged by sending a coldstart acknowledge (CA) frame. If multiple CS frames are received, only the last CS frame received will be acknowledged. After the transmission of a CA frame the synchronization master will wait to receive a CA frame back within a specifiable interval after the round-trip delay of the CA frame through the network. When the synchronization master receives a CA frame in this interval it will transit to *SM_WAIT_4_CYCLE_START_CS*. However, if the synchronization master receives another CS frame after the transmission of the CA frame and before it receives a CA frame back, it will re-enter the *SM_FLOOD* state, thereby ignoring all previous actions in *SM_FLOOD*. The CS frame reception in *SM_FLOOD* state is, thus, history-less.

9.2.3.2 Transition Summary

Table X.3 - SM_FLOOD state transition summary

	SM_FLOOD						
	Guard			Command			
	RX PCF	local_timer	flood_aux_state	next state	TX_PCF	local_timer	flood_aux_state
1		TO	WAIT_AFTER_CS_RX		CA	smc_scheduled_pit - ca_acceptance_window / 2	WAIT_AFTER_CA_TX
2		TO	WAIT_AFTER_CA_TX			ca_acceptance_window	ACCEPT_CA_RX
3		TO	ACCEPT_CA_RX	SM_UNSYNC		sm_coldstart_timeout	
4	CS					cs_offset	WAIT_AFTER_CS_RX
5	CA		ACCEPT_CA_RX	SM_WAIT_4_CYCLE_START_CS		ca_offset	

9.2.4 *SM_WAIT_4_CYCLE_START_CS* State

9.2.4.1 Description

The *SM_WAIT_4_CYCLE_START_CS* state is an intermediate state in which the synchronization master waits for a configurable time (the *ca_offset*) before it enters the synchronous operation phase by sending the first integration frame (IN frame).

9.2.4.2 Transition Summary

Table X.4 - SM_WAIT_4_CYCLE_START_CS state transition summary

	SM_WAIT_4_CYCLE_START_CS						
	Guard		Command				
	RX PCF	local_timer	next state	TX PCF	local_timer	local_integration_cycle	flood_aux_state
1		TO	SM_TENTATIVE_SYNC	IN		initial_integration_cycle	
2	CS		SM_FLOOD		cs_offset		WAIT_AFTER_CS_RX
3	CA				ca_offset		

9.2.5 *SM_TENTATIVE_SYNC* State

9.2.5.1 Description

The *SM_TENTATIVE_SYNC* state is a synchronous state; hence the clock synchronization service and the clique detection services are running. The clock synchronization service ensures that the *local_clock* and *local_integration_cycle* are synchronously updated in the synchronization masters, synchronization clients, and compression masters in the time-triggered Ethernet network. The clique detection services maintain the *local_sync_membership* and *local_async_membership* in order to identify the loss of synchronization of the time-triggered Ethernet device executing the clique detection service.

The *SM_TENTATIVE_SYNC* state is the “weakest” of the synchronous states. A synchronization master will transit to the next “better” synchronous state when the number of bits set in its *local_sync_membership* bit vector reaches or exceeds the *sm_tentative_sync_threshold_sync* threshold, which is configured offline.

NOTE: The synchronization master will remain in *SM_TENTATIVE_SYNC* state for the duration of the roundtrip of the first IN frame transmitted after coldstart: with the transition from *WAIT_4_CYCLE_START_CS* to *SM_TENTATIVE_SYNC* state the synchronization master sends the first IN frame (when *local_clock* = 0). The individual IN frames sent from different synchronization masters are then compressed in the compression master and the compressed IN frame is sent back to the synchronization masters. Depending on the number of bits set in the *pcf_membership_new* field in the compressed IN frame, the synchronization masters will either transit back to an unsynchronized state or forward to the *SM_SYNC* state.

In the *SM_TENTATIVE_SYNC* state the dispatch of TT-transferred frames can be enabled/disabled via configuration.

9.2.5.2 Transition Summary

The transition summary uses the function *pre(arg)* in line 7. This is used to elevate priority of *local_time* showing value *arg* over all other guards (in particular over a PCF being received at *local_time arg*) that also evaluate *local_time* to the same value *arg*. This changes the default priority defined in 9.1 for the respective guard.

In line 7 pre(*smc_scheduled_pit - acceptance_window*/2) is used to update *local_integration_cycle* just at the start of the acceptance window ensuring that an IN PCF received exactly at the start of the acceptance window will be checked against the new value at line 8. A CA PCF being received at *local_time smc_scheduled_pit - acceptance_window*/2 will still be handled with higher priority, though. I.e., in this latter case *local_integration_cycle* will be set to 0 and the transition to *SM_WAIT_4_CYCLE_START_CS* will be taken while line 7 will not be executed.

Table X.5 - SM_TENTATIVE_SYNC state transition summary

	SM_TENTATIVE_SYNC																
	Guard						Command										
	RX PCF	local_clock	pcf_integration_cycle	local_sync_membership	best_pcf.membership	PCF channel	next state	TX PCF	local_timer	local_clock	local_integration_cycle	local_sync_membership	local_async_membership	best_pcf.membership	best_pcf.permanence	clock_corr
1		smc_scheduled_pit + acceptance_window/2		w(.) < sm_tentative_sync_threshold_sync			SM_UNSYNC		sm_restart_timeout	0	0	0	0	0	0	
2		smc_scheduled_pit + acceptance_window/2		w(.) >= sm_tentative_sync_threshold_sync			SM_SYNC							0	0	FTA(best_pcf)
3	CA						SM_WAIT_4_CYCLE_START_CS		ca_offset	0	0	0	0	0	0	
4	IN	< smc_scheduled_pit - acceptance_window/2											.[ae]]= pcf_membership_new			
5	IN	>= smc_scheduled_pit - acceptance_window/2	!= local_integration_cycle										.[ae]]= pcf_membership_new			
6	IN	>= smc_scheduled_pit - acceptance_window/2 AND <= smc_scheduled_pit + acceptance_window/2	== local_integration_cycle	w(.) <= w(pcf_membership_new)								pcf_membership_new				
7		pre(smc_scheduled_pit - acceptance_window/2)									(local_integration_cycle + 1) % max_integration_cycle					
8	IN	>= smc_scheduled_pit - acceptance_window/2 AND <= smc_scheduled_pit + acceptance_window/2	== local_integration_cycle		w(.[N]) <= w(pcf_membership_new)	N								.[N] = pcf_membership_new	.[N] = local_clock	

	startup and restart service
	asynchronous clique detection
	synchronous clique detection
	clock synchronization

9.2.6 *SM_SYNC* State

9.2.6.1 Description

The *SM_SYNC* state is a synchronous state; hence the clock synchronization service and the clique detection services are running. The clock synchronization service ensures that the *local_clock* and *local_integration_cycle* are synchronously updated in the synchronization masters, synchronization clients, and compression masters in the time-triggered Ethernet network. The clique detection services maintain the *local_sync_membership* and *local_async_membership* in order to identify the loss of synchronization of the time-triggered Ethernet device executing the clique detection service.

The *SM_SYNC* state is a synchronous state in which a synchronization master will reside for at least a configurable number (*num_stable_cycles*) of integration cycles. When the synchronization master has resided in the *SM_SYNC* state for *num_stable_cycles* it will transit to *SM_STABLE* state.

NOTE: The motivation of *SM_SYNC* is a “testing” phase for the synchronized global time which has just been established.

NOTE: In the configuration of time-triggered Ethernet network for tolerance of two faulty time-triggered Ethernet devices, the minimum number of bits that must be set in the *local_sync_membership* vector of a synchronization master (*sm_sync_threshold_sync*) is set to 1. Hence, in this configuration cliques will only be detected via the asynchronous and the relative clique detection mechanisms.

In the *SM_SYNC* state, the dispatch of TT-transferred frames can be enabled/disabled via configuration.

9.2.6.2 Transition Summary

The transition summary uses the function *pre(arg)* in lines 2, 9, and 14. This is used to elevate priority of *local_time* showing value *arg* over all other guards (in particular over a PCF being received at *local_time arg*) that also evaluate *local_time* to the same value *arg*. This changes the default priority defined in 9.1 for the respective guard.

In lines 2 and 9 this is used to execute asynchronous clique detection at *local_clock* 0 (which may result in taking the transition to *SM_UNSYNC* state or just an update of *local_async_membership[ae]* and *ae*) before dispatching an IN PCF at *local_time* 0 (line 11) and also before adding any out-of-schedule IN PCF received at *local_time* 0 to the respective *local_async_membership* variable (line 10). A CA PCF being received at *local_time* 0 will still be handled with higher priority, though; i.e., in this latter case, the transition to *SM_WAIT_4_CYCLE_START_CS* will be taken while neither line 2 nor line 9 will be executed.

In line 14 pre(*smc_scheduled_pit* - *acceptance_window*/2) is used to re-set *local_sync_membership* and to advance *local_integration_cycle* just at the start of the acceptance window ensuring that an IN PCF received exactly at the start of the acceptance window will be checked against the new value of *local_integration_cycle* and, if in-schedule, will be added to *local_sync_membership* only after re-set. Again a CA PCF being received at *local_time smc_scheduled_pit* - *acceptance_window*/2 will be handled with higher priority, though. I.e., in this latter case the transition to *SM_WAIT_4_CYCLE_START_CS* will be taken while line 14 will not be executed.

Table X.6 - SM_SYNC state transition summary

	SM_SYNC																					
	Guard								Command													
	RX PCF	local_clock	pcf_integration_cycle	local_sync_membership	local_async_membership	stable_cycle_ctr	best_pcf.membership	PCF channel	next state	TX PCF	local_timer	local_clock	local_integration_cycle	local_sync_membership	local_async_membership	stable_cycle_ctr	ae	best_pcf.membership	best_pcf.permanence	clock_corr
1		0								IN											
2		pre(0)			w(.) >= sm_sync_threshold_async OR w(.) >= w(local_sync_membership)				SM_UNSYNC		sm_restart_timeout	0	0	0	0	0	0	0	0	0	0
3		smc_scheduled_pit + acceptance_window/2		w(.) < sm_sync_threshold_sync AND w(.) > 0					SM_UNSYNC		sm_restart_timeout	0	0	0	0	0	0	0	0	0
4		smc_scheduled_pit + acceptance_window/2		w(.) = 0					SM_INTEGRATE		sm_restart_timeout	0	0	0	0	0	0	0	0	0
5		smc_scheduled_pit + acceptance_window/2		w(.) >= sm_sync_threshold_sync		< num_stable_cycles										+= 1		0	0	FTA(best_pcf)	
6		smc_scheduled_pit + acceptance_window/2		w(.) >= sm_sync_threshold_sync		>= num_stable_cycles			SM_STABLE							0		0	0	FTA(best_pcf)	
7		smc_scheduled_pit + clock_corr_delay										+= clock_corr									0
8	CA								SM_WAIT_4_CYCLE_START_CS		ca_offset	0	0	0	0	0	0	0	0	0	0
9		pre(0)		w(.) > w(local_async_membership)	w(.) < sm_sync_threshold_async										[ae] = 0		[ae + 1)%2				
10	IN	< smc_scheduled_pit - acceptance_window/2 OR > smc_scheduled_pit + acceptance_window/2													[ae] = pcf_membership_new						
11	IN	>= integration_cycle_duration - acceptance_window/2													[(ae+1)%2] = pcf_membership_new						
12	IN	>= smc_scheduled_pit - acceptance_window/2 AND <= smc_scheduled_pit + acceptance_window/2	!= local_integration_cycle												[ae] = pcf_membership_new						
13	IN	>= smc_scheduled_pit - acceptance_window/2 AND <= smc_scheduled_pit + acceptance_window/2	== local_integration_cycle	w(.) <= w(pcf_membership_new)										pcf_membership_new						
14		pre(smc_scheduled_pit - acceptance_window/2)											(local_integration_cycle + 1) % max_integration_cycle	0							
15	IN	>= smc_scheduled_pit - acceptance_window/2 AND <= smc_scheduled_pit + acceptance_window/2	== local_integration_cycle				w(.[N]) <= w(pcf_membership_new)	N											.[N] = pcf_membership_new	.[N] = local_clock	

9.2.7 *SM_STABLE* State

9.2.7.1 Description

The *SM_STABLE* state is a synchronous state; hence the clock synchronization service and the clique detection services are running. The clock synchronization service ensures that the *local_clock* and *local_integration_cycle* are synchronously updated in the synchronization masters, synchronization clients, and compression masters in the time-triggered Ethernet network. The clique detection services maintain the *sm_sync_membership* and *sm_async_membership* in order to identify the loss of synchronization of the time-triggered Ethernet device executing the clique detection service.

9.2.7.2 Transition Summary

Table X.7 - SM_STABLE state transition summary

	SM_STABLE																			
	Guard									Command										
	RX PCF	local_clock	pcf_integration_cycle	local_sync_membership	local_async_membership	stable_cycle_ctr	best_pcf.membership	PCF channel	stable_ca_enabled	next state	TX PCF	local_timer	local_clock	local_integration_cycle	local_sync_membership	local_async_membership	stable_cycle_ctr	ae	best_pcf.membership	best_pcf.permanence	clock_corr
1		0									IN										
2		pre(0)			w(.) >= sm_stable_threshold_async OR (w(.) > 0 AND w(.) >= w(local_sync_membership))					SM_INTEGRATE		sm_restart_timtout	0	0	0	0	0	0	0	0	0
3		smc_scheduled_pit + acceptance_window/2		w(.) < sm_stable_threshold_sync		>= num_unstable_cycles				SC_INTEGRATE		sm_restart_timeout	0	0	0	0	0	0	0	0	0
4		smc_scheduled_pit + acceptance_window/2		w(.) < sm_stable_threshold_sync		< num_unstable_cycles											+= 1		0	0	FTA(best_pcf)
5		smc_scheduled_pit + acceptance_window/2		w(.) >= sc_stable_threshold_sync													0		0	0	FTA(best_pcf)
6		smc_scheduled_pit + clock_corr_delay											+= clock_corr								0
7	CA								TRUE	SM_WAIT_4_CYCLE_START_CS		ca_offset	0	0	0	0	0	0	0	0	0
8		pre(0)		w(.) > w(local_async_membership) OR w(local_async_membership) = 0	w(.) < sm_stable_threshold_async																
9	IN	< smc_scheduled_pit - acceptance_window/2 OR > smc_scheduled_pit + acceptance_window/2														[ae] = 0		[ae + 1)%2			
10	IN	>= integration_cycle_duration - acceptance_window/2														[ae] = pcf_membership_new					
11	IN	>= smc_scheduled_pit - acceptance_window/2 AND <= smc_scheduled_pit + acceptance_window/2	!= local_integration_cycle													[(ae+1)%2] = pcf_membership_new					
12	IN	>= smc_scheduled_pit - acceptance_window/2 AND <= smc_scheduled_pit + acceptance_window/2	== local_integration_cycle	w(.) <= w(pcf_membership_new)																	
13		pre(smc_scheduled_pit - acceptance_window/2)												(local_integration_cycle + 1) % max_integration_cycle	pcf_membership_new	0					
14	IN	>= smc_scheduled_pit - acceptance_window/2 AND <= smc_scheduled_pit + acceptance_window/2	== local_integration_cycle				w[.[N]] <= w(pcf_membership_new)	N											[N] = pcf_membership_new	[N] = local_clock	

NOTE: Table X.8, which contained an optional state, was removed and does not appear in this document revision.

9.3 Synchronization Client Protocol State Machine

Figure 29 depicts the protocol state machine executed in a synchronization client. The synchronization client differentiates between one unsynchronized state and two synchronized states. The unsynchronized state is *SC_INTEGRATE* state. Synchronized states include *SC_SYNC* state and *SC_STABLE* state.

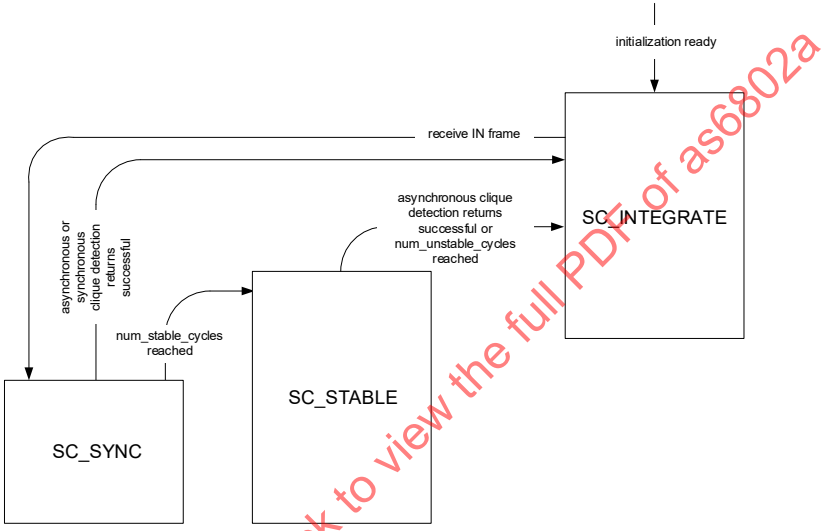


Figure 29 - Synchronization client protocol state machine

9.3.1 *SC_INTEGRATE* State

9.3.1.1 Description

The *SC_INTEGRATE* state is the first state entered after successful power-on of a time-triggered Ethernet device, *TTE_1*, say, that realizes the synchronization client functionality. The main purpose of this state is to identify whether there are already synchronized time-triggered Ethernet devices up and running. If there is indeed a set of time-triggered Ethernet devices communicating, *TTE_1* will receive protocol control frames from this set. Depending on the number of membership bits set in the *pcf_membership_new field*, *TTE_1*, may decide to transit to a synchronized state. In this case we say that *TTE_1* has integrated to the running synchronized global time.

9.3.1.2 Transition Summary

Table X.9 - SC_INTEGRATE state transition summary

SC_INTEGRATE					
Guard		**Command**			
RX PCF	pcf_membership_new	next state	local_clock	local_integration_cycle	local_sync_membership
IN	w(.) >= sc_integrate_to_sync_threshold	SC_SYNC	smc_scheduled_pit	pcf_integration_cycle	pcf_membership_new

9.3.2 *SC_SYNC* State

9.3.2.1 Description

The *SC_SYNC* state is a synchronous state; hence the clock synchronization service and the clique detection services are running. The clock synchronization service ensures that the *local_clock* and *local_integration_cycle* are synchronously updated in the synchronization masters, synchronization clients, and compression masters in the time-triggered Ethernet network. The clique detection services maintain the *local_sync_membership* and *local_async_membership* in order to identify the loss of synchronization of the time-triggered Ethernet device executing the clique detection service.

The *SC_SYNC* state is a synchronous state in which a synchronization client will reside for at least a configurable number (*num_stable_cycles*) of integration cycles. When the synchronization client has resided in the *SC_SYNC* state for *num_stable_cycles* it will transit to *SC_STABLE*.

NOTE: The motivation of *SC_SYNC* is a “testing” phase for the synchronized global time which has just been established.

In the *SC_SYNC* state, the dispatch of TT-transferred frames can be enabled/disabled via configuration.

9.3.2.2 Transition Summary

Table X.10 - SC_SYNC state transition summary

		SC_SYNC																
		Guard							Command									
	RX PCF	local_clock	pcf_integration_cycle	local_sync_membership	local_async_membership	stable_cycle_ctr	best_pcf.membership	PCF channel	next state	local_clock	local_integration_cycle	local_sync_membership	local_async_membership	stable_cycle_ctr	ae	best_pcf.membership	best_pcf.permanence	clock_corr
1		pre(0)			w(.) >= sc_sync_threshold_async				SC_INTEGRATE	0	0	0	0	0	0	0	0	0
2		smc_scheduled_pit + acceptance_window/2		w(.) < sc_sync_threshold_sync					SC_INTEGRATE	0	0	0	0	0	0	0	0	0
3		smc_scheduled_pit + acceptance_window/2		w(.) >= sc_sync_threshold_sync		< num_stable_cycles								+= 1		0	0	FTA(best_pcf)
4		smc_scheduled_pit + acceptance_window/2		w(.) >= sc_sync_threshold_sync		>= num_stable_cycles			SC_STABLE					0		0	0	FTA(best_pcf)
5		smc_scheduled_pit + clock_corr_delay								+= clock_corr								0
6		pre(0)			w(.) < sc_sync_threshold_async										(ae + 1)%2			
7	IN	< smc_scheduled_pit - acceptance_window/2 OR > smc_scheduled_pit + acceptance_window/2											.[ae] != pcf_membership_new					
8	IN	>= integration_cycle_duration - acceptance_window/2											.[(ae+1)%2] != pcf_membership_new					
9	IN	>= smc_scheduled_pit - acceptance_window/2 AND <= smc_scheduled_pit + acceptance_window/2	!= local_integration_cycle										.[ae] != pcf_membership_new					
10	IN	>= smc_scheduled_pit - acceptance_window/2 AND <= smc_scheduled_pit + acceptance_window/2	== local_integration_cycle	w(.) <= w(pcf_membership_new)								pcf_membership_new						
11		pre(smc_scheduled_pit - acceptance_window/2)									(local_integration_cycle + 1) % max_integration_cycle	0						
12	IN	>= smc_scheduled_pit - acceptance_window/2 AND <= smc_scheduled_pit + acceptance_window/2	== local_integration_cycle				w(.[N]) <= w(pcf_membership_new)	N								.[N] = pcf_membership_new	.[N] = local_clock	

9.3.3 *SC_STABLE* State

9.3.3.1 Description

The *SC_STABLE* state is a synchronous state; hence the clock synchronization service and the clique detection services are running. The clock synchronization service ensures that the *local_clock* and *local_integration_cycle* are synchronously updated in the synchronization masters, synchronization clients, and compression masters in the time-triggered Ethernet network. The clique detection services maintain the *sm_sync_membership* and *sm_async_membership* in order to identify the loss of synchronization of the time-triggered Ethernet device executing the clique detection service.

9.3.3.2 Transition Summary

Table X.11 - SC_STABLE state transition summary

	SC_STABLE																		
	Guard								Command										
	RX PCF	local_clock	pcf_integration_cycle	local_sync_membership	local_async_membership	stable_cycle_ctr	best_pcf.membership	PCF channel	next state	local_clock	local_integration_cycle	local_sync_membership	local_async_membership	stable_cycle_ctr	ae	best_pcf.membership	best_pcf.permanence	clock_corr
1		pre(0)			w(.) >= sc_stable_threshold_async				SC_INTEGRATE	0	0	0	0	0	0	0	0	0
2		smc_scheduled _pit + acceptance_window/2		w(.) < sc_stable_threshold_sync		>= num_unstable_cycles			SC_INTEGRATE	0	0	0	0	0	0	0	0	0
3		smc_scheduled _pit + acceptance_window/2		w(.) < sc_stable_threshold_sync		< num_unstable_cycles								+= 1		0	0	FTA(best_pcf)
4		smc_scheduled _pit + acceptance_window/2		w(.) >= sc_stable_threshold_sync										0		0	0	FTA(best_pcf)
5		smc_scheduled_pit + clock_corr_delay								+= clock_corr								0
6		pre(0)			w(.) < sc_sync_threshold_async										(ae + 1)%2			
7	IN	< smc_scheduled _pit - acceptance_window/2 OR > smc_scheduled _pit + acceptance_window/2																
8	IN	>= integration_cycle_duration - acceptance_window/2																
9	IN	>= smc_scheduled _pit - acceptance_window/2 AND <= smc_scheduled _pit + acceptance_window/2	!= local_integration_cycle															
10	IN	>= smc_scheduled _pit - acceptance_window/2 AND <= smc_scheduled _pit + acceptance_window/2	== local_integration_cycle	w(.) <= w(pcf_membership_new)								pcf_membership_new						
11		pre(smc_scheduled _pit - acceptance_window/2)									(local_integration_cycle + 1) % max_integration_cycle	0						
12	IN	>= smc_scheduled _pit - acceptance_window/2 AND <= smc_scheduled _pit + acceptance_window/2	== local_integration_cycle				w(.[N]) <= w(pcf_membership_new)	N								.[N] = pcf_membership_new	.[N] = local_clock	

9.4 Compression Master Protocol State Machine for High-Integrity Synchronization Masters

Figure 30 depicts the protocol state machine executed in a compression master for high-integrity synchronization masters. The compression master differentiates unsynchronized states and synchronized states. Unsynchronized states include *CM_INTEGRATE* state, *CM_WAIT_4_CYCLE_START* state, and *CM_UNSYNC* state. Synchronized states include *CM_TENTATIVE_SYNC* state, *CM_SYNC* state, and *CM_STABLE* state.

When the compression master configuration operates with high-integrity synchronization masters, it dispatches:

a. All coldstart frames, only when the compression master is either in *CM_UNSYNC* state, *CM_WAIT_4_CYCLE_START*, or *CM_TENTATIVE_SYNC* state; it discards coldstart frames received in other states.

b. All received coldstart acknowledge frames.

c. All received integration frames as compressed integration frames.

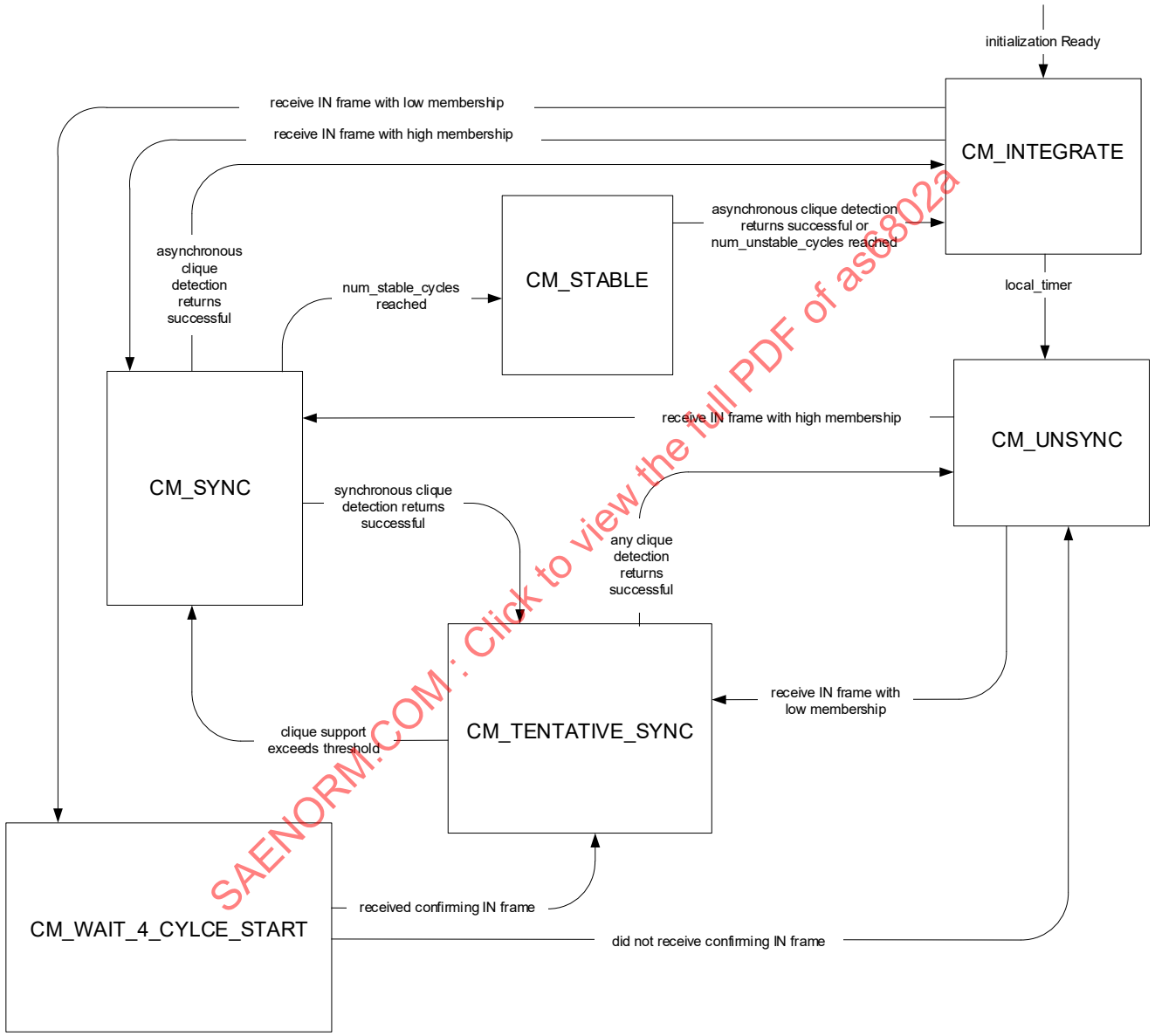


Figure 30 - Compression master protocol state machine for high-integrity synchronization masters

9.4.1 *CM_INTEGRATE* State

9.4.1.1 Description

The *CM_INTEGRATE* state is the first state entered after successful power on of a time-triggered Ethernet device, *TTE_1*, say, configured as compression master. After entering this state, *TTE_1* identifies whether there are already synchronized time-triggered Ethernet devices running. If there are time-triggered Ethernet devices synchronized, *TTE_1* will receive protocol control frames from this set. Depending on the number of membership bits set in the *pcf_membership_new* field, *TTE_1* may decide to transit to a synchronized state. If this occurs, we say that *TTE_1* has integrated to the running synchronized global time.

Conversely, there may not be a set of operational time-triggered Ethernet devices, or there may be an insufficient number of operational time-triggered Ethernet devices. In these cases, *TTE_1* would not receive a protocol control frame with a sufficiently high number of membership bits set in the *pcf_membership_new* field for a sufficiently long duration, and therefore not synchronize to global time.

9.4.1.2 Transition Summary

Table X.12 - CM_INTEGRATE state transition summary

	CM_INTEGRATE								
	Guard			Command					
	RX PCF	local_timer	pcf_membership_new	next state	TX_PCF	local_timer	local_clock	local_integration_cycle	local_sync_membership
1	CA				CA				
2	IN				IN				
3		TO		CM_UNSYNC					
4	IN		w(.) >= cm_integrate_to_wait_threshold AND w(.) < cm_integrate_to_sync_threshold	CM_WAIT_4_CYCLE_START		OFF	cm_scheduled_pit	pcf_integration_cycle	pcf_membership_new
5	IN		w(.) >= cm_integrate_to_sync_threshold	CM_SYNC		OFF	cm_scheduled_pit	pcf_integration_cycle	pcf_membership_new
	Setting local_timer to OFF will de-activate the timer and delete any pending timeout condition TO (in case reception of a PCF happens at the same time as the timeout condition TO).								

9.4.2 *CM_WAIT_4_CYCLE_START* State

9.4.2.1 Description

In the *CM_WAIT_4_CYCLE_START* state, the compression master verifies whether the integration frame that it used to integrate was correct or not. This verification is done by checking whether it receives an in-schedule integration frame in the integration cycle succeeding the integration. If it receives such an integration frame it transits to the *CM_TENTATIVE_SYNC* state. If not then it transits to *CM_UNSYNC* state. When it receives an integration frame with a sufficiently high number of bits in the *pcf_membership_new* field, while it is residing in the *CM_WAIT_4_CYCLE_START* state and this integration frame is received out-of-schedule it resynchronizes to this newly received integration frame.

9.4.2.2 Transition Summary

Table X.13 - CM_WAIT_4_CYCLE_START state transition summary

	CM_WAIT_4_CYCLE_START											
	Guard					Command						
	RX PCF	local_clock	pcf_integration_cycle	pcf_membership_new	num_cycles	next state	TX_PCF	num_cycles	local_clock	local_integration_cycle	local_sync_membership	clock_corr
1	CS						CS					
2	CA						CA					
3	IN						IN					
4		cm_scheduled _pit + acceptance_window/2			1	CM_UNSYNC		0	0	0		
5	IN	>= cm_scheduled _pit - acceptance_window/2 AND <= cm_scheduled _pit + acceptance_window/2	== local_integration_cycle		1	CM_TENTATIVE_SYNC		0	cm_scheduled_pit		pcf_membership_new	
6	IN	>= cm_scheduled _pit - acceptance_window/2 AND <= cm_scheduled _pit + acceptance_window/2	== local_integration_cycle	w(.) >= local_sync_membership	0						pcf_membership_new	cm_scheduled_pit - local_clock
7	IN	< cm_scheduled _pit - acceptance_window/2 OR > cm_scheduled _pit + acceptance_window/2		w(.) >= cm_wait_threshold_sync				0	cm_scheduled_pit	pcf_integration_cycle	pcf_membership_new	0
8	IN	>= cm_scheduled _pit - acceptance_window/2 AND <= cm_scheduled _pit + acceptance_window/2	!= local_integration_cycle	w(.) >= cm_wait_threshold_sync				0	cm_scheduled_pit	pcf_integration_cycle	pcf_membership_new	0
9		cm_scheduled_pit + clock_corr_delay							+= clock_corr			0

9.4.3 *CM_UNSYNC* State

9.4.3.1 Description

In the *CM_UNSYNC* state, the compression master is unsynchronized and waits to receive a compressed integration frame with a sufficient number of bits set in the *pcf_membership_new* field.

9.4.3.2 Transition Summary

Table X.14 - CM_UNSYNC state transition summary

	CM_UNSYNC						
	Guard		Command				
	RX PCF	pcf_membership_new	next state	TX_PCF	local_clock	local_integration_cycle	local_sync_membership
1	CS			CS			
2	CA			CA			
3	IN			IN			
4	IN	w(.) >= cm_unsync_to_sync_threshold	CM_SYNC		cm_scheduled_pit	pcf_integration_cycle	pcf_membership_new
5	IN	w(.)< cm_unsync_to_sync_threshold AND w(.) >= cm_unsync_to_tentative_sync_threshold	CM_TENTATIVE_SYNC		cm_scheduled_pit	pcf_integration_cycle	pcf_membership_new

9.4.4 *CM_TENTATIVE_SYNC* State

9.4.4.1 Description

In the *CM_TENTATIVE_SYNC* state the compression master is synchronized to a set of synchronization masters. When it perceives an increase in the synchronization quality it will transit to *CM_SYNC* state. On the other hand, when it diagnoses a degradation in the synchronization quality or even a synchronization loss, it transits back to *CM_UNSYNC* state.

9.4.4.2 Transition Summary

Table X.15 - CM_TENTATIVE_SYNC state transition summary

	CM_TENTATIVE_SYNC												
	Guard					Command							
	RX PCF	local_clock	pcf_integration_cycle	local_sync_membership	local_async_membership	next state	TX PCF	local_clock	local_integration_cycle	local_sync_membership	local_async_membership	ae	clock_corr
1	CS						CS						
2	CA						CA						
3	IN						IN						
4		pre(cm_scheduled_pit + acceptance_window/2)			w(.) >= cm_tentative_sync_threshold_async	CM_UNSYNC		0	0	0	0	0	0
5					w(.) < cm_tentative_sync_threshold_async						.[ae] = 0	(ae + 1)%2	
6		cm_scheduled_pit + acceptance_window/2		w(.) < cm_tentative_sync_threshold_sync		CM_UNSYNC		0	0	0	0	0	0
7				w(.) >= cm_tentative_sync_threshold_sync AND w(.) < cm_tentative_sync_to_sync_threshold									
8				w(.) >= cm_tentative_sync_to_sync_threshold		CM_SYNC							
9		cm_scheduled_pit + clock_corr_delay						+= clock_corr					0
10	IN	< cm_scheduled_pit - acceptance_window/2 OR > cm_scheduled_pit + acceptance_window/2									.[ae] |= pcf_membership_new		
11	IN	>= cm_scheduled_pit - acceptance_window/2 AND <= cm_scheduled_pit + acceptance_window/2	!= local_integration_cycle								.[ae] |= pcf_membership_new		
12	IN	>= cm_scheduled_pit AND < cm_scheduled_pit + acceptance_window/2	!= local_integration_cycle								.[(ae+1)%2] |= pcf_membership_new		
13	IN	>= cm_scheduled_pit - acceptance_window/2 AND <= cm_scheduled_pit + acceptance_window/2	== local_integration_cycle	w(.) <= w(pcf_membership_new)						pcf_membership_new			
14		pre(cm_scheduled_pit - acceptance_window/2)							(local_integration_cycle + 1) % max_integration_cycle	0			
15	IN	>= cm_scheduled_pit - acceptance_window/2 AND <= cm_scheduled_pit + acceptance_window/2	== local_integration_cycle	w(.) <= w(pcf_membership_new)									cm_scheduled_pit - local_clock

9.4.5 *CM_SYNC* State

9.4.5.1 Description

In the *CM_SYNC* state, the compression master is synchronized. An increase in the synchronization quality may bring the compression master into the *CM_STABLE* state; a decrease in the quality into the *CM_INTEGRATE* state.

9.4.5.2 Transition Summary

Table X.16 - CM_SYNC state transition summary

	CM_SYNC																
	Guard						Command										
	RX PCF	local_clock	pcf_integration_cycle	local_sync_membership	local_async_membership	stable_cycle_ctr	next state	TX PCF	local_timer	local_clock	local_integration_cycle	local_sync_membership	local_async_membership	ae	stable_cycle_ctr	clock_corr
1	CA							CA								
2	IN							IN								
3		pre(cm_scheduled_pit + acceptance_window/2)			w(.) >= cm_sync_threshold_async		CM_INTEGRATE		cm_restart_timeout	0	0	0	0	0	0	0
4					w(.) < cm_sync_threshold_async								.[ae] = 0	(ae + 1)%2		
5				w(.) < cm_sync_threshold_sync			CM_TENTATIVE_SYNC								0	
6		cm_scheduled_pit + acceptance_window/2		w(.) >= cm_sync_threshold_sync		< num_stable_cycles									+= 1	
7				w(.) >= cm_sync_threshold_sync		>= num_stable_cycles	CM_STABLE								0	
8		cm_scheduled_pit + clock_corr_delay								+= clock_corr						0
9	IN	< cm_scheduled_pit - acceptance_window/2 OR > cm_scheduled_pit + acceptance_window/2											.[ae] |= pcf_membership_new			
10	IN	>= cm_scheduled_pit - acceptance_window/2 AND <= cm_scheduled_pit + acceptance_window/2	!= local_integration_cycle										.[ae] |= pcf_membership_new			
11	IN	>= cm_scheduled_pit AND < cm_scheduled_pit + acceptance_window/2	!= local_integration_cycle										.[(ae+1)%2] |= pcf_membership_new			
12	IN	>= cm_scheduled_pit - acceptance_window/2 AND <= cm_scheduled_pit + acceptance_window/2	== local_integration_cycle	w(.) <= w(pcf_membership_new)								pcf_membership_new				
13		pre(cm_scheduled_pit - acceptance_window/2)									(local_integration_cycle + 1) % max_integration_cycle	0				
14	IN	>= cm_scheduled_pit - acceptance_window/2 AND <= cm_scheduled_pit + acceptance_window/2	== local_integration_cycle	w(.) <= w(pcf_membership_new)												cm_scheduled_pit - local_clock

9.4.6 *CM_STABLE* State

9.4.6.1 Description

A compression master in *CM_STABLE* state has reached the highest level of synchronization quality. In this state, a compression master can be configured to sustain the temporary loss of all protocol control frames in the network.

9.4.6.2 Transition Summary

Table X.17 - CM_STABLE state transition summary

	CM_STABLE																
	Guard						Command										
	RX PCF	local_clock	pcf_integration_cycle	local_sync_membership	local_async_membership	stable_cycle_ctr	next state	TX PCF	local_timer	local_clock	local_integration_cycle	local_sync_membership	local_async_membership	ae	stable_cycle_ctr	clock_corr
1	CA							CA								
2	IN							IN								
3		pre(cm_scheduled_pit + acceptance_window/2)			w(.) >= cm_stable_threshold_async		CM_INTEGRATE		cm_restart_timeout	0	0	0	0	0	0	0
					w(.) < cm_stable_threshold_async								.[ae] = 0	(ae + 1)%2		
4		cm_scheduled_pit + acceptance_window/2		w(.) < cm_stable_threshold_sync		>= num_unstable_cycles	CM_INTEGRATE		cm_restart_timeout	0	0	0	0	0	0	0
5				w(.) < cm_stable_threshold_sync		< num_unstable_cycles									+= 1	
6				w(.) >= cm_stable_threshold_sync											0	
7		cm_scheduled_pit + clock_corr_delay								+= clock_corr						0
8	IN	< cm_scheduled_pit - acceptance_window/2 OR > cm_scheduled_pit + acceptance_window/2											.[ae] |= pcf_membership_new			
10	IN	>= cm_scheduled_pit - acceptance_window/2 AND <= cm_scheduled_pit + acceptance_window/2	!= local_integration_cycle										.[ae] |= pcf_membership_new			
9	IN	>= cm_scheduled_pit AND < cm_scheduled_pit + acceptance_window/2	!= local_integration_cycle										.[(ae+1)%2] |= pcf_membership_new			
11	IN	>= cm_scheduled_pit - acceptance_window/2 AND <= cm_scheduled_pit + acceptance_window/2	== local_integration_cycle	w(.) <= w(pcf_membership_new)								pcf_membership_new				
12		pre(cm_scheduled_pit - acceptance_window/2)									(local_integration_cycle + 1) % max_integration_cycle	0				
13	IN	>= cm_scheduled_pit - acceptance_window/2 AND <= cm_scheduled_pit + acceptance_window/2	== local_integration_cycle	w(.) <= w(pcf_membership_new)												cm_scheduled_pit - local_clock

9.5 Compression Master Protocol State Machine for Standard-Integrity Synchronization Masters

Figure 31 depicts the protocol state machine executed in a compression master for standard-integrity synchronization masters. The compression master differentiates between unsynchronized states and synchronized states. Unsynchronized states include *CM_INTEGRATE* state, *CM_UNSYNC* state, *CM_CA_ENABLED* state, and *CM_WAIT_4_IN* state. Synchronized states include *CM_SYNC* state and *CM_STABLE* state.

A compression master that operates with standard integrity synchronization masters relays coldstart frames when the compression master is in *CM_UNSYNC* state, and blocks coldstart frames in all other states.

A compression master that operates with standard integrity synchronization masters relays coldstart acknowledge frames when the compression master is in *CM_UNSYNC* state, provided that the value of the *pcf_membership_new* is sufficiently high or when the compression master is in *CM_CA_ENABLED* state and blocks coldstart acknowledge frames in all other states.

When operating with standard-integrity synchronization masters, a compression master relays integration frames in *CM_SYNC* state and in *CM_STABLE* state, but only if the integration frame triggered a transition from any of *CM_INTEGRATE* state, *CM_UNSYNC* state, or *CM_WAIT_4_IN* state to *CM_SYNC* state or the integration frame provides the clock synchronization process in *CM_SYNC* state or *CM_STABLE* state. The compression master blocks all other integration frames.

RX PCF *equals cm_compressed_pit*.

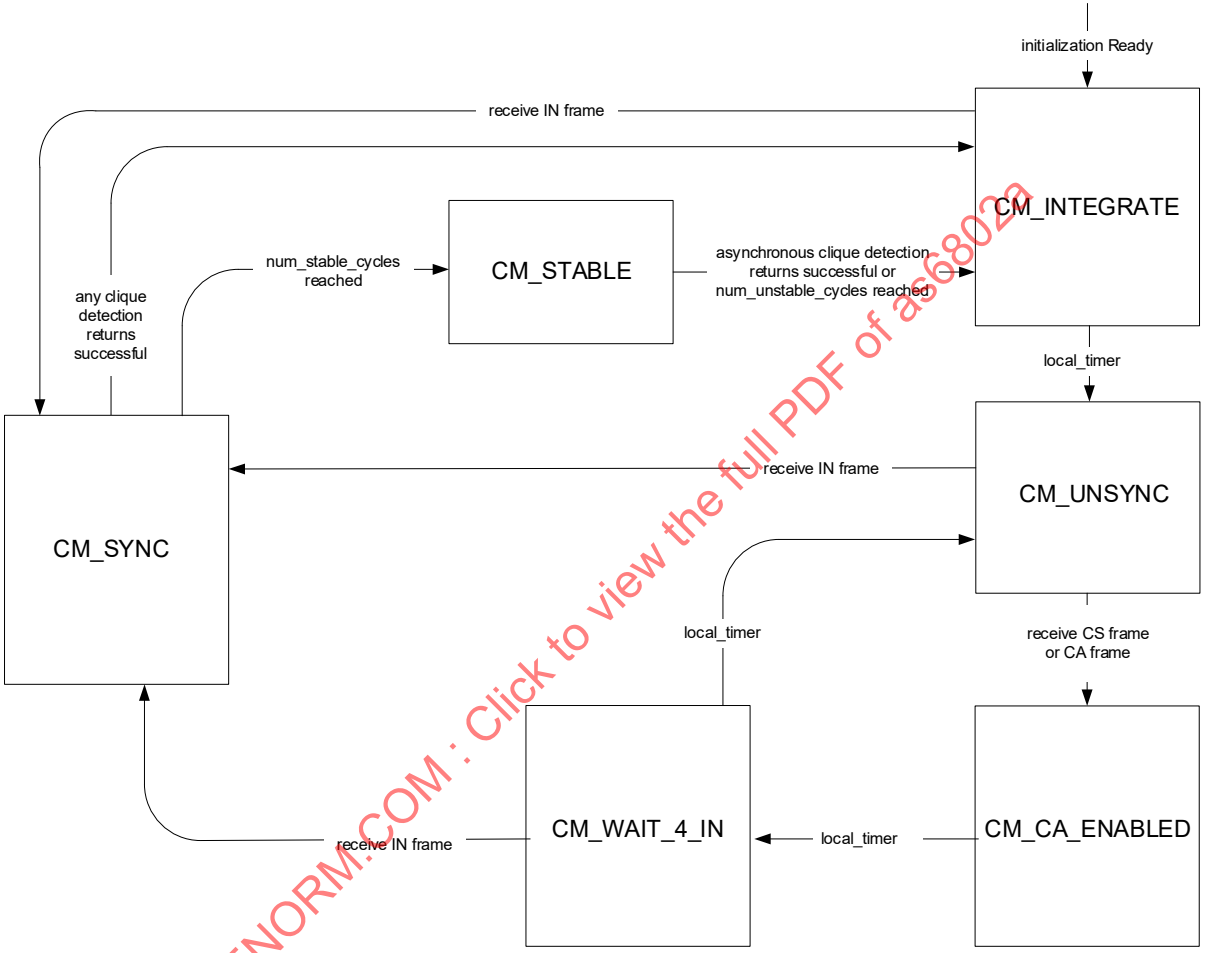


Figure 31 - Compression master protocol state machine for standard-integrity synchronization

9.5.1 *CM_INTEGRATE* State

9.5.1.1 Description

The *CM_INTEGRATE* state is the first state entered after successful power on of a time-triggered Ethernet device, *TTE_1*, say, configured as a compression master. After entering this state, *TTE_1* identifies whether there are already synchronized time-triggered Ethernet devices running. In the case that there are time-triggered Ethernet devices synchronized, *TTE_1* will receive protocol control frames from this set. Depending on the number of membership bits set in the *pcf_membership_new* field, *TTE_1* may decide to transit to a synchronized state. If this occurs, we say that *TTE_1* has integrated to the running synchronized global time.

Conversely, there may not be a set of time-triggered Ethernet devices operational, or there may be an insufficient number of time-triggered Ethernet devices operational. In these cases, *TTE_1* would not receive a protocol control frame with a sufficiently high number of membership bits set in the *pcf_membership_new* field for a sufficiently long duration, and therefore not synchronize to global time.

9.5.1.2 Transition Summary

Table X.18 - CM_INTEGRATE state transition summary

	CM_INTEGRATE							
	Guard			Command				
	RX PCF	local_timer	pcf_membership_new	next state	local_timer	local_clock	local_integration_cycle	local_sync_membership
1		TO		CM_UNSYNC				
2	IN		w(.) >= cm_integrate_to_sync_threshold	CM_SYNC	OFF	cm_scheduled_pit	pcf_integration_cycle	pcf_membership_new

9.5.2 *CM_UNSYNC* State

9.5.2.1 Description

In the *CM_UNSYNC* state, the compression master is unsynchronized and waits to receive a compressed integration frame with a sufficient number of bits set in the *pcf_membership_new* field. In this state, the compression master also reacts to the reception of a CS frame or a CA frame with a sufficiently high number of bits set in the *pcf_membership_new* field in order to guard the execution of the coldstart algorithm.

9.5.2.2 Transition Summary

Table X.19 - CM_UNSYNC state transition summary

	CM_UNSYNC							
	Guard		Command					
	RX PCF	pcf_membership_new	next state	TX_PCF	local_timer	local_clock	local_integration_cycle	local_sync_membership
1	CS		CM_CA_ENABLED	CS	cm_ca_enabled_timeout			
2	CA	w(.) >= cm_sync_threshold_sync	CM_CA_ENABLED	CA	cm_ca_enabled_timeout			
3	IN	w(.) >= cm_unsync_to_sync_threshold	CM_SYNC			cm_scheduled_pit	pcf_integration_cycle	pcf_membership_new

9.5.3 *CM_CA_ENABLED* State

9.5.3.1 Description

In the *CM_CA_ENABLED* state the compression master relays CA frames for a configured duration after which it transits to *CM_WAIT_4_IN* state.

9.5.3.2 Transition Summary

Table X.20 - CM_CA_ENABLED state transition summary

	CM_CA_ENABLED				
	Guard		Command		
	RX PCF	local_timer	next state	TX_PCF	local_timer
1	CA			CA	
2		TO	CM_WAIT_4_IN		cm_wait_4_in_timeout

9.5.4 *CM_WAIT_4_IN* State

9.5.4.1 Description

In the *CM_WAIT_4_IN* state the compression master waits for the reception of a compressed IN frame with a sufficiently high number of bits set in the *pcf_membership_new* field. When it receives such an IN frame, it integrates to *CM_SYNC* state, otherwise it re-enters the *CM_UNSYNC* state.