

TICTOC Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2013

S. Davari
A. Oren
Broadcom Corp.
M. Bhatia
P. Roberts
Alcatel-Lucent
L. Montini
Cisco Systems
October 22, 2012

Transporting Timing messages over MPLS Networks
draft-ietf-tictoc-1588overmpls-03

Abstract

This document defines the method for transporting Timing messages such as PTP and NTP over an MPLS network. The method allows for the easy identification of these PDUs at the port level to allow for port level processing of these PDUs in both LERs and LSRs.

The basic idea is to transport Timing messages inside dedicated MPLS LSPs. These LSPs only carry timing messages and possibly Control and Management packets, but they do not carry customer traffic.

Two methods for transporting Timing messages over MPLS are defined. The first method is to transport Timing messages directly over the dedicated MPLS LSP via UDP/IP encapsulation, which is suitable for MPLS networks. The second method is to transport Timing messages inside a PW via Ethernet encapsulation, which is suitable for both MPLS and MPLS-TP networks.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 25, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	6
2. Terminology	8
3. Problem Statement	9
4. Timing over MPLS Architecture	10
5. Dedicated LSPs for Timing messages	12
6. Timing over LSP Encapsulation	13
6.1. Timing over UDP/IP over MPLS Encapsulation	13
6.2. Timing over PW Encapsulation	13
6.3. Other Timing Encapsulation methods	14
7. Timing Message Processing	15
8. Protection and Redundancy	16
9. ECMP	17
10. PHP	18
11. Entropy	19
12. OAM, Control and Management	20
13. QoS Considerations	21
14. FCS Recalculation	22
15. UDP Checksum Correction	23
16. Routing extensions for Timing-aware Routers	24
17. Signaling Extensions for Creating Timing LSPs	25
18. Behavior of LER/LSR	26
18.1. Behavior of Timing-capable/aware LER	26
18.2. Behavior of Timing-capable/aware LSR	26
18.3. Behavior of non-Timing-capable/aware LSR	26
19. Other considerations	28
20. Security Considerations	29

21. Applicability Statement	30
22. Acknowledgements	31
23. IANA Considerations	32
24. References	33
24.1. Normative References	33
24.2. Informative References	33
Authors' Addresses	36

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119 [RFC2119].

When used in lower case, these words convey their typical use in common language, and are not to be interpreted as described in RFC2119 [RFC2119].

1. Introduction

The objective of Precision Time Protocol (PTP) and Network Timing Protocol (NTP) are to synchronize independent clocks running on separate nodes of a distributed system.

[IEEE] defines PTP messages for frequency, phase and time synchronization. The PTP messages include PTP PDUs over UDP/IP (Annex D and E of [IEEE]) and PTP PDUs over Ethernet (Annex F of [IEEE- 1588]). This document defines mapping and transport of the PTP messages defined in [IEEE] over MPLS/MPLS-TP networks. PTP defines several clock types: ordinary clocks, boundary clocks, end-to-end transparent clocks, and peer-to-peer transparent clocks. Transparent clocks require intermediate nodes to update correction field inside PTP message that reflects the transit time in the node.

[RFC5905] defines NTP messages for clock and time synchronization. The PTP messages (PDUs) are transported over UDP/IP. This document defines mapping and transport of the NTP messages defined in [RFC5905] over MPLS networks.

One key attribute of all of these Timing messages is that the Timestamp processing should occur as close as possible to the actual transmission and reception at the physical port interface. This targets optimal time and/or frequency recovery by avoiding variable delay introduced by queues internal to the clocks.

To facilitate the fast and efficient recognition of Timing messages at the port level when the Timing messages are carried over MPLS LSPs, this document defines the specific encapsulations that should be used. In addition, it can be expected that there will exist LSR/ LERs where only a subset of the physical ports will have the port-based Timing message processing capabilities. In order to ensure that the LSPs carrying Timing packets always enter and exit ports with this capability, routing extensions are defined to advertise this capability on a port basis and to allow for the establishment of LSPs that only transit such ports. While this path establishment restriction may be applied only at the LER Ingress and/or egress ports, it becomes more important when using transparent clock capable LSRs in the path.

Port based Timing message processing involves Timing message recognition. Once the Timing messages are recognized they can be modified based on the reception or transmission timestamp.

This document provides two methods for transporting Timing messages over MPLS. One is applicable to MPLS environment and the other one is applicable to both MPLS and MPLS-TP environment.

The solution involves transporting Timing messages over dedicated LSPs called Timing LSPs. These LSPs carry Timing messages and MAY carry Management and control messages, but not data plane client traffic. Timing LSPs can be established statically or via signaling. Extensions to control plane (OSPF, ISIS, etc) is required to enable routers to distribute their Timing processing capabilities over MPLS to other routers. However such extensions are outside the scope of this document.

Extensions to signaling protocols (e.g., RSVP-TE) are required for establishing PTP LSPs. However such extensions are outside the scope of this document.

While the techniques included herein allow for the establishment of paths optimized to include Time-stamping capable links, the performance of the Slave clocks is outside the scope of this document.

2. Terminology

1588: The timing and synchronization as defined by IEEE 1588.

NTP: The timing and synchronization protocol defined by IETF RFC-1305 and RFC-5905.

PTP: The timing and synchronization protocol used by 1588.

Master Clock: The source of 1588 timing to a set of slave clocks.

Master Port: A port on a ordinary or boundary clock that is in Master state. This is the source of timing toward slave ports.

Slave Clock: A receiver of 1588 timing from a master clock.

Slave Port: A port on a boundary clock or ordinary clock that is receiving timing from a master clock.

Ordinary Clock: A device with a single PTP port.

Transparent Clock. A device that measures the time taken for a PTP event message to transit the device and then updates the correctionField of the message with this transit time.

Boundary Clock: A device with more than one PTP port. Generally boundary clocks will have one port in slave state to receive timing and then other ports in master state to re-distribute the timing.

PTP LSP: An LSP dedicated to carry PTP messages

PTP PW: A PW within a PTP LSP that is dedicated to carry PTP messages.

CW: Pseudowire Control Word

LAG: Link Aggregation

ECMP: Equal Cost Multipath

CF: Correction Field, a field inside certain PTP messages (message type 0-3) that holds the accumulative transit time inside intermediate switches

3. Problem Statement

[IEEE] has defined methods for transporting PTP messages over Ethernet and IP networks. [RFC5905] has defined the method of transporting NTP messages over IP networks. There is a need to transport Timing messages over MPLS networks while supporting the Transparent Clock (TC), Boundary Clock (BC) and Ordinary Clock (OC) functionality in the LER and LSRs in the MPLS network.

There are multiple ways of transporting Timing over MPLS. However, there is a requirement to limit the possible encapsulation options to simplify the Timing message identification and processing required at the port level.

When Timing-awareness is needed, Timing messages should not be transported over LSPs or PWs that are carrying customer traffic because LSRs perform Label switching based on the top label in the stack. To detect Timing messages inside such LSPs require special hardware to do deep packet inspection at line rate. Even if such hardware exists, the payload can't be deterministically identified by LSRs because the payload type is a context of the PW label, and the PW label and its context are only known to the Edge routers (PEs/LERs); LSRs don't know what is a PW's payload (Ethernet, ATM, FR, CES, etc). Even if one restricts an LSP to only carry Ethernet PWs, the LSRs don't have the knowledge of whether PW Control Word (CW) is present or not and therefore can not deterministically identify the payload.

A generic method is defined in this document that does not require deep packet inspection at line rate, and can deterministically identify Timing messages. The generic method is applicable to MPLS and MPLS-TP networks.

4. Timing over MPLS Architecture

Timing messages are exchange between Timing ports on ordinary and boundary clocks. Boundary clocks terminate the Timing messages and act as master for other boundary clocks or for slave clocks. Transparent clocks do not terminate the Timing messages but they do modify the contents of the Timing messages as they transit across the transparent clock.

Master/Slave clock could be integrated in the LERs. An example is shown in Figure 1, where the LERs act as Ordinary Clock (OC) and are the initiating/terminating point for Timing messages. The ingress LER encapsulates the Timing messages in Timing LSP and the Egress LER terminates the Timing LSP. The LSRs act as Transparent Clock (TC) and just update the Timing field in the Timing messages.

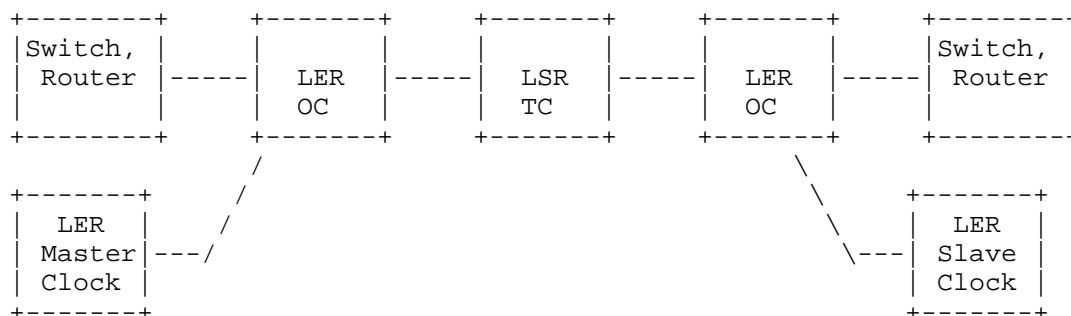


Figure (1) - Deployment example 1 of timing over MPLS/MPLS-TP network

LERs could also act as Boundary Clock (BC). This is shown in Figure 2, where LERs terminate the Timing messages received from switch/routers that are outside of the MPLS network acting as OC or BC. In this example LERs regenerate the clock and initiate timing messages encapsulated in Timing LSP toward the MPLS network, while the LSRs act as Transparent Clock (TC) and just update the Timing field in the Timing messages, which are already encapsulated in Timing LSPs.

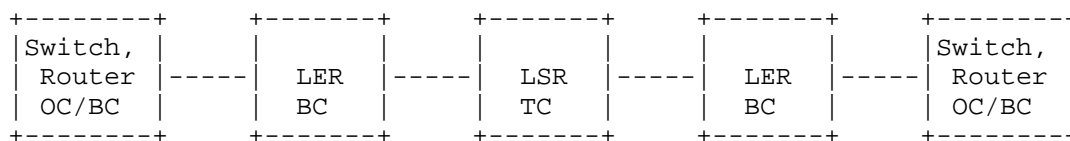


Figure (2) - Deployment example 2 of timing over MPLS/MPLS-TP network

LERs could also act as Transparent Clock (TC). This is shown in Figure 3, where LERs do not terminate the Timing messages received from switch/routers that are outside of the MPLS network acting as OC, TC or BC. The LERs act as TC and update the Timing field in the Timing messages as they transit the LER, while encapsulating them in timing LSP. The LSRs also act as Transparent Clock (TC) and just update the Timing field in the Timing messages which are already encapsulated in Timing LSPs.

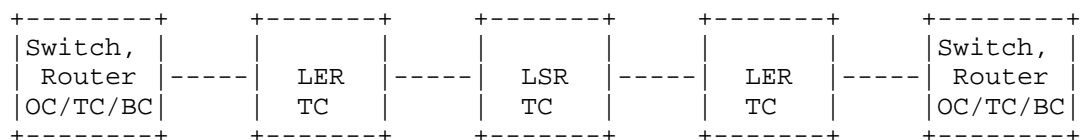


Figure (3) - Deployment example 3 of timing over MPLS/MPLS-TP network

5. Dedicated LSPs for Timing messages

Many methods have been considered for identifying the Timing messages when they are encapsulated in MPLS such as using GAL/ACH or a new reserved label. These methods were not attractive since they either required deep packet inspection at line rate in the intermediate LSRs or they required use of a scarce new reserved label. Also one of the goals was to reuse existing OAM mechanisms.

The method defined in this document can be used by LER and LSRs to identify Timing messages in MPLS tunnels by just looking at the top label in the MPLS label stack, which only carry Timing messages as well as OAM, but not data plane client traffic.

Compliant implementations MUST use dedicated LSPs to carry Timing messages over MPLS. These LSPs are herein referred to as "Timing LSPs" and the labels associated with these LSPs as "Timing LSP labels". The Timing LSPs that runs between Ingress and Egress LERs MUST be co-routed. Alternatively, a single bidirectional co-routed LSP can be used.

Co-routing of the two directions is required to limit the difference in the delays in the Master clock to Slave clock direction compared to the Slave clock to Master clock direction. The Timing LSP MAY be MPLS LSP or MPLS-TP LSP.

The Timing LSPs could be configured or signaled via RSVP-TE/GMPLS. New Extensions to RSVP-TE/GMPLS TLVs are required; however they are outside the scope of this document.

The Timing LSPs MAY carry essential MPLS/MPLS-TP OAM traffic such as BFD and LSP Ping but the LSP data plane client plane traffic MUST be Timing packets only.

6. Timing over LSP Encapsulation

This document defines two methods for carrying Timing messages over MPLS. The first method is carrying UDP/IP encapsulated Timing messages over Timing LSPs, which is suitable for MPLS networks and the second method, is carrying Ethernet encapsulated Timing messages over Ethernet PWs inside Timing LSPs, which is suitable for MPLS and MPLS-TP networks.

6.1. Timing over UDP/IP over MPLS Encapsulation

The simplest method of transporting Timing messages over MPLS is to encapsulate Timing PDUs in UDP/IP and then encapsulate them in Timing LSP. This format is shown in Figure 4.

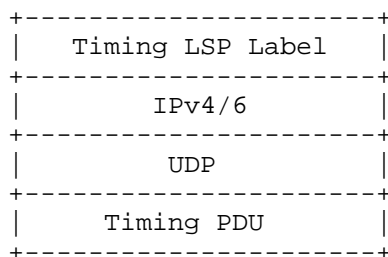


Figure (4) - Timing over UDP/IP over MPLS Encapsulation

This encapsulation is very simple and is useful when the network between Timing Master Clock and Slave Clock is MPLS network.

In order for an LER/LSR to process Timing messages, the Timing LSP Label must be at the top label of the label stack. The LER/LSR MUST know that the Timing LSP Label is used for carrying Timing messages. This can be accomplished via static configuration or via RSVP-TE signaling.

The UDP/IP encapsulation of PTP MUST follow Annex D and E of [IEEE]. While the UDP/IP encapsulation of NTP MUST follow [RFC5905].

6.2. Timing over PW Encapsulation

Another method of transporting Timing over MPLS networks is by encapsulating Timing PDUs in PW which in turn is transported over Timing LSPs. In case of PTP, Ethernet PW encapsulation [RFC4448], shown in Fig 5(A) MUST be used and the Ethernet encapsulation of PTP MUST follow Annex F of [IEEE].

The Tagged mode defined in [RFC-4448] MUST be used and the Payload MUST have 2 VLAN tags (S-VLAN and C-VLAN). The Timing over PW encapsulation MUST use the Control Word (CW) as specified in [RFC4448] to ensure proper detection of PTP messages inside the MPLS packets for Timing over LSP and Timing over PW encapsulation. The use of Sequence Number in the CW is optional.

Timing over PW encapsulation for NTP MUST use NTP over UDP/IP over PW (the IP PW discussed in [RFC4447]) shown in Fig 5(B).

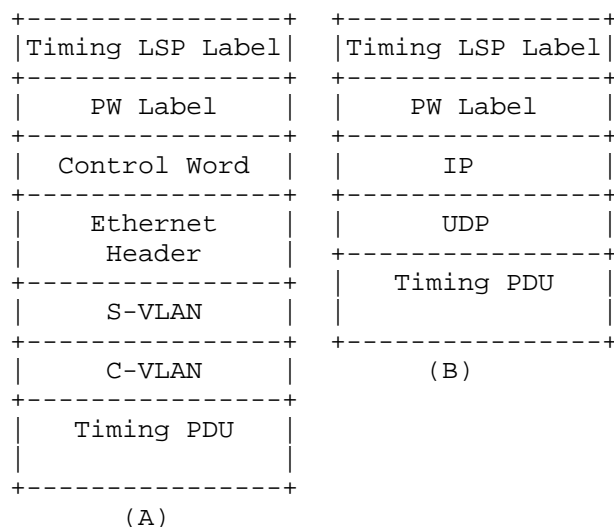


Figure (5) - Timing over PW Encapsulations

In order for an LSR to process PTP messages, the top label of the label stack (the Tunnel Label) MUST be a Timing label.

The PW method of transporting Timing over MPLS is applicable to both MPLS and MPLS-TP networks.

6.3. Other Timing Encapsulation methods

In future other timing encapsulation methods may be introduced, such as a new shim header after the Bottom of Stack to carry the Timing information. Such new encapsulations are outside the scope of this document. The control and signaling requirements in this document are defined generically enough to accommodate any such new encapsulations.

7. Timing Message Processing

Each Timing protocol such as PTP and NTP, define their set of Timing messages. For example PTP defines SYNC, DELAY_REQ, DELAY_RESP, FOLLOW_UP, etc messages.

Some of the Timing messages require time stamping at port level and some dont. It is the job of the LER/LSR to parse the timing message and find out the type of the Timing message and decide whether and how to Time- stamp it (e.g., BC) or modify existing timestamp in it (e.g., TC).

For example the following PTP messages (called Event messages) require time-stamping, while other Non-Event PTP messages do not need time-stamping.

- o Announce
- o SYNC
- o DELAY_REQ (Delay Request)
- o PDELAY_REQ (Peer Delay Request)
- o PDELAY_RESP (Peer Delay Response)

SYNC and DELAY_REQ are exchanged between Master Clock and Slave Clock and MUST be transported over PTP LSPs. PDELAY_REQ and PDELAY_RESP are exchanged between adjacent PTP clocks (i.e. Master, Slave, Boundary, or Transparent) and MAY be transported over single hop PTP LSPs. If Two Step PTP clocks are present, then the FOLLOW_UP, DELAY_RESP, and PDELAY_RESP_FOLLOW_UP messages must also be transported over the PTP LSPs.

For a given instance of 1588 protocol, SYNC and DELAY_REQ MUST be transported over two PTP LSPs that are in opposite directions. These PTP LSPs, which are in opposite directions MUST be congruent and co-routed. Alternatively, a single bidirectional co-routed LSP can be used.

Except as indicated above for the two-step PTP clocks, Non-Event PTP message types do not need to be processed by intermediate routers. These message types MAY be carried in PTP Tunnel LSPs.

8. Protection and Redundancy

In order to ensure continuous uninterrupted operation of slave clocks, usually as a general practice, slave clocks (or ports) track redundant master clocks.

It is the responsibility of the network operator to ensure that physically disjoint Timing LSPs are established between a slave clock (or port) and redundant master clocks (or ports).

When a slave clock (or port) listens to redundant master clocks or ports, any prolonged Timing LSP outage will trigger the slave clock or port to switch to a redundant master clock or port.

LSP/PW protection such as Linear protection Switching (1:1, 1+1), Ring protection switching or MPLS Fast Reroute (FRR) generally switch alternative path that usually cause a change in delay, which if undetected by slave clock can reduce accuracy of the slave clock. However it is expected that most Slave clocks could detect the change in delay. Therefore this specification recommends that protection switching SHOULD be used, unless the operator knows that the protection switching may have adverse effect on the slave clock.

Note that any protection or reroute mechanism that adds additional MPLS label to the label stack, such as Facility Backup Fast Reroute, MUST ensure that the pushed label is also a Timing Label to ensure recognition of the MPLS frame as containing Timing messages, as it transits the backup path.

9. ECMP

To ensure the optimal operation of slave clocks and avoid error introduced by forward and reverse path delay asymmetry, the physical path for Timing messages from master clock to slave Clock and vice versa must be the same for all Event Timing messages listed in section 7.

Therefore the Timing LSPs MUST not be subject to ECMP (Equal Cost Multipath).

10. PHP

To ensure that the label on the top of the label stack is the Timing LSP Label, PHP MUST not be used.

11. Entropy

To ensure all Timing messages in a Timing LSP take the same path, Entropy MUST NOT be used for the Timing LSP [mpls-entropy] and Entropy MSUT NOT be used for the PWs that are carried inside Timing LSP [RFC6391].

12. OAM, Control and Management

In order to manage Timing LSPs and their encapsulated PWs, they MUST be able to carry OAM and management messages. These management messages MUST be differentiated from Timing messages via already defined IETF methods.

For example BFD [RFC5880], [RFC5884] and LSP-Ping [RFC4389] MAY run over PTP LSPs via UDP/IP encapsulation or via GAL/G-ACH. These Management protocols can easily be identified by the UDP Destination Port number or by GAL/ACH respectively.

Also BFD, LSP-Ping and other management messages MAY run over the PWs encapsulated in Timing LSP via one of the defined VCCVs (Type 1, 3 or 4) [RFC5085] (note that VCCV Type 2 using Router Alert Label is going to be deprecated by IETF). In this case G-ACH, PW label (TTL=1) or GAL-ACH are used to identify such management messages.

13. QoS Considerations

In network deployments where not every LSR/LER is Timing-aware, it is important to reduce the impact of the non-Timing-aware LSR/LERs on the timing recovery in the slave clock. The Timing messages are time critical and must be treated with the highest priority. Therefore Timing over MPLS messages must be treated with the highest priority in the routers. This can be achieved by proper setup of Timing LSPs.

It is recommended that the Timing LSPs are setup or configured properly to indicate EF-PHB [RFC3246] for the CoS and Green [RFC2697] for drop eligibility.

14. FCS Recalculation

When timestamp generation and timing packet adjustment is performed near the physical port hardware, the process MUST include recalculation of the Ethernet FCS. Also FCS retention for the payload Ethernet described in [RFC4720] MUST NOT be used.

15. UDP Checksum Correction

For UDP/IP encapsulation mode of Timing over MPLS, the UDP checksum is optional when used for IPv4 encapsulation and mandatory in case of IPv6.

When UDP checksum is used, each Timing-aware LER/LSR must either incrementally update the UDP checksum after Time stamping or Correction Field update or verify the UDP checksum on reception from upstream and recalculate the checksum completely on transmission to downstream node after Time stamping or Correction Field update.

16. Routing extensions for Timing-aware Routers

MPLS-TE routing relies on extensions to OSPF [RFC2328] [RFC5340] and IS-IS [ISO] [RFC1195] in order to advertise Traffic Engineering (TE) link information used for constraint-based routing.

Indeed, it is useful to advertise data plane TE router link capabilities, such as the capability for a router to be Timing-aware. This capability **MUST** then be taken into account during path computation to prefer or even require links that advertise themselves as Timing-aware. In this way the path can ensure the entry and exit points into the LERs and, if desired, the links into the LSRs are able to perform port based timestamping thus minimizing their impact on the performance of the slave clock.

extensions are required to OSPF and IS-IS in order to advertise Timing-aware capabilities of a link. Such extensions are outside the scope of this document; however such extension **SHOULD** be able to signal the following information per Router Link:

- o Capable of processing PTP, NTP or other Timing flows
- o Capable of performing Transparent Clock operation
- o Capable of performing Boundary Clock operation

17. Signaling Extensions for Creating Timing LSPs

RSVP-TE signaling MAY be used to setup the timing LSPs. When RSVP-TE is used to setup Timing LSPs, some information that indicates that the LSP is carrying Timing flows MUST be included in the new Extensions to RSVP-TE:

The following information MAY also be included in the new Extensions to RSVP-TE:

- o Offset from Bottom of Stack (BoS) to the start of the Timestamp field
- o Number of VLANs in case of PW encapsulation
- o Timestamp field Type
 - * Correction Field, Timestamp
- o Timestamp Field format
 - * 64-bit PTPv1, 80-bit PTPv2, 32-bit NTP, 64-bit NTP, 128-bit NTP, etc.

Note that in case the above optional information is signaled with RSVP-TE for a Timing LSP, all the Timing packets carried in that LSP must have the same signaled characteristics. For example if Timestamp format is signaled as 64-bit PTPv1, then all Timing packets must use 64-bit PTPv1 timestamp.

18. Behavior of LER/LSR

Timing-capable/aware LERs and LSRs are routers that have one or more interfaces that can perform Timing operations (OC/BC/TC) on Timing packets and are configured to do so. Timing-capable/aware LERs and LSRs can advertise their Timing-capability per-interface via control plane such as OSPF or IS-IS. The Timing-capable/aware LERs can then signal Timing LSPs via RSVP-TE signaling. Alternatively the Timing capability of LER and LSRs may be configured in a centralized controller and the Timing LSP may be setup using manual configuration or other methods such as SDN.

18.1. Behavior of Timing-capable/aware LER

When a Timing-capable/aware LER behaves as a Transparent clock and receives a Timing message from a Timing-capable/aware non-MPLS interface, the LER updates the Correction Field (CF) and encapsulates and forwards the timing message over previously established Timing LSP. Also when a Timing message is received from a Timing-capable/aware MPLS interface, LER updates the Correction Filed (CF) and decapsulates the MPLS encapsulation and forwards the timing message to a non-MPLS interface.

When a Timing-capable/aware LER behaves as a Boundary clock and receives a Timing message from a Timing-capable/aware non MPLS interface, the LER Timestamps the Timing packet and sends it to the LERs Boundary clock processing module. Also when a Timing message is received from a Timing-capable/aware MPLS interface, the LER Timestamps the Timing packet and sends it to the LERs Boundary clock processing module.

When a Timing-capable/aware LER behaves as an Ordinary Clock toward the MPLS network, and receives a Timing message from a Timing-capable/aware MPLS interface, the LER Timestamps the Timing packet and sends it to the LERs Ordinary clock processing module.

18.2. Behavior of Timing-capable/aware LSR

A Timing-capable/aware LSR behaves as a Transparent clock and receives a Timing message from a Timing-capable/aware MPLS interface. The LSR updates the Correction Filed (CF) and forwards the timing message over another MPLS interface.

18.3. Behavior of non-Timing-capable/aware LSR

It is most beneficial when all LSRs in the path of a Timing LSP be timing-Capable/aware LSRs. This would ensure the highest quality time and clock synchronization by Timing Slave Clocks. However, this

specification does not mandate that all LSRs in path of a Timing LSP be Timing- capable/aware.

Non-Timing-capable/aware LSRs just switch the packets encapsulated in Timing LSPs and dont perform any Timing operation (TC). However as explained in QoS section the Timing8 over MPLS packets MUST be still be treated with the highest priority based on their Traffic Class (TC) marking.

19. Other considerations

[IEEE] Tdefines an optional peer-to-peer Transparent clocking that requires peer delay measurement between two adjacent Timing-capable/ware routers/switches. Peer delay measurement messages need to be time stamped and terminated by the Timing-capable/aware routers/switches. This means that two adjacent LSRs may be engaged in a peer delay measurement. Such peer delay measurement messages SHALL NOT use the Timing LSP that runs between two LERs. For transporting such peer delay measurement messages there are two options. Either a single-hop LSP needs to be created between the two adjacent LSRs engaged in peer delay measurement to carry peer delay measurement messages, or other methods such as PTP transport over Ethernet may be used for transporting peer delay measurement messages if the link between the two routers is Ethernet.

The use of Explicit Null Label (Label= 0 or 2) is acceptable as long as either the Explicit Null label is the bottom of stack label (applicable only to UDP/IP encapsulation) or the label below the Explicit Null label is a PTP label.

20. Security Considerations

MPLS PW security considerations in general are discussed in [RFC3985] and [RFC4447], and those considerations also apply to this document.

An experimental security protocol is defined in [IEEE]. The PTP security extension and protocol provides group source authentication, message integrity, and replay attack protection for PTP messages.

21. Applicability Statement

The Timing over MPLS transport methods described in this document apply to the following network Elements:

- o An LER receives IP or Ethernet Encapsulated Timing messages from a non-MPLS interface and forwards them as MPLS encapsulated Timing messages over Timing LSP, while performing Transparent Clock functionality
- o An LER receives MPLS encapsulated Timing messages from a Timing LSP and forwards them to non-MPLS interface as IP or Ethernet Encapsulated Timing messages, while performing Transparent Clock functionality
- o An LER receives MPLS encapsulated Timing messages from a Timing LSP and terminates them, while performing the OC or BC functionality
- o An LSR receives MPLS encapsulated Timing messages from a Timing LSP and forwards them to another MPLS interface, while performing the TC functionality

This document also supports the case where not all LSRs are Timing-capable/aware. It also supports the case where not all LER/LSR interfaces are Timing-capable/aware.

22. Acknowledgements

The authors would like to thank Luca Martini, Ron Cohen, Yaakov Stein, Tal Mizrahi, Stefano Ruffini, Luca Moniti and other members of IETF for reviewing and providing feedback on this draft.

23. IANA Considerations

There are no IANA requirements in this specification.

24. References

24.1. Normative References

- [IEEE] IEEE 1588-2008, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems".
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3985] Bryant, S. and P. Pate, "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture", RFC 3985, March 2005.
- [RFC4389] Thaler, D., Talwar, M., and C. Patel, "Neighbor Discovery Proxies (ND Proxy)", RFC 4389, April 2006.
- [RFC4447] Martini, L., Rosen, E., El-Aawar, N., Smith, T., and G. Heron, "Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)", RFC 4447, April 2006.
- [RFC4448] Martini, L., Rosen, E., El-Aawar, N., and G. Heron, "Encapsulation Methods for Transport of Ethernet over MPLS Networks", RFC 4448, April 2006.
- [RFC4720] Malis, A., Allan, D., and N. Del Regno, "Pseudowire Emulation Edge-to-Edge (PWE3) Frame Check Sequence Retention", RFC 4720, November 2006.
- [RFC5085] Nadeau, T. and C. Pignataro, "Pseudowire Virtual Circuit Connectivity Verification (VCCV): A Control Channel for Pseudowires", RFC 5085, December 2007.
- [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, June 2010.
- [RFC5884] Aggarwal, R., Kompella, K., Nadeau, T., and G. Swallow, "Bidirectional Forwarding Detection (BFD) for MPLS Label Switched Paths (LSPs)", RFC 5884, June 2010.

24.2. Informative References

- [I-D.ietf-pwe3-fat-pw] Bryant, S., Filsfils, C., Drafz, U., Kompella, V., Regan, J., and S. Amante, "Flow Aware Transport of Pseudowires over an MPLS Packet Switched Network", draft-ietf-pwe3-fat-pw-07 (work in progress), July 2011.

- [ISO] ISO/IEC 10589:1992, "Intermediate system to Intermediate system routing information exchange protocol for use in conjunction with the Protocol for providing the Connectionless-mode Network Service (ISO 8473)".
- [RFC1195] Callon, R., "Use of OSI IS-IS for routing in TCP/IP and dual environments", RFC 1195, December 1990.
- [RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, April 1998.
- [RFC2697] Heinanen, J. and R. Guerin, "A Single Rate Three Color Marker", RFC 2697, September 1999.
- [RFC3246] Davie, B., Charny, A., Bennet, J., Benson, K., Le Boudec, J., Courtney, W., Davari, S., Firoiu, V., and D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)", RFC 3246, March 2002.
- [RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, September 2003.
- [RFC3784] Smit, H. and T. Li, "Intermediate System to Intermediate System (IS-IS) Extensions for Traffic Engineering (TE)", RFC 3784, June 2004.
- [RFC4970] Lindem, A., Shen, N., Vasseur, JP., Aggarwal, R., and S. Shaffer, "Extensions to OSPF for Advertising Optional Router Capabilities", RFC 4970, July 2007.
- [RFC4971] Vasseur, JP., Shen, N., and R. Aggarwal, "Intermediate System to Intermediate System (IS-IS) Extensions for Advertising Router Information", RFC 4971, July 2007.
- [RFC5120] Przygienda, T., Shen, N., and N. Sheth, "M-ISIS: Multi Topology (MT) Routing in Intermediate System to Intermediate Systems (IS-ISs)", RFC 5120, February 2008.
- [RFC5305] Li, T. and H. Smit, "IS-IS Extensions for Traffic Engineering", RFC 5305, October 2008.
- [RFC5329] Ishiguro, K., Manral, V., Davey, A., and A. Lindem, "Traffic Engineering Extensions to OSPF Version 3", RFC 5329, September 2008.
- [RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, July 2008.

- [RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.
- [RFC6391] Bryant, S., Filsfils, C., Drafz, U., Kompella, V., Regan, J., and S. Amante, "Flow-Aware Transport of Pseudowires over an MPLS Packet Switched Network", RFC 6391, November 2011.

Authors' Addresses

Shahram Davari
Broadcom Corp.
San Jose, CA 95134
USA

Email: davari@broadcom.com

Amit Oren
Broadcom Corp.
San Jose, CA 95134
USA

Email: amito@broadcom.com

Manav Bhatia
Alcatel-Lucent
Bangalore,
India

Email: manav.bhatia@alcatel-lucent.com

Peter Roberts
Alcatel-Lucent
Kanata,
Canada

Email: peter.roberts@alcatel-lucent.com

Laurent Montini
Cisco Systems
San Jose CA
USA

Email: lmontini@cisco.com

TICTOC Working Group
INTERNET DRAFT
Intended status: Standards Track
Expires: January 30, 2013

Vinay Shankarkumar
Laurent Montini
Cisco Systems

Tim Frost
Greg Dowd
Symmetricom

July 30, 2012

Precision Time Protocol Version 2 (PTPv2)
Management Information Base
draft-ietf-tictoc-ntp-mib-03.txt

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

This Internet-Draft will expire on January 30, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Abstract

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in TCP/IP-based internets. In particular, it defines objects for managing networks using Precision Time Protocol.

This memo specifies a MIB module in a manner that is both compliant to the SNMPv2 SMI, and semantically identical to the peer SNMPv1 definitions.

Table of Contents

1. Introduction.....	2
1.1. Relationship to other Profiles and MIBs.....	3
1.2. Change Log.....	3
2. The SNMP Management Framework.....	4
3. Overview.....	5
4. IETF PTP MIB Definition.....	5
5. Security Considerations.....	61
6. IANA Considerations.....	61
7. References.....	61
7.1. Normative References.....	61
7.2. Informative References.....	61
8. Acknowledgements.....	63
9. Author's Addresses.....	64
10. ANNEX A: Extended Fields Addendum.....	65

1. Introduction

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in the Internet Community. In particular, it describes managed objects used for managing PTP devices including the ordinary clock, transparent clock, boundary clocks.

This MIB is restricted to reading standard PTP data elements, as described in [IEEE 1588-2008]. It is envisioned this MIB will complement other managed objects to be defined to monitor, measure the performance of the PTP devices and telecom clocks. Those objects are considered out of scope for the current draft.

Similarly, this MIB is read-only and not intended to provide the ability to configure PTP clocks. Since PTP clocks are often embedded in other network elements such as routers, switches and gateways, this ability is generally provided via the configuration interface for the network element.

1.1. Relationship to other Profiles and MIBs

This MIB is intended to be used with the default PTP profile described in [IEEE 1588-2008], and the Telecom Profile described in [G.8265.1], when running over the IP network layer. As stated above, it is envisioned this MIB will complement other managed objects to be defined to monitor, measure the performance of the PTP devices and telecom clocks.

Some other PTP profiles have their own MIBs defined as part of the profile, and this MIB is not intended to replace those MIBs.

1.2. Change Log

This section tracks changes made to the revisions of the Internet Drafts of this document. It will be **deleted** when the document is published as an RFC. This section tracks changes made to the revisions of the Internet Drafts of this document. It will be **deleted** when the document is published as an RFC.

draft-vinay-tictoc-ntp-mib

-00 Mar 11 Initial version; showed structure of MIB

draft-ietf-tictoc-ntp-mib

-00 Jul 11 First full, syntactically correct and compileable MIB

-01 Jan 12 Revised following comments from Bert Wijnen:

- revised introduction to clarify the scope, and the relationship to other MIBs and profiles
- changed name to "ntpbases"
- corrected some data types
- corrected references and typos

-02 Jul 12 Revised following comment at IETF83:

- changed "ntpbasesClockPortRunningIPversion" to the more generic "ntpbasesClockPortRunningTransport", covering all transport types defined in [IEEE 1588-2008] (i.e. IPv4, IPv6, Ethernet, DeviceNet and ControlNet).
- changed addresses associated with transports from "InetAddress" (for the IP transport) to a string, to allow for the different transport types.

-03 Jul 31 Revised following comments at IETF83 and from Andy Bierman:

- corrected minor compiling errors and typos from -01
- corrected breaking compiling error from -02
- moved the OBJECT-GROUPS and MODULE-COMPLIANCES to the end
- edited description clauses of MIBCompliances 1,2,3 & 4
- renamed ptptimeMIBCompliances1 to 4:
 - ptptimeMIBCompliancesSystemInfo
 - ptptimeMIBCompliancesClockInfo
 - ptptimeMIBCompliancesClockPortInfo
 - ptptimeMIBCompliancesTransparentClockInfo
- added Annex with the extended structures introduced in -01
- edited minor corrections from -02

2. The SNMP Management Framework

The SNMP Management Framework presently consists of five major components:

1. An overall architecture, described in STD62, [RFC 3411].
2. Mechanisms for describing and naming objects and events for the purpose of management. The first version of this Structure of Management Information (SMI) is called SMIv1 and described in STD 16: [RFC 1155], [RFC 1212] and [RFC 1215]. The second version, called SMIv2, is described in STD 58: [RFC 2578], [RFC 2579] and [RFC 2580].
3. Message protocols for transferring management information. The first version of the SNMP message protocol is called SNMPv1 and described in STD 15 [RFC 1157]. A second version of the SNMP message protocol, which is not an Internet standards track protocol, is called SNMPv2c and described in [RFC 1901] and [RFC 1906]. The third version of the message protocol is called SNMPv3 and described in STD62: [RFC 3417], [RFC 3412] and [RFC 3414].
4. Protocol operations for accessing management information. The first set of protocol operations and associated PDU formats is described in STD 15 [RFC 1157]. A second set of protocol operations and associated PDU formats is described in STD 62 [RFC 3416].
5. A set of fundamental applications described in STD 62 [RFC 3413] and the view-based access control mechanism described in STD 62 [RFC 3415].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using the mechanisms defined in the SMI.

This memo specifies a MIB module that is compliant to the SMIV2. A MIB conforming to the SMIV1 can be produced through the appropriate translations. The resulting translated MIB must be semantically equivalent, except where objects or events are omitted because no translation is possible (e.g., use of Counter64). Some machine readable information in SMIV2 will be converted into textual descriptions in SMIV1 during the translation process. However, this loss of machine readable information is not considered to change the semantics of the MIB.

3. Overview

The objects defined in this MIB are to be used when describing the Precision Time Protocol (PTPv2).

4. IETF PTP MIB Definition

```
PTPBASE-MIB DEFINITIONS ::= BEGIN
```

IMPORTS

```
    MODULE-IDENTITY,
    OBJECT-TYPE,
    Integer32,
    Gauge32,
    Unsigned32,
    Counter32,
    Counter64,
    mib-2
        FROM SNMPv2-SMI
    OBJECT-GROUP,
    MODULE-COMPLIANCE
        FROM SNMPv2-CONF
    TEXTUAL-CONVENTION,
    TruthValue,
    DisplayString
        FROM SNMPv2-TC
    InterfaceIndexOrZero
        FROM IF-MIB;
```

ptpbasemib MODULE-IDENTITY

```
    LAST-UPDATED      "201207230000Z"
    ORGANIZATION      "TICTOC Working Group"
    CONTACT-INFO
        "WG Email: tictoc@ietf.org
```

Vinay Shankarkumar

Cisco Systems,
Email: vinays@cisco.com

Laurent Montini,
Cisco Systems,
Email: lmontini@cisco.com

Tim Frost,
Symmetricom Inc.,
Email: tfrost@symmetricom.com

Greg Dowd,
Symmetricom Inc.,
Email: gdowd@symmetricom.com"

DESCRIPTION

"The MIB module for PTP version 2 (IEEE Std. 1588(TM)-2008)

Overview of PTP version 2 (IEEE Std. 1588(TM)-2008)

[IEEE 1588-2008] defines a protocol enabling precise synchronization of clocks in measurement and control systems implemented with packet-based networks, the Precision Time Protocol Version 2 (PTPv2). This MIB does not address the earlier version IEEE Std. 1588(TM)-2002 (PTPv1). The protocol is applicable to network elements communicating using IP. The protocol enables heterogeneous systems that include clocks of various inherent precision, resolution, and stability to synchronize to a grandmaster clock.

The protocol supports system-wide synchronization accuracy in the sub-microsecond range with minimal network and local clock computing resources. [IEEE 1588-2008] uses UDP/IP or Ethernet and can be adapted to other mappings. It includes formal mechanisms for message extensions, higher sampling rates, correction for asymmetry, a clock type to reduce error accumulation in large topologies, and specifications on how to incorporate the resulting additional data into the synchronization protocol. The [IEEE 1588-2008] defines conformance and management capability also.

MIB description

This MIB is to support the Precision Time Protocol version 2 (PTPv2, hereafter designated as PTP) features of network element system devices, when using the default PTP profile described in [IEEE 1588-2008], or the Telecom Profile described in [G.8265.1], when running over the IP network layer.

It is envisioned this MIB will complement other managed objects to be defined to monitor, measure the performance of the PTP devices and telecom clocks.

Some other PTP profiles have their own MIBs defined as part of the profile, and this MIB is not intended to replace those MIBs.

Acronyms:

ARB	Arbitrary Timescale
E2E	End-to-End
EUI	Extended Unique Identifier.
GPS	Global Positioning System
IANA	Internet Assigned Numbers Authority
IP	Internet Protocol
MAC	Media Access Control
	according to [IEEE 802.3-2008]
NIST	National Institute of Standards and Technology
NTF	Network Time Protocol (see IETF [RFC 5905])
OUI	Organizational Unique Identifier (allocated by the IEEE)
P2P	Peer-to-Peer
PTP	Precision Time Protocol
TAI	International Atomic Time
TC	Transparent Clock
UDP	User Datagram Protocol
UTC	Coordinated Universal Time

References:

- [IEEE 1588-2008] IEEE Standard for A Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, IEEE Std. 1588(TM)-2008, 24 July 2008.
- [G.8265.1] Precision Time Protocol Telecom Profile for Frequency Synchronization, ITU-T Recommendation G.8265.1, October 2010.

As defined in [IEEE 1588-2008]:

Accuracy:

The mean of the time or frequency error between the clock under test and a perfect reference clock, over an ensemble of measurements. Stability is a measure of how the mean varies with respect to variables such as time, temperature, and so on, while the precision is a measure of the deviation of the error from the mean.

Atomic process:

A process is atomic if the values of all inputs to the process are not permitted to change until all of the results of the process are instantiated, and the outputs of the process are not visible to other processes until the processing of each output is complete.

Boundary clock:

A clock that has multiple Precision Time Protocol (PTP) ports in

a domain and maintains the timescale used in the domain. It may serve as the source of time, i.e., be a master clock, and may synchronize to another clock, i.e., be a slave clock.

Boundary node clock:

A clock that has multiple Precision Time Protocol (PTP) ports in a domain and maintains the timescale used in the domain. It differs from a boundary clock in that the clock roles can change.

Clock:

A node participating in the Precision Time Protocol (PTP) that is capable of providing a measurement of the passage of time since a defined epoch.

Domain:

A logical grouping of clocks that synchronize to each other using the protocol, but that are not necessarily synchronized to clocks in another domain.

End-to-end transparent clock:

A transparent clock that supports the use of the end-to-end delay measurement mechanism between slave clocks and the master clock. Each node must measure the residence time of PTP event messages and accumulate it in Correction Field.

Epoch:

The origin of a timescale.

Event:

An abstraction of the mechanism by which signals or conditions are generated and represented.

Foreign master:

An ordinary or boundary clock sending Announce messages to another clock that is not the current master recognized by the other clock.

Grandmaster clock:

Within a domain, a clock that is the ultimate source of time for clock synchronization using the protocol.

Holdover:

A clock previously synchronized/syntonized to another clock (normally a primary reference or a master clock) but now free-running based on its own internal oscillator, whose frequency is being adjusted using data acquired while it had been synchronized/syntonized to the other clock. It is said to be in holdover or in the holdover mode, as long as it is within its accuracy requirements.

Link:

A network segment between two Precision Time Protocol ports supporting the peer delay mechanism of this standard. The peer delay mechanism is designed to measure the propagation time over such a link.

Management node:

A device that configures and monitors clocks.

Master clock:

In the context of a single Precision Time Protocol communication path, a clock that is the source of time to which all other clocks on that path synchronize.

Message timestamp point:

A point within a Precision Time Protocol event message serving as a reference point in the message. A timestamp is defined by the instant a message timestamp point passes the reference plane of a clock.

Multicast communication:

A communication model in which each Precision Time Protocol message sent from any PTP port is capable of being received and processed by all PTP ports on the same PTP communication path.

Node:

A device that can issue or receive Precision Time Protocol communications on a network.

One-step clock:

A clock that provides time information using a single event message.

On-pass support:

Indicates that each node in the synchronization chain from master to slave can support IEEE-1588.

Ordinary clock:

A clock that has a single Precision Time Protocol port in a domain and maintains the timescale used in the domain. It may serve as a source of time, i.e., be a master clock, or may synchronize to another clock, i.e., be a slave clock.

Parent clock:

The master clock to which a clock is synchronized.

Peer-to-peer transparent clock:

A transparent clock that, in addition to providing Precision Time Protocol event transit time information, also provides corrections for the propagation delay of the link connected to the port receiving the PTP event message. In the presence of peer-to-peer transparent clocks, delay measurements between slave clocks and the master clock are performed using the

peer-to-peer delay measurement mechanism.

Phase change rate:

The observed rate of change in the measured time with respect to the reference time. The phase change rate is equal to the fractional frequency offset between the measured frequency and the reference frequency.

PortNumber:

An index identifying a specific Precision Time Protocol port on a PTP node.

Primary reference:

A source of time and or frequency that is traceable to international standards.

Profile:

The set of allowed Precision Time Protocol features applicable to a device.

Precision Time Protocol communication:

Information used in the operation of the protocol, transmitted in a PTP message over a PTP communication path.

Precision Time Protocol communication path:

The signaling path portion of a particular network enabling direct communication among ordinary and boundary clocks.

Precision Time Protocol node:

PTP ordinary, boundary, or transparent clock or a device that generates or parses PTP messages.

Precision Time Protocol port:

A logical access point of a clock for PTP communications to the communications network.

Recognized standard time source:

A recognized standard time source is a source external to Precision Time Protocol that provides time and/or frequency as appropriate that is traceable to the international standards laboratories maintaining clocks that form the basis for the International Atomic Time and Universal Coordinated Time timescales. Examples of these are GPS, NTP, and NIST timeservers.

Requestor:

The port implementing the peer-to-peer delay mechanism that initiates the mechanism by sending a Pdelay_Req message.

Responder:

The port responding to the receipt of a Pdelay_Req message as part of the operation of the peer-to-peer delay mechanism.

Synchronized clocks:

Two clocks are synchronized to a specified uncertainty if they have the same epoch and their measurements of the time of a single event at an arbitrary time differ by no more than that uncertainty.

Syntonized clocks:

Two clocks are syntonized if the duration of the second is the same on both, which means the time as measured by each advances at the same rate. They may or may not share the same epoch.

Timeout:

A mechanism for terminating requested activity that, at least from the requester's perspective, does not complete within the specified time.

Timescale:

A linear measure of time from an epoch.

Traceability:

A property of the result of a measurement or the value of a standard whereby it can be related to stated references, usually national or international standards, through an unbroken chain of comparisons all having stated uncertainties.

Translation device:

A boundary clock or, in some cases, a transparent clock that translates the protocol messages between regions implementing different transport and messaging protocols, between different versions of [IEEE 1588-2008], or different PTP profiles.

Transparent clock:

A device that measures the time taken for a Precision Time Protocol event message to transit the device and provides this information to clocks receiving this PTP event message.

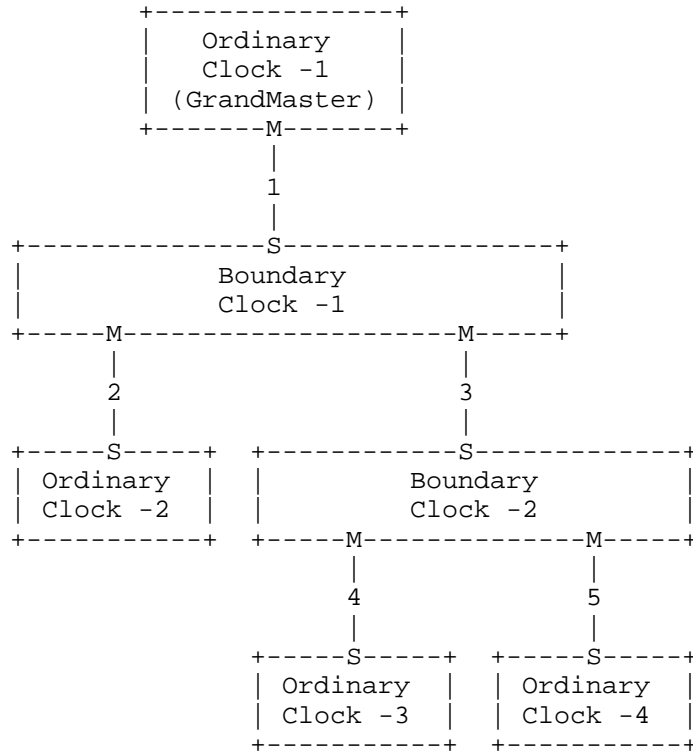
Two-step clock:

A clock that provides time information using the combination of an event message and a subsequent general message.

The below table specifies the object formats of the various textual conventions used.

Data type mapping	Textual Convention	SYNTAX
5.3.2 TimeInterval	ClockTimeInterval	OCTET STRING(SIZE(1..255))
5.3.3 Timestamp	ClockTimestamp	OCTET STRING(SIZE(6))
5.3.4 ClockIdentity	ClockIdentity	OCTET STRING(SIZE(1..255))
5.3.5 PortIdentity	ClockPortNumber	INTEGER(1..65535)
5.3.7 ClockQuality	ClockQualityClassType	

Simple master-slave hierarchy, section 6.6.2.4 [IEEE 1588-2008]:



Grandmaster

Boundary Clock(0-N) Ordinary Clocks(0-N)

Ordinary Clocks(0-N)

Relationship cardinality:

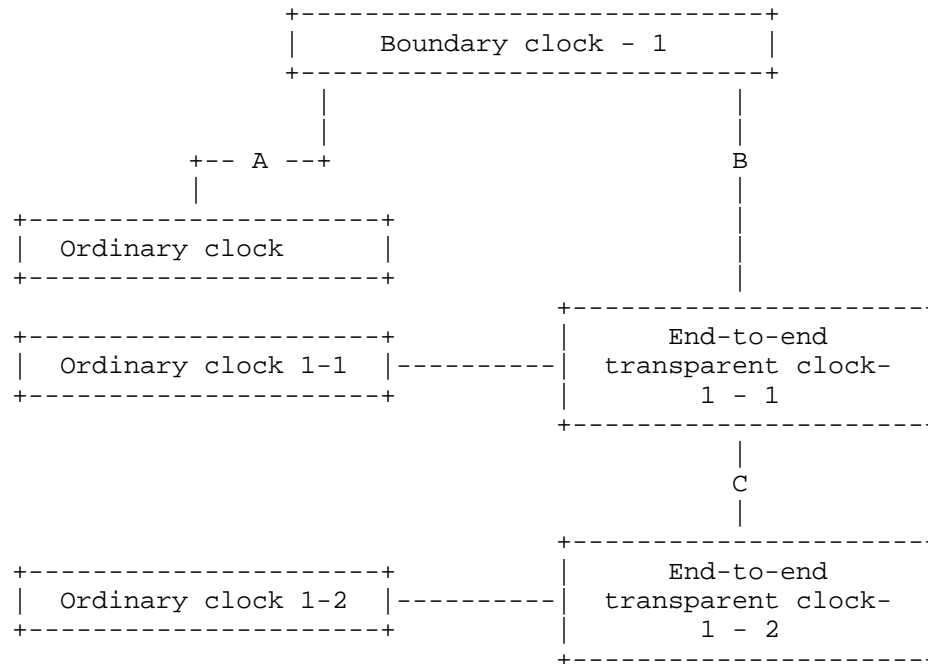
PTP system 1 : N PTP Clocks

PTP Clock 1 : 1 Domain

PTP Clock 1 : N PTP Ports

PTP Ports N : M Physical Ports (interface in IF-MIB)

Transparent clock diagram, section 6.7.1.3 of [IEEE 1588-2008]:



The MIB refers to the sections of [IEEE 1588-2008]."

-- revision log

::= { mib-2 XXX } -- XXX to be assigned by IANA

ClockDomainType ::= TEXTUAL-CONVENTION

DISPLAY-HINT "d"

STATUS current

DESCRIPTION

"The Domain is identified by an integer, the domainNumber, in the range of 0 to 255. An integer value that is used to assign each PTP device to a particular domain. The following values define the valid domains.

Value	Definition
-----	-----
0	Default domain
1	Alternate domain 1
2	Alternate domain 2
3	Alternate domain 3
4 - 127	User-defined domains
128 - 255	Reserved"

REFERENCE "Section 7.1 Domains, Table 2 of [IEEE 1588-2008]"

SYNTAX Unsigned32 (0..255)

ClockIdentity ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"The clock Identity is an 8-octet array and will be presented in the form of a character array. The value of the ClockIdentity should be taken from the IEEE EUI-64 individual assigned numbers as indicated in Section 7.5.2.2.2 of [IEEE 1588-2008]. The EUI-64 address is divided into the following fields:

OUI bytes (0-2)

Extension identifier bytes (3-7)

The clock identifier can be constructed from existing EUI-48 assignments and here is an abbreviated example extracted from section 7.5.2.2.2 [IEEE 1588-2008].

Company EUI-48 = 0xACDE4823456716

EUI-64 = ACDE48FFFE23456716

It is important to note the IEEE Registration Authority has deprecated the use of MAC-48 in any new design."

REFERENCE "Section 7.5.2.2.1 of [IEEE 1588-2008]"

SYNTAX OCTET STRING (SIZE (1..255))

ClockIntervalBase2 ::= TEXTUAL-CONVENTION

DISPLAY-HINT "d"

STATUS current

DESCRIPTION

"The interval included in message types Announce, Sync, Delay_Req, and Pdelay_Req as indicated in section 7.7.2.1 of [IEEE 1588-2008].

The mean time interval between successive messages shall be represented as the logarithm to the base 2 of this time interval measured in seconds on the local clock of the device sending the message. The values of these logarithmic attributes shall be selected from integers in the range -128 to 127 subject to further limits established in an applicable PTP profile."

REFERENCE "Section 7.7.2.1 General interval specification of [IEEE 1588-2008]"

SYNTAX Integer32 (-128..127)

ClockMechanismType ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"The clock type based on whether End to End or peer to peer mechanisms are used. The mechanism used to calculate the Mean Path Delay as indicated in Table 9 of [IEEE 1588-2008].

Delay mechanism	Value(hex)	Specification
-----	-----	-----
E2E	01	The port is configured to use the delay request-response mechanism.
P2P	02	The port is configured to use the peer delay mechanism.
DISABLED	FE	The port does not implement the delay mechanism."

REFERENCE "Sections 8.2.5.4.4, 6.6.4, 7.4.2 of [IEEE 1588-2008]."

SYNTAX INTEGER {
 e2e(1),
 p2p(2),
 disabled(254)
 }

ClockInstanceType ::= TEXTUAL-CONVENTION

DISPLAY-HINT "d"

STATUS current

DESCRIPTION

"The instance of the Clock of a given clock type in a given domain."

SYNTAX Unsigned32 (0..255)

ClockPortNumber ::= TEXTUAL-CONVENTION

DISPLAY-HINT "d"

STATUS current

DESCRIPTION

"An index identifying a specific Precision Time Protocol (PTP) port on a PTP node."

REFERENCE "Sections 7.5.2.3 and 5.3.5 of [IEEE 1588-2008]"

SYNTAX Unsigned32 (0..65535)

ClockPortState ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"This is the value of the current state of the protocol engine associated with this port.

Port state	Value	Description
-----	-----	-----

initializing	1	In this state a port initializes its data sets, hardware, and communication facilities.
faulty	2	The fault state of the protocol.
disabled	3	The port shall not place any messages on its communication path.
listening	4	The port is waiting for the

		announceReceiptTimeout to expire or to receive an Announce message from a master.
preMaster	5	The port shall behave in all respects as though it were in the MASTER state except that it shall not place any messages on its communication path except for Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up, signaling, or management messages.
master	6	The port is behaving as a master port.
passive	7	The port shall not place any messages on its communication path except for Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up, or signaling messages, or management messages that are a required response to another management message
uncalibrated	8	The local port is preparing to synchronize to the master port.
slave	9	The port is synchronizing to the selected master port."

REFERENCE "Section 8.2.5.3.1 portState and 9.2.5 of [IEEE 1588-2008]"

SYNTAX INTEGER {
initializing(1),
faulty(2),
disabled(3),
listening(4),
preMaster(5),
master(6),
passive(7),
uncalibrated(8),
slave(9)
}

ClockProfileType ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"Clock Profile used. A profile is the set of allowed Precision Time Protocol (PTP) features applicable to a device."

REFERENCE "Section 3.1.30 and 19.3 PTP profiles of [IEEE 1588-2008]"

SYNTAX INTEGER {
default(1),
telecom(2),
vendorspecific(3)
}

ClockQualityAccuracyType ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"The ClockQuality as specified in section 5.3.7, 7.6.2.5 and Table 6 of [IEEE 1588-2008].

The following values are not represented in the enumerated values.

0x01-0x1F Reserved

0x32-0x7F Reserved

It is important to note that section 7.1.1 RFC2578 allows for gaps and enumerate values to start with zero when indicated by the protocol."

REFERENCE "Section 5.3.7, 7.6.2.5 and Table 6 of [IEEE 1588-2008]"

SYNTAX INTEGER {
 reserved00(1), -- 0
 nanoSecond25(32), -- 0x20
 nanoSecond100(33), -- 0x21
 nanoSecond250(34), -- 0x22
 microSec1(35), -- 0x23
 microSec2dot5(36), -- 0x24
 microSec10(37), -- 0x25
 microSec25(38), -- 0x26
 microSec100(39), -- 0x27
 microSec250(40), -- 0x28
 milliSec1(41), -- 0x29
 milliSec2dot5(42), -- 0x2A
 milliSec10(43), -- 0x2B
 milliSec25(44), -- 0x2C
 milliSec100(45), -- 0x2D
 milliSec250(46), -- 0x2E
 second1(47), -- 0x2F
 second10(48), -- 0x30
 secondGreater10(49), -- 0x31
 unknown(254), -- 0xFE
 reserved255(255) -- 0xFF
 }

ClockQualityClassType ::= TEXTUAL-CONVENTION

DISPLAY-HINT "d"

STATUS current

DESCRIPTION

"The ClockQuality as specified in section 5.3.7, 7.6.2.4 and Table 5 of [IEEE 1588-2008].

Value	Description
0	Reserved to enable compatibility with future versions.

- 1-5 Reserved
- 6 Shall designate a clock that is synchronized to a primary reference time source. The timescale distributed shall be PTP. A clockClass 6 clock shall not be a slave to another clock in the domain.
- 7 Shall designate a clock that has previously been designated as clockClass 6 but that has lost the ability to synchronize to a primary reference time source and is in holdover mode and within holdover specifications. The timescale distributed shall be PTP. A clockClass 7 clock shall not be a slave to another clock in the domain.
- 8 Reserved.
- 9-10 Reserved to enable compatibility with future versions.
- 11-12 Reserved.
- 13 Shall designate a clock that is synchronized to an application-specific source of time. The timescale distributed shall be ARB. A clockClass 13 clock shall not be a slave to another clock in the domain.
- 14 Shall designate a clock that has previously been designated as clockClass 13 but that has lost the ability to synchronize to an application-specific source of time and is in holdover mode and within holdover specifications. The timescale distributed shall be ARB. A clockClass 14 clock shall not be a slave to another clock in the domain.
- 15-51 Reserved.
- 52 Degradation alternative A for a clock of clockClass 7 that is not within holdover specification. A clock of clockClass 52 shall not be a slave to another clock in the domain.
- 53-57 Reserved.
- 58 Degradation alternative A for a clock of clockClass 14 that is not within holdover specification. A clock of clockClass 58 shall not be a slave to another clock in the domain.
- 59-67 Reserved.
- 68-122 For use by alternate PTP profiles.
- 123-127 Reserved.
- 128-132 Reserved.
- 133-170 For use by alternate PTP profiles.
- 171-186 Reserved.
- 187 Degradation alternative B for a clock of clockClass 7 that is not within holdover specification. A clock of clockClass 187 may

be a slave to another clock in the domain.

188-192 Reserved.

193 Degradation alternative B for a clock of clockClass 14 that is not within holdover specification. A clock of clockClass 193 may be a slave to another clock in the domain.

194-215 Reserved.

216-232 For use by alternate PTP profiles.

233-247 Reserved.

248 Default. This clockClass shall be used if none of the other clockClass definitions apply.

249-250 Reserved.

251 Reserved for version 1 compatibility; see Clause 18.

252-254 Reserved.

255 Shall be the clockClass of a slave-only clock; see 9.2.2."

REFERENCE "Section 5.3.7, 7.6.2.4 and Table 5 of [IEEE 1588-2008]."

SYNTAX Unsigned32 (0..255)

ClockRoleType ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"The Clock Role. The protocol generates a Master Slave relationship among the clocks in the system.

Clock Role	Value	Description
Master clock	1	A clock that is the source of time to which all other clocks on that path synchronize.
Slave clock	2	A clock which synchronizes to another clock (master)."

SYNTAX INTEGER {
 master(1),
 slave(2)
}

ClockStateType ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"The clock state returned by PTP engine.

Clock State	Value	Description
Freerun state	1	Applies to a slave device that is not locked to a master. This is the initial state a slave starts out with when it is not getting any PTP packets from the master or because of some other input

error (erroneous packets, etc).

- | | | |
|---------------------|---|---|
| Holdover state | 2 | In this state the slave device is locked to a master but communication with the master is lost or the timestamps in the ntp packets are incorrect. But since the slave was locked to the master, it can run with the same accuracy for sometime. The slave can continue to operate in this state for some time. If communication with the master is not restored for a while, the device is moved to the FREERUN state. |
| Acquiring state | 3 | The slave device is receiving packets from a master and is trying to acquire a lock. |
| Freq_locked state | 4 | Slave device is locked to the Master with respect to frequency, but not phase aligned |
| Phase_aligned state | 5 | Locked to the master with respect to frequency and phase." |

```
SYNTAX          INTEGER {
                    freerun(1),
                    holdover(2),
                    acquiring(3),
                    frequencyLocked(4),
                    phaseAligned(5)
                  }
```

ClockTimeSourceType ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"The ClockQuality as specified in section 5.3.7, 7.6.2.6 and Table 7 of [IEEE 1588-2008].

The following values are not represented in the enumerated values.

0xF0-0xFE	For use by alternate PTP profiles
0xFF	Reserved

It is important to note that section 7.1.1 RFC2578 allows for gaps and enumerate values to start with zero when indicated by the protocol."

REFERENCE "Section 5.3.7, 7.6.2.6 and Table 7 of [IEEE 1588-2008]."

```
SYNTAX          INTEGER {
```

```
        atomicClock(16), -- 0x10
        gps(32), -- 0x20
        terrestrialRadio(48), -- 0x22
        ptp(64), -- 0x40
        ntp(80), -- 0x50
        handSet(96), -- 0x60
        other(144), -- 0x90
        internalOscillator(160) -- 0xA0
    }
```

ClockTimeInterval ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"This textual convention corresponds to the TimeInterval structure indicated in section 5.3.2 of [IEEE 1588-2008]. It will be presented in the form of a character array.

The TimeInterval type represents time intervals.

```
    struct TimeInterval
    {
        Integer64 scaledNanoseconds;
    };
```

The scaledNanoseconds member is the time interval expressed in units of nanoseconds and multiplied by 2**16.

Positive or negative time intervals outside the maximum range of this data type shall be encoded as the largest positive and negative values of the data type, respectively.

For example, 2.5 ns is expressed as 0000 0000 0002 8000 in Base16."

REFERENCE

"Section 5.3.2 and setion 7.7.2.1 Timer interval specification of [IEEE 1588-2008]"

SYNTAX OCTET STRING (SIZE (1..255))

ClockTxModeType ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"Transmission mode.

unicast. Using unicast communication channel.

multicast. Using Multicast communication channel.

multicast-mix. Using multicast-unicast communication channel"

```
SYNTAX INTEGER {
    unicast(1),
    multicast(2),
```

```
        multicastmix(3)
    }
```

ClockType ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"The clock types as defined in the MIB module description."

REFERENCE "Section 6.5.1 of [IEEE 1588-2008]."

SYNTAX INTEGER {
 ordinaryClock(1),
 boundaryClock(2),
 transparentClock(3),
 boundaryNode(4)
}

ClockPortTransportType ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"The Clock port transport type."

REFERENCE "Annex D (IPv4), Annex E (IPv6), Annex F (Ethernet),
Annex G (DeviceNET), Annex H (ControlNET) and
Annex I (IEC61158) of [IEEE 1588-2008]"

SYNTAX INTEGER {
 ipversion4(1),
 ipversion6(2),
 ethernet(3),
 deviceNET(4),
 controlNET(5),
 iec61158(6)
}

ClockPortTransportTypeAddress ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"The Clock port transport protocol address used for this
communication between the clock nodes. This is a string
corresponding to the address type as specified by the
ClockPortTransportType.

This can be address of type IP version 4, IP version 6,
Ethernet, DeviceNET, ControlNET and IEC61158."

REFERENCE "Annex D (IPv4), Annex E (IPv6), Annex F (Ethernet),
Annex G (DeviceNET), Annex H (ControlNET) and
Annex I (IEC61158) of [IEEE 1588-2008]"

SYNTAX OCTET STRING (SIZE (1..255))

ptpbasesMIBClockInfo OBJECT IDENTIFIER

::= { ptpbasesMIBObjects 2 }

```
ptpbaseSystemTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF PtpbaseSystemEntry
    MAX-ACCESS   not-accessible
    STATUS      current
    DESCRIPTION
        "Table of count information about the PTP system for all
        domains."
    ::= { ptpbaseMIBSystemInfo 1 }

ptpbaseSystemEntry OBJECT-TYPE
    SYNTAX      PtpbaseSystemEntry
    MAX-ACCESS   not-accessible
    STATUS      current
    DESCRIPTION
        "An entry in the table, containing count information about a
        single domain. New row entries are added when the PTP clock for
        this domain is configured, while the unconfiguration of the PTP
        clock removes it."
    INDEX
        {
            ptpDomainIndex,
            ptpInstanceIndex
        }
    ::= { ptpbaseSystemTable 1 }

PtpbaseSystemEntry ::= SEQUENCE {
    ptpDomainIndex      ClockDomainType,
    ptpInstanceIndex     ClockInstanceType,
    ptpDomainClockPortsTotal Gauge32
}

ptpDomainIndex OBJECT-TYPE
    SYNTAX      ClockDomainType
    MAX-ACCESS   not-accessible
    STATUS      current
    DESCRIPTION
        "This object specifies the domain number used to create logical
        group of PTP devices. The Clock Domain is a logical group of
        clocks and devices that synchronize with each other using the
        PTP protocol."

        0          Default domain
        1          Alternate domain 1
        2          Alternate domain 2
        3          Alternate domain 3
        4 - 127     User-defined domains
        128 - 255   Reserved"
    ::= { ptpbaseSystemEntry 1 }

ptpInstanceIndex OBJECT-TYPE
    SYNTAX      ClockInstanceType
    MAX-ACCESS   not-accessible
    STATUS      current
```

DESCRIPTION

"This object specifies the instance of the Clock for this domain."

::= { ptptimeSystemEntry 2 }

ptptimeDomainClockPortsTotal OBJECT-TYPE

SYNTAX Gauge32

UNITS "ntp ports"

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object specifies the total number of clock ports configured within a domain."

::= { ptptimeSystemEntry 3 }

ptptimeSystemDomainTable OBJECT-TYPE

SYNTAX SEQUENCE OF PtpptimeSystemDomainEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"Table of information about the PTP system for all clock modes -- ordinary, boundary or transparent."

::= { ptptimeMIBSystemInfo 2 }

ptptimeSystemDomainEntry OBJECT-TYPE

SYNTAX PtpptimeSystemDomainEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"An entry in the table, containing information about a single clock mode for the PTP system. A row entry gets added when PTP clocks are configured on the router."

INDEX { ptptimeSystemDomainClockTypeIndex }

::= { ptptimeSystemDomainTable 1 }

```
PtpptimeSystemDomainEntry ::= SEQUENCE {  
    ptptimeSystemDomainClockTypeIndex ClockType,  
    ptptimeSystemDomainTotals          Unsigned32  
}
```

ptptimeSystemDomainClockTypeIndex OBJECT-TYPE

SYNTAX ClockType

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"This object specifies the clock type as defined in the Textual convention description."

::= { ptptimeSystemDomainEntry 1 }

ptptimeSystemDomainTotals OBJECT-TYPE

```

SYNTAX          Unsigned32
UNITS           "domains"
MAX-ACCESS      read-only
STATUS          current
DESCRIPTION
    "This object specifies the total number of PTP domains for this
    particular clock type configured in this node."
 ::= { ptpbaseSystemDomainEntry 2 }

```

ptpbaseSystemProfile OBJECT-TYPE

```

SYNTAX          ClockProfileType
MAX-ACCESS      read-only
STATUS          current
DESCRIPTION
    "This object specifies the PTP Profile implemented on the
    system."
REFERENCE       "Section 19.3 PTP profiles of [IEEE 1588-2008]"
 ::= { ptpbaseMIBSystemInfo 3 }

```

ptpbaseClockCurrentDSTable OBJECT-TYPE

```

SYNTAX          SEQUENCE OF PtpbaseClockCurrentDSEntry
MAX-ACCESS      not-accessible
STATUS          current
DESCRIPTION
    "Table of information about the PTP clock Current Datasets for
    all domains."
 ::= { ptpbaseMIBClockInfo 1 }

```

ptpbaseClockCurrentDSEntry OBJECT-TYPE

```

SYNTAX          PtpbaseClockCurrentDSEntry
MAX-ACCESS      not-accessible
STATUS          current
DESCRIPTION
    "An entry in the table, containing information about a single
    PTP clock Current Datasets for a domain."
REFERENCE       "1588 Version 2.0 Section 8.2.2 currentDS data set member
    specifications of [IEEE 1588-2008]"
INDEX          {
                ptpbaseClockCurrentDSDomainIndex,
                ptpbaseClockCurrentDSClockTypeIndex,
                ptpbaseClockCurrentDSInstanceIndex
            }
 ::= { ptpbaseClockCurrentDSTable 1 }

```

```

PtpbaseClockCurrentDSEntry ::= SEQUENCE {
    ptpbaseClockCurrentDSDomainIndex      ClockDomainType,
    ptpbaseClockCurrentDSClockTypeIndex   ClockType,
    ptpbaseClockCurrentDSInstanceIndex    ClockInstanceType,
    ptpbaseClockCurrentDSStepsRemoved     Unsigned32,

```

```

    ptptimeClockCurrentDSOffsetFromMaster ClockTimeInterval,
    ptptimeClockCurrentDSMeanPathDelay    ClockTimeInterval
}

ptptimeClockCurrentDSDomainIndex OBJECT-TYPE
    SYNTAX          ClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the domain number used to create logical
        group of PTP devices."
    ::= { ptptimeClockCurrentDSEntry 1 }

ptptimeClockCurrentDSClockTypeIndex OBJECT-TYPE
    SYNTAX          ClockType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the clock type as defined in the
        Textual convention description."
    ::= { ptptimeClockCurrentDSEntry 2 }

ptptimeClockCurrentDSInstanceIndex OBJECT-TYPE
    SYNTAX          ClockInstanceType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the instance of the clock for this clock
        type in the given domain."
    ::= { ptptimeClockCurrentDSEntry 3 }

ptptimeClockCurrentDSStepsRemoved OBJECT-TYPE
    SYNTAX          Unsigned32
    UNITS           "Steps"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The current clock dataset StepsRemoved value.

        This object specifies the distance measured by the number of
        Boundary clocks between the local clock and the Foreign master
        as indicated in the stepsRemoved field of Announce messages."
    REFERENCE       "1588 Version 2.0 Section 8.2.2.2 stepsRemoved"
    ::= { ptptimeClockCurrentDSEntry 4 }

ptptimeClockCurrentDSOffsetFromMaster OBJECT-TYPE
    SYNTAX          ClockTimeInterval
    UNITS           "Time Interval"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the current clock dataset ClockOffset

```

value. The value of the computation of the offset in time between a slave and a master clock."

REFERENCE "1588 Version 2.0 Section 8.2.2.3 of
[IEEE 1588-2008]"

::= { ntpbaseClockCurrentDSEntry 5 }

ntpbaseClockCurrentDSMeanPathDelay OBJECT-TYPE

SYNTAX ClockTimeInterval

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object specifies the current clock dataset MeanPathDelay value.

The mean path delay between a pair of ports as measure by the delay request-response mechanism."

REFERENCE "1588 Version 2.0 Section 8.2.2.4 mean path delay"

::= { ntpbaseClockCurrentDSEntry 6 }

ntpbaseClockParentDSTable OBJECT-TYPE

SYNTAX SEQUENCE OF NtpbaseClockParentDSEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"Table of information about the PTP clock Parent Datasets for all domains."

::= { ntpbaseMIBClockInfo 2 }

ntpbaseClockParentDSEntry OBJECT-TYPE

SYNTAX NtpbaseClockParentDSEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"An entry in the table, containing information about a single PTP clock Parent Datasets for a domain."

REFERENCE

"Section 8.2.3 parentDS data set member specifications of [IEEE 1588-2008]"

INDEX {
 ntpbaseClockParentDSDomainIndex,
 ntpbaseClockParentDSClockTypeIndex,
 ntpbaseClockParentDSInstanceIndex
}

::= { ntpbaseClockParentDSTable 1 }

NtpbaseClockParentDSEntry ::= SEQUENCE {

ntpbaseClockParentDSDomainIndex	ClockDomainType,
ntpbaseClockParentDSClockTypeIndex	ClockType,
ntpbaseClockParentDSInstanceIndex	ClockInstanceType,
ntpbaseClockParentDSParentPortIdentity	OCTET STRING,


```

    ptptimeClockParentDSParentStats      TruthValue,
    ptptimeClockParentDSOffset            ClockIntervalBase2,
    ptptimeClockParentDSClockPhChRate     Integer32,
    ptptimeClockParentDSGMClockIdentity   ClockIdentity,
    ptptimeClockParentDSGMClockPriority1   Unsigned32,
    ptptimeClockParentDSGMClockPriority2   Unsigned32,
    ptptimeClockParentDSGMClockQualityClass ClockQualityClassType,
    ptptimeClockParentDSGMClockQualityAccuracy ClockQualityAccuracyType,
    ptptimeClockParentDSGMClockQualityOffset Unsigned32
}

```

ptptimeClockParentDSDomainIndex OBJECT-TYPE

SYNTAX ClockDomainType

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"This object specifies the domain number used to create logical group of PTP devices."

::= { ptptimeClockParentDSEntry 1 }

ptptimeClockParentDSClockTypeIndex OBJECT-TYPE

SYNTAX ClockType

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"This object specifies the clock type as defined in the Textual convention description."

::= { ptptimeClockParentDSEntry 2 }

ptptimeClockParentDSInstanceIndex OBJECT-TYPE

SYNTAX ClockInstanceType

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"This object specifies the instance of the clock for this clock type in the given domain."

::= { ptptimeClockParentDSEntry 3 }

ptptimeClockParentDSParentPortIdentity OBJECT-TYPE

SYNTAX OCTET STRING(SIZE(1..256))

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object specifies the value of portIdentity of the port on the master that issues the Sync messages used in synchronizing this clock."

REFERENCE

"Section 8.2.3.2 parentDS.parentPortIdentity of [IEEE 1588-2008]"

::= { ptptimeClockParentDSEntry 4 }

ptptimeClockParentDSParentStats OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object specifies the Parent Dataset ParentStats value.

This value indicates whether the values of ParentDSOffset and ParentDSClockPhChRate have been measured and are valid. A TRUE value shall indicate valid data."

REFERENCE "Section 8.2.3.3 parentDS.parentStats of [IEEE 1588-2008]"

::= { ptptimeClockParentDSEntry 5 }

ptptimeClockParentDSOffset OBJECT-TYPE

SYNTAX ClockIntervalBase2 (-128..127)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object specifies the Parent Dataset ParentOffsetScaledLogVariance value.

This value is the variance of the parent clocks phase as measured by the local clock."

REFERENCE

"Section 8.2.3.4 parentDS.observedParentOffsetScaledLogVariance [IEEE 1588-2008]"

::= { ptptimeClockParentDSEntry 6 }

ptptimeClockParentDSClockPhChRate OBJECT-TYPE

SYNTAX Integer32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object specifies the clock's parent dataset ParentClockPhaseChangeRate value.

This value is an estimate of the parent clocks phase change rate as measured by the slave clock."

REFERENCE

"Section 8.2.3.5 parentDS.observedParentClockPhaseChangeRate of [IEEE 1588-2008]"

::= { ptptimeClockParentDSEntry 7 }

ptptimeClockParentDSGMClockIdentity OBJECT-TYPE

SYNTAX ClockIdentity

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object specifies the parent dataset Grandmaster clock identity."

REFERENCE

"Section 8.2.3.6 parentDS.grandmasterIdentity of
[IEEE 1588-2008]"

::= { ptpbaseClockParentDSEntry 8 }

ptpbaseClockParentDSGMClockPriority1 OBJECT-TYPE

SYNTAX Unsigned32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object specifies the parent dataset Grandmaster clock
priority1."

REFERENCE

"Section 8.2.3.8 parentDS.grandmasterPriority1 of
[IEEE 1588-2008]"

::= { ptpbaseClockParentDSEntry 9 }

ptpbaseClockParentDSGMClockPriority2 OBJECT-TYPE

SYNTAX Unsigned32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object specifies the parent dataset grandmaster clock
priority2."

REFERENCE

"Section 8.2.3.9 parentDS.grandmasterPriority2 of
[IEEE 1588-2008]"

::= { ptpbaseClockParentDSEntry 10 }

ptpbaseClockParentDSGMClockQualityClass OBJECT-TYPE

SYNTAX ClockQualityClassType (0..255)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object specifies the parent dataset grandmaster clock
quality class."

REFERENCE

"Section 8.2.3.7 parentDS.grandmasterClockQuality of
[IEEE 1588-2008]"

::= { ptpbaseClockParentDSEntry 11 }

ptpbaseClockParentDSGMClockQualityAccuracy OBJECT-TYPE

SYNTAX ClockQualityAccuracyType

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object specifies the parent dataset grandmaster clock
quality accuracy."

REFERENCE

"Section 8.2.3.7 parentDS.grandmasterClockQuality of
[IEEE 1588-2008]"

::= { ptpbaseClockParentDSEntry 12 }

ptpbasedClockParentDSGMClockQualityOffset OBJECT-TYPE

SYNTAX Unsigned32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object specifies the parent dataset grandmaster clock quality offset."

REFERENCE

"Section 8.2.3.7 parentDS.grandmasterClockQuality of [IEEE 1588-2008]"

::= { ptpbasedClockParentDSEntry 13 }

ptpbasedClockDefaultDSTable OBJECT-TYPE

SYNTAX SEQUENCE OF PtpbasedClockDefaultDSEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"Table of information about the PTP clock Default Datasets for all domains."

::= { ptpbasedMIBClockInfo 3 }

ptpbasedClockDefaultDSEntry OBJECT-TYPE

SYNTAX PtpbasedClockDefaultDSEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"An entry in the table, containing information about a single PTP clock Default Datasets for a domain."

INDEX {
 ptpbasedClockDefaultDSDomainIndex,
 ptpbasedClockDefaultDSClockTypeIndex,
 ptpbasedClockDefaultDSInstanceIndex
 }

::= { ptpbasedClockDefaultDSTable 1 }

PtpbasedClockDefaultDSEntry ::= SEQUENCE {

ptpbasedClockDefaultDSDomainIndex	ClockDomainType,
ptpbasedClockDefaultDSClockTypeIndex	ClockType,
ptpbasedClockDefaultDSInstanceIndex	ClockInstanceType,
ptpbasedClockDefaultDSTwoStepFlag	TruthValue,
ptpbasedClockDefaultDSClockIdentity	ClockIdentity,
ptpbasedClockDefaultDSPriority1	Unsigned32,
ptpbasedClockDefaultDSPriority2	Unsigned32,
ptpbasedClockDefaultDSSlaveOnly	TruthValue,
ptpbasedClockDefaultDSQualityClass	ClockQualityClassType,
ptpbasedClockDefaultDSQualityAccuracy	ClockQualityAccuracyType,
ptpbasedClockDefaultDSQualityOffset	Integer32

}

ptpbasedefaultDSDomainIndex OBJECT-TYPE
SYNTAX ClockDomainType
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
 "This object specifies the domain number used to create logical
 group of PTP devices."
 ::= { ptpbasedefaultDSEntry 1 }

ptpbasedefaultDSClockTypeIndex OBJECT-TYPE
SYNTAX ClockType
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
 "This object specifies the clock type as defined in the
 Textual convention description."
 ::= { ptpbasedefaultDSEntry 2 }

ptpbasedefaultDSInstanceIndex OBJECT-TYPE
SYNTAX ClockInstanceType
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
 "This object specifies the instance of the clock for this clock
 type in the given domain."
 ::= { ptpbasedefaultDSEntry 3 }

ptpbasedefaultDSTwoStepFlag OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "This object specifies whether the Two Step process is used."
 ::= { ptpbasedefaultDSEntry 4 }

ptpbasedefaultDSClockIdentity OBJECT-TYPE
SYNTAX ClockIdentity
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "This object specifies the default Datasets clock identity."
 ::= { ptpbasedefaultDSEntry 5 }

ptpbasedefaultDSPriority1 OBJECT-TYPE
SYNTAX Unsigned32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "This object specifies the default Datasets clock Priority1."
 ::= { ptpbasedefaultDSEntry 6 }

ptpbasedefaultDSPriority2 OBJECT-TYPE

```
SYNTAX          Unsigned32
MAX-ACCESS      read-only
STATUS          current
DESCRIPTION
    "This object specifies the default Datasets clock Priority2."
 ::= { ptptimeClockDefaultDSEntry 7 }

ptptimeClockDefaultDSSlaveOnly OBJECT-TYPE
SYNTAX          TruthValue
MAX-ACCESS      read-only
STATUS          current
DESCRIPTION
    "Whether the SlaveOnly flag is set."
 ::= { ptptimeClockDefaultDSEntry 8 }

ptptimeClockDefaultDSQualityClass OBJECT-TYPE
SYNTAX          ClockQualityClassType (0..255)
MAX-ACCESS      read-only
STATUS          current
DESCRIPTION
    "This object specifies the default dataset Quality Class."
 ::= { ptptimeClockDefaultDSEntry 9 }

ptptimeClockDefaultDSQualityAccuracy OBJECT-TYPE
SYNTAX          ClockQualityAccuracyType
MAX-ACCESS      read-only
STATUS          current
DESCRIPTION
    "This object specifies the default dataset Quality Accuracy."
 ::= { ptptimeClockDefaultDSEntry 10 }

ptptimeClockDefaultDSQualityOffset OBJECT-TYPE
SYNTAX          Integer32
MAX-ACCESS      read-only
STATUS          current
DESCRIPTION
    "This object specifies the default dataset Quality offset."
 ::= { ptptimeClockDefaultDSEntry 11 }

ptptimeClockRunningTable OBJECT-TYPE
SYNTAX          SEQUENCE OF PtpbaseClockRunningEntry
MAX-ACCESS      not-accessible
STATUS          current
DESCRIPTION
    "Table of information about the PTP clock Running Datasets for
     all domains."
 ::= { ptptimeMIBClockInfo 4 }

ptptimeClockRunningEntry OBJECT-TYPE
SYNTAX          PtpbaseClockRunningEntry
```

```

MAX-ACCESS      not-accessible
STATUS          current
DESCRIPTION
    "An entry in the table, containing information about a single
    PTP clock running Datasets for a domain."
INDEX           {
                ptptimeClockRunningDomainIndex,
                ptptimeClockRunningClockTypeIndex,
                ptptimeClockRunningInstanceIndex
                }
 ::= { ptptimeClockRunningTable 1 }

PtpptimeClockRunningEntry ::= SEQUENCE {
    ptptimeClockRunningDomainIndex      ClockDomainType,
    ptptimeClockRunningClockTypeIndex   ClockType,
    ptptimeClockRunningInstanceIndex    ClockInstanceType,
    ptptimeClockRunningState             ClockStateType,
    ptptimeClockRunningPacketsSent      Counter64,
    ptptimeClockRunningPacketsReceived Counter64
}

ptptimeClockRunningDomainIndex OBJECT-TYPE
    SYNTAX      ClockDomainType
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This object specifies the domain number used to create logical
        group of PTP devices."
    ::= { ptptimeClockRunningEntry 1 }

ptptimeClockRunningClockTypeIndex OBJECT-TYPE
    SYNTAX      ClockType
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This object specifies the clock type as defined in the
        Textual convention description."
    ::= { ptptimeClockRunningEntry 2 }

ptptimeClockRunningInstanceIndex OBJECT-TYPE
    SYNTAX      ClockInstanceType
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This object specifies the instance of the clock for this clock
        type in the given domain."
    ::= { ptptimeClockRunningEntry 3 }

ptptimeClockRunningState OBJECT-TYPE
    SYNTAX      ClockStateType
    MAX-ACCESS  read-only
    STATUS      current

```

DESCRIPTION

"This object specifies the Clock state returned by PTP engine which was described earlier.

Freerun state. Applies to a slave device that is not locked to a master. This is the initial state a slave starts out with when it is not getting any PTP packets from the master or because of some other input error (erroneous packets, etc).

Holdover state. In this state the slave device is locked to a master but communication with the master is lost or the timestamps in the ptp packets are incorrect. But since the slave was locked to the master, it can run with the same accuracy for sometime. The slave can continue to operate in this state for some time. If communication with the master is not restored for a while, the device is moved to the FREERUN state.

Acquiring state. The slave device is receiving packets from a master and is trying to acquire a lock.

Freq_locked state. Slave device is locked to the Master with respect to frequency, but not phase aligned

Phase_aligned state. Locked to the master with respect to frequency and phase."

```
::= { ptpbaseClockRunningEntry 4 }
```

ptpbaseClockRunningPacketsSent OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object specifies the total number of all packet Unicast and multicast that have been sent out for this clock in this domain for this type."

```
::= { ptpbaseClockRunningEntry 5 }
```

ptpbaseClockRunningPacketsReceived OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object specifies the total number of all packet Unicast and multicast that have been received for this clock in this domain for this type."

```
::= { ptpbaseClockRunningEntry 6 }
```

ptpbaseClockTimePropertiesDSTable OBJECT-TYPE

SYNTAX SEQUENCE OF PtpbaseClockTimePropertiesDSEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
 "Table of information about the PTP clock Timeproperties
 Datasets for all domains."
 ::= { ptpbaseMIBClockInfo 5 }

ptpbaseClockTimePropertiesDSEntry OBJECT-TYPE

SYNTAX PtpbaseClockTimePropertiesDSEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
 "An entry in the table, containing information about a single
 PTP clock timeproperties Datasets for a domain."
 REFERENCE "Section 8.2.4 of [IEEE 1588-2008]"
 INDEX {
 ptpbaseClockTimePropertiesDSDomainIndex,
 ptpbaseClockTimePropertiesDSClockTypeIndex,
 ptpbaseClockTimePropertiesDSInstanceIndex
 }
 ::= { ptpbaseClockTimePropertiesDSTable 1 }

PtpbaseClockTimePropertiesDSEntry ::= SEQUENCE {
 ptpbaseClockTimePropertiesDSDomainIndex ClockDomainType,
 ptpbaseClockTimePropertiesDSClockTypeIndex ClockType,
 ptpbaseClockTimePropertiesDSInstanceIndex ClockInstanceType,
 ptpbaseClockTimePropertiesDSCurrentUTCOffsetValid TruthValue,
 ptpbaseClockTimePropertiesDSCurrentUTCOffset Integer32,
 ptpbaseClockTimePropertiesDSLeap59 TruthValue,
 ptpbaseClockTimePropertiesDSLeap61 TruthValue,
 ptpbaseClockTimePropertiesDSTimeTraceable TruthValue,
 ptpbaseClockTimePropertiesDSFreqTraceable TruthValue,
 ptpbaseClockTimePropertiesDSPTPTimescale TruthValue,
 ptpbaseClockTimePropertiesDSSource ClockTimeSourceType
 }

ptpbaseClockTimePropertiesDSDomainIndex OBJECT-TYPE

SYNTAX ClockDomainType
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
 "This object specifies the domain number used to create logical
 group of PTP devices."
 ::= { ptpbaseClockTimePropertiesDSEntry 1 }

ptpbaseClockTimePropertiesDSClockTypeIndex OBJECT-TYPE

SYNTAX ClockType
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
 "This object specifies the clock type as defined in the

Textual convention description."
 ::= { ptptimePropertiesDSEntry 2 }

ptptimePropertiesDSInstanceIndex OBJECT-TYPE
SYNTAX ClockInstanceType
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
 "This object specifies the instance of the clock for this clock
 type in the given domain."
 ::= { ptptimePropertiesDSEntry 3 }

ptptimePropertiesDSCurrentUTCOffsetValid OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "This object specifies the timeproperties dataset value of
 whether current UTC offset is valid."
REFERENCE "Section 8.2.4.2 of [IEEE 1588-2008]"
 ::= { ptptimePropertiesDSEntry 4 }

ptptimePropertiesDSCurrentUTCOffset OBJECT-TYPE
SYNTAX Integer32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "This object specifies the timeproperties dataset value of
 current UTC offset.

 In PTP systems whose epoch is the PTP epoch, the value of
 timePropertiesDS.currentUtcOffset is the offset
 between TAI and UTC; otherwise the value has no meaning. The
 value shall be in units of seconds.
 The initialization value shall be selected as follows:
 a) If the timePropertiesDS.ptpTimescale (see 8.2.4.8) is TRUE,
 the value is the value obtained from a
 primary reference if the value is known at the time of
 initialization, else.
 b) The value shall be the current number of leap seconds (7.2.3)
 when the node is designed."
REFERENCE "Section 8.2.4.3 of [IEEE 1588-2008]"
 ::= { ptptimePropertiesDSEntry 5 }

ptptimePropertiesDSLeap59 OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "This object specifies the Leap59 value in the clock Current
 Dataset."
REFERENCE "Section 8.2.4.4 of [IEEE 1588-2008]"

```
 ::= { ptptimeClockTimePropertiesDSEntry 6 }

ptptimeClockTimePropertiesDSLeap61 OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "This object specifies the Leap61 value in the clock Current
        Dataset."
    REFERENCE    "Section 8.2.4.5 of [IEEE 1588-2008]"
    ::= { ptptimeClockTimePropertiesDSEntry 7 }

ptptimeClockTimePropertiesDSTimeTraceable OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "This object specifies the Timetraceable value in the clock
        Current Dataset."
    REFERENCE    "Section 8.2.4.6 of [IEEE 1588-2008]"
    ::= { ptptimeClockTimePropertiesDSEntry 8 }

ptptimeClockTimePropertiesDSFreqTraceable OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "This object specifies the Frequency Traceable value in the
        clock Current Dataset."
    REFERENCE    "Section 8.2.4.7 of [IEEE 1588-2008]"
    ::= { ptptimeClockTimePropertiesDSEntry 9 }

ptptimeClockTimePropertiesDSPTPTimescale OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "This object specifies the PTP Timescale value in the clock
        Current Dataset."
    REFERENCE    "Section 8.2.4.8 of [IEEE 1588-2008]"
    ::= { ptptimeClockTimePropertiesDSEntry 10 }

ptptimeClockTimePropertiesDSSource OBJECT-TYPE
    SYNTAX      ClockTimeSourceType
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "This object specifies the Timesource value in the clock Current
        Dataset."
    REFERENCE    "Section 8.2.4.9 of [IEEE 1588-2008]"
    ::= { ptptimeClockTimePropertiesDSEntry 11 }
```

```
ptpbasedClockTransDefaultDSTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF PtpbasedClockTransDefaultDSEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Table of information about the PTP Transparent clock Default
        Datasets for all domains."
    ::= { ptpbaseMIBClockInfo 6 }

ptpbasedClockTransDefaultDSEntry OBJECT-TYPE
    SYNTAX          PtpbasedClockTransDefaultDSEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry in the table, containing information about a single
        PTP Transparent clock Default Datasets for a domain."
    REFERENCE       "Section 8.3.2 of [IEEE 1588-2008]"
    INDEX           {
                    ptpbasedClockTransDefaultDSDomainIndex,
                    ptpbasedClockTransDefaultDSInstanceIndex
                }
    ::= { ptpbasedClockTransDefaultDSTable 1 }

PtpbasedClockTransDefaultDSEntry ::= SEQUENCE {
    ptpbasedClockTransDefaultDSDomainIndex  ClockDomainType,
    ptpbasedClockTransDefaultDSInstanceIndex ClockInstanceType,
    ptpbasedClockTransDefaultDSClockIdentity ClockIdentity,
    ptpbasedClockTransDefaultDSNumOfPorts   Counter32,
    ptpbasedClockTransDefaultDSDelay        ClockMechanismType,
    ptpbasedClockTransDefaultDSPrimaryDomain Integer32
}

ptpbasedClockTransDefaultDSDomainIndex OBJECT-TYPE
    SYNTAX          ClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the domain number used to create logical
        group of PTP devices."
    ::= { ptpbasedClockTransDefaultDSEntry 1 }

ptpbasedClockTransDefaultDSInstanceIndex OBJECT-TYPE
    SYNTAX          ClockInstanceType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the instance of the clock for this clock
        type in the given domain."
    ::= { ptpbasedClockTransDefaultDSEntry 2 }
```

ptpbasedClockTransDefaultDSClockIdentity OBJECT-TYPE
SYNTAX ClockIdentity
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "This object specifies the value of the clockIdentity attribute
 of the local clock."
REFERENCE "Section 8.3.2.2.1 of [IEEE 1588-2008]"
::= { ptpbasedClockTransDefaultDSEntry 3 }

ptpbasedClockTransDefaultDSNumOfPorts OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "This object specifies the number of PTP ports of the device."
REFERENCE "Section 8.3.2.2.2 of [IEEE 1588-2008]"
::= { ptpbasedClockTransDefaultDSEntry 4 }

ptpbasedClockTransDefaultDSDelay OBJECT-TYPE
SYNTAX ClockMechanismType
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "This object, if the transparent clock is an end-to-end
 transparent clock, has the value shall be E2E; If the
 transparent clock is a peer-to-peer transparent clock, the
 value
 shall be P2P."
REFERENCE "Section 8.3.2.3.1 of [IEEE 1588-2008]"
::= { ptpbasedClockTransDefaultDSEntry 5 }

ptpbasedClockTransDefaultDSPrimaryDomain OBJECT-TYPE
SYNTAX Integer32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "This object specifies the value of the primary syntonization
 domain. The initialization value shall be 0."
REFERENCE "Section 8.3.2.3.2 of [IEEE 1588-2008]"
::= { ptpbasedClockTransDefaultDSEntry 6 }

ptpbasedClockPortTable OBJECT-TYPE
SYNTAX SEQUENCE OF PtpbasedClockPortEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
 "Table of information about the clock ports for a particular
 domain."
::= { ptpbasedMIBClockInfo 7 }

```

ptpbasedClockPortEntry OBJECT-TYPE
    SYNTAX          PtpbasedClockPortEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry in the table, containing information about a single
        clock port."
    INDEX
        {
            ptpbasedClockPortDomainIndex,
            ptpbasedClockPortClockTypeIndex,
            ptpbasedClockPortClockInstanceIndex,
            ptpbasedClockPortTablePortNumberIndex
        }
    ::= { ptpbasedClockPortTable 1 }

PtpbasedClockPortEntry ::= SEQUENCE {
    ptpbasedClockPortDomainIndex      ClockDomainType,
    ptpbasedClockPortClockTypeIndex   ClockType,
    ptpbasedClockPortClockInstanceIndex ClockInstanceType,
    ptpbasedClockPortTablePortNumberIndex ClockPortNumber,
    ptpbasedClockPortName             DisplayString,
    ptpbasedClockPortRole              ClockRoleType,
    ptpbasedClockPortSyncOneStep       TruthValue,
    ptpbasedClockPortCurrentPeerAddressType ClockPortTransportType,
    ptpbasedClockPortCurrentPeerAddress ClockPortTransportTypeAddress,
    ptpbasedClockPortNumOfAssociatedPorts Gauge32
}

ptpbasedClockPortDomainIndex OBJECT-TYPE
    SYNTAX          ClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the domain number used to create logical
        group of PTP devices."
    ::= { ptpbasedClockPortEntry 1 }

ptpbasedClockPortClockTypeIndex OBJECT-TYPE
    SYNTAX          ClockType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the clock type as defined in the
        Textual convention description."
    ::= { ptpbasedClockPortEntry 2 }

ptpbasedClockPortClockInstanceIndex OBJECT-TYPE
    SYNTAX          ClockInstanceType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION

```

```
        "This object specifies the instance of the clock for this clock
        type in the given domain."
 ::= { ptptimeClockPortEntry 3 }

ptptimeClockPortTablePortNumberIndex OBJECT-TYPE
    SYNTAX      ClockPortNumber (1..65535)
    MAX-ACCESS   not-accessible
    STATUS      current
    DESCRIPTION
        "This object specifies the PTP Portnumber for this port."
 ::= { ptptimeClockPortEntry 4 }

ptptimeClockPortName OBJECT-TYPE
    SYNTAX      DisplayString (SIZE (1..64))
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "This object specifies the PTP clock port name configured on the
        router."
 ::= { ptptimeClockPortEntry 5 }

ptptimeClockPortRole OBJECT-TYPE
    SYNTAX      ClockRoleType
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "This object describes the current role (slave/master) of the
        port."
 ::= { ptptimeClockPortEntry 6 }

ptptimeClockPortSyncOneStep OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "This object specifies that one-step clock operation between
        the PTP master and slave device is enabled."
 ::= { ptptimeClockPortEntry 7 }

ptptimeClockPortCurrentPeerAddressType OBJECT-TYPE
    SYNTAX      ClockPortTransportType
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "This object specifies the current peer's network address used
        for PTP communication."
 ::= { ptptimeClockPortEntry 8 }

ptptimeClockPortCurrentPeerAddress OBJECT-TYPE
    SYNTAX      ClockPortTransportTypeAddress
    MAX-ACCESS   read-only
    STATUS      current
```

DESCRIPTION

"This object specifies the current peer's network address used for PTP communication."

::= { ptpbaseClockPortEntry 9 }

ptpbaseClockPortNumOfAssociatedPorts OBJECT-TYPE

SYNTAX Gauge32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object specifies -

For a master port - the number of PTP slave sessions (peers) associated with this PTP port.

For a slave port - the number of masters available to this slave port (might or might not be peered)."

::= { ptpbaseClockPortEntry 10 }

ptpbaseClockPortDSTable OBJECT-TYPE

SYNTAX SEQUENCE OF PtpbaseClockPortDSEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"Table of information about the clock ports dataset for a particular domain."

::= { ptpbaseMIBClockInfo 8 }

ptpbaseClockPortDSEntry OBJECT-TYPE

SYNTAX PtpbaseClockPortDSEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"An entry in the table, containing port dataset information for a single clock port."

```
INDEX {
    ptpbaseClockPortDSDomainIndex,
    ptpbaseClockPortDSClockTypeIndex,
    ptpbaseClockPortDSClockInstanceIndex,
    ptpbaseClockPortDSPortNumberIndex
}
```

::= { ptpbaseClockPortDSTable 1 }

PtpbaseClockPortDSEntry ::= SEQUENCE {

ptpbaseClockPortDSDomainIndex	ClockDomainType,
ptpbaseClockPortDSClockTypeIndex	ClockType,
ptpbaseClockPortDSClockInstanceIndex	ClockInstanceType,
ptpbaseClockPortDSPortNumberIndex	ClockPortNumber,
ptpbaseClockPortDSName	DisplayString,
ptpbaseClockPortDSPortIdentity	OCTET STRING,
ptpbaseClockPortDSAnnouncementInterval	Integer32,
ptpbaseClockPortDSAnnounceRctTimeout	Integer32,


```
    ptpbaseClockPortDSSyncInterval          Integer32,
    ptpbaseClockPortDSMinDelayReqInterval   Integer32,
    ptpbaseClockPortDSPeerDelayReqInterval  Integer32,
    ptpbaseClockPortDSDelayMech             ClockMechanismType,
    ptpbaseClockPortDSPeerMeanPathDelay     ClockTimeInterval,
    ptpbaseClockPortDSGrantDuration         Unsigned32,
    ptpbaseClockPortDSPTPVersion            Integer32
}

ptpbaseClockPortDSDomainIndex OBJECT-TYPE
    SYNTAX          ClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the domain number used to create logical
        group of PTP devices."
    ::= { ptpbaseClockPortDSEntry 1 }

ptpbaseClockPortDSClockTypeIndex OBJECT-TYPE
    SYNTAX          ClockType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the clock type as defined in the
        Textual convention description."
    ::= { ptpbaseClockPortDSEntry 2 }

ptpbaseClockPortDSClockInstanceIndex OBJECT-TYPE
    SYNTAX          ClockInstanceType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the instance of the clock for this clock
        type in the given domain."
    ::= { ptpbaseClockPortDSEntry 3 }

ptpbaseClockPortDSPortNumberIndex OBJECT-TYPE
    SYNTAX          ClockPortNumber (1..65535)
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the PTP portnumber associated with this
        PTP port."
    ::= { ptpbaseClockPortDSEntry 4 }

ptpbaseClockPortDSName OBJECT-TYPE
    SYNTAX          DisplayString (SIZE (1..64))
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the PTP clock port name."
    ::= { ptpbaseClockPortDSEntry 5 }
```

ptpbasedClockPortDSPortIdentity OBJECT-TYPE
SYNTAX OCTET STRING(SIZE(1..256))
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This object specifies the PTP clock port Identity."
::= { ptpbasedClockPortDSEntry 6 }

ptpbasedClockPortDSAnnouncementInterval OBJECT-TYPE
SYNTAX Integer32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This object specifies the Announce message transmission
interval associated with this clock port."
::= { ptpbasedClockPortDSEntry 7 }

ptpbasedClockPortDSAnnounceRctTimeout OBJECT-TYPE
SYNTAX Integer32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This object specifies the Announce receipt timeout associated
with this clock port."
::= { ptpbasedClockPortDSEntry 8 }

ptpbasedClockPortDSSyncInterval OBJECT-TYPE
SYNTAX Integer32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This object specifies the Sync message transmission interval."
::= { ptpbasedClockPortDSEntry 9 }

ptpbasedClockPortDSMinDelayReqInterval OBJECT-TYPE
SYNTAX Integer32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This object specifies the Delay_Req message transmission
interval."
::= { ptpbasedClockPortDSEntry 10 }

ptpbasedClockPortDSPeerDelayReqInterval OBJECT-TYPE
SYNTAX Integer32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This object specifies the Pdelay_Req message transmission
interval."
::= { ptpbasedClockPortDSEntry 11 }

ptpbasedClockPortDSDelayMech OBJECT-TYPE
SYNTAX ClockMechanismType
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "This object specifies the delay mechanism used. If the clock
 is an end-to-end clock, the value of the is e2e, else if the
 clock is a peer to-peer clock, the value shall be p2p."
 ::= { ptpbasedClockPortDSEntry 12 }

ptpbasedClockPortDSPeerMeanPathDelay OBJECT-TYPE
SYNTAX ClockTimeInterval
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "This object specifies the peer meanPathDelay."
 ::= { ptpbasedClockPortDSEntry 13 }

ptpbasedClockPortDSGrantDuration OBJECT-TYPE
SYNTAX Unsigned32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "This object specifies the grant duration allocated by the
 master."
 ::= { ptpbasedClockPortDSEntry 14 }

ptpbasedClockPortDSPTPVersion OBJECT-TYPE
SYNTAX Integer32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "This object specifies the PTP version being used."
 ::= { ptpbasedClockPortDSEntry 15 }

ptpbasedClockPortRunningTable OBJECT-TYPE
SYNTAX SEQUENCE OF PtpbasedClockPortRunningEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
 "Table of information about the clock ports running dataset for
 a particular domain."
 ::= { ptpbasedMIBClockInfo 9 }

ptpbasedClockPortRunningEntry OBJECT-TYPE
SYNTAX PtpbasedClockPortRunningEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION

"An entry in the table, containing running dataset information about a single clock port."

```

INDEX      {
            ptptimeClockPortRunningDomainIndex,
            ptptimeClockPortRunningClockTypeIndex,
            ptptimeClockPortRunningClockInstanceIndex,
            ptptimeClockPortRunningPortNumberIndex
            }
 ::= { ptptimeClockPortRunningTable 1 }

PtpbaseClockPortRunningEntry ::= SEQUENCE {
    ptptimeClockPortRunningDomainIndex      ClockDomainType,
    ptptimeClockPortRunningClockTypeIndex   ClockType,
    ptptimeClockPortRunningClockInstanceIndex ClockInstanceType,
    ptptimeClockPortRunningPortNumberIndex  ClockPortNumber,
    ptptimeClockPortRunningName             DisplayString,
    ptptimeClockPortRunningState            ClockPortState,
    ptptimeClockPortRunningRole             ClockRoleType,
    ptptimeClockPortRunningInterfaceIndex   InterfaceIndexOrZero,
    ptptimeClockPortRunningTransport        ClockPortTransportType,
    ptptimeClockPortRunningEncapsulationType Integer32,
    ptptimeClockPortRunningTxMode           ClockTxModeType,
    ptptimeClockPortRunningRxMode           ClockTxModeType,
    ptptimeClockPortRunningPacketsReceived  Counter64,
    ptptimeClockPortRunningPacketsSent     Counter64
}

ptptimeClockPortRunningDomainIndex OBJECT-TYPE
    SYNTAX      ClockDomainType
    MAX-ACCESS   not-accessible
    STATUS      current
    DESCRIPTION
        "This object specifies the domain number used to create logical
        group of PTP devices."
    ::= { ptptimeClockPortRunningEntry 1 }

ptptimeClockPortRunningClockTypeIndex OBJECT-TYPE
    SYNTAX      ClockType
    MAX-ACCESS   not-accessible
    STATUS      current
    DESCRIPTION
        "This object specifies the clock type as defined in the
        Textual convention description."
    ::= { ptptimeClockPortRunningEntry 2 }

ptptimeClockPortRunningClockInstanceIndex OBJECT-TYPE
    SYNTAX      ClockInstanceType
    MAX-ACCESS   not-accessible
    STATUS      current
    DESCRIPTION
        "This object specifies the instance of the clock for this clock
        type in the given domain."

```

```
::= { ptptimeClockPortRunningEntry 3 }
```

```
ptptimeClockPortRunningPortNumberIndex OBJECT-TYPE
```

```
SYNTAX          ClockPortNumber (1..65535)
```

```
MAX-ACCESS      not-accessible
```

```
STATUS          current
```

```
DESCRIPTION
```

```
"This object specifies the PTP portnumber associated with this
clock port."
```

```
::= { ptptimeClockPortRunningEntry 4 }
```

```
ptptimeClockPortRunningName OBJECT-TYPE
```

```
SYNTAX          DisplayString (SIZE (1..64))
```

```
MAX-ACCESS      read-only
```

```
STATUS          current
```

```
DESCRIPTION
```

```
"This object specifies the PTP clock port name."
```

```
::= { ptptimeClockPortRunningEntry 5 }
```

```
ptptimeClockPortRunningState OBJECT-TYPE
```

```
SYNTAX          ClockPortState
```

```
MAX-ACCESS      read-only
```

```
STATUS          current
```

```
DESCRIPTION
```

```
"This object specifies the port state returned by PTP engine."
```

```
initializing - In this state a port initializes
               its data sets, hardware, and
               communication facilities.
faulty       - The fault state of the protocol.
disabled     - The port shall not place any
               messages on its communication path.
listening    - The port is waiting for the
               announceReceiptTimeout to expire or
               to receive an Announce message from
               a master.
preMaster    - The port shall behave in all respects
               as though it were in the MASTER state
               except that it shall not place any
               messages on its communication path
               except for Pdelay_Req, Pdelay_Resp,
               Pdelay_Resp_Follow_Up, signaling, or
               management messages.
master       - The port is behaving as a master port.
passive      - The port shall not place any
               messages on its communication path
               except for Pdelay_Req, Pdelay_Resp,
               Pdelay_Resp_Follow_Up, or signaling
               messages, or management messages
               that are a required response to
               another management message
uncalibrated - The local port is preparing to
```

```

        synchronize to the master port.
    slave      - The port is synchronizing to the
                  selected master port."
 ::= { ptpbaseClockPortRunningEntry 6 }

ptpbaseClockPortRunningRole OBJECT-TYPE
    SYNTAX      ClockRoleType
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "This object specifies the Clock Role."
 ::= { ptpbaseClockPortRunningEntry 7 }

ptpbaseClockPortRunningInterfaceIndex OBJECT-TYPE
    SYNTAX      InterfaceIndexOrZero
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "This object specifies the interface on the router being used by
         the PTP Clock for PTP communication."
 ::= { ptpbaseClockPortRunningEntry 8 }

ptpbaseClockPortRunningTransport OBJECT-TYPE
    SYNTAX      ClockPortTransportType
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "This object specifies the transport protocol being used for PTP
         communication (the mapping used)."
 ::= { ptpbaseClockPortRunningEntry 9 }

ptpbaseClockPortRunningEncapsulationType OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "This object specifies the type of encapsulation if the
         interface is adding extra layers (eg. VLAN, Pseudowire
         encapsulation...) for the PTP messages."
 ::= { ptpbaseClockPortRunningEntry 10 }

ptpbaseClockPortRunningTxMode OBJECT-TYPE
    SYNTAX      ClockTxModeType
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "This object specifies the clock transmission mode as

        unicast:      Using unicast communication channel.
        multicast:     Using Multicast communication channel.
        multicast-mix: Using multicast-unicast communication channel"
 ::= { ptpbaseClockPortRunningEntry 11 }
```

ptpbaseClockPortRunningRxMode OBJECT-TYPE

SYNTAX ClockTxModeType
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
 "This object specifies the clock receive mode as

unicast: Using unicast communication channel.
 multicast: Using Multicast communication channel.
 multicast-mix: Using multicast-unicast communication channel"

::= { ptpbaseClockPortRunningEntry 12 }

ptpbaseClockPortRunningPacketsReceived OBJECT-TYPE

SYNTAX Counter64
 UNITS "packets"
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
 "This object specifies the packets received on the clock port
 (cumulative)."

::= { ptpbaseClockPortRunningEntry 13 }

ptpbaseClockPortRunningPacketsSent OBJECT-TYPE

SYNTAX Counter64
 UNITS "packets"
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
 "This object specifies the packets sent on the clock port
 (cumulative)."

::= { ptpbaseClockPortRunningEntry 14 }

ptpbaseClockPortTransDSTable OBJECT-TYPE

SYNTAX SEQUENCE OF PtpbaseClockPortTransDSEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
 "Table of information about the Transparent clock ports running
 dataset for a particular domain."

::= { ptpbaseMIBClockInfo 10 }

ptpbaseClockPortTransDSEntry OBJECT-TYPE

SYNTAX PtpbaseClockPortTransDSEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
 "An entry in the table, containing clock port Transparent
 dataset information about a single clock port"

INDEX {

```

        ptpbaseClockPortTransDSDomainIndex,
        ptpbaseClockPortTransDSInstanceIndex,
        ptpbaseClockPortTransDSPortNumberIndex
    }
    ::= { ptpbaseClockPortTransDSTable 1 }

PtpbaseClockPortTransDSEntry ::= SEQUENCE {
    ptpbaseClockPortTransDSDomainIndex      ClockDomainType,
    ptpbaseClockPortTransDSInstanceIndex    ClockInstanceType,
    ptpbaseClockPortTransDSPortNumberIndex  ClockPortNumber,
    ptpbaseClockPortTransDSPortIdentity     ClockIdentity,
    ptpbaseClockPortTransDSlogMinPdelayReqInt Integer32,
    ptpbaseClockPortTransDSFaultyFlag      TruthValue,
    ptpbaseClockPortTransDSPeerMeanPathDelay ClockTimeInterval
}

ptpbaseClockPortTransDSDomainIndex OBJECT-TYPE
    SYNTAX      ClockDomainType
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This object specifies the domain number used to create logical
        group of PTP devices."
    ::= { ptpbaseClockPortTransDSEntry 1 }

ptpbaseClockPortTransDSInstanceIndex OBJECT-TYPE
    SYNTAX      ClockInstanceType
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This object specifies the instance of the clock for this clock
        type in the given domain."
    ::= { ptpbaseClockPortTransDSEntry 2 }

ptpbaseClockPortTransDSPortNumberIndex OBJECT-TYPE
    SYNTAX      ClockPortNumber (1..65535)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This object specifies the PTP port number associated with this
        port."
    REFERENCE   "Section 7.5.2 Port Identity [IEEE 1588-2008]"
    ::= { ptpbaseClockPortTransDSEntry 3 }

ptpbaseClockPortTransDSPortIdentity OBJECT-TYPE
    SYNTAX      ClockIdentity
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object specifies the value of the PortIdentity
        attribute of the local port."
    REFERENCE   "Section 8.3.3.2.1 of [IEEE 1588-2008]"

```



```
::= { ptpbaseClockPortTransDSEntry 4 }
```

```
ptpbaseClockPortTransDSlogMinPdelayReqInt OBJECT-TYPE
```

```
SYNTAX Integer32
```

```
MAX-ACCESS read-only
```

```
STATUS current
```

```
DESCRIPTION
```

```
"This object specifies the value of the logarithm to the
base 2 of the minPdelayReqInterval."
```

```
REFERENCE "Section 8.3.3.3.1 of [IEEE 1588-2008]"
```

```
::= { ptpbaseClockPortTransDSEntry 5 }
```

```
ptpbaseClockPortTransDSFaultyFlag OBJECT-TYPE
```

```
SYNTAX TruthValue
```

```
MAX-ACCESS read-only
```

```
STATUS current
```

```
DESCRIPTION
```

```
"This object specifies the value TRUE if the port is faulty
and FALSE if the port is operating normally."
```

```
REFERENCE "Section 8.3.3.3.2 of [IEEE 1588-2008]"
```

```
::= { ptpbaseClockPortTransDSEntry 6 }
```

```
ptpbaseClockPortTransDSPeerMeanPathDelay OBJECT-TYPE
```

```
SYNTAX ClockTimeInterval
```

```
MAX-ACCESS read-only
```

```
STATUS current
```

```
DESCRIPTION
```

```
"This object specifies, (if the delayMechanism used is P2P) the
value is the estimate of the current one-way propagation delay,
i.e., <meanPathDelay> on the link attached to this port
computed
using the peer delay mechanism. If the value of the
delayMechanism
used is E2E, then the value will be zero."
```

```
REFERENCE "Section 8.3.3.3.3 of [IEEE 1588-2008]"
```

```
::= { ptpbaseClockPortTransDSEntry 7 }
```

```
ptpbaseClockPortAssociateTable OBJECT-TYPE
```

```
SYNTAX SEQUENCE OF PtpbaseClockPortAssociateEntry
```

```
MAX-ACCESS not-accessible
```

```
STATUS current
```

```
DESCRIPTION
```

```
"Table of information about a given port's associated ports.
```

```
For a master port - multiple slave ports which have established
sessions with the current master port.
```

```
For a slave port - the list of masters available for a given
slave port.
```

```
Session information (pkts, errors) to be displayed based on
```

```

        availability and scenario."
 ::= { ptpbaseMIBClockInfo 11 }

ptpbaseClockPortAssociateEntry OBJECT-TYPE
    SYNTAX          PtpbaseClockPortAssociateEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry in the table, containing information about a single
        associated port for the given clockport."
    INDEX
        {
            ptpClockPortCurrentDomainIndex,
            ptpClockPortCurrentClockTypeIndex,
            ptpClockPortCurrentClockInstanceIndex,
            ptpClockPortCurrentPortNumberIndex,
            ptpbaseClockPortAssociatePortIndex
        }
 ::= { ptpbaseClockPortAssociateTable 1 }

PtpbaseClockPortAssociateEntry ::= SEQUENCE {
    ptpClockPortCurrentDomainIndex      ClockDomainType,
    ptpClockPortCurrentClockTypeIndex   ClockType,
    ptpClockPortCurrentClockInstanceIndex ClockInstanceType,
    ptpClockPortCurrentPortNumberIndex  ClockPortNumber,
    ptpbaseClockPortAssociatePortIndex  Unsigned32,
    ptpbaseClockPortAssociateAddressType ClockPortTransportType,
    ptpbaseClockPortAssociateAddress    ClockPortTransportTypeAddress,
    ptpbaseClockPortAssociatePacketsSent Counter64,
    ptpbaseClockPortAssociatePacketsReceived Counter64,
    ptpbaseClockPortAssociateInErrors    Counter64,
    ptpbaseClockPortAssociateOutErrors   Counter64
}

ptpClockPortCurrentDomainIndex OBJECT-TYPE
    SYNTAX          ClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the given port's domain number."
 ::= { ptpbaseClockPortAssociateEntry 1 }

ptpClockPortCurrentClockTypeIndex OBJECT-TYPE
    SYNTAX          ClockType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the given port's clock type."
 ::= { ptpbaseClockPortAssociateEntry 2 }

ptpClockPortCurrentClockInstanceIndex OBJECT-TYPE
    SYNTAX          ClockInstanceType
    MAX-ACCESS      not-accessible

```

```
STATUS          current
DESCRIPTION
    "This object specifies the instance of the clock for this clock
    type in the given domain."
 ::= { ptpbaseClockPortAssociateEntry 3 }

ptpClockPortCurrentPortNumberIndex OBJECT-TYPE
SYNTAX          ClockPortNumber
MAX-ACCESS      not-accessible
STATUS          current
DESCRIPTION
    "This object specifies the PTP Port Number for the given port."
 ::= { ptpbaseClockPortAssociateEntry 4 }

ptpbaseClockPortAssociatePortIndex OBJECT-TYPE
SYNTAX          Unsigned32 (1..65535)
MAX-ACCESS      not-accessible
STATUS          current
DESCRIPTION
    "This object specifies the associated port's serial number in
    the current port's context."
 ::= { ptpbaseClockPortAssociateEntry 5 }

ptpbaseClockPortAssociateAddressType OBJECT-TYPE
SYNTAX          ClockPortTransportType
MAX-ACCESS      read-only
STATUS          current
DESCRIPTION
    "This object specifies the peer port's network address type used
    for PTP communication."
 ::= { ptpbaseClockPortAssociateEntry 6 }

ptpbaseClockPortAssociateAddress OBJECT-TYPE
SYNTAX          ClockPortTransportTypeAddress
MAX-ACCESS      read-only
STATUS          current
DESCRIPTION
    "This object specifies the peer port's network address used for
    PTP communication."
 ::= { ptpbaseClockPortAssociateEntry 7 }

ptpbaseClockPortAssociatePacketsSent OBJECT-TYPE
SYNTAX          Counter64
UNITS           "packets"
MAX-ACCESS      read-only
STATUS          current
DESCRIPTION
    "The number of packets sent to this peer port from the current
    port."
 ::= { ptpbaseClockPortAssociateEntry 8 }

ptpbaseClockPortAssociatePacketsReceived OBJECT-TYPE
```

```
SYNTAX          Counter64
UNITS           "packets"
MAX-ACCESS      read-only
STATUS          current
DESCRIPTION
    "The number of packets received from this peer port by the
    current port."
 ::= { ptpbaseClockPortAssociateEntry 9 }

ptpbaseClockPortAssociateInErrors OBJECT-TYPE
SYNTAX          Counter64
UNITS           "packets"
MAX-ACCESS      read-only
STATUS          current
DESCRIPTION
    "This object specifies the input errors associated with the
    peer port."
 ::= { ptpbaseClockPortAssociateEntry 10 }

ptpbaseClockPortAssociateOutErrors OBJECT-TYPE
SYNTAX          Counter64
UNITS           "packets"
MAX-ACCESS      read-only
STATUS          current
DESCRIPTION
    "This object specifies the output errors associated with the
    peer port."
 ::= { ptpbaseClockPortAssociateEntry 11 }

ptpbaseMIBNotifs OBJECT IDENTIFIER
 ::= { ptpbaseMIB 0 }

ptpbaseMIBObjects OBJECT IDENTIFIER
 ::= { ptpbaseMIB 1 }

ptpbaseMIBConformance OBJECT IDENTIFIER
 ::= { ptpbaseMIB 2 }

ptpbaseMIBSystemInfo OBJECT IDENTIFIER
 ::= { ptpbaseMIBObjects 1 }

-- Conformance Information Definition

ptpbaseMIBCompliances OBJECT IDENTIFIER
 ::= { ptpbaseMIBConformance 1 }

ptpbaseMIBGroups OBJECT IDENTIFIER
 ::= { ptpbaseMIBConformance 2 }

ptpbaseMIBCompliancesSystemInfo MODULE-COMPLIANCE
```

```
STATUS          current
DESCRIPTION
    "Compliance statement for agents that provide read-only support
    for PTPBASE-MIB to provide system level information of clock devices.
    Such devices can only be monitored using this MIB module.

    The Module is implemented with support for read-only. In other
    words, only monitoring is available by implementing this
    MODULE-COMPLIANCE."
MODULE          -- this module
MANDATORY-GROUPS { ptpbaseMIBSystemInfoGroup }
::= { ptpbaseMIBCompliances 1 }

ptpbaseMIBCompliancesClockInfo MODULE-COMPLIANCE
STATUS          current
DESCRIPTION
    "Compliance statement for agents that provide read-only support
    for PTPBASE-MIB to provide clock related information. Such devices can
    only be monitored using this MIB module.

    The Module is implemented with support for read-only. In other
    words, only monitoring is available by implementing this
    MODULE-COMPLIANCE."
MODULE          -- this module
MANDATORY-GROUPS {
    ptpbaseMIBClockCurrentDSGroup,
    ptpbaseMIBClockParentDSGroup,
    ptpbaseMIBClockDefaultDSGroup,
    ptpbaseMIBClockRunningGroup,
    ptpbaseMIBClockTimepropertiesGroup
}
::= { ptpbaseMIBCompliances 2 }

ptpbaseMIBCompliancesClockPortInfo MODULE-COMPLIANCE
STATUS          current
DESCRIPTION
    "Compliance statement for agents that provide read-only support
    for PTPBASE-MIB to provide clock port related information. Such devices
    can only be monitored using this MIB module.

    The Module is implemented with support for read-only. In other
    words, only monitoring is available by implementing this
    MODULE-COMPLIANCE."
MODULE          -- this module
MANDATORY-GROUPS {
    ptpbaseMIBClockPortGroup,
    ptpbaseMIBClockPortDSGroup,
    ptpbaseMIBClockPortRunningGroup,
    ptpbaseMIBClockPortAssociateGroup
}
::= { ptpbaseMIBCompliances 3 }
```

```
ptpbaseMIBCompliancesTransparentClockInfo MODULE-COMPLIANCE
    STATUS          current
    DESCRIPTION
        "Compliance statement for agents that provide read-only support
        for PTPBASE-MIB to provide Transparent clock related information. Such
        devices can only be monitored using this MIB module.

        The Module is implemented with support for read-only. In other
        words, only monitoring is available by implementing this
        MODULE-COMPLIANCE."
    MODULE           -- this module
    MANDATORY-GROUPS {
        ptpbaseMIBClockTransparentDSGroup,
        ptpbaseMIBClockPortTransDSGroup
    }
    ::= { ptpbaseMIBCompliances 4 }

ptpbaseMIBSystemInfoGroup OBJECT-GROUP
    OBJECTS          {
        ptpbaseSystemDomainTotals,
        ptpDomainClockPortsTotal,
        ptpbaseSystemProfile
    }
    STATUS          current
    DESCRIPTION
        "Group which aggregates objects describing system-wide
        information"
    ::= { ptpbaseMIBGroups 1 }

ptpbaseMIBClockCurrentDSGroup OBJECT-GROUP
    OBJECTS          {
        ptpbaseClockCurrentDSStepsRemoved,
        ptpbaseClockCurrentDSOffsetFromMaster,
        ptpbaseClockCurrentDSMeanPathDelay
    }
    STATUS          current
    DESCRIPTION
        "Group which aggregates objects describing PTP Current Dataset
        information"
    ::= { ptpbaseMIBGroups 2 }

ptpbaseMIBClockParentDSGroup OBJECT-GROUP
    OBJECTS          {
        ptpbaseClockParentDSParentPortIdentity,
        ptpbaseClockParentDSParentStats,
        ptpbaseClockParentDSOffset,
        ptpbaseClockParentDSClockPhChRate,
        ptpbaseClockParentDSGMClockIdentity,
        ptpbaseClockParentDSGMClockPriority1,
        ptpbaseClockParentDSGMClockPriority2,
        ptpbaseClockParentDSGMClockQualityClass,
        ptpbaseClockParentDSGMClockQualityAccuracy,
```

```

        ptpbaseClockParentDSGMClockQualityOffset
    }
    STATUS          current
    DESCRIPTION
        "Group which aggregates objects describing PTP Parent Dataset
        information"
    ::= { ptpbaseMIBGroups 3 }

ptpbaseMIBClockDefaultDSGroup OBJECT-GROUP
    OBJECTS          {
        ptpbaseClockDefaultDSTwoStepFlag,
        ptpbaseClockDefaultDSClockIdentity,
        ptpbaseClockDefaultDSPriority1,
        ptpbaseClockDefaultDSPriority2,
        ptpbaseClockDefaultDSSlaveOnly,
        ptpbaseClockDefaultDSQualityClass,
        ptpbaseClockDefaultDSQualityAccuracy,
        ptpbaseClockDefaultDSQualityOffset
    }
    STATUS          current
    DESCRIPTION
        "Group which aggregates objects describing PTP Default Dataset
        information"
    ::= { ptpbaseMIBGroups 4 }

ptpbaseMIBClockRunningGroup OBJECT-GROUP
    OBJECTS          {
        ptpbaseClockRunningState,
        ptpbaseClockRunningPacketsSent,
        ptpbaseClockRunningPacketsReceived
    }
    STATUS          current
    DESCRIPTION
        "Group which aggregates objects describing PTP running state
        information"
    ::= { ptpbaseMIBGroups 5 }

ptpbaseMIBClockTimepropertiesGroup OBJECT-GROUP
    OBJECTS          {
        ptpbaseClockTimePropertiesDSCurrentUTCOffsetValid,
        ptpbaseClockTimePropertiesDSCurrentUTCOffset,
        ptpbaseClockTimePropertiesDSLeap59,
        ptpbaseClockTimePropertiesDSLeap61,
        ptpbaseClockTimePropertiesDSTimeTraceable,
        ptpbaseClockTimePropertiesDSFreqTraceable,
        ptpbaseClockTimePropertiesDSPTPTimescale,
        ptpbaseClockTimePropertiesDSSource
    }
    STATUS          current
    DESCRIPTION
        "Group which aggregates objects describing PTP Time Properties
        information"
```

```
 ::= { ptptimeMIBGroups 6 }

ptptimeMIBClockTransparentDSGroup OBJECT-GROUP
    OBJECTS      {
        ptptimeClockTransDefaultDSClockIdentity,
        ptptimeClockTransDefaultDSNumOfPorts,
        ptptimeClockTransDefaultDSDelay,
        ptptimeClockTransDefaultDSPrimaryDomain
    }
    STATUS        current
    DESCRIPTION   "Group which aggregates objects describing PTP Transparent
        Dataset
        information"
    ::= { ptptimeMIBGroups 7 }

ptptimeMIBClockPortGroup OBJECT-GROUP
    OBJECTS      {
        ptptimeClockPortName,
        ptptimeClockPortSyncOneStep,
        ptptimeClockPortCurrentPeerAddress,
        ptptimeClockPortNumOfAssociatedPorts,
        ptptimeClockPortCurrentPeerAddressType,
        ptptimeClockPortRole
    }
    STATUS        current
    DESCRIPTION   "Group which aggregates objects describing information for a
        given PTP Port."
    ::= { ptptimeMIBGroups 8 }

ptptimeMIBClockPortDSGroup OBJECT-GROUP
    OBJECTS      {
        ptptimeClockPortDSName,
        ptptimeClockPortDSPortIdentity,
        ptptimeClockPortDSAnnouncementInterval,
        ptptimeClockPortDSAnnounceRctTimeout,
        ptptimeClockPortDSSyncInterval,
        ptptimeClockPortDSMinDelayReqInterval,
        ptptimeClockPortDSPeerDelayReqInterval,
        ptptimeClockPortDSDelayMech,
        ptptimeClockPortDSPeerMeanPathDelay,
        ptptimeClockPortDSGrantDuration,
        ptptimeClockPortDSPTPVersion
    }
    STATUS        current
    DESCRIPTION   "Group which aggregates objects describing PTP Port Dataset
        information"
    ::= { ptptimeMIBGroups 9 }

ptptimeMIBClockPortRunningGroup OBJECT-GROUP
```



```
OBJECTS      {
    ptptimeClockPortRunningName,
    ptptimeClockPortRunningState,
    ptptimeClockPortRunningRole,
    ptptimeClockPortRunningInterfaceIndex,
    ptptimeClockPortRunningTransport,
    ptptimeClockPortRunningEncapsulationType,
    ptptimeClockPortRunningTxMode,
    ptptimeClockPortRunningRxMode,
    ptptimeClockPortRunningPacketsReceived,
    ptptimeClockPortRunningPacketsSent
}
STATUS      current
DESCRIPTION
    "Group which aggregates objects describing PTP running interface
    information"
 ::= { ptptimeMIBGroups 10 }

ptptimeMIBClockPortTransDSGroup OBJECT-GROUP
OBJECTS      {
    ptptimeClockPortTransDSPortIdentity,
    ptptimeClockPortTransDSlogMinPdelayReqInt,
    ptptimeClockPortTransDSFaultyFlag,
    ptptimeClockPortTransDSPeerMeanPathDelay
}
STATUS      current
DESCRIPTION
    "Group which aggregates objects describing PTP TransparentDS
    Dataset
    information"
 ::= { ptptimeMIBGroups 11 }

ptptimeMIBClockPortAssociateGroup OBJECT-GROUP
OBJECTS      {
    ptptimeClockPortAssociatePacketsSent,
    ptptimeClockPortAssociatePacketsReceived,
    ptptimeClockPortAssociateAddress,
    ptptimeClockPortAssociateAddressType,
    ptptimeClockPortAssociateInErrors,
    ptptimeClockPortAssociateOutErrors
}
STATUS      current
DESCRIPTION
    "Group which aggregates objects describing information on peer
    PTP ports for a given PTP clock-port."
 ::= { ptptimeMIBGroups 12 }

END
```

5. Security Considerations

This MIB contains readable objects whose values provide information related to PTP objects. While unauthorized access to the readable objects is relatively innocuous, unauthorized access to the writeable objects could cause a denial of service, or could cause unauthorized creation and/or manipulation of tunnels. Hence, the support for SET operations in a non-secure environment without proper protection can have a negative effect on network operations.

SNMPv1 by itself is such an insecure environment. Even if the network itself is secure (for example by using IPSec), even then, there is no control as to who on the secure network is allowed to access and SET (change/create/delete) the objects in this MIB.

It is recommended that the implementers consider the security features as provided by the SNMPv3 framework. Specifically, the use of the User-based Security Model [RFC 3414] and the View-based Access Control Model [RFC 3415] is recommended.

It is then a customer/user responsibility to ensure that the SNMP entity giving access to this MIB, is properly configured to give access to those objects only to those principals (users) that have legitimate rights to access them.

6. IANA Considerations

To be added.

7. References

7.1. Normative References

[IEEE 1588-2008] "IEEE Standard for A Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", IEEE Std. 1588(TM)-2008, 24 July 2008

7.2. Informative References

[RFC 1155] Rose, M., and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based Internets", STD 16, RFC 1155, Performance Systems International, Hughes LAN Systems, May 1990

[RFC 1157] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple Network Management Protocol", STD 15, RFC 1157, SNMP Research, Performance Systems International, Performance Systems International, MIT Laboratory for Computer Science, May 1990.

[RFC 1212] Rose, M., and K. McCloghrie, "Concise MIB Definitions", STD 16, RFC 1212, Performance Systems International, Hughes LAN Systems, March 1991

[RFC 1215] M. Rose, "A Convention for Defining Traps for use with the SNMP", RFC 1215, Performance Systems International, March 1991

[RFC 1901] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Introduction to Community-based SNMPv2", RFC 1901, SNMP Research, Inc., Cisco Systems, Inc., Dover Beach Consulting, Inc., International Network Services, January 1996.

[RFC 1906] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1906, SNMP Research, Inc., Cisco Systems, Inc., Dover Beach Consulting, Inc., International Network Services, January 1996.

[RFC 2578] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[RFC 2579] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.

[RFC 2580] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.

[RFC 3411] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, Enterasys Networks, BMC Software, Inc., Lucent Technologies, December 2002

[RFC 3412] Case, J., Harrington D., Presuhn R., and B. Wijnen, "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3412, SNMP Research, Inc., Enterasys Networks, BMC Software, Inc., Lucent Technologies, December 2002.

[RFC 3413] Levi, D., Meyer, P., and B. Stewart, "Simple Network Management Protocol (SNMP) Applications", STD 62, RFC 3413, Nortel Networks, Secure Computing Corporation, December 2002.

[RFC 3414] Blumenthal, U., and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", STD 62, RFC 3414, Lucent Technologies, December 2002.

[RFC 3415] Wijnen, B., Presuhn, R., and K. McCloghrie, "View-based Access Control Model (VACM) for the Simple Network Management

Protocol (SNMP)", STD 62, RFC 3415, Lucent Technologies, BMC Software, Inc., Cisco Systems, Inc., December 2002.

[RFC 3416] Presuhn, R. (Ed.), "Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3416, BMC Software, Inc., December 2002.

[RFC 3417] Presuhn, R. (Ed.), "Transport Mappings for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3417, BMC Software, Inc., December 2002.

[RFC 5905] David L. Mills, " Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, University of Delaware, June 2010.

[IEEE 802.3-2008] "IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and Metropolitan area networks - Specific requirements Part 3: Carrier sense multiple access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications", IEEE Std. 802.3 - 2008, 26 December 2008

[G.8265.1] "Precision time protocol telecom profile for frequency synchronization", ITU-T Recommendation G.8265.1, October 2010.

8. Acknowledgements

Thanks to John Linton and Danny Lee for valuable comments.

9. Author's Addresses

Vinay Shankarkumar
Cisco Systems,
7025-4 Kit Creek Road,
Research Triangle Park,
NC 27560,
USA.
Email: vinays@cisco.com

Laurent Montini,
Cisco Systems,
11, rue Camille Desmoulins,
92782 Issy-les-Moulineaux,
France.
Email: lmontini@cisco.com

Tim Frost,
Symmetricom Inc.,
2300 Orchard Parkway,
San Jose,
CA 95131,
USA.
Email: tfrost@symmetricom.com

Greg Dowd,
Symmetricom Inc.,
2300 Orchard Parkway,
San Jose,
CA 95131,
USA.
Email: gdowd@symmetricom.com

10. ANNEX A: Extended Fields Addendum

Some structures in the MIB have been extended. The MIB is thus able to cover the structures defined in the IEEE standards and is extensible as well.

- o ClockIdentity is defined in the standard as 8-octet array. The MIB defines it as OCTET string of length (1..255).
- o ClockPortNumber is defined in the standard as ranging from 1, 2, ... till FFFF (16 bits); FFFF is used as the 'all-ports' indicator in Management messages and in signalling messages. The MIB defines it as Unsigned32 ranging in value (0..65535).
- o ClockTimeInterval is defined in the standard as of length 64 bits (Integer64). The MIB defines it as OCTET string of length (1..255).
- o ptpbaseClockParentDSClockPhChRate (parentDS.observedParentClockPaseChangeRate) is defined in the standard as 16 bits. The MIB defines it as Integer32.

TICTOC Working Group
Internet Draft
Intended status: Informational
Expires: March 2013

Tal Mizrahi
Marvell
September 14, 2012

TICTOC Security Requirements
draft-ietf-tictoc-security-requirements-03.txt

Abstract

As time synchronization protocols are becoming increasingly common and widely deployed, concern about their exposure to various security threats is increasing. This document defines a set of security requirements for time synchronization protocols, focusing on the Precision Time Protocol (PTP) and the Network Time Protocol (NTP). This document also discusses the security impacts of time synchronization protocol practices, the time synchronization performance implications of external security practices, the dependencies between other security services and time synchronization.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on March 14, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Conventions Used in this Document	4
2.1. Terminology	4
2.2. Terms & Abbreviations	5
3. Security Threats	5
3.1. Threat Model	5
3.1.1. Internal vs. External Attackers	6
3.1.2. Man in the Middle (MITM) vs. Packet Injector	6
3.2. Threat Analysis.....	6
3.2.1. Packet Interception and Manipulation	6
3.2.2. Spoofing	6
3.2.3. Replay Attack	7
3.2.4. Rogue Master Attack	7
3.2.5. Packet Interception and Removal	7
3.2.6. Packet Delay Manipulation	7
3.2.7. Cryptographic Performance Attacks	7
3.2.8. L2/L3 DoS Attacks	8
3.2.9. Master Time Source Spoofing (e.g. GPS fraud)	8
3.3. Threat Analysis Summary	8
4. Security Requirements	9
4.1. Clock Identity Authentication	9
4.1.1. Authentication of Masters	10
4.1.2. Recursive Authentication of Masters (Chain of Trust).....	10
4.1.3. Authentication of Slaves	11
4.1.4. PTP: Authentication of Transparent Clocks.....	11
4.1.5. PTP: Authentication of Announce Messages	11
4.2. Data integrity	12
4.2.1. PTP: Hop-by-hop vs. End-to-end Integrity Protection	12
4.2.1.1. Hop by Hop Integrity Protection	12
4.2.1.2. End to End Integrity Protection	13

4.3. Availability	13
4.4. Replay Protection	14
4.5. Cryptographic Keys & Security Associations	14
4.5.1. Security Association	14
4.5.2. Unicast and Multicast	14
4.5.3. Key Freshness	14
4.6. Performance	15
4.7. Confidentiality.....	15
4.8. Protection against packet delay attacks	16
4.9. Combining Secured with Unsecured Nodes	16
4.9.1. Secure Mode	17
4.9.2. Hybrid Mode	17
5. Summary of Requirements	18
6. Additional security implications	19
6.1. Security and on-the-fly Timestamping	19
6.2. Security and Two-Step Timestamping	20
6.3. Intermediate Clocks	20
6.4. The Effect of External Security Protocols on Time Synchronization	21
6.5. External Security Services Requiring Time Synchronization	21
7. Issues for Further Discussion	21
8. Security Considerations	21
9. IANA Considerations	22
10. Acknowledgments	22
11. References	22
11.1. Normative References	22
11.2. Informative References	22
12. Contributing Authors	24

1. Introduction

As time synchronization protocols are becoming increasingly common and widely deployed, concern about the resulting exposure to various security threats is increasing. If a time synchronization protocol is compromised, the applications it serves are prone to a range of possible attacks including Denial-of-Service or incorrect behavior.

This document focuses on the security aspects of the Precision Time Protocol (PTP) [IEEE1588] and the Network Time Protocol [NTPv4]. The Network Time Protocol was defined with an inherent security protocol, defined in [NTPv4] and in [AutoKey]. The IEEE 1588 includes an experimental security protocol, defined in Annex K of the standard, but this Annex was never formalized into a fully defined security protocol.

Many of the existing packet timing deployments do not use any security mechanisms. The absence of a standard security solution for

PTP undoubtedly contributed to the wide deployment of unsecured time synchronization solutions. However, in some cases security mechanisms may not be strictly necessary, e.g., due to other security practices in place, or due to the architecture of the network. A time synchronization security solution, much like any security solution, is comprised of various building blocks, and must be carefully tailored for the specific system it is deployed in. Based on a system-specific threat assessment, the benefits of a security solution must be weighed against the potential risks, and based on this tradeoff an optimal security solution can be selected.

This document attempts to add clarity to the time synchronization protocol security requirements discussion by addressing a series of questions:

- (1) What are the threats that need to be addressed for the time synchronization protocol, and thus what security services need to be provided? (e.g. a malicious NTP server or PTP master)
- (2) What external security practices impact the security and performance of time keeping, and what can be done to mitigate these impacts? (e.g. an IPSec tunnel in the synchronization traffic path)
- (3) What are the security impacts of time synchronization protocol practices? (e.g. on-the-fly modification of timestamps)
- (4) What are the dependencies between other security services and time synchronization? (e.g. which comes first - the certificate or the timestamp?)

In light of the questions above, this document defines a set of requirements for security solutions for time synchronization protocols, focusing on PTP and NTP.

2. Conventions Used in this Document

2.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [KEYWORDS].

This document describes security requirements, and thus requirements are phrased in the document in the form "the security mechanism MUST/SHOULD/...". Note, that the phrasing does not imply that this document defines a specific security mechanism, but defines the requirements that every security mechanism should comply to.

This document refers to both PTP and NTP. For the sake of consistency, throughout the document the term "master" applies to both a PTP master and an NTP server. Similarly, the term "slave" applies to both PTP slaves and NTP clients. The general term "clock" refers to masters, slaves and PTP Transparent Clocks (TC). The term "protocol packets" is refers generically to PTP and NTP messages.

2.2. Terms & Abbreviations

BC	Boundary Clock
MITM	Man In The Middle
NTP	Network Time Protocol
OC	Ordinary Clock
PTP	Precision Time Protocol
Secured clock	A clock that supports a security mechanism that complies to the requirements in this document
TC	Transparent Clock
Unsecured clock	A clock that does not support a security mechanism according to the requirments in this document

3. Security Threats

This section discusses the possible attacker types, and analyzes various attacks against time synchronization protocols.

The literature is rich with security threats of time synchronization protocols, e.g., [Traps], [AutoKey], [TM], [SecPTP], and [SecSen]. The threat analysis in this document is mostly based on [TM].

3.1. Threat Model

A time synchronization protocol can be attacked by various types of attackers.

The analysis in this documents classifies attackers according to 2 criteria, as described in 3.1.1. and 3.1.2.

3.1.1. Internal vs. External Attackers

In the context of internal and external attackers, the underlying assumption is that the time synchronization protocol is secured either by an encryption or an authentication mechanism.

Internal attackers either have access to a trusted segment of the network, or possess the encryption or authentication keys. External attackers, on the other hand, do not have the keys, and are exposed only to the encrypted or authenticated traffic. Thus, an internal attacker can maliciously tamper with legitimate traffic in the network, as well as generate its own traffic and make it appear legitimate to its attacked nodes.

Obviously, in the absence of a security mechanism there is no distinction between internal and external attackers, since all attackers are internal in practice.

3.1.2. Man in the Middle (MITM) vs. Packet Injector

MITM attackers are located in a position that allows interception and modification of in-flight protocol packets.

A traffic injector is not located in an MITM position, but can attack by generating protocol packets. An injector can also potentially eavesdrop to protocol packets sent as multicast, record them and replay them later.

3.2. Threat Analysis

3.2.1. Packet Interception and Manipulation

A packet interception and manipulation attack results when a Man-In-The-Middle (MITM) attacker intercepts timing protocol packets, alters them and relays them to their destination, allowing the attacker to maliciously tamper with the protocol. This can result in a situation where the time protocol is apparently operational but providing intentionally inaccurate information.

3.2.2. Spoofing

In spoofing, an attacker masquerades as a legitimate node in the network by generating and transmitting protocol packets. For example, an attacker can impersonate the master, allowing malicious distribution of false timing information. As with packet interception and manipulation, this can result in a situation where the time

protocol is apparently operational but providing intentionally inaccurate information.

3.2.3. Replay Attack

In a replay attack, an attacker records protocol packets and replays them at a later time without any modification. This can also result in a situation where the time protocol is apparently operational but providing intentionally inaccurate information.

3.2.4. Rogue Master Attack

In a rogue master attack, an attacker causes other nodes in the network to believe it is a legitimate master. As opposed to the spoofing attack, in the Rouge Master attack the attacker does not fake its identity, but rather manipulates the master election process. For example, in PTP, an attacker can manipulate the Best Master Clock Algorithm (BMCA), and cause other nodes in the network to believe it is the most eligible candidate to be a grandmaster.

3.2.5. Packet Interception and Removal

A packet interception and removal attack results when a Man-In-The-Middle attacker intercepts and drops protocol packets, preventing the destination node from receiving the timing information.

3.2.6. Packet Delay Manipulation

In a packet delay manipulation scenario, a Man-In-The-Middle attacker intercepts protocol packets, and relays them to their destination after adding a maliciously computed delay.

Note that the attackee still receives one copy of each packet, contrary to the replay attack, where a packet is received by the attackee more than once.

3.2.7. Cryptographic Performance Attacks

In cryptographic performance attacks, an attacker transmits fake protocol packet, causing high utilization of the cryptographic engine at the receiver, which attempts to verify the integrity of these fake packets.

3.2.8. L2/L3 DoS Attacks

There are many possible Layer 2 and Layer 3 Denial of Service attacks. As the target's availability is compromised, the timing protocol is affected accordingly.

3.2.9. Master Time Source Spoofing (e.g. GPS fraud)

In time source spoofing, an attacker spoofs the accurate time source of the master. For example, if the master uses a GPS based clock as its reference source, an attacker can spoof the GPS satellites, causing the master to use a false reference time.

3.3. Threat Analysis Summary

The two key factors to a threat analysis are the severity and the likelihood of each of the analyzed attacks.

Table 1 summarizes the security attacks presented in 3.2. For each attack, the table specifies its impact, and its applicability to each of the attacker types presented in 3.1.

The Impact column provides an intuition to the severity of each attack, and the relevant Attacker Type columns provide an intuition about the how difficult each attack is to implement, and hence about the likelihood of each attack.

The impact column in Table 1 can have one of 3 values:

- o DoS - the attack causes a denial of service to the attacked node, the impact of which is not restricted to the time synchronization protocol.
- o False time - slaves align to a false time or frequency value due to the attack. Note that if the time synchronization service aligns to a false time, it may cause denial of service to other applications that rely on accurate time. However, for the purpose of the analysis in this section we distinguish this implication from "DoS", which refers to a DoS attack that is not necessarily aimed at the time synchronization protocol.
- o Accuracy degradation - the attack yields a degradation in the slave accuracy, but does not completely compromise the slaves' time and frequency.

The Attacker Type columns refer to the 4 possible combinations of the attacker types defined in 3.1.

Attack	Impact			Attacker Type			
	False Time	Accuracy Degrad.	DoS	Internal MITM	External Inj.	Internal MITM	External Inj.
Interception and manipulation	+			+			
Spoofing	+			+	+		
Replay attack	+			+	+		
Rogue master attack	+			+	+		
Interception and Removal		+		+		+	
Packet delay manipulation	+			+		+	
Crypt. performance attacks			+	+	+	+	+
DoS attacks			+	+	+	+	+
Master Time source spoofing (e.g. GPS spoofing)	+			+	+	+	+

Table 1 Threat Analysis - Summary

4. Security Requirements

This section defines a set of requirements from the security mechanisms used for PTP and NTP. These requirements are phrased in the form "the security mechanism MUST/SHOULD/MAY...". However, this document does not specify how these requirements can be met; While these requirements can be satisfied by extending the time protocols, at least a subset of the requirements can be met by applying common security practices to the network or by using existing security protocols, such as IPsec or MACsec. Thus, security solutions that address these requirements are outside the scope of this document.

4.1. Clock Identity Authentication

Requirement

The security mechanism MUST provide a means for each clock to authenticate the sender of a protocol packet.

Discussion

In the context of this document, authentication refers to:

- o Identification: verifying the identity of the peer clock.
- o Authorization: verifying that the peer clock is permitted to play the role that it plays in the protocol. For example, some nodes may be permitted to be masters, while other nodes are only permitted to be slaves or TCs.

The following subsections describe 4 distinct cases of clock authentication.

4.1.1. Authentication of Masters

Requirement

The security mechanism MUST support an authentication mechanism, allowing slave clocks to authenticate the identity of master clocks.

4.1.2. Recursive Authentication of Masters (Chain of Trust)

Requirement

The security mechanism MUST support recursive authentication of the master, to be used in cases where end-to-end authentication is not possible.

Discussion

Clocks authenticate masters in order to ensure the authenticity of the time source.

In some cases a slave is connected to an intermediate master, that is not the primary time source. For example, in PTP a slave can be connected to a Boundary Clock (BC), which in turn is connected to a grandmaster. A similar example in NTP is when a client is connected to a stratum 2 server, which is connected to a stratum 1 server. In both the PTP and the NTP cases, the slave authenticates the intermediate master, and the intermediate master authenticates the primary master. This inductive authentication process is referred to in [AutoKey] as proventionation.

4.1.3. Authentication of Slaves

Requirement

The security mechanism SHOULD provide a means for a master to authenticate its slaves.

Discussion

Slaves are authenticated by masters in order to verify that the slave is authorized to receive timing services from the master.

Authentication of slaves prevents unauthorized clocks from receiving time services, and also reduces unnecessary load on the master clock, by preventing the master from serving unauthorized clocks. It could be argued that the authentication of slaves could put a higher load on the master than serving the unauthorized clock, and hence this requirement is a SHOULD.

4.1.4. PTP: Authentication of Transparent Clocks

Requirement

The security mechanism for PTP SHOULD provide a means for a master to authenticate the identity of the P2P TCs directly connected to it.

Discussion

P2P TCs that are one hop from the master use the PDelay_Req and PDelay_Resp handshake to compute the link delay between the master and TC. These TCs are authenticated by the master.

Authentication of TCs, much like authentication of slaves, reduces unnecessary load on the master clock and peer TCs, by preventing the master from serving unauthorized clocks.

4.1.5. PTP: Authentication of Announce Messages

Requirement

The security mechanism for PTP MUST support authentication of Announce messages.

Discussion

Master election is performed in PTP using the Best Master Clock Algorithm (BMCA). Each Ordinary Clock (OC) announces its clock

attributes using Announce messages, and the best master is elected based on the information gathered from all the candidates. Announce messages must be authenticated in order to prevent malicious master attacks.

Note, that this subsection specifies a requirement that is not necessarily included in 4.1.1. or in 4.1.3. , since the BMCA is initiated before clocks have been defined as masters or slaves.

4.2. Data integrity

Requirement

The security mechanism MUST protect the integrity of protocol packets.

Discussion

While subsection 4.1. refers to ensuring WHO sent the protocol packet, this subsection refers to ensuring that the packet arrived intact. The integrity protection mechanism ensures the authenticity and completeness of data from the data originator.

4.2.1. PTP: Hop-by-hop vs. End-to-end Integrity Protection

Requirement

A security mechanism for PTP MUST support hop-by-hop integrity protection.

Requirement

A security mechanism for PTP SHOULD support end-to-end integrity protection.

Discussion

Specifically in PTP, when protocol packets are subject to modification by TCs, the integrity protection can be enforced in one of two approaches, end-to-end or hop-by-hop.

4.2.1.1. Hop by Hop Integrity Protection

Each hop that needs to modify a protocol packet:

- o Verifies its integrity.

- o Modifies the packet, i.e., modifies the correctionField.
- o Re-generates the integrity protection, e.g., re-computes a Message Authentication Code.

In the hop-by-hop approach, the integrity of protocol packets is protected by induction on the path from the originator to the receiver.

This approach is simple, but allows malicious TCs to modify protocol packets.

4.2.1.2. End to End Integrity Protection

In this approach, the integrity protection is maintained on the path from the originator of a protocol packet to the receiver. This allows the receiver to validate the protocol packet without the ability of intermediate TCs to manipulate the packet.

Since TCs need to modify the correctionField, a separate integrity protection mechanism is used specifically for the correctionField.

The end-to-end approach limits the TC's impact to the correctionField alone, while the rest of the protocol packet is protected on an end-to-end basis. It should be noted that this approach is more difficult to implement than the hop-by-hop approach, as it requires separate layers of protection for the correctionField and for the rest of the packet, using different cryptographic mechanisms and keys.

4.3. Availability

Requirement

The security mechanism MUST protect the time synchronization protocol from DoS attacks by external attackers.

Discussion

The protocol availability can be compromised by several different attacks. An attacker can inject protocol messages to implement the spoofing attack (Section 3.2.2.) or the rogue master attack (Section 3.2.4.), causing denial of service to the attackee. An authentication mechanism (Section 4.1.) limits these attacks strictly to internal attackers, and thus prevents external attackers from performing them.

Note that a security mechanism applied at the time synchronization layer cannot, by itself, prevent DoS attacks described in Section 3.2.8. DoS attacks at lower layers of the protocol stack (Section 3.2.8.) can still be implemented by external attackers even in the presence of an authentication mechanism.

4.4. Replay Protection

Requirement

Protocol messages MUST be resistant to replay attacks.

4.5. Cryptographic Keys & Security Associations

4.5.1. Security Association

Requirement

The security protocol SHOULD support an association protocol where:

- o Two or more clocks authenticate each other.
- o The clocks generate and agree on a cryptographic session key.

Discussion

The security requirements in 4.1. and 4.2. require usage of cryptographic mechanisms, deploying cryptographic keys. A security association is an essential building block in these mechanisms.

4.5.2. Unicast and Multicast

Requirement

The security mechanism SHOULD support security association protocols for unicast and for multicast associations.

Discussion

A unicast protocol requires an association protocol between two clocks, whereas a multicast protocol requires an association protocol among two or more clocks, where one of the clocks is a master.

4.5.3. Key Freshness

Requirement

The cryptographic keys MUST be refreshed periodically.

Requirement

The association protocol MUST be invoked periodically, where each instance of the association protocol MUST produce a different session key.

4.6. Performance

Requirement

The security mechanism MUST be designed in such a way that it does not degrade the quality of the time transfer.

Requirement

The mechanism SHOULD be relatively lightweight, as client restrictions often dictate a low processing and memory footprint, and because the server may have extensive fan-out.

Requirement

The mechanism also SHOULD not require excessive storage of client state in the master, nor significantly increase bandwidth consumption.

Discussion

Note that the performance requirements refer to a time-synchronization-specific security mechanism. In systems where a security protocol is used for other types of traffic as well, this document does not place any performance requirements on the security protocol performance. For example, if IPsec encryption is used for securing all information between the master and slave node, including information that is not part of the time protocol, the requirements in this subsection are not necessarily applicable.

4.7. Confidentiality

Requirement

The security mechanism MAY provide confidentiality protection of the protocol packets.

Discussion

In the context of time synchronization, confidentiality is typically of low importance, since timing information is typically not considered secret information.

Confidentiality can play an important role when service providers charge payment for time synchronization services, but these cases are rather esoteric.

Confidentiality can also prevent an MITM attacker from identifying protocol packets. Thus, confidentiality can assist in protecting the timing protocol against packet delay attacks, where the attacker selectively adds delay to time protocol packets. Note, that time protocols have predictable behavior such as packet transmission rates and packet lengths, and thus packet encryption does not prevent delay attacks, but rather makes these attacks more difficult to implement.

4.8. Protection against packet delay attacks

Requirement

The security mechanism MAY include a means to detect packet delay attacks.

Requirement

The security mechanism MAY include a redundancy mechanism that allows a node that detects a delay attack to switch over to a secondary master.

Discussion

While this document does not define specific security solutions, we note that common practices for protection against delay attacks use redundant masters (e.g. [NTPv4]), or redundant paths between the master and slave (e.g. [DelayAtt]). If one of the time sources indicates a time value that is significantly different than the other sources, it is assumed to be erroneous or under attack, and is therefore ignored.

This requirement is a "may" requirement since both master redundancy and path redundancy are not necessarily possible in all network topologies.

4.9. Combining Secured with Unsecured Nodes

Integrating a security mechanism into a time synchronized system is a complex process, and in some cases may require a gradual process,

where new equipment supports the security mechanism, and is required to interoperate with legacy equipment without the security features.

4.9.1. Secure Mode

Requirement

The security mechanism **MUST** support a secure mode, where only secured clocks are permitted to take part in the synchronization protocol. A protocol packet received from an unsecured clock **MUST** be discarded.

Discussion

While the requirement in this subsection is a bit similar to the one in 4.1. , it explicitly defines the secure mode, as opposed to the hybrid mode presented in the next subsection.

4.9.2. Hybrid Mode

Requirement

The security protocol **MAY** support a hybrid mode, where both secured and unsecured clocks are permitted to take part in the protocol.

Discussion

The hybrid mode allows both secured and unsecured clocks to take part in the synchronization protocol. NTP, for example, allows a mixture of secured and unsecured nodes.

Requirement

A master in the hybrid mode **SHOULD** be a secured clock.

A secured slave in the hybrid mode **SHOULD** discard all protocol packets received from unsecured clocks.

Discussion

This requirement ensures that the existence of unsecured clocks does not compromise the security provided to secured clocks. Hence, secured slaves only "trust" protocol packets received from a secured clock. An unsecured clock can receive protocol packets from either secured clocks, or unsecured clocks.

Note that the security scheme in [NTPv4] with [AutoKey] does not satisfy this requirement, since nodes prefer the server with the best

clock, and not necessarily the server that supports authentication. For example, a stratum 2 server is connected to two stratum 1 servers, Server A, supporting authentication, and server B, without authentication. If server B has a more accurate clock than A, the stratum 2 server chooses server B, in spite of the fact it does not support authentication.

5. Summary of Requirements

Section	Requirement	Type
4.1.	Authentication of sender.	MUST
	Authentication of master.	MUST
	Recursive authentication.	MUST
	Authentication of slaves.	SHOULD
	PTP: Authentication of TCs.	SHOULD
	PTP: Authentication of Announce messages.	SHOULD
4.2.	Integrity protection.	MUST
	PTP: hop-by-hop integrity protection.	MUST
	PTP: end-to-end integrity protection.	SHOULD
4.3.	Protection against DoS attacks.	MUST
4.4.	Replay protection.	MUST
4.5.	Security association.	SHOULD
	Unicast and multicast associations.	SHOULD
	Key freshness.	MUST
4.6.	Performance: no degradation in	MUST

	quality of time transfer.	
	Performance: lightweight.	SHOULD
	Performance: storage, bandwidth.	MUST
4.7.	Confidentiality protection.	MAY
4.8.	Protection against delay attacks.	MAY
4.9.	Secure mode.	MUST
	Hybrid mode.	MAY

Table 2 Summary of Security Requirements

6. Additional security implications

This section discusses additional implications of the interaction between time synchronization protocols and security mechanisms.

This section refers to time synchronization security mechanisms, as well as to "external" security mechanisms, i.e., security mechanisms that are not strictly related to the time synchronization protocol.

6.1. Security and on-the-fly Timestamping

Time synchronization protocols often require protocol packets to be modified during transmission and reception. Both NTP and PTP in one-step mode require clocks to modify protocol packets with the time of transmission or reception.

In the presence of a security mechanism, whether encryption or integrity protection:

- o During transmission the security protocol must be applied after integrating the timestamp into the packet.
- o During reception, the encryption or integrity check must be performed before modifying the packet with the time of reception.

To allow high accuracy, timestamping is typically performed as close to the transmission or reception time as possible. However, since the security engine must be placed between the timestamping function and the physical interface, in some cases it may introduce non-

deterministic latency that causes accuracy degradation. These performance aspects have been analyzed in the literature, e.g., in [1588IPsec] and [Tunnel].

6.2. Security and Two-Step Timestamping

PTP supports a two-step mode of operation, where the time of transmission and the time of reception of protocol packets are measured without modifying the packets. As opposed to one-step mode, two step timestamping can be performed at the physical interface even in the presence of a security mechanism.

Note that if an encryption mechanism such as IPsec is used, it presents a challenge to the timestamping mechanism, since time protocol packets are encrypted when traversing the physical interface, and are thus impossible to identify. A possible solution to this problem [IPsecSync] is to include an indication in the encryption header that identifies time synchronization packets.

6.3. Intermediate Clocks

A time synchronization protocol allows slaves to receive time information from an accurate time source. Time information is sent over a path that often traverses one or more intermediate clocks.

- o In NTP, time information originated from a stratum 1 server can be distributed to stratum 2 servers, and in turn distributed from the stratum 2 servers to NTP clients. In this case, the stratum 2 servers are a layer of intermediate clocks.
- o In PTP, BCs and TCs are intermediate nodes used to improve the accuracy of time information conveyed between the grandmaster and the slaves.

A common rule of thumb in network security is that end-to-end security is the best policy, as it secures the entire path between the data originator and its receiver. The usage of intermediate nodes implies that if a security mechanism is deployed in the network, all intermediate nodes must be exposed to the security key since they must be able to send time information to the slaves, or to modify time information sent through them.

This inhehrent property of using intermediate clocks increases the system's exposure to internal threats, as there is a large number of nodes that are exposed to the security keys.

6.4. The Effect of External Security Protocols on Time Synchronization

Time synchronization protocols are often deployed in systems that use security mechanisms and protocols.

A typical example is the 3GPP Femtocell network [3GPP], where IPsec is used for securing traffic between a Femtocell and the Femto Gateway. In some cases, all traffic between these two nodes may be secured by IPsec, including the time synchronization protocol traffic. This use-case is thoroughly discussed in [IPsecSync].

Another typical example is the usage of MACsec encryption in L2 networks that deploy time synchronization [AvbAssum].

The usage of external security mechanisms may affect time synchronization protocols as follows:

- o Timestamping accuracy can be affected, as described in 6.1.
- o If traffic is secured between two nodes in the network, no intermediate clocks can be used between these two nodes. In the [3GPP] example, if traffic between the Femtocell and the Femto Gateway is encrypted, then time protocol packets are sent over the underlying network without modification, and thus cannot enjoy the improved accuracy provided by intermediate clock nodes.

6.5. External Security Services Requiring Time Synchronization

Certificate validation requires the sender and receiver to be roughly time synchronized. Thus, synchronization is required for establishing security protocols such as IKEv2 and TLS.

An even stronger interdependence between a time synchronization protocol and a security mechanism is defined in [AutoKey], which defines mutual dependence between the acquired time information, and the authentication protocol that secures it.

7. Issues for Further Discussion

- o The key distribution is outside the scope of this document. Although this is a cardinal element in any security system, it is not a security requirement, and is thus not described here.

8. Security Considerations

The security considerations of network timing protocols are presented throughout this document.

9. IANA Considerations

There are no new IANA considerations implied by this document.

10. Acknowledgments

The authors gratefully acknowledge Stefano Ruffini, Dieter Sibold and Dan Grossman for their thorough review and helpful comments. The authors would also like to thank members of the TICTOC WG for providing feedback on the TICTOC mailing list.

This document was prepared using 2-Word-v2.0.template.dot.

11. References

11.1. Normative References

- [KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [NTPv4] Mills, D., Martin, J., Burbank, J., Kasch, W., "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.
- [AutoKey] Haberman, B., Mills, D., "Network Time Protocol Version 4: Autokey Specification", RFC 5906, June 2010.
- [IEEE1588] IEEE TC 9 Test and Measurement Society 2000, "1588 IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems Version 2", IEEE Standard, 2008.

11.2. Informative References

- [Traps] Treytl, A., Gaderer, G., Hirschler, B., Cohen, R., "Traps and pitfalls in secure clock synchronization" in Proceedings of 2007 International Symposium for Precision Clock Synchronization for Measurement, Control and Communication, ISPCS 2007, pp. 18-24, 2007.
- [TM] T. Mizrahi, "Time synchronization security using IPsec and MACsec", ISPCS 2011, pp. 38-43, 2011.

- [SecPTP] J. Tsang, K. Beznosov, "A security analysis of the precise time protocol (short paper)," 8th International Conference on Information and Communication Security (ICICS 2006), pp. 50-59, 2006.
- [SecSen] S. Ganeriwal, C. Popper, S. Capkun, M. B. Srivastava, "Secure Time Synchronization in Sensor Networks", ACM Trans. Info. and Sys. Sec., Volume 11, Issue 4, July 2008.
- [AvbAssum] D. Pannell, "Audio Video Bridging Gen 2 Assumptions", IEEE 802.1 AVB Plenary, work in progress, May 2012.
- [IPsecSync] Y. Xu, "IPsec security for packet based synchronization", IETF, draft-xu-tictoc-ipsec-security-for-synchronization (work in progress), 2011.
- [3GPP] 3GPP, "Security of Home Node B (HNB) / Home evolved Node B (HeNB)", 3GPP TS 33.320 10.4.0 (work in progress), 2011.
- [1588IPsec] A. Treytl, B. Hirschler, "Securing IEEE 1588 by IPsec tunnels - An analysis", in Proceedings of 2010 International Symposium for Precision Clock Synchronization for Measurement, Control and Communication, ISPCS 2010, pp. 83-90, 2010.
- [Tunnel] A. Treytl, B. Hirschler, and T. Sauter, "Secure tunneling of high precision clock synchronisation protocols and other timestamped data", in Proceedings of the 8th IEEE International Workshop on Factory Communication Systems (WFCS), vol. ISBN 978-1-4244-5461-7, pp. 303-313, 2010.
- [DelayAtt] T. Mizrahi, "A Game Theoretic Analysis of Delay Attacks against Time Synchronization Protocols", accepted, to appear in Proceedings of the International IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication, ISPCS, 2012.

12. Contributing Authors

Karen O'Donoghue
ISOC

Email: odonoghue@isoc.org

Authors' Addresses

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam, 20692 Israel

Email: talmi@marvell.com

NTP Working Group
Internet Draft
Intended status: Informational
Expires: April 2013

T. Mizrahi
Marvell
October 15, 2012

Using NTP Extension Field without Authentication
draft-mizrahi-ntp-extension-field-00.txt

Abstract

The Network Time Protocol Version 4 (NTPv4) defines the optional usage of extension fields. An extension field is an optional field that resides at the end of the NTP header, and can be used to add optional capabilities or additional information that is not conveyed in the standard NTP header. The current definition of extension fields in NTPv4 is somewhat ambiguous regarding the connection between extension fields and the presence of a Message Authentication Code (MAC). This draft clarifies the usage of extension fields in the presence and in the absence of a MAC, while maintaining interoperability with existing implementations.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on April 15, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Conventions Used in this Document	3
2.1. Terminology	3
2.2. Terms & Abbreviations	3
3. NTP Extension Field Usage with and without a MAC	4
3.1. Extension Field Format	4
3.2. Extension Fields in the Presence of a MAC	4
3.3. Extension Fields in the Absence of a MAC	4
3.4. Interoperability with Current Implementations	4
3.5. Interoperability with Current Implementations	4
4. Security Considerations	5
5. IANA Considerations	5
6. Acknowledgments	5
7. References	5
7.1. Normative References	5
Appendix A. Requirements from NTPv4 and Autokey	5
A.1. NTP Extension Field for Future Extensions	5
A.2. NTP Extension Field in the Presence of a MAC	6
A.3. The NTP Extension Field Format	6
A.4. NTP Extension Field in Autokey	6

1. Introduction

The NTP header format consists of a set of fixed fields that may be followed by some optional fields. Two types of optional fields are defined, Message Authentication Codes (MAC), and extension fields.

If a MAC is used, it resides at the end of the packet. This field can contain either a 20-octet digest, a 16-octet digest, or a 4-octet crypto-NAK.

NTP extension fields were defined in [NTPv4] as a generic mechanism that allows to add future extensions and features without modifying the NTP header format.

The only currently defined extension field is the one used by the AutoKey protocol [AutoKey].

The NTP specification is somewhat ambiguous with regards to the connection between using extension fields and the presence of a MAC.

- o The definition of the NTP extension field implies that it was intended to be a generic mechanism that can be used for various future features of the protocol (see Section A.1.).
- o On the other hand, the NTP extension field description in [NTPv4] states that a MAC is always present when an extension field is present (see Section A.2.).

The last two quotes seem to be in contradiction; since the extension field was defined as a generic future-compatible building block, it seems unlikely to bind it to a specific feature in the protocol.

Moreover, the extension field parsing rules presented in [AutoKey] imply that an extension field can be present without a MAC, provided that the extension field is at least 7 words long.

This document attempts to resolve the ambiguity with regards to the connection between NTP extension fields and MACs, and describes the usage of extension fields in the absence of a MAC in a way that is interoperable with current implementations.

2. Conventions Used in this Document

2.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [KEYWORDS].

2.2. Terms & Abbreviations

NTPv4	Network Time Protocol Version 4
MAC	Message Authentication Code

3. NTP Extension Field Usage with and without a MAC

3.1. Extension Field Format

The NTP extension field is defined in [NTPv4]. The extension field format is quoted here in Section A.3.

The minimal length of an extension field, as defined in [NTPv4], is 16 octets.

3.2. Extension Fields in the Presence of a MAC

The usage of extension fields in the presence of a MAC is specified in [NTPv4] and in [AutoKey].

3.3. Extension Fields in the Absence of a MAC

Extension fields can be used when a MAC is not present in the NTP packet. In this case, the extension fields must comply to the parsing rules in Section A.4. Specifically:

- o If the packet includes a single extension field, the length of the extension field MUST be at least 7 words, i.e., at least 28 bytes.
- o If the packet includes more than one extension field, the length of the last extension field MUST be at least 28 octets. The length of the other extension fields in this case MUST be at least 16 octets each, as defined in [NTPv4].

A host that supports NTP extension fields MUST parse NTP extension fields as described in Section A.4.

3.4. Interoperability with Current Implementations

The behavior described in Section 3.3. is compliant to [AutoKey], and thus should be compatible with existing implementations that support NTP extension fields.

3.5. Interoperability with Current Implementations

This document currently clarifies the usage of extension fields in the absence of a MAC, in accordance with the definitions in [NTPv4] and [AutoKey].

A future version of this document may define a more generic and flexible usage of extension fields.

4. Security Considerations

The security considerations of the network time protocol are discussed in [NTPv4]. This document clarifies some ambiguity with regards to the usage of the NTP extension field, and thus the behavior described in this document does not introduce new security considerations.

5. IANA Considerations

There are no new IANA considerations implied by this document.

6. Acknowledgments

The author thanks Dave Mills and Danny Mayer for their insightful comments.

This document was prepared using 2-Word-v2.0.template.dot.

7. References

7.1. Normative References

- | | |
|------------|---|
| [KEYWORDS] | Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997. |
| [NTPv4] | Mills, D., Martin, J., Burbank, J., Kasch, W., "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010. |
| [AutoKey] | Haberman, B., Mills, D., "Network Time Protocol Version 4: Autokey Specification", RFC 5906, June 2010. |

Appendix A. Requirements from NTPv4 and Autokey

A.1. NTP Extension Field for Future Extensions

The following paragraph is quoted from [NTPv4], Section 16.

This document introduces NTP extension fields allowing for the development of future extensions to the protocol, where a particular extension is to be identified by the Field Type sub-field within the extension field.

A.2. NTP Extension Field in the Presence of a MAC

The following paragraph is quoted from [NTPv4], Section 7.5.

In NTPv4, one or more extension fields can be inserted after the header and before the MAC, which is always present when an extension field is present.

A.3. The NTP Extension Field Format

Figure 1 specifies the NTP extension field format, and is quoted from [NTPv4]. For further details refer to [NTPv4].

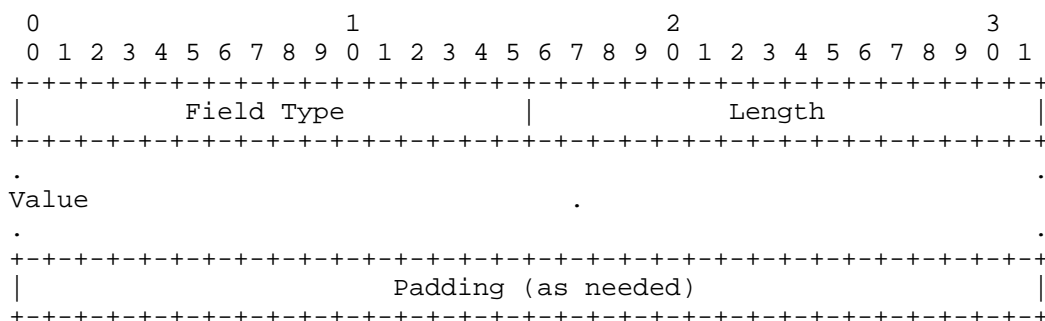


Figure 1 The NTP Extension Field Format

A.4. NTP Extension Field in Autokey

The following paragraph is quoted from [AutoKey], Section 10.

One or more extension fields follow the NTP packet header and the last followed by the MAC. The extension field parser initializes a pointer to the first octet beyond the NTP packet header and calculates the number of octets remaining to the end of the packet. If the remaining length is 20 (128-bit digest plus 4-octet key ID) or 22 (160-bit digest plus 4-octet key ID), the remaining data are the MAC and parsing is complete. If the remaining length is greater than 22, an extension field is present. If the remaining length is less than 8 or not a multiple of 4, a format error has occurred and the packet is discarded; otherwise, the parser increments the pointer by the extension field length and then uses the same rules as above to determine whether a MAC is present or another extension field.

Authors' Addresses

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam, 20692 Israel

Email: talmi@marvell.com

L2VPN Working Group
Internet-Draft
Intended Status: Experimental RFC
Expires: February 2013

Shankar Raman
Balaji Venkat Venkataswami
Gaurav Raina
I.I.T Madras
Bhargav Bhikkaji
Dell-Force10
August 17, 2012

Securing Model-C Inter-Provider VPLS L2 VPNs with Label Hopping and TicToc
draft-mjsraman-l2vpn-vpls-tictoc-label-hop-01

Abstract

In certain models of inter-provider Multi- Protocol Label Switching (MPLS) based Virtual Private Networks (VPNs) spoofing attack against VPN sites is a key concern. For example, MPLS-based VPN inter-provider model "C" for VPLS is not favoured, owing to security concerns in the dataplane, even though it can scale with respect to maintenance of routing state. Since the inner labels associated with VPN sites are not encrypted during transmission, a man-in-the-middle attacker can spoof packets to a specific VPLS site. In this paper, we propose a label-hopping technique which uses a set of randomized labels and a method for hopping amongst these labels using the time instant the packet leaves the port from a sending Provider Edge Router. To prevent the attacker from identifying the labels in polynomial time, we also use an additional label. The proposed technique can be applied to other variants of inter-provider MPLS based VPNs where Multi-Protocol exterior-BGP (MP-eBGP) multi-hop is used. As we address a key security concern, we can make a case for the deployment of MPLS based VPLS inter-provider model "C". Specifically we use the TicToc based Precision Time Protocol LSP to provide the timing for determining the time instant at which the packet is sent from the remote end Provider Edge Router and for calculating when it must have left the that peer at the Provider Edge Router at the near end / receiving end.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1	Introduction	4
1.1	Terminology	5
2.	Methodology of the proposal	5
2.1	PRE-REQUISITES FOR THE LABEL-HOPPING SCHEME	5
2.1.1	MPLS VPLS VPN model "C"	5
2.1.2	PE configuration	6
2.1.3	Control and data-plane flow	6
2.2	LABEL-HOPPING TECHNIQUE	7
2.2.1	Algorithm 1 Control-plane PEne algorithm	8
2.2.2	Algorithm 2 Control-plane PEfa algorithm	10
2.2.3	Algorithm 3 Data-plane PEfa algorithm	11
2.2.4	Algorithm 4 Data-plane PEne algorithm	12
2.2.1	Illustration	13
2.3	SIMULATION AND IMPLEMENTATION	14
2.3.1	Simulation	14
2.3.2	Implementation	14
2.4	CONCLUSION AND FUTURE WORK	15

2.5	ACKNOWLEDGEMENTS	15
3	Security Considerations	16
4	IANA Considerations	16
5	References	16
5.1	Normative References	16
5.2	Informative References	16
	Authors' Addresses	18

1 Introduction

Multi-Protocol Label Switching (MPLS) [6] technology uses fixed size labels to forward data packets between routers. By stacking labels, specific customer services such as Layer 2 Virtual Private Networks (L2-VPNs) such as VPLS (Virtual Private Lan Service) based on Border Gateway Protocol (BGP) extensions are widely deployed in the Internet. BGP-based MPLS L2-VPN services are provided either on a single Internet Service Provider (ISP) core or across multiple ISP cores. The latter cases are known as inter-provider MPLS L2-VPNs which are broadly categorized and referred to as models: "A", "B" and "C".

Model "A" uses back-to-back VPN Routing and Forwarding (VRF) connections between Autonomous System Border Routers (ASBRs). Model "B" uses eBGP redistribution of labelled VPLS routes from Autonomous Systems (AS) to neighbouring AS. Model "C" uses multi-hop MP-eBGP redistribution of labelled VPLS MAC routes and eBGP redistribution of VPLS MAC routes from an AS to a neighbouring AS. Model "C" is more scalable for maintaining routing states and hence preferred for deployment in the Internet; refer to [2] for more details. Security issues in MPLS, especially MPLS-based VPNs has attracted attention [1]. The security of model "A" matches the single-AS standard proposed in [9]. Model "B" can be secured well on the control-plane, but on the data-plane the validity of the outer-most label (Label Distribution or Resource Reservation Protocol label) is not checked. This weakness could be exploited to inject crafted packets from inside an MPLS network core. A solution for this problem is proposed in [2]. Model "C" can be secured on the control-plane but has a security weakness on the data-plane. The Autonomous System Border Routers (ASBRs) do not have any VPN information and hence the inner-most label cannot be validated. In this case, the solution used for Model "B" cannot be applied. An attacker can exploit this weakness to send unidirectional packets into the VPN sites connected to the other AS. Therefore, ISPs using model "C" must either trust each other or not deploy it [4].

Control plane security issue in model "C" can be resolved by using IPSec. If IPSec is used in the data-plane then configuring and maintaining key associations could be extremely cumbersome. Even though model "C" is highly scalable for carrying VPN Routing and Forwarding (VRF) VPLS MAC routes, the vulnerability of the data-plane renders it unusable. The current recommendation is that model "C" must not be used. In model "C", there are at least two labels for each packet: the Provider Edge (PE) label, which defines the Label Switched Path (LSP) to the egress PE, and the VPN label, which defines the VPN associated with the packet on the PE.

In [5], the authors propose encryption techniques, such as IPSec, for securing the provider edge (PE) of the network. The authors also highlight that the processing capacity could be over-burdened. Further, if an attacker is located at the core of the network, or in the network between the providers that constitute an inter-provider MPLS VPN, then spoofing attacks are possible. The vulnerability of MPLS against spoofing attacks and performance impact of IPSec has been discussed in [3]. If the inner labels that identify packets going towards a L2 VPLS VPN site are spoofed, then sensitive information related to services available within the organizational servers can be compromised. As far as we know, there is no scheme available for installing an antispoofing mechanism for these VPLS VPN service labels.

This paper outlines a label-hopping technique that helps to alleviate the data-plane security problem in model "C". We propose a scheme that changes the inner VPLS VPN labels dynamically based on the time instant the packet is sent from the remote-end PE router. By using a mix of algorithms and randomized labels, we can guard against spoofing and related attacks. The advantage of our scheme is that it can be used wherever Multiprotocol-external BGP (MP-eBGP) multi-hop scenarios arise.

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Methodology of the proposal

2.1 PRE-REQUISITES FOR THE LABEL-HOPPING SCHEME

In this section, we briefly review the network topology for model "C", the PE configuration and the control-plane exchanges needed for our proposed scheme.

2.1.1 MPLS VPLS VPN model "C"

The reference MPLS-eBGP based VPLS VPN network for model "C" as described in [11] is shown in Figure 1, which also shows the control plane exchanges. The near-end PE (PEne) and far-end PE (PEfa) are connected through the inter-provider MPLS core. The VPN connectivity is established through a set of routers from different Autonomous Systems (AS) and their ASBRs. In the VPLS VPN, MP-eBGP updates are exchanged for a set of MAC based Forward Equivalence Classes (FECs). These FECs, which have to be protected, originate from the MAC

addresses / FECs behind PEne in a VPLS VPN site or a set of VPLS VPN sites.

2.1.2 PE configuration

Various configurations are needed on the PEs to implement the label hopping scheme. A set of "m" algorithms that generate collision-free labels (universal hashing algorithms) are initially implemented in the PEs. Each algorithm is mapped to an index $A = (a_1; a_2; \dots a_m)$ where $m \geq 1$. The bit-selection pattern used by the PEs for generating the additional label is also configured. PEne must be configured for a FEC or a set of FECs represented by an aggregate label (per VRF label) which will use the label-hopping scheme. For each FEC or a set of FECs, a set of valid labels used for hopping, $K = (k_1; k_2; k_3; \dots k_n)$ where $n \geq 1$ and, $k_i \neq k_j$ if $i \neq j$, is configured in PEne. For the set of labels K time slices $TS = (TS_1; TS_2; TS_3 \dots TS_n)$ are also exchanged. These time slices can be periodically changed and a new set of TS ranging from TS_1 to TS_n can be exchanged after a time duration $TS_Exchange_Interval$ which itself can be randomized from time to time. In the case of bi-directional security, the roles of the PEs can be reversed. In addition to these data sets a random seed is also exchanged. This Random Seed which we will henceforth as $Rseed$ is used to generate the label for the next time slot.

2.1.3 Control and data-plane flow

Initially, set K , set TS and the bit-selection pattern used by the PEs are exchanged securely over the control-plane. Optionally an index from A , representing a hash-algorithm, could also be exchanged. We propose that only the index is exchanged between the PEs, as it enhances the security, for two reasons. First, the algorithm itself is masked from the attacker. Second, the algorithm can be changed frequently, and it would be difficult for the attacker to identify the final mapping that generates the label to be used for a packet. Figure 1 depicts this unidirectional exchange from PEne to PEfa.

The control plane exchanges also involve a-priori constructing a Precision Time Protocol (PTP) LSP for deriving the clock at the PEne and PEfa for a forwarding direction. For the reverse direction another PTP LSP can be constructed as well. In the example that we illustrate we discuss about only a single forwarding direction. The PTP LSP port assigned for a forwarding direction is tied in with the configuration that goes into the inter-PEne-PEfa exchanges to setup the labelling control plane. So each pair of PEne and PEfa knows which PTP port and corresponding PTP LSP as per [12] to be used for the traffic. The PTP LSP is intended for providing the clocking between a pair of PEne and PEfa. The clock / timestamp derived from

this PTP LSP is used in the data plane operation to determine which label is valid at that time instant as will be seen in the Algorithms provided below.

Once the secure control-plane exchanges are completed, we apply the label-hopping technique, and PEfa forwards the labelled traffic towards PEne through the intermediate routers using the label-stacking technique (Figure 2). The stacked labels along with the payload are transferred between the AS and ASBRs before they reach PEne. Using the label-hopping algorithm PEne verifies the integrity of labels. Upon validation, PEne uses the label information to forward the packets to the appropriate VPLS VPN service instance or site. This data-plane exchange from PEfa and PEne is depicted in Figure 3. We now present the label-hopping scheme.

2.2 LABEL-HOPPING TECHNIQUE

In this section, we describe the label-hopping technique and discuss some implementation aspects. Once a data packet destined to the PEne arrives at the PEfa (a) a first-label is chosen using set K and set TS, and the random seed Rseed, and a first-label selected. Next (b) a selected number of bytes from the payload is chosen as input to the hashing algorithm. The hash-digest obtained as a result is used to obtain the additional label for the packet. The agreed bit-selection pattern is then applied on the hash-digest to obtain an additional label, which is then concatenated with the first label. Once PEne receives these packets it verifies both the labels.

The implementation steps for the control-plane at the PEne and PEfa are given by Algorithms 1 and 2. The implementation steps for the data-plane at the PEfa and PEne are given by Algorithms 3 and 4.

2.2.1 Algorithm 1 Control-plane PEne algorithm

Require:

- * FEC[] Forward Equivalence Classes,
- * K[] valid labels,
- * TS[] valid time slices,
- * A[i] hash algorithm instance,
- * I[] the bit-selection pattern chosen for the inner label.
- * Random seed "Rseed" which is used for generating the index into set K (set of labels).
- * PTP port and PTP LSP information

Begin

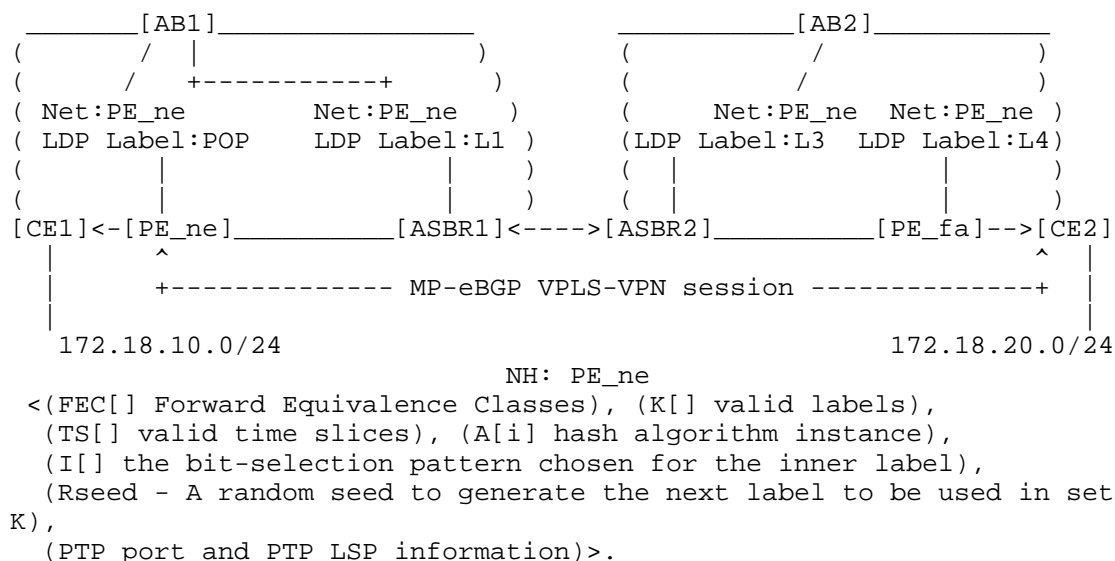
packet = makepacket(FEC,K, TS, A[i], I, Rseed);

CP-SendPacket(PEfa, MP-eBGP, packet);

End

Note: The values in K need not be contiguous and can be randomly chosen from a pool of labels to remove coherence in the label space. Also the algorithms used could be either vendor dependent or a set of standard algorithms mapped the same way by the PEne and PEfa. If the two PEs involved are from different vendors we assume that a set of standard algorithms are used.

Note: Also the values in set TS should be of a coarse granularity of seconds recommended to be higher than 2 seconds.



Exchange all details as per Algorithm 1.

Figure 1: Control-plane exchanges for model C [11]

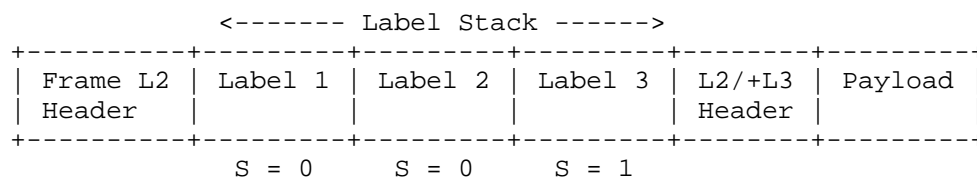


Figure 2: Label stack using scheme outlined for Model "C"

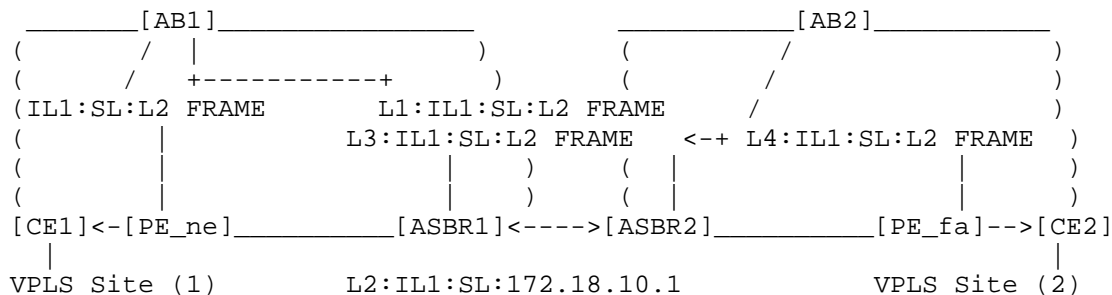


Figure 3: Data-plane flow for model C [11]

2.2.2 Algorithm 2 Control-plane PEfa algorithm

```
Require: None
Begin
packet = CP-ReceivePacket(PEn); // from PEn
FEC[] = ExtractFEC(packet); // extract FECs
K[] = ExtractLabels(packet); // extract the labels
TS[] = ExtractTimeSlices(packet); // extract the time slices
Rseed = ExtractRandomSeed(packet); // extract the Rseed value.
selectHashAlgorithm(A[i]); // hash algorithm to use
RecordValues(FEC); // information for PEfa
RecordValues(K);
RecordValues(TS);
RecordValues(I); // bit-selection pattern to be used
RecordValue(Rseed);
End
```

2.2.3 Algorithm 3 Data-plane PEfa algorithm

```
Require: None

Begin
Initialization :

One Time Init :

BeginInit

CurrentTimeSliceIndex = 0;

CurrentMasterClock = PTP LSP Master Clock Timestamp;

CurrentTimeInstant = CurrentMasterClock;

NextTimeInstant = CurrentMasterClock + TS[CurrentTimeSliceIndex];

EndInit

packet = DP-ReceivePacket(Interface);
match = CheckFEC(packet); // Is the algorithm enabled?
if match == 0 then
    return; // no match
end if
hash-digest = calculateHash(A[i],packet);
if (CurrentTimeInstant <= NextTimeInstant ((+ or -) configured
seconds)) then
    // do nothing;
else
    CurrentTimeSliceIndex++;
    if CurrentTimeSliceIndex == n then // check to wrap around
        CurrentTimeSliceIndex = 0;
    end if
    CurrentTimeInstant = NextTimeInstant;
    NextTimeInstant = CurrentTimeInstant + TS[CurrentTimeSliceIndex];
end if
first-label = K[GenerateRandom(Rseed) MOD n(K)];
end if
additional-label = process(hash-digest,I)
DP-SendPacket(PEne, first-label, additional-label, packet);
End
```

2.2.4 Algorithm 4 Data-plane PENE algorithm

```
Require: None
Initialization :
One Time Init :

BeginInit
CurrentTimeSliceIndex = 0;
CurrentMasterClock = PTP LSP Clock Timestamp;
CurrentTimeInstant = CurrentMasterClock;
NextTimeInstant = CurrentMasterClock + TS[CurrentTimeSliceIndex];
EndInit

Begin
packet = DP-ReceivePacket(Interface);
match = CheckFEC(packet);
if match == 0 then
    return; //no match
end if

label-in-packet=extractPacket(packet, LABEL);
inner-label=extractPacket(packet, INNER-LABEL);
hash-digest=calculateHash(A[i],packet);
if (CurrentTimeInstant <= NextTimeInstant ((+ or -) configured
seconds)) then
    // do nothing;
else
    CurrentTimeSliceIndex++;
    // Save the old RseedIndex into set K
    OldRseedIndex = RseedIndex;
    RseedIndex = (GenerateRandom(Rseed) MOD n(K));
    NextRseedIndex =
        LookAheadRseedIndex(GenerateRandom(Rseed) MOD n(K));
    RollbackRseed(Rseed by 1);
    if CurrentTimeSliceIndex == n then // check to wrap around
        CurrentTimeSliceIndex = 0;
    end if
    CurrentTimeInstant = NextTimeInstant;
    NextTimeInstant = CurrentTimeInstant + TS[CurrentTimeSliceIndex];
end if
// Check if label used before in the previous | current or future
// time slot can be used
// Check with OldRseedIndex, RseedIndex and NextRseedIndex
first-label-range = K[RseedIndex (+or- 1)];
additional-label = process(hash-digest,I)
if label-in-packet ! in first-label-range then
    error(); return;
end if
```

```
if inner-label != additional-label then
    error(); return;
end if
DP-SendPacket(CE1, NULL, NULL, packet);
End
```

Here configured seconds could be a fraction as well.

In order to avoid too many processing cycles in the line cards of PEne and PEfa, the hash- digest is calculated over a predefined size of the payload. An additional inner label is further added to enhance protection against spoofing attacks. With an increased label size, an attacker spends more than polynomial time to guess the VPN instance label for the site behind PEne. There could be two hash-digests that generate the same label. In this case, the two hash-digests is differentiated using the additional label. Collisions can be avoided by re-hashing or any other suitable techniques that are proposed in the literature [8]. If collisions exceed a certain number, then Algorithms 1 and 2 can be executed with a set of new labels.

Note :

It is to be noted that the change in the algorithm to randomly pick up a label for the next time slot will help in avoiding man-in-the-middle attackers from synchronizing with the time slots and the labels which in the previous version of the algorithm was predictable if a large number of packets were observed. The Random seed agreed upon will generate in lock step with the time slots at both the PEfa and PEne, the correct label to be used and that will throw off the attacker from synchronizing with such label changes. Thus even replay attacks may be harder to attempt in such a case.

2.2.1 Illustration

We now briefly illustrate the label-hopping scheme. In Figure 1, using Algorithms 1 and 2, a set of labels are forwarded from PEne to PEfa. The roles of PEne and PEfa are interchanged for reverse traffic. Figure 2 shows a packet from the data-plane for model "C", with the proposed scheme. In the figure, "Label 1" refers to the outermost label, while "Label 2" refers to the label generated from the set K and set TS and "Label 3" refers to an additional label generated as in Algorithm 3. This additional label has bottom of stack bit (denoted by S in Figure 2) set. These labels are stacked immediately onto the packet and the path labels for routing the packets to appropriate intermediary PEs are added. Figure 3 also shows these path labels used by the data packet to reach PEne. When the packet passes through the core of an intermediary AS involved in model "C", or through the network connecting the intermediary AS, the

intruder or the attacker has the capability to inspect the labels and the payload. However, the proposed scheme prevents the attacker from guessing the right combination of the labels. We can increase the size of the additional inner-labels thereby reducing threats from polynomial time attacks.

2.3 SIMULATION AND IMPLEMENTATION

In this section, we present the preliminary simulation results on performance, comparing the label-hopping technique with deep packet inspection where we encrypt and decrypt the complete packet. We also briefly highlight some implementation issues.

2.3.1 Simulation

Implementing the label-hopping scheme for all set of FECs belonging to any or all VPN service instances may cause throughput degradation. This is because the hashdigest computation and derivation of the inner-label / additional inner label calculation can be computation intensive. We therefore compared our technique by choosing a part of the payload as input to our hashing algorithm. We simulated our algorithm on a 2.5 GHz processor Intel dual processor quad core machine. We compared the performance of the label-hopping technique with a deep packet inspection technique where the complete packet was encrypted before transmission and decrypted on reception. These simulation figures indicate that we were able to process 10 million packets per second when we used 64-byte for hashing on a payload of size 1024 bytes. For a hash using 128-byte, we were able to process about 6.3 million packets per second. However with a deep packet inspection where we encrypted and decrypted the complete packet, we were able to process only about 1 million packets per second. In cases where performance becomes a bottleneck, this label-hopping scheme can be applied to specific traffic which are mission-critical, sensitive and most likely need to be protected as they travel from the PEfa to the PEne. Selective application of this service which could be offered as a premium for a selected set of FECs is a suitable option, thereby protecting the traffic of organizations that are paranoid about the integrity of the switched traffic into their VPN sites.

2.3.2 Implementation

One of the concerns in the scheme is the use of payload for generating the random inner label / additional label. If the payload does not vary between two packets then the control-plane exchanges have to be renegotiated with a different algorithm to be used for the hashing for the subsequent packets. The other concern in the scheme is to tackle the problem of fragmentation that can occur along the

path from PE_{fa} to PE_{ne}. We can fragment the packet at PE_{fa} and ensure that the size of the packet is fixed before transmission. We could also employ the Path Maximum Transfer Unit (Path-MTU) discovery process so that packets do not get split into multiple fragments. If packets are fragmented this scheme fails. However, networks usually employ the Path-MTU discovery process to prevent fragmentation and hence this problem may not occur.

2.4 CONCLUSION AND FUTURE WORK

In this paper, we proposed a label-hopping scheme for inter-provider BGP-based MPLS VPLS VPNs that employ MPE-BGP multi-hop control-plane exchanges. In such an environment, without label-hopping, the data-plane is subject to spoofing attacks.

The technique proposed uses a time-based label hopping scheme in addition to the use of the payload to generate an inner label to prevent attackers from easily deciphering labels and their respective VPNs. The scheme is less computationally intensive than encryption-based methods. It prevents the spoofed packets from getting into a VPN site even if the attacker is in the core or at an intervening link between ISPs. In our scheme, we chose the time instant that the packet leaves the first Provider Edge on the far end and this time instant serves as the variable component that the attacker cannot decipher. This requires the use of time synchronization mechanism. This is provided by the PTP LSP constructed for this purpose.

2.5 ACKNOWLEDGEMENTS

The authors would like to acknowledge the UK EP-SRC Digital Economy Programme and the Government of India Department of Science and Technology (DST) for funding given to the IU-ATC. The authors would also like to thank Chandrasekhar.R and Narayana Swamy for his review and valuable comments during the writing of this draft.

3 Security Considerations

The main objective of this proposal is to secure the Inter-Provider MPLS VPLS VPN Model-C data plane by preventing spoofing attacks and other unidirectional attacks against the customer site in this model. The suggestions and algorithms provided will mitigate these attacks to a large extent. The attacker will have many barriers to break through before he/she can successfully mount an attack against the customer site in this model with these algorithms implemented. The availability of TicToc as a method of clocking helps a great deal in this direction.

4 IANA Considerations

Appropriate IANA indicators would have to be provided to exchange the set of values that Algorithm 1 outlines in order to implement this scheme.

5 References

5.1 Normative References

- [KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC1776] Crocker, S., "The Address is the Message", RFC 1776, April 1 1995.
- [TRUTHS] Callon, R., "The Twelve Networking Truths", RFC 1925, April 1 1996.

5.2 Informative References

- [1] S. Alouneh, A. En-Nouaary and A. Agarwal, "MPLS security: an approach for unicast and multicast environments", Annals of Telecommunications, Springer, vol. 64, no. 5, June 2009, pp. 391-400, doi:10.1007/s12243-009-0089-y.
- [2] M. H. Behringer and M. J. Morrow, "MPLS VPN security", Cisco Press, June 2005, ISBN-10: 1587051834.
- [3] B. Daugherty and C. Metz, "Multiprotocol Label Switching and IP, Part 1, MPLS VPNS over IP Tunnels", IEEE Internet Computing, May-June 2005, pp. 68-72, doi:

10.1109/MIC.2005.61.

[4] L. Fang, N. Bitar, J. L. Le Roux and J. Miles, "Interprovider IP-MPLS services: requirements, implementations, and challenges", IEEE Communications Magazine, vol. 43, no. 6, June 2005, pp. 119-128, doi: 10.1109/MCOM.2005.1452840.

[5] C. Lin and W. Guowei, "Security research of VPN technology based on MPLS", Proceedings of the Third International Symposium on Computer Science and Computational Technology (ISCST 10), August 2010, pp. 168-170, ISBN- 13:9789525726107.

[6] Y. Rekhter, B. Davie, E. Rosen, G. Swallow, D. Farinacci and D. Katz, "Tag switching architecture overview", Proceedings of the IEEE, vol. 85, no. 12, December 1997, pp. 1973-1983, doi:10.1109/5.650179.

[7] E. Rosen and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, Standard Track, February, 2006.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, "Introduction to algorithms", 3rd edition, MIT Press, September 2009, ISBN-10:0262033844.

[9] C. Semeria, "RFC 2547bis: BGP/MPLS VPN fundamentals", Juniper Networks white paper, March 2001.

[10] Advance MPLS VPN Security Tutorials [Online], Available:
"http://etutorials.org/Networking/MPLS+VPN+security/Part+II+Advanced+MPLS+VPN+Security+Issues/", [Accessed: 10th December 2011]

[11] Inter-provider MPLS VPN models [Online], Available:
"http://mpls-configuration-on-cisco-iossoftware.org.ua/1587051990/ ch07lev1sec4.html", [Accessed 10th December 2011]

[12] Davari.S et.al, Transporting PTP messages (1588) over MPLS networks, "http://datatracker.ietf.org/doc/draft-ietf-tictoc-1588overmpls/?include_text=1", Work in Progress, October 2011.

[EVILBIT] Bellovin, S., "The Security Flag in the IPv4 Header",

RFC 3514, April 1 2003.

[RFC5513] Farrel, A., "IANA Considerations for Three Letter Acronyms", RFC 5513, April 1 2009.

[RFC5514] Vyncke, E., "IPv6 over Social Networks", RFC 5514, April 1 2009.

Authors' Addresses

Shankar Raman
Department of Computer Science and Engineering
I.I.T Madras,
Chennai - 600036
TamilNadu,
India.

EMail: mjsraman@cse.iitm.ac.in

Balaji Venkat Venkataswami
Department of Electrical Engineering,
I.I.T Madras,
Chennai - 600036,
TamilNadu,
India.

EMail: balajivenkat299@gmail.com

Prof.Gaurav Raina
Department of Electrical Engineering,
I.I.T Madras,
Chennai - 600036,
TamilNadu,
India.

EMail: gaurav@ee.iitm.ac.in

Bhargav Bhikkaji
Dell-Force10,
350 Holger Way,
San Jose, CA

U.S.A

Email: Bhargav_Bhikkaji@dell.com

L3VPN Working Group
Internet-Draft
Intended Status: Experimental RFC
Expires: February 2013

Shankar Raman
Balaji Venkat Venkataswami
Gaurav Raina
I.I.T, Madras
August 17, 2012

Securing Model-C Inter-Provider VPNs with Label Hopping and TicToc
draft-mjsraman-l3vpn-tictoc-label-hop-01

Abstract

In certain models of inter-provider Multi- Protocol Label Switching (MPLS) based Virtual Private Networks (VPNs) spoofing attack against VPN sites is a key concern. For example, MPLS-based VPN inter-provider model "C" is not favoured, owing to security concerns in the dataplane, even though it can scale with respect to maintenance of routing state. Since the inner labels associated with VPN sites are not encrypted during transmission, a man-in-the-middle attacker can spoof packets to a specific VPN site. In this paper, we propose a label-hopping technique which uses a set of randomized labels and a method for hopping amongst these labels using the time instant the packet leaves the port from a sending Provider Edge Router. To prevent the attacker from identifying the labels in polynomial time, we also use an additional label. The proposed technique can be applied to other variants of inter-provider MPLS based VPNs where Multi-Protocol exterior-BGP (MP-eBGP) multi-hop is used. As we address a key security concern, we can make a case for the deployment of MPLS based VPN inter-provider model "C". Specifically we use the TicToc based Precision Time Protocol LSP to provide the timing for determining the time instant at which the packet is sent from the remote end Provider Edge Router and for calculating when it must have left the that peer at the Provider Edge Router at the near end / receiving end.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1	Introduction	4
1.1	Terminology	5
2.	Methodology of the proposal	5
2.1	PRE-REQUISITES FOR THE LABEL-HOPPING SCHEME	5
2.1.1	MPLS VPN model "C"	5
2.1.2	PE configuration	6
2.1.3	Control and data-plane flow	6
2.2	LABEL-HOPPING TECHNIQUE	7
2.2.1	Algorithm 1 Control-plane PEne algorithm	8
2.2.2	Algorithm 2 Control-plane PEfa algorithm	10
2.2.3	Algorithm 3 Data-plane PEfa algorithm	11
2.2.4	Algorithm 4 Data-plane PEne algorithm	12
2.2.1	Illustration	13
2.3	SIMULATION AND IMPLEMENTATION	14
2.3.1	Simulation	14
2.3.2	Implementation	14
2.4	CONCLUSION AND FUTURE WORK	15
2.5	ACKNOWLEDGEMENTS	15
3	Security Considerations	16

4	IANA Considerations	16
5	References	16
5.1	Normative References	16
5.2	Informative References	16
	Authors' Addresses	18

1 Introduction

Multi-Protocol Label Switching (MPLS) [6] technology uses fixed size labels to forward data packets between routers. By stacking labels, specific customer services such as Layer 3 Virtual Private Networks (L3-VPNs) based on Border Gateway Protocol (BGP) extensions are widely deployed in the Internet. BGP-based MPLS L3-VPN services are provided either on a single Internet Service Provider (ISP) core or across multiple ISP cores. The latter cases are known as inter-provider MPLS VPNs which are broadly categorized and referred to as models: "A", "B" and "C" [10].

Model "A" uses back-to-back VPN Routing and Forwarding (VRF) connections between Autonomous System Border Routers (ASBRs). Model "B" uses eBGP redistribution of labelled VPN-IPv4 routes from Autonomous Systems (AS) to neighbouring AS. Model "C" uses multi-hop MP-eBGP redistribution of labelled VPN-IPv4 routes and eBGP redistribution of IPv4 routes from an AS to a neighbouring AS. Model "C" is more scalable for maintaining routing states and hence preferred for deployment in the Internet; refer to [2] for more details. Security issues in MPLS, especially MPLS-based VPNs has attracted attention [1]. The security of model "A" matches the single-AS standard proposed in [9]. Model "B" can be secured well on the control-plane, but on the data-plane the validity of the outer-most label (Label Distribution or Resource Reservation Protocol label) is not checked. This weakness could be exploited to inject crafted packets from inside an MPLS network core. A solution for this problem is proposed in [2]. Model "C" can be secured on the control-plane but has a security weakness on the data-plane. The Autonomous System Border Routers (ASBRs) do not have any VPN information and hence the inner-most label cannot be validated. In this case, the solution used for Model "B" cannot be applied. An attacker can exploit this weakness to send unidirectional packets into the VPN sites connected to the other AS. Therefore, ISPs using model "C" must either trust each other or not deploy it [4].

Control plane security issue in model "C" can be resolved by using IPsec. If IPsec is used in the data-plane then configuring and maintaining key associations could be extremely cumbersome. Even though model "C" is highly scalable for carrying VPN Routing and Forwarding (VRF) routes, the vulnerability of the data-plane renders it unusable. The current recommendation is that model "C" must not be used. A simple solution to this problem is to filter all IP traffic with the exception of the required eBGP peering between the ASBRs, thereby preventing a large number of potential IP traffic-related attacks. However, controlling labelled packets is difficult. In model "C", there are at least two labels for each packet: the Provider Edge (PE) label, which defines the Label Switched Path (LSP) to the egress

PE, and the VPN label, which defines the VPN associated with the packet on the PE.

In [5], the authors propose encryption techniques, such as IPSec, for securing the provider edge (PE) of the network. The authors also highlight that the processing capacity could be over-burdened. Further, if an attacker is located at the core of the network, or in the network between the providers that constitute an inter-provider MPLS VPN, then spoofing attacks are possible. The vulnerability of MPLS against spoofing attacks and performance impact of IPSec has been discussed in [3]. If the inner labels that identify packets going towards a L3 VPN site are spoofed, then sensitive information related to services available within the organizational servers can be compromised. As far as we know, there is no scheme available for installing an antispoofing mechanism for these VPN service labels.

This paper outlines a label-hopping technique that helps to alleviate the data-plane security problem in model "C". We propose a scheme that changes the inner VPN labels dynamically based on the time instant the packet is sent from the remote-end PE router. By using a mix of algorithms and randomized labels, we can guard against spoofing and related attacks. The advantage of our scheme is that it can be used wherever Multiprotocol-external BGP (MP-eBGP) multi-hop scenarios arise.

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Methodology of the proposal

2.1 PRE-REQUISITES FOR THE LABEL-HOPPING SCHEME

In this section, we briefly review the network topology for model "C", the PE configuration and the control-plane exchanges needed for our proposed scheme.

2.1.1 MPLS VPN model "C"

The reference MPLS-eBGP based VPN network for model "C" as described in [11] is shown in Figure 1, which also shows the control plane exchanges. The near-end PE (PEne) and far-end PE (PEfa) are connected through the inter-provider MPLS core. The VPN connectivity is established through a set of routers from different Autonomous Systems (AS) and their ASBRs. In the VPN, MP-eBGP updates are

exchanged for a set of Forward Equivalence Classes (FECs). These FECs, which have to be protected, originate from the prefixes behind PEne in a VPN site or a set of VPN sites.

2.1.2 PE configuration

Various configurations are needed on the PEs to implement the label hopping scheme. A set of "m" algorithms that generate collision-free labels (universal hashing algorithms) are initially implemented in the PEs. Each algorithm is mapped to an index $A = (a_1; a_2; \dots a_m)$ where $m \geq 1$. The bit-selection pattern used by the PEs for generating the additional label is also configured. PEne must be configured for a FEC or a set of FECs represented by an aggregate label (per VRF label) which will use the label-hopping scheme. For each FEC or a set of FECs, a set of valid labels used for hopping, $K = (k_1; k_2; k_3; \dots k_n)$ where $n \geq 1$ and, $k_i \neq k_j$ if $i \neq j$, is configured in PEne. For the set of labels K time slices $TS = (TS_1; TS_2; TS_3 \dots TS_n)$ are also exchanged. These time slices can be periodically changed and a new set of TS ranging from TS_1 to TS_n can be exchanged after a time duration $TS_Exchange_Interval$ which itself can be randomized from time to time. In the case of bi-directional security, the roles of the PEs can be reversed. In addition to these data sets a random seed is also exchanged. This Random Seed which we will henceforth as $Rseed$ is used to generate the label for the next time slot.

2.1.3 Control and data-plane flow

Initially, set K , set TS and the bit-selection pattern used by the PEs are exchanged securely over the control-plane. Optionally an index from A , representing a hash-algorithm, could also be exchanged. We propose that only the index is exchanged between the PEs, as it enhances the security, for two reasons. First, the algorithm itself is masked from the attacker. Second, the algorithm can be changed frequently, and it would be difficult for the attacker to identify the final mapping that generates the label to be used for a packet. Figure 1 depicts this unidirectional exchange from PEne to PEfa.

The control plane exchanges also involve a-priori constructing a Precision Time Protocol (PTP) LSP for deriving the clock at the PEne and PEfa for a forwarding direction. For the reverse direction another PTP LSP can be constructed as well. In the example that we illustrate we discuss about only a single forwarding direction. The PTP LSP port assigned for a forwarding direction is tied in with the configuration that goes into the inter-PEne-PEfa exchanges to setup the labelling control plane. So each pair of PEne and PEfa knows which PTP port and corresponding PTP LSP as per [12] to be used for the traffic. The PTP LSP is intended for providing the clocking

between a pair of PEne and PEfa. The clock / timestamp derived from this PTP LSP is used in the data plane operation to determine which label is valid at that time instant as will be seen in the Algorithms provided below.

Once the secure control-plane exchanges are completed, we apply the label-hopping technique, and PEfa forwards the labelled traffic towards PEne through the intermediate routers using the label-stacking technique (Figure 2). The stacked labels along with the payload are transferred between the AS and ASBRs before they reach PEne. Using the label-hopping algorithm PEne verifies the integrity of labels. Upon validation, PEne uses the label information to forward the packets to the appropriate VPN service instance or site. This data-plane exchange from PEfa and PEne is depicted in Figure 3. We now present the label-hopping scheme.

2.2 LABEL-HOPPING TECHNIQUE

In this section, we describe the label-hopping technique and discuss some implementation aspects. Once a data packet destined to the PEne arrives at the PEfa (a) a first-label is chosen using set K and set TS, and the random seed Rseed, and a first-label selected. Next (b) a selected number of bytes from the payload is chosen as input to the hashing algorithm. The hash-digest obtained as a result is used to obtain the additional label for the packet. The agreed bit-selection pattern is then applied on the hash-digest to obtain an additional label, which is then concatenated with the first label. Once PEne receives these packets it verifies both the labels.

The implementation steps for the control-plane at the PEne and PEfa are given by Algorithms 1 and 2. The implementation steps for the data-plane at the PEfa and PEne are given by Algorithms 3 and 4.

2.2.1 Algorithm 1 Control-plane PEne algorithm

Require:

- * FEC[] Forward Equivalence Classes,
- * K[] valid labels,
- * TS[] valid time slices,
- * Random seed "Rseed" which is used for generating the index into set K (set of labels).
- * A[i] hash algorithm instance,
- * I[] the bit-selection pattern chosen for the inner label.
- * PTP port and PTP LSP information

Begin

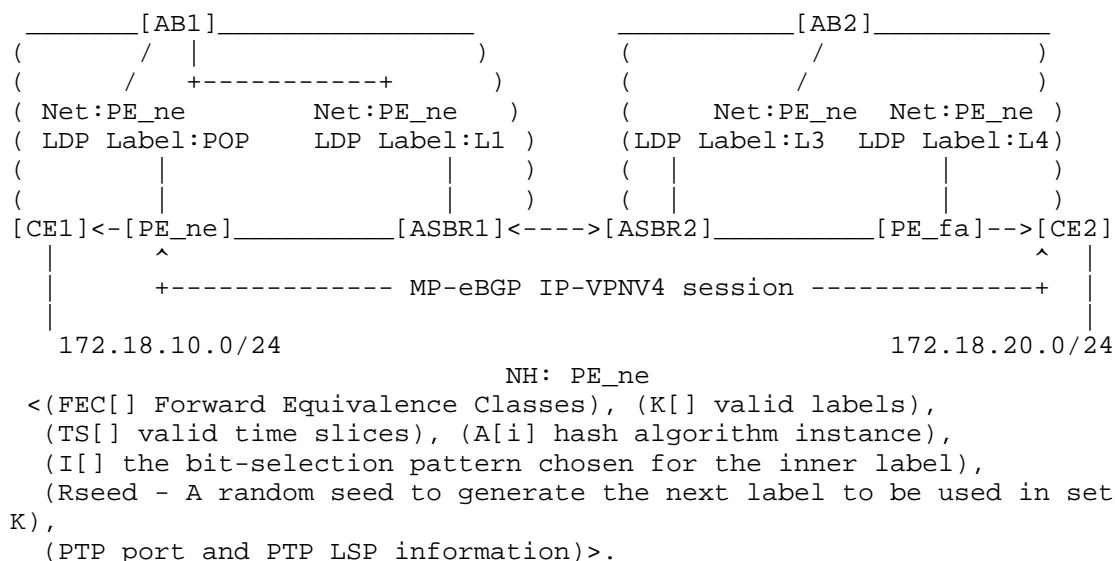
packet = makepacket(FEC,K, TS, A[i], I, Rseed);

CP-SendPacket(PEfa, MP-eBGP, packet);

End

Note: The values in K need not be contiguous and can be randomly chosen from a pool of labels to remove coherence in the label space. Also the algorithms used could be either vendor dependent or a set of standard algorithms mapped the same way by the PEne and PEfa. If the two PEs involved are from different vendors we assume that a set of standard algorithms are used.

Note: Also the values in set TS should be of a coarse granularity of seconds recommended to be higher than 2 seconds.



Exchange all details as per Algorithm 1.

Figure 1: Control-plane exchanges for model C [11]

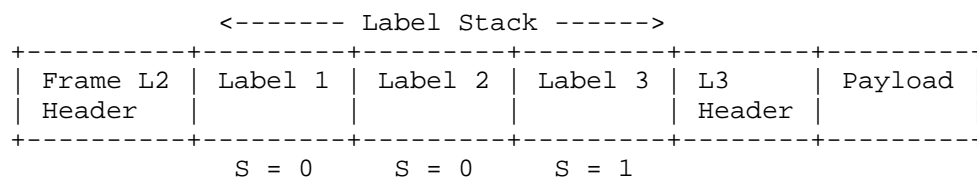


Figure 2: Label stack using scheme outlined for Model "C"

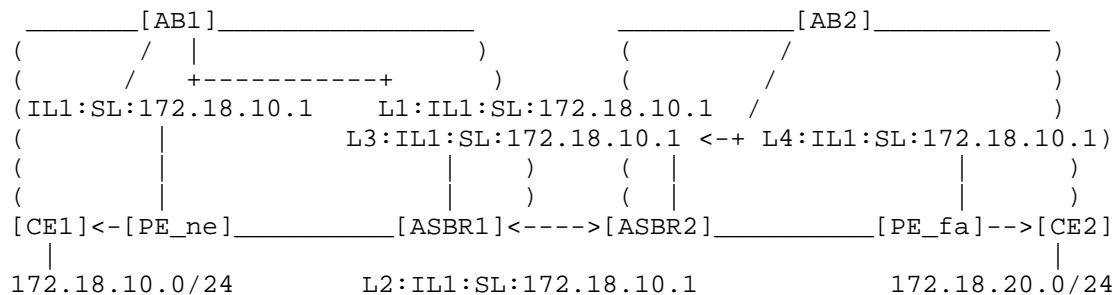


Figure 3: Data-plane flow for model C [11]

2.2.2 Algorithm 2 Control-plane PEfa algorithm

```
Require: None
Begin
packet = CP-ReceivePacket(PEnet); // from PEnet
FEC[] = ExtractFEC(packet); // extract FECs
K[] = ExtractLabels(packet); // extract the labels
TS[] = ExtractTimeSlices(packet); // extract the time slices
Rseed = ExtractRandomSeed(packet); // extract the Rseed value.
selectHashAlgorithm(A[i]); // hash algorithm to use
RecordValues(FEC); // information for PEfa
RecordValues(K);
RecordValues(TS);
RecordValues(I); // bit-selection pattern to be used
RecordValue(Rseed);
End
```

2.2.3 Algorithm 3 Data-plane PEfa algorithm

```
Require: None

Begin
Initialization :

One Time Init :

BeginInit

CurrentTimeSliceIndex = 0;

CurrentMasterClock = PTP LSP Master Clock Timestamp;

CurrentTimeInstant = CurrentMasterClock;

NextTimeInstant = CurrentMasterClock + TS[CurrentTimeSliceIndex];

EndInit

packet = DP-ReceivePacket(Interface);
match = CheckFEC(packet); // Is the algorithm enabled?
if match == 0 then
    return; // no match
end if
hash-digest = calculateHash(A[i],packet);
if (CurrentTimeInstant <= NextTimeInstant ((+ or -) configured
seconds)) then
    // do nothing;
else
    CurrentTimeSliceIndex++;
    if CurrentTimeSliceIndex == n then // check to wrap around
        CurrentTimeSliceIndex = 0;
    end if
    CurrentTimeInstant = NextTimeInstant;
    NextTimeInstant = CurrentTimeInstant + TS[CurrentTimeSliceIndex];
end if
first-label = K[GenerateRandom(Rseed) MOD n(K)];
end if
additional-label = process(hash-digest,I)
DP-SendPacket(PEne, first-label, additional-label, packet);
End
```

2.2.4 Algorithm 4 Data-plane PENE algorithm

```

Require: None
Initialization :
One Time Init :

BeginInit
CurrentTimeSliceIndex = 0;
CurrentMasterClock = PTP LSP Clock Timestamp;
CurrentTimeInstant = CurrentMasterClock;
NextTimeInstant = CurrentMasterClock + TS[CurrentTimeSliceIndex];
EndInit

Begin
packet = DP-ReceivePacket(Interface);
match = CheckFEC(packet);
if match == 0 then
    return; //no match
end if

label-in-packet=extractPacket(packet, LABEL);
inner-label=extractPacket(packet, INNER-LABEL);
hash-digest=calculateHash(A[i],packet);
if (CurrentTimeInstant <= NextTimeInstant ((+ or -) configured
seconds)) then
    // do nothing;
else
    CurrentTimeSliceIndex++;
    // Save the old RseedIndex into set K
    OldRseedIndex = RseedIndex;
    RseedIndex = (GenerateRandom(Rseed) MOD n(K));
    NextRseedIndex =
        LookAheadRseedIndex(GenerateRandom(Rseed) MOD n(K));
    RollbackRseed(Rseed by 1);
    if CurrentTimeSliceIndex == n then // check to wrap around
        CurrentTimeSliceIndex = 0;
    end if
    CurrentTimeInstant = NextTimeInstant;
    NextTimeInstant = CurrentTimeInstant + TS[CurrentTimeSliceIndex];
end if
// Check if label used before in the previous | current or future
// time slot can be used
// Check with OldRseedIndex, RseedIndex and NextRseedIndex
first-label-range = K[RseedIndex (+or- 1)];
additional-label = process(hash-digest,I)
if label-in-packet ! in first-label-range then
    error(); return;
end if

```



```
if inner-label != additional-label then
    error(); return;
end if
DP-SendPacket(CE1, NULL, NULL, packet);
End
```

Here configured seconds could be a fraction as well.

In order to avoid too many processing cycles in the line cards of PEne and PEfa, the hash- digest is calculated over a predefined size of the payload. An additional inner label is further added to enhance protection against spoofing attacks. With an increased label size, an attacker spends more than polynomial time to guess the VPN instance label for the site behind PEne. There could be two hash-digests that generate the same label. In this case, the two hash-digests is differentiated using the additional label. Collisions can be avoided by re-hashing or any other suitable techniques that are proposed in the literature [8]. If collisions exceed a certain number, then Algorithms 1 and 2 can be executed with a set of new labels.

Note :

It is to be noted that the change in the algorithm to randomly pick up a label for the next time slot will help in avoiding man-in-the-middle attackers from synchronizing with the time slots and the labels which in the previous version of the algorithm was predictable if a large number of packets were observed. The Random seed agreed upon will generate in lock step with the time slots at both the PEfa and PEne, the correct label to be used and that will throw off the attacker from synchronizing with such label changes. Thus even replay attacks may be harder to attempt in such a case.

2.2.1 Illustration

We now briefly illustrate the label-hopping scheme. In Figure 1, using Algorithms 1 and 2, a set of labels are forwarded from PEne to PEfa. The roles of PEne and PEfa are interchanged for reverse traffic. Figure 2 shows a packet from the data-plane for model "C", with the proposed scheme. In the figure, "Label 1" refers to the outermost label, while "Label 2" refers to the label generated from the set K and set TS and "Label 3" refers to an additional label generated as in Algorithm 3. This additional label has bottom of stack bit (denoted by S in Figure 2) set. These labels are stacked immediately onto the packet and the path labels for routing the packets to appropriate intermediary PEs are added. Figure 3 also shows these path labels used by the data packet to reach PEne. When the packet passes through the core of an intermediary AS involved in model "C", or through the network connecting the intermediary AS, the

intruder or the attacker has the capability to inspect the labels and the payload. However, the proposed scheme prevents the attacker from guessing the right combination of the labels. We can increase the size of the additional inner-labels thereby reducing threats from polynomial time attacks.

2.3 SIMULATION AND IMPLEMENTATION

In this section, we present the preliminary simulation results on performance, comparing the label-hopping technique with deep packet inspection where we encrypt and decrypt the complete packet. We also briefly highlight some implementation issues.

2.3.1 Simulation

Implementing the label-hopping scheme for all set of FECs belonging to any or all VPN service instances may cause throughput degradation. This is because the hashdigest computation and derivation of the inner-label / additional inner label calculation can be computation intensive. We therefore compared our technique by choosing a part of the payload as input to our hashing algorithm. We simulated our algorithm on a 2.5 GHz processor Intel dual processor quad core machine. We compared the performance of the label-hopping technique with a deep packet inspection technique where the complete packet was encrypted before transmission and decrypted on reception. These simulation figures indicate that we were able to process 10 million packets per second when we used 64-byte for hashing on a payload of size 1024 bytes. For a hash using 128-byte, we were able to process about 6.3 million packets per second. However with a deep packet inspection where we encrypted and decrypted the complete packet, we were able to process only about 1 million packets per second. In cases where performance becomes a bottleneck, this label-hopping scheme can be applied to specific traffic which are mission-critical, sensitive and most likely need to be protected as they travel from the PEfa to the PEne. Selective application of this service which could be offered as a premium for a selected set of FECs is a suitable option, thereby protecting the traffic of organizations that are paranoid about the integrity of the switched traffic into their VPN sites.

2.3.2 Implementation

We are modifying the open source Quagga router software on Linux to implement our scheme. One of the concerns in the scheme is the use of payload for generating the random inner label / additional label. If the payload does not vary between two packets then the control-plane exchanges have to be renegotiated with a different algorithm to be used for the hashing for the subsequent packets. The other concern in

the scheme is to tackle the problem of fragmentation that can occur along the path from PEfa to Pene. We can fragment the packet at PEfa and ensure that the size of the packet is fixed before transmission. We could also employ the Path Maximum Transfer Unit (Path-MTU) discovery process so that packets do not get split into multiple fragments. If packets are fragmented this scheme fails. However, networks usually employ the Path-MTU discovery process to prevent fragmentation and hence this problem may not occur.

2.4 CONCLUSION AND FUTURE WORK

In this paper, we proposed a label-hopping scheme for inter-provider BGP-based MPLS VPNs that employ MPe-BGP multi-hop control-plane exchanges. In such an environment, without label-hopping, the data-plane is subject to spoofing attacks.

The technique proposed uses a time-based label hopping scheme in addition to the use of the payload to generate an inner label to prevent attackers from easily deciphering labels and their respective VPNs. The scheme is less computationally intensive than encryption-based methods. It prevents the spoofed packets from getting into a VPN site even if the attacker is in the core or at an intervening link between ISPs. In our scheme, we chose the time instant that the packet leaves the first Provider Edge on the far end and this time instant serves as the variable component that the attacker cannot decipher. This requires the use of time synchronization mechanism. This is provided by the PTP LSP constructed for this purpose.

2.5 ACKNOWLEDGEMENTS

The authors would like to acknowledge the UK EP-SRC Digital Economy Programme and the Government of India Department of Science and Technology (DST) for funding given to the IU-ATC.

3 Security Considerations

The main objective of this proposal is to secure the Inter-Provider MPLS VPN Model-C data plane by preventing spoofing attacks and other unidirectional attacks against the customer site in this model. The suggestions and algorithms provided will mitigate these attacks to a large extent. The attacker will have many barriers to break through before he/she can successfully mount an attack against the customer site in this model with these algorithms implemented. The availability of TicToc as a method of clocking helps a great deal in this direction.

4 IANA Considerations

Appropriate IANA indicators would have to be provided to exchange the set of values that Algorithm 1 outlines in order to implement this scheme.

5 References

5.1 Normative References

- [KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC1776] Crocker, S., "The Address is the Message", RFC 1776, April 1 1995.
- [TRUTHS] Callon, R., "The Twelve Networking Truths", RFC 1925, April 1 1996.

5.2 Informative References

- [1] S. Alouneh, A. En-Nouaary and A. Agarwal, "MPLS security: an approach for unicast and multicast environments", *Annals of Telecommunications*, Springer, vol. 64, no. 5, June 2009, pp. 391-400, doi:10.1007/s12243-009-0089-y.
- [2] M. H. Behringer and M. J. Morrow, "MPLS VPN security", Cisco Press, June 2005, ISBN-10: 1587051834.
- [3] B. Daugherty and C. Metz, "Multiprotocol Label Switching and IP, Part 1, MPLS VPNS over IP Tunnels", *IEEE Internet Computing*, May-June 2005, pp. 68-72, doi:

10.1109/MIC.2005.61.

[4] L. Fang, N. Bitar, J. L. Le Roux and J. Miles, "Interprovider IP-MPLS services: requirements, implementations, and challenges", IEEE Communications Magazine, vol. 43, no. 6, June 2005, pp. 119-128, doi: 10.1109/MCOM.2005.1452840.

[5] C. Lin and W. Guowei, "Security research of VPN technology based on MPLS", Proceedings of the Third International Symposium on Computer Science and Computational Technology (ISCST 10), August 2010, pp. 168-170, ISBN- 13:9789525726107.

[6] Y. Rekhter, B. Davie, E. Rosen, G. Swallow, D. Farinacci and D. Katz, "Tag switching architecture overview", Proceedings of the IEEE, vol. 85, no. 12, December 1997, pp. 1973-1983, doi:10.1109/5.650179.

[7] E. Rosen and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, Standard Track, February, 2006.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, "Introduction to algorithms", 3rd edition, MIT Press, September 2009, ISBN-10:0262033844.

[9] C. Semeria, "RFC 2547bis: BGP/MPLS VPN fundamentals", Juniper Networks white paper, March 2001.

[10] Advance MPLS VPN Security Tutorials [Online], Available:
"http://etutorials.org/Networking/MPLS+VPN+security/Part+II+Advanced+MPLS+VPN+Security+Issues/", [Accessed: 10th December 2011]

[11] Inter-provider MPLS VPN models [Online], Available:
"http://mpls-configuration-on-cisco-iossoftware.org.ua/1587051990/ ch07lev1sec4.html", [Accessed 10th December 2011]

[12] Davari.S et.al, Transporting PTP messages (1588) over MPLS networks, "http://datatracker.ietf.org/doc/draft-ietf-tictoc-1588overmpls/?include_text=1", Work in Progress, October 2011.

[EVILBIT] Bellovin, S., "The Security Flag in the IPv4 Header",

RFC 3514, April 1 2003.

[RFC5513] Farrel, A., "IANA Considerations for Three Letter Acronyms", RFC 5513, April 1 2009.

[RFC5514] Vyncke, E., "IPv6 over Social Networks", RFC 5514, April 1 2009.

Authors' Addresses

Shankar Raman
Department of Computer Science and Engineering
I.I.T Madras,
Chennai - 600036
TamilNadu,
India.

EMail: mjsraman@cse.iitm.ac.in

Balaji Venkat Venkataswami
Department of Electrical Engineering,
I.I.T Madras,
Chennai - 600036,
TamilNadu,
India.

EMail: balajivenkat299@gmail.com

Prof.Gaurav Raina
Department of Electrical Engineering,
I.I.T Madras,
Chennai - 600036,
TamilNadu,
India.

EMail: gaurav@ee.iitm.ac.in

Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 3, 2012

D. Mills
University of Delaware
K. O'Donoghue, Ed.
Internet Society
D. Hart

H. Stenn
Network Time Foundation, Inc.
October 31, 2011

Control Messages Protocol for Use with Network Time Protocol Version 4
draft-odonoghue-ntp4-control-01

Abstract

This document describes the structure of the control messages used with the Network Time Protocol. These control messages can be used to monitor and control the Network Time Protocol application running on any IP network attached computer. The information in this informational RFC was originally described in Appendix B of RFC 1305.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 3, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	4
2. NTP Control Message Format	6
3. Status Words	8
3.1. System Status Word	8
3.2. Peer Status Word	10
3.3. Clock Status Word	12
3.4. Error Status Word	13
4. Commands	14
5. IANA Considerations	17
6. Security Considerations	17
7. Acknowledgements	17
8. References	17
8.1. Normative References	17
8.2. Informative References	17
Authors' Addresses	18

1. Introduction

Editor's Note (to be removed prior to publication): The text below is taken directly from RFC 1305. Input is requested to update the text to reflect current practice. This is required to fully obsolete RFC 1305.

In a comprehensive network-management environment, facilities are presumed available to perform routine NTP control and monitoring functions, such as setting the leap-indicator bits at the primary servers, adjusting the various system parameters and monitoring regular operations. Ordinarily, these functions can be implemented using a network-management protocol such as SNMP and suitable extensions to the MIB database. However, in those cases where such facilities are not available, these functions can be implemented using special NTP control messages described herein. These messages are intended for use only in systems where no other management facilities are available or appropriate, such as in dedicated-function bus peripherals. Support for these messages is not required in order to conform to this specification.

The NTP Control Message has the value 6 specified in the mode field of the first octet of the NTP header and is formatted as shown below. The format of the data field is specific to each command or response; however, in most cases the format is designed to be constructed and viewed by humans and so is coded in free-form ASCII. This facilitates the specification and implementation of simple management tools in the absence of fully evolved network-management facilities. As in ordinary NTP messages, the authenticator field follows the data field. If the authenticator is used the data field is zero-padded to a 32-bit boundary, but the padding bits are not considered part of the data field and are not included in the field count.

IP hosts are not required to reassemble datagrams larger than 576 octets; however, some commands or responses may involve more data than will fit into a single datagram. Accordingly, a simple reassembly feature is included in which each octet of the message data is numbered starting with zero. As each fragment is transmitted the number of its first octet is inserted in the offset field and the number of octets is inserted in the count field. The more-data (M) bit is set in all fragments except the last.

Most control functions involve sending a command and receiving a response, perhaps involving several fragments. The sender chooses a distinct, nonzero sequence number and sets the status field and R and E bits to zero. The responder interprets the opcode and additional information in the data field, updates the status field, sets the R bit to one and returns the three 32-bit words of the header along

with additional information in the data field. In case of invalid message format or contents the responder inserts a code in the status field, sets the R and E bits to one and, optionally, inserts a diagnostic message in the data field.

Some commands read or write system variables and peer variables for an association identified in the command. Others read or write variables associated with a radio clock or other device directly connected to a source of primary synchronization information. To identify which type of variable and association a 16-bit association identifier is used. System variables are indicated by the identifier zero. As each association is mobilized a unique, nonzero identifier is created for it. These identifiers are used in a cyclic fashion, so that the chance of using an old identifier which matches a newly created association is remote. A management entity can request a list of current identifiers and subsequently use them to read and write variables for each association. An attempt to use an expired identifier results in an exception response, following which the list can be requested again.

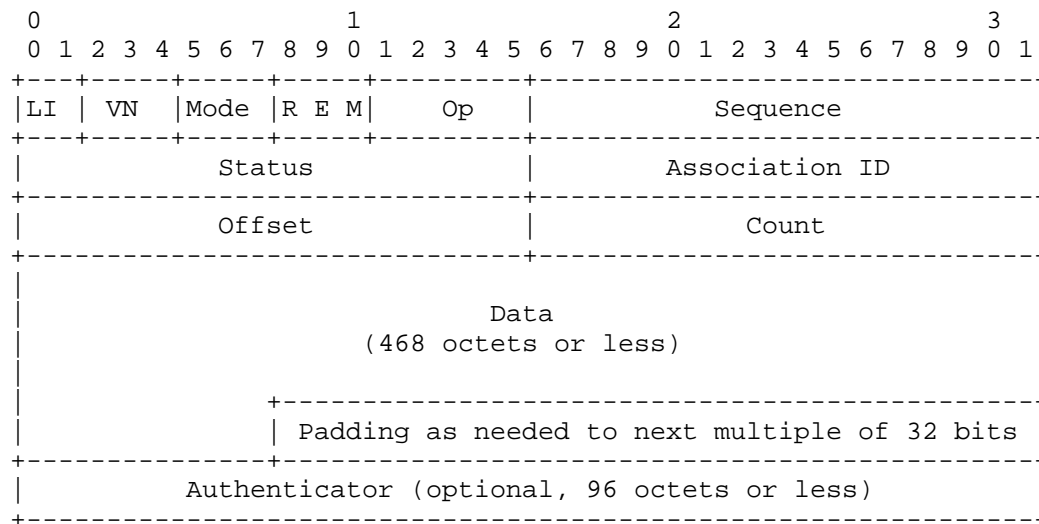
Some exception events, such as when a peer becomes reachable or unreachable, occur spontaneously and are not necessarily associated with a command. An implementation may elect to save the event information for later retrieval or to send an asynchronous response (called a trap) or both. In case of a trap the IP address and port number is determined by a previous command and the sequence field is set as described below. Current status and summary information for the latest exception event is returned in all normal responses. Bits in the status field indicate whether an exception has occurred since the last response and whether more than one exception has occurred.

Commands need not necessarily be sent by an NTP peer, so ordinary access-control procedures may not apply; however, the optional mask/match mechanism suggested in [RFC5905] provides the capability to limit mode 6 processing to selected address ranges.

The Network Time Protocol reference implementation maintained by the University of Delaware and ntp.org provides a utility program, ntpq which enables management and configuration of the ntpd daemon using NTP Control Messages (mode 6). A related utility program, ntpdc, uses an earlier, deprecated implementation-specific binary management protocol using NTP mode 7 datagrams. Due to the implementation complexity of the earlier protocol, the reference implementation has added support for all operations that previously were exposed only via mode 7 to the preferred mode 6 interface. Support for mode 7 requests is likely to be disabled by default in the reference implementation's daemon.

2. NTP Control Message Format

The format of the NTP Control Message header, which immediately follows the UDP header, is shown below. Following is a description of its fields. Bit positions marked as zero are reserved and should always be transmitted as zero.



LI: This is a two-bit integer that must be zero for control message requests and responses. The Leap Indicator value used at this position in most NTP modes is in the System Status Word provided in some control message responses.

Version Number (VN): This is a three-bit integer indicating a minimum NTP version number. NTP servers should not respond to control messages with an unrecognized version number. Requests may intentionally use a lower version number to enable interoperability with earlier versions. The reference implementation utility `ntpq` uses version 2 by default. Responses must carry the same version as the corresponding request.

Mode: This is a three-bit integer indicating the mode. It must have the value 6, indicating an NTP control message.

Response Bit (R): Set to zero for commands, one for responses.

Error Bit (E): Set to zero for normal response, one for error response.

More Bit (M): Set to zero for last fragment, one for all others.

Operation Code (Op): This is a five-bit integer specifying the command function. The values are:

Code	Meaning
0	reserved
1	read status command/response
2	read variables command/response
3	write variables command/response
4	read clock variables command/response
5	write clock variables command/response
6	set trap address/port command/response
7	trap response
8	runtime configuration command/response
9	export configuration to file command/response
10	retrieve remote address stats command/response
11	retrieve local address stats command/response
12	request client-specific nonce command/response
13-30	reserved for future use
31	unset trap address/port command/response

Sequence: This is a 16-bit integer indicating the sequence number. Each request should use a different sequence number. Each response carries the same sequence number as its corresponding request. For asynchronous trap responses, the responder increments the sequence number by one each response, allowing trap receivers to detect missing trap responses. Note the sequence number of each fragment in a multiple-datagram response carries the same sequence number, copied from the request.

Status: This is a 16-bit code indicating the current status of the system, peer or clock, with values coded as described in following sections.

Association ID: This is a 16-bit unsigned integer identifying a valid association, or zero for the system clock.

Offset: This is a 16-bit unsigned integer indicating the offset, in octets, of the first octet in the data area. The offset must be zero in requests. Responses spanning multiple datagrams use a positive offset in all but the first datagram.

Count: This is a 16-bit unsigned integer indicating the length of the data, in octets

Data: This contains the message data for the command or response.

The maximum number of data octets is 468.

Padding: Contains zero to three octets with value zero, as needed to ensure the overall control message size is a multiple of 4 octets.

Authenticator (optional): When an NTP authentication mechanism is used, this contains the message authenticator information defined in section 7.3 of [RFC5905].

3. Status Words

Status words indicate the present status of the system, associations and clock. They are designed to be interpreted by network-monitoring programs and are in one of four 16-bit formats shown in Figure 6 and described in this section. System and peer status words are associated with responses for all commands except the read clock variables, write clock variables and set trap address/port commands. The association identifier zero specifies the system status word, while a nonzero identifier specifies a particular peer association. The status word returned in response to read clock variables and write clock variables commands indicates the state of the clock hardware and decoding software. A special error status word is used to report malformed command fields or invalid values.

3.1. System Status Word

The system status word appears in the status field of the response to a read status or read variables command with a zero association identifier. The format of the system status word is as follows:

```

      0                               1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+-----+-----+-----+-----+
|LI | ClockSrc | Count | Code |
+---+-----+-----+-----+-----+

```

Leap Indicator (LI): This is a two-bit code warning of an impending leap second to be inserted/deleted in the last minute of the current day, with bit 0 and bit 1, respectively, coded as follows: (EDITOR: this could refer to RFC 5905 section 7.3 figure 9 instead.)

LI	Meaning
00	no warning
01	insert second after 23:59:59 of the current day
10	delete second 23:59:59 of the current day
11	unsynchronized

ClockSrc: This is a six-bit integer indicating the current synchronization source, with values coded as follows:

Code	Meaning
0	unspecified or unknown
1	Calibrated atomic clock (e.g., PPS,, HP 5061)
2	VLF (band 4) or LF (band 5) radio (e.g., OMEGA,, WWVB)
3	HF (band 7) radio (e.g., CHU,, MSF,, WWV/H)
4	UHF (band 9) satellite (e.g., GOES,, GPS)
5	local net (e.g., DCN,, TSP,, DTS)
6	UDP/NTP
7	UDP/TIME
8	eyeball-and-wristwatch
9	telephone modem (e.g., NIST)
10-63	reserved

System Event Counter: This is a four-bit integer indicating the number of system events occurring since the last time the System Event Code changed. Upon reaching 15, subsequent events with the same code are not counted.

System Event Code: This is a four-bit integer identifying the latest system exception event, with new values overwriting previous values, and coded as follows:

Code	Meaning
0	unspecified
1	frequency correction (drift) file not available
2	frequency correction started (frequency stepped)
3	spike detected and ignored, starting stepout timer
4	frequency training started
5	clock synchronized
6	system restart
7	panic stop (required step greater than panic threshold)
8	no system peer
9	leap second insertion/deletion armed for end of current month
10	leap second disarmed
11	leap second inserted or deleted
12	clock stepped (stepout timer expired)
13	kernel loop discipline status changed
14	leapseconds table loaded from file
15	leapseconds table outdated, updated file needed

3.2. Peer Status Word

A peer status word is returned in the status field of a response to a read status, read variables or write variables command and appears also in the list of association identifiers and status words returned by a read status command with a zero association identifier. The format of a peer status word is as follows:

```

0                               1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-----+-----+-----+-----+
|  Flags  |  Sel  |  Count  |  Code  |
+-----+-----+-----+-----+
```

Peer Status Flags: This is a set of five bits indicating the status of the peer determined by the packet procedure, with bits assigned as follows:

Peer Status Flag Bit	Value	Meaning
0	0x8000	configured (peer.config)
1	0x4000	authentication enabled (peer.authenable)
2	0x2000	authentication okay (peer.authentic)
3	0x1000	reachable (peer.reach != 0)
4	0x0800	broadcast association

Peer Selection (Sel): This is a three-bit integer indicating the status of the peer determined by the clock-selection procedure, with values coded as follows:

Peer Sel	Meaning
0	rejected
1	discarded by intersection algorithm
2	discarded by table overflow (not currently used)
3	discarded by the cluster algorithm
4	included by the combine algorithm
5	backup source (with more than sys.maxclock survivors)
6	system peer (synchronization source)
7	PPS (pulse per second) peer

Peer Event Counter: This is a four-bit integer indicating the number of peer events that occurred since the last time the peer event code changed. Upon reaching 15, subsequent events with the same code are not counted.

Peer Event Code: This is a four-bit integer identifying the latest peer exception event, with new values overwriting previous values, and coded as follows:

Peer Event Code	Meaning
0	unspecified
1	association mobilized
2	association demobilized
3	peer unreachable
4	peer reachable
5	association restarted or timed out
6	no reply (used only with one-shot ntpd -q, known as ntpdate mode)
7	peer rate limit exceeded (kiss code RATE received)
8	access denied (kiss code DENY received), not currently implemented
9	leap second insertion/deletion at month's end armed by peer vote
10	became system peer (sys.peer)
11	reference clock event (see clock status word)
12	authentication failed
13	popcorn spike suppressed by peer clock filter register
14	entering interleaved mode
15	recovered from interleave error

3.3. Clock Status Word

There are two ways a reference clock can be attached to a NTP service host, as an dedicated device managed by the operating system and as a synthetic peer managed by NTP. As in the read status command, the association identifier is used to identify which one, zero for the system clock and nonzero for a peer clock. Only one system clock is supported by the protocol, although many peer clocks can be supported. A system or peer clock status word appears in the status field of the response to a read clock variables or write clock variables command. This word can be considered an extension of the system status word or the peer status word as appropriate. The format of the clock status word is as follows:

```

0                               1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-----+-----+-----+
|  Reserved   | Count | Code |
+-----+-----+-----+
```

Reserved: An eight-bit integer that should be ignored by requesters and zeroed by responders.

Clock Event Counter: This is a four-bit integer indicating the number of clock events that occurred since the last time the clock event

code changed. Upon reaching 15, subsequent events with the same code are not counted.

Clock Event Code: This is a four-bit integer indicating the current clock status, with values coded as follows:

Clock Status	Meaning
0	clock operating within nominals
1	reply timeout
2	bad reply format
3	hardware or software fault
4	propagation failure (loss of signal)
5	bad date format or value
6	bad time format or value
7-15	reserved

3.4. Error Status Word

An error status word is returned in the status field of an error response as the result of invalid message format or contents. Its presence is indicated when the E (error) bit is set along with the response (R) bit in the response. The format of the Error Status Word is:

0	1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5	
Error Code	Reserved

Error code: an eight-bit integer coded as follows:

Error Status	Meaning
0	unspecified
1	authentication failure
2	invalid message length or format
3	invalid opcode
4	unknown association identifier
5	unknown variable name
6	invalid variable value
7	administratively prohibited
8-255	reserved

Reserved: Responders should use zero. Requesters should ignore the Reserved value to preserve the possibility of future use.

4. Commands

Commands consist of the header and optional data field shown in Section 3. When present, the data field contains a list of identifiers or assignments in the form `<<identifier>>[=<<value>>],<<identifier>>[=<<value>>],...` where `<<identifier>>` is the ASCII name of a system or peer variable specified in Sections 9.1 and 11.1 of RFC 5905 and `<<value>>` is expressed as a decimal, hexadecimal or string constant in the syntax of the C programming language. Where no ambiguity exists, the "s." or "p." prefixes shown in Figure 5 of Section 7.1 of RFC 5905 [RFC5905] can be suppressed. Whitespace (ASCII nonprinting format effectors) can be added to improve readability for simple monitoring programs that do not reformat the data field. Internet Protocol version 4 addresses are represented as four decimal octets without leading zeros, separated by dots. Internet Protocol version 6 addresses are represented as mandated by [RFC5952], without surrounding square brackets unless a port specification is combined with the address. Timestamps, including reference, originate, receive and transmit values, as well as the logical clock, are represented in units of seconds and fractions, preferably in hexadecimal notation, while delay, offset, dispersion and distance values are represented in units of milliseconds and fractions, preferably in decimal notation. All other values are represented as-is, preferably in decimal notation.

Implementations may define variables other than those listed in Figures 6, 7, 16, 17, 18, 19, 27 and 29 of RFC 5905. Called extramural variables, these are distinguished by the inclusion of some character type other than alphanumeric or "." in the name. For those commands that return a list of assignments in the response data field, if the command data field is empty, it is expected that all available variables defined in Figures 6, 7 and 17 of RFC 5905 will be included in the response. For the read commands, if the command data field is nonempty, an implementation may choose to process this field to individually select which variables are to be returned.

Commands are interpreted as follows:

Read Status (1): The command data field is empty or contains a list of identifiers separated by commas. The command operates in two ways depending on the value of the association identifier. If this identifier is nonzero, the response includes the peer identifier and status word. Optionally, the response data field may contain other

information, such as described in the Read Variables command. If the association identifier is zero, the response includes the system identifier (0) and status word, while the data field contains a list of binary-coded pairs <<association identifier>> <<status word>>, one for each currently defined association.

Read Variables (2): The command data field is empty or contains a list of identifiers separated by commas. If the association identifier is nonzero, the response includes the requested peer identifier and status word, while the data field contains a list of peer variables and values as described above. If the association identifier is zero, the data field contains a list of system variables and values. If a peer has been selected as the synchronization source, the response includes the peer identifier and status word; otherwise, the response includes the system identifier (0) and status word.

Write Variables (3): The command data field contains a list of assignments as described above. The variables are updated as indicated. The response is as described for the Read Variables command.

Read Clock Variables (4): The command data field is empty or contains a list of identifiers separated by commas. The association identifier selects the system clock variables or peer clock variables in the same way as in the Read Variables command. The response includes the requested clock identifier and status word and the data field contains a list of clock variables and values, including the last timecode message received from the clock.

Write Clock Variables (5): The command data field contains a list of assignments as described above. The clock variables are updated as indicated. The response is as described for the Read Clock Variables command. The reference implementation daemon requires authentication for this command.

Set Trap Address/Port (6): The command association identifier, status and data fields are ignored. The address and port number for subsequent trap messages are taken from the source address and port of the control message itself. The initial trap counter for trap response messages is taken from the sequence field of the command. The response association identifier, status and data fields are not significant. Implementations should include sanity timeouts which prevent trap transmissions if the monitoring program does not renew this information after a lengthy interval.

Trap Response (7): This command differs from the others described here, which are initiated by a management agent (such as ntpq) and

responded to by a NTP daemon. Trap Response is sent by a NTP daemon to any registered trap receivers when a system, peer or clock exception event occurs. The opcode field is 7 and the R bit is set. The trap counter is incremented by one for each trap sent and the sequence field set to that value. The trap message is sent using the IP address and port fields established by the set trap address/port command. If a system trap the association identifier field is set to zero and the status field contains the system status word. If a peer trap the association identifier field is set to that peer and the status field contains the peer status word. Optional ASCII-coded information can be included in the data field.

Configure (8): The command data is parsed and applied as if supplied in the daemon configuration file. The reference implementation daemon requires authentication for this command.

Save Configuration (9): Write a snapshot of the current configuration to the file name supplied as the command data. The reference implementation daemon requires authentication for this command. Further, the command is refused unless a directory in which to store the resulting files has been explicitly configured by the operator.

Read MRU (10): Retrieves records of recently seen remote addresses and associated statistics. Command data consists of name=value pairs controlling the selection of records, as well as a requestor-specific nonce previously retrieved using this command or opcode 12, Request Nonce. The response consists of name=value pairs where some names can appear multiple times using a dot followed by a zero-based index to distinguish them, and to associate elements of the same record with the same index. A new nonce is provided with each successful response.

Read local address stats (11): Retrieves the local network addresses of the daemon with status and counters for each. Command data is not used in the request. Similar to Read MRU, some response information uses zero-based indexes as part of the variable name preceding the equals sign and value, where each index relates information for a single local address. The reference implementation daemon requires authentication for this command.

Request Nonce (12): Retrieves a 96-bit nonce specific to the requesting remote address, which is valid for a limited period. Command data is not used in the request. The nonce consists of a 64-bit NTP timestamp and 32 bits of hash derived from that timestamp, the remote address, and salt known only to the server which varies between daemon runs. The reference implementation honors nonces which were issued less than 16 seconds prior. Regurgitation of the nonce by a management agent demonstrates to the server that the agent

can receive datagrams sent to the source address of the request, making source address "spoofing" more difficult in a similar way as TCP's three-way handshake.

Unset Trap (31): Removes the requesting remote address and port from the list of trap receivers. Command data is not used in the request. If the address and port are not in the list of trap receivers, the error code is 4, bad association.

5. IANA Considerations

Editor's Note: To be reviewed by the working group prior to completion.

6. Security Considerations

Editor's Note: To be supplied by the working group prior to completion.

7. Acknowledgements

8. References

8.1. Normative References

- [RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.
- [RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, August 2010.

8.2. Informative References

- [RFC5906] Haberman, B. and D. Mills, "Network Time Protocol Version 4: Autokey Specification", RFC 5906, June 2010.
- [RFC5907] Gerstung, H., Elliott, C., and B. Haberman, "Definitions of Managed Objects for Network Time Protocol Version 4 (NTPv4)", RFC 5907, June 2010.
- [RFC5908] Gayraud, R. and B. Lourdelet, "Network Time Protocol (NTP) Server Option for DHCPv6", RFC 5908, June 2010.

Authors' Addresses

Dr. David L. Mills
University of Delaware
Newark, Delaware
US

Email: mills@udel.edu

Karen O'Donoghue (editor)
Internet Society
King George, Virginia
US

Email: odonoghue@isoc.org

David L. Hart
Redmond, Washington
US

Email: hart@ntp.org

Harlan M. Stenn
Network Time Foundation, Inc.
Talent, Oregon
US

Email: stenn@ntp.org

Network Working Group
Internet Draft
Intended status: Experimental
Expires: April 2013

Technion - Israel Institute of Technology
A. Shpiner
R. Tse
C. Schelp
PMC-Sierra
T. Mizrahi
Marvell

October 15, 2012

Multi-Path Time Synchronization
draft-shpiner-multi-path-synchronization-00.txt

Abstract

Clock synchronization protocols are very widely used in IP-based networks. The Network Time Protocol (NTP) has been commonly deployed for many years, and the last few years have seen an increasingly rapid deployment of the Precision Time Protocol (PTP). As time-sensitive applications evolve, clock accuracy requirements are becoming increasingly stringent, requiring the time synchronization protocols to provide high accuracy. Slave Diversity is a recently introduced approach, where the master and slave clocks (also known as server and client) are connected through multiple network paths, and the slave combines the information received through all paths to obtain a higher clock accuracy compared to the conventional one-path approach. This document describes a multi-path approach to PTP and NTP over IP networks, allowing the protocols to run concurrently over multiple communication paths between the master and slave clocks. The multi-path approach can significantly contribute to clock accuracy, security and fault protection. The Multi-Path Precision Time Protocol (MPPTP) and Multi-Path Network Time Protocol (MPNTP) define an additional layer that extends the existing PTP and NTP without the need to modify these protocols. MPPTP and MPNTP also allow backward compatibility with nodes that do not support the multi-path extension.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on April 15, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Conventions Used in this Document	5
2.1. Abbreviations	5
2.2. Terminology	5
3. Multiple Paths in IP Networks	5
3.1. Load Balancing	5
3.2. Using Multiple Paths Concurrently	5
3.3. Two-Way Paths	6
4. Solution Overview	6
4.1. Path Configuration and Identification	6
4.2. Combining	7
5. Multi-Path Time Synchronization Protocols over IP Networks ...	7
5.1. One-Way Multi-Path Synchronization	8
5.1.1. One-Way MPPTP Synchronization Message Exchange	8
5.1.2. One-Way MPNTP Synchronization Message Exchange	9
5.2. Two-Way Multi-Path Synchronization	9
5.2.1. Two-Way MPPTP Synchronization Message Exchange	10

5.2.2. Two-Way MPNTP Synchronization Message Exchange	10
5.3. Using Traceroute for Path Discovery	11
6. Combining Algorithm	11
6.1. Averaging	11
6.2. Switching / Dynamic Algorithm	12
6.3. NTP-like Filtering-Clustering-Combining Algorithm	12
7. Security Considerations	13
8. IANA Considerations	13
9. Acknowledgments	13
10. References	13
10.1. Normative References	13
10.2. Informative References	13

1. Introduction

The two most common time synchronization protocols in IP networks are the Network Time Protocol [NTP], and the Precision Time Protocol (PTP), defined in the IEEE 1588 standard [IEEE1588]. The accuracy of the time synchronization protocols directly depends on the stability and the symmetry of propagation delays on both directions between the master and slave clocks. Depending on the nature of the underlying network, time synchronization protocol packets can be subject to variable network latency or path asymmetry (e.g. [ASYMMETRY], [ASYMMETRY2]). As time sensitive applications evolve, accuracy requirements are becoming increasingly stringent.

Using a single network path in a clock synchronization protocol closely ties the slave clock accuracy to the behavior of the specific path, which may suffer from temporal congestion, faults or malicious attacks. Relying on multiple clock servers as in NTP solves these problems, but requires active maintenance of multiple accurate sources in the network, which is not always possible. The usage of Transparent Clocks (TC) in PTP solves the congestion problem by eliminating the queueing time from the delay calculations, but requires the intermediate routers and switches to support the TC functionality, which is not always the case.

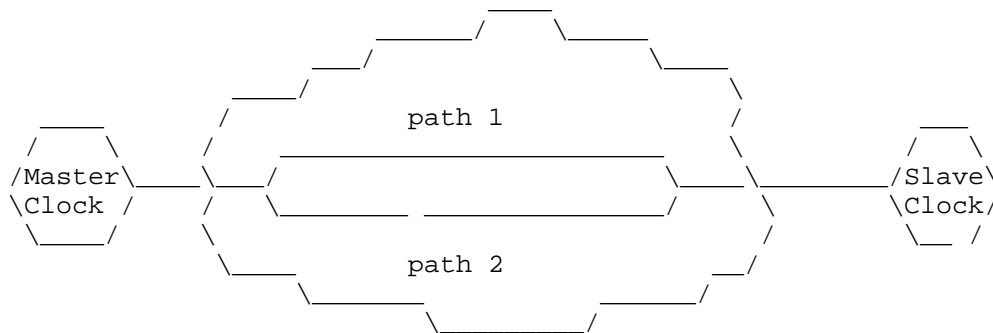


Figure 1 Multi-Path Connection

Since master and slave clocks are often connected through more than one path in the network, as shown in Figure 1, [SLAVEDIV] suggested that a time synchronization protocol can be run over multiple paths, providing several advantages. First, it can significantly increase the clock accuracy as shown in [SLAVEDIV]. Second, this approach provides additional security, allowing to mitigate man-in-the-middle attacks against the time synchronization protocol [DELAY-ATT]. Third, using multiple paths concurrently provides an inherent failure protection mechanism with a negligible recovery time.

This document introduces Multi-Path PTP (MPPTP) and Multi-Path NTP (MPNTP), respectively. These extensions are defined at the network layer, and do not require any changes in the PTP or in the NTP protocols.

MPPTP and MPNTP are defined over IP networks. As IP networks typically combine ECMP routing, this property is leveraged for the multiple paths used in MPPTP and MPNTP. The key property of the multi-path extensions is that clocks in the network can use more than one IP address. Each {master IP, slave IP} address pair defines a path. Depending on the network topology and configuration, the IP combination pairs can form multiple diverse paths used by the multi-path synchronization protocols.

This document introduces two variants for each of the two multi-path protocols; a variant that requires all nodes to support the multi-path protocol, referred to as the two-way variant, and a backward compatible variant that allows a multi-path clock to connect to a conventional single-path clock, referred to as the one-way variant.

2. Conventions Used in this Document

2.1. Abbreviations

ECMP	Equal Cost Multiple Path
LAN	Local Area Network
MPNTP	Multi-Path Network Time Protocol
MPPTP	Multi-Path Precision Time Protocol
NTP	Network Time Protocol
PTP	Precision Time Protocol

2.2. Terminology

In the NTP terminology, a time synchronization protocol is run between a client and a server, while PTP uses the terms master and slave. Throughout this document, the sections that refer to both PTP and NTP generically use the terms master and slave.

3. Multiple Paths in IP Networks

3.1. Load Balancing

Traffic sent across IP networks is often load balanced across multiple paths. The load balancing decisions are typically based on packet header fields: source and destination addresses, Layer 4 ports, the Flow Label field in IPv6, etc. Three common load balancing criteria are per-destination, per-flow and per-packet. The per-destination load balancers take a load balancing decision based on the destination IP address. Per-flow load balancers use various fields in the packet header, e.g., IP addresses and Layer 4 ports, for the load balancing decision. Per-packet load balancers use flow-blind techniques such as round-robin without basing the choice on the packet content.

3.2. Using Multiple Paths Concurrently

To utilize the diverse paths that traverse per-destination load-balancers or per-flow load-balancers, the packet transmitter can vary the IP addresses in the packet header. The analysis in [PARIS2] shows that a significant majority of the flows on the internet traverse per-destination or per-flow load-balancing. It presents statistics

that 72% of the flows traverse per-destination load balancing and 39% of the flows traverse per-flow load-balancing, while only a negligible part of the flows traverse per-packet load balancing. These statistics show that the vast majority of the traffic on the internet is load balanced based on packet header fields.

The approaches in this draft are based on varying the source and destination IP addresses in the packet header. Possible extensions have been considered that also vary the UDP ports. However some of the existing implementations of PTP and NTP use fixed UDP port values in both the source and destination UDP port fields, and thus do not allow this approach.

3.3. Two-Way Paths

A key property of IP networks is that packets forwarded from A to B do not necessarily traverse the same path as packets from B to A. Thus, we define a two-way path for a master-slave connection as a pair of one-way paths: the first from master to slave and the second from slave to master.

In a locally administered network, a traffic engineering approach can be used to verify that time synchronization traffic is always forwarded through bidirectional two-way paths, i.e., that each two way path uses the same route on the forward and reverse directions. However, in the general case two-way paths do not necessarily use the same path for the forward and reverse directions.

4. Solution Overview

The multi-path time synchronization protocols we present are comprised of two building blocks; one is the path configuration and identification, and the other is the algorithm used by the slave to combine the information received from the various paths.

4.1. Path Configuration and Identification

The master and slave clocks must be able to determine the path of transmitted protocol packets, and to identify the path of incoming protocol packets. A path is determined by a {master IP, slave IP} address pair. The synchronization protocol message exchange is run independently through each path.

Each IP address pair defines a two-way path, and thus allows the clocks to bind a transmitted packet to a specific path, or to identify the path of an incoming packet.

In locally administered IP networks, the routing tables across the network can be configured with multiple traffic engineered paths between the pair of clocks. By carefully configuring the routers in such networks it is possible to create diverse paths for each of the IP address pairs between two clocks in the network. However, in public and provider networks the load balancing behavior is hidden from the end users. In this case the actual number of paths may be less than the number of IP address pairs, since some of the address pairs may share common paths.

4.2. Combining

Various methods can be used for combining the time information received from the different paths. This document surveys several combining methods in Section 6. The output of the combining algorithm is the accurate time offset.

5. Multi-Path Time Synchronization Protocols over IP Networks

This section presents two variants of MPPTP and MPNTP; one-way multi-path time synchronization and two-way multi-path time synchronization. In the first variant the multi-path protocol is run only by the slave, and the master is not aware of its usage. In the second variant all clocks must support the multi-path protocol.

The two-way protocol provides higher path diversity by using multiple IP addresses at both ends, the master and slave, while the one-way protocol only uses multiple addresses at the slave. On the other hand, the two-way protocol can only be deployed in networks where all the clocks support this protocol, while the one-way protocol can be used in hybrid networks.

Multi-path time synchronization, in both variants, requires clocks to use multiple IP addresses. This approach introduces a tradeoff; using a large number of IP addresses allows a large number of diverse paths, providing the advantages of slave diversity discussed in Section 1. On the other hand, a large number of IP addresses is more costly, requires the network topology to be more redundant and yields a management overhead.

The descriptions in this section refer to the end-to-end scheme of PTP, but are similarly applicable to the peer-to-peer scheme. The MPNTP protocol described in this document refers to the NTP client-server mode, although the concepts described here can be extended to include the symmetric variant as well.

Multi-path synchronization protocols by nature require protocol messages to be sent as unicast. Specifically in PTP, the following messages must be sent as unicast in MPPTP: Sync, Delay_Req, Delay_Resp, PDelay_Req, PDelay_Resp, Follow_Up, and PDelay_Resp_Follow_Up. Note that [IEEE1588] allows these messages to be sent either as multicast or as unicast.

5.1. One-Way Multi-Path Synchronization

In the one-way approach, only the slave is aware of the fact that multiple paths are used, while the master is agnostic to the usage of multiple paths. This approach allows a hybrid network, where some of the clocks are multi-path clocks, and others are conventional one-path clocks. A one-way multi-path clock presents itself to the network as N independent clocks, using N IP addresses, and N clock identity values. Thus, the usage of multiple slave identities by a slave clock is transparent from the master's point of view, such that it treats each of the identities as a separate slave clock.

5.1.1. One-Way MPPTP Synchronization Message Exchange

The one-way MPPTP message exchange procedure is as follows.

- o Each one-way MPPTP clock has a fixed set of N IP addresses and N corresponding clockIdentities . One of the IP addresses and clockIdentity values are defined as the clock primary identity.
- o The BMC algorithm determines the master.
- o Every one-way MPPTP port that is not in the 'slave' state (i.e., either a master or a clock that has just joined the network) periodically sends Announce messages using its primary identity.
- o A one-way MPPTP clock that is in the 'slave' state periodically transmits a set of N Announce messages using its N identities, while a clock in the 'master' state only transmits Announce messages using its primary identity.
- o The master periodically sends unicast Sync messages from its primary identity address to each of the slaves identified by the sourcePortIdentity and IP address.
- o The slave, upon receiving a Sync message, identifies its path according to the destination IP address. In response to the Sync message the slave sends a Delay_Req unicast message to the primary identity of the master. This message is sent using the slave identity corresponding to the path the Sync was received through.

- o The master, in response to a Delay_Req message from the slave, responds with a Delay_Resp message using the IP address and sourcePortIdentity from the Delay_Req message.
- o Upon receiving the Delay_Resp message, the slave identifies the path using the destination IP address and the requestingPortIdentity. The slave can then compute the corresponding path delay and the offset from the master.

5.1.2. One-Way MPNTP Synchronization Message Exchange

The one-way MPNTP message exchange procedure is as follows.

- o A one-way MPNTP client has N separate identities, i.e., N IP addresses, and N corresponding Reference IDs.
- o A one-way MPNTP client initiates the NTP protocol with an NTP server N times, using each of its N identities.
- o The NTP protocol is maintained between the server and each of the N client identities.
- o The client sends NTP messages to the master using each of its N identities.
- o The server responds to the client's NTP messages using the IP address from the received NTP packet.
- o The client, upon receiving an NTP packet, uses the IP destination address to identify the path it came through, and uses the time information accordingly.

5.2. Two-Way Multi-Path Synchronization

In two-way multi-path synchronization each clock has N IP addresses. Time synchronization messages are exchanged between each combination of {master IP, slave IP} addresses, allowing multiple paths between the master and slave. Note that the actual number of paths between the master and slave may be less than the number of {master, slave} IP address pairs.

Once the multiple two-way connections are established, a separate synchronization protocol exchange instance is run through each of them.

5.2.1. Two-Way MPPTP Synchronization Message Exchange

The two-way MPPTP message exchange procedure is as follows.

- o Every clock periodically sends a set of N Announce messages, using its N addresses as the source IP address. The sourcePortIdentity field in the PTP header remains the same for all PTP messages of a given clock.
- o The BMC algorithm determines the master. Clocks are identified by the sourcePortIdentity and not by the IP address.
- o Each clock learns the multiple IP addresses of other clocks from the source IP addresses of the Announce packets it receives.
- o The master periodically sends unicast Sync messages from each of its N_m IP addresses to each of the slave's N_s IP addresses.
- o The slave, upon receiving a Sync message, identifies its path according to the {source, destination} IP addresses. In response to the Sync message the slave sends a Delay_Req unicast message, swapping the source and destination IP addresses from the Sync message.
- o The master, in response to a Delay_Req message from the slave, responds with a Delay_Resp message using the sourcePortIdentity from the Delay_Req message, and swapping the IP addresses from the Delay_Req.
- o Upon receiving the Delay_Resp message, the slave identifies the path using the {source, destination} IP address pair. The slave can then compute the corresponding path delay and the offset from the master.
- o The PTP protocol messages are sent through each of the $N_m \times N_s$ paths, and the slave combines the information from all these paths.

5.2.2. Two-Way MPNTP Synchronization Message Exchange

The MPNTP message exchange procedure is as follows.

- o Each NTP clock has a set of N IP addresses. The assumption is that the server information, including its multiple IP addresses is known to the NTP clients.

- o The MPNTP client initiates the $N_s \times N_c$ instances of the protocol, one for each {server IP, client IP} pair, allowing the client to combine the information from the $N_s \times N_c$ paths.
(N_s and N_c indicate the number of IP addresses used by the server and client, respectively.)
- o The client sends NTP messages to the master using each of the source-destination address combinations.
- o The server responds to the client's NTP messages using the IP address combination from the received NTP packet.
- o Using the {source, destination} IP address pair in the received packets, the client identifies the path, and performs its computations for each of the paths accordingly.

5.3. Using Traceroute for Path Discovery

The protocols presented above use multiple IP addresses in a single clock to create multiple paths. However, although each two-way path is defined by a different {master, slave} address pair, some of the IP address pairs may traverse exactly the same network path, making them redundant. Traceroute-based path discovery can be used for filtering only the IP addresses that obtain diverse paths. 'Paris Traceroute' [PARIS] and 'TraceFlow' [TRACEFLOW] are examples of tools that discover the paths between two points in the network.

The Traceroute-based filtering can be implemented by both master and slave nodes, or it can be restricted to run only on slave nodes to reduce the overhead on the master.

6. Combining Algorithm

Previous sections discussed the methods of creating the multiple paths and obtaining the time information required by the slave algorithm. This section discusses the algorithm used to combine this information into a single accurate time estimate. Note that the choice of the combining algorithm is local to the slave, and does not affect the interoperability of the protocol. Several combining methods are examined next.

6.1. Averaging

In the first method the slave performs an autonomous time computation for each of the master-slave paths, and obtains the combined time by simply averaging the separate instances. This method can be further

enhanced by adding weights to each of the paths. For example, a reasonable weighting choice is to use an inverse of the round-trip delay between the peers. Another option is to use the inverse of the path delay variance. , which is approximately the maximum likelihood estimator under certain assumptions [WEIGHT-MEAN].

6.2. Switching / Dynamic Algorithm

The switching and dynamic algorithms are presented in [SLAVEDIV]. The switching algorithm periodically chooses a primary path, and performs all time computations based on the protocol packets received through the primary path. The primary path is defined as the path with the minimal distance between the sampled delay and the average delay. The dynamic algorithm dynamically chooses between the result of the switching algorithm and the averaging.

6.3. NTP-like Filtering-Clustering-Combining Algorithm

NTP ([NTP], [NTP2]) provides an efficient algorithm of combining offset samples from multiple peers. The same approach can be used in MPPTP and MPNTP.

In the MPNTP, the selection and combining algorithms treat the offset samples from multiple paths as NTP treats samples from distinct peers. The rest of the selection and combining algorithms, as well as clock control logic is the same as in conventional NTP. In MPPTP, a similar approach to NTP can be adopted.

The combining algorithm [NTP3] contains three steps: filtering, selection and clustering.

In the filtering step, the best of the last n (usually $n=8$) samples of each peer is chosen. The choice criterion is the combination of a round trip delay estimate of the sample and the distance from the average offset of all n samples of a peer.

In the selection step the peers are divided into two groups: true-chimers and false tickers.

The clustering step chooses a subset of the true-chimers, whose peer jitter (the variance of peer offset samples) is smaller than the total select jitter of all selected peer offsets (the variance of the best offset of the selected peers).

The offset samples that passed through the three steps are combined by a weighted average into a single offset estimate. Detailed explanations are provided in [NTP2],[NTP3].

7. Security Considerations

The security aspects of time synchronization protocols are discussed in detail in [TICTOCSEC]. The methods describe in this document propose to run a time synchronization protocol through redundant paths, and thus allow to detect and mitigate man-in-the-middle attacks, as described in [DELAY-ATT].

8. IANA Considerations

There are no IANA actions required by this document.

RFC Editor: please delete this section before publication.

9. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

10. References

10.1. Normative References

- [IEEE1588] IEEE Instrumentation and Measurement Society, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", IEEE Std 1588, 2008.
- [NTP] D. Mills, J. Martin, J. Burbank, W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", IETF, RFC 5905, 2010.

10.2. Informative References

- [ASSYMETRY] Yihua He and Michalis Faloutsos and Srikanth Krishnamurthy and Bradley Huffaker, "On routing asymmetry in the internet", IEEE Globecom, 2005.
- [ASSYMETRY2] Abhinav Pathak, Himabindu Pucha, Ying Zhang, Y. Charlie Hu, and Z. Morley Mao, "A measurement study of internet delay asymmetry", PAM'08, 2008.
- [DELAY-ATT] T. Mizrahi, "A Game Theoretic Analysis of Delay Attacks against Time Synchronization Protocols", ISPCS, 2012.

- [NTP2] Mills, D.L., "Internet time synchronization: the Network Time Protocol", IEEE Trans. Communications COM-39, 10 (October 1991), 1482-1493.
- [NTP3] Mills, D.L., "Improved algorithms for synchronizing computer network clocks", IEEE/ACM Trans. Networks 3, 3(June 1995), 245-254.
- [PARIS] Brice Augustin, Timur Friedman and Renata Teixeira, "Measuring Load-balanced Paths in the Internet", IMC, 2007.
- [PARIS2] B. Augustin, T. Friedman, and R. Teixeira, "Measuring Multipath Routing in the Internet", IEEE/ACM Transactions on Networking, 19(3), p. 830 - 840, June 2011.
- [SLAVEDIV] T. Mizrahi, "Slave Diversity: Using Multiple Paths to Improve the Accuracy of Clock Synchronization Protocols", ISPCS, 2012.
- [TICTOCSEC] T. Mizrahi, K. O'Donoghue, "TICTOC Security Requirements", IETF, draft-ietf-tictoc-security-requirements, work in progress, 2012.
- [TRACEFLOW] J. Narasimhan, B. V. Venkataswami, R. Groves and P. Hoose, "Traceflow", IETF, draft-janapath-intarea-traceflow, work in progress, 2012.
- [WEIGHT-MEAN] http://en.wikipedia.org/wiki/Weighted_mean#Dealing_with_variance

Authors' Addresses

Alex Shpiner
Department of Electrical Engineering
Technion - Israel Institute of Technology
Haifa, 32000 Israel

Email: shalex@tx.technion.ac.il

Richard Tse
PMC-Sierra
8555 Baxter Place
Burnaby, BC
Canada
V5A 4V7

Email: Richard.Tse@pmcs.com

Craig Schelp
PMC-Sierra
8555 Baxter Place
Burnaby, BC
Canada
V5A 4V7

Email: craig.schelp@pmcs.com

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam, 20692 Israel

Email: talmi@marvell.com

NTP Working Group
Internet-Draft
Obsoletes: 5906 (if approved)
Intended status: Standards Track
Expires: January 29, 2013

D. Sibold
PTB
S. Roettger
TU-BS
July 30, 2012

Network Time Protocol: autokey Version 2 Specification
draft-sibold-autokey-00

Abstract

This document describes a security protocol that enables authenticated time synchronization using Network Time Protocol (NTP). Autokey Version 2 obsoletes NTP autokey protocol (RFC 5906) which suffers from various security vulnerabilities. Its design considers the special requirements that are related to the task of precise timekeeping.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 29, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Differences from the original autokey	3
2. Security Threats	3
3. Objectives	4
4. Terms and abbreviations	4
5. Autokey Overview	4
6. Protocol Sequence	5
6.1. Association Message	5
6.2. Certificate Message	5
6.3. Cookie Message	6
6.4. Time request message	6
7. Hash and MAC algorithms	6
7.1. Hash Function for Cookie and Autokey	6
7.2. Hash Function for the Message Authentication Code	6
8. Server Seed Considerations	6
8.1. Server Seed Function	6
8.2. Server Seed Live Time	6
9. IANA Considerations	6
10. Security Considerations	7
11. Acknowledgements	7
12. References	7
12.1. Normative References	7
12.2. Informative References	8
Appendix A. TICTOC Security Requirements	8
Authors' Addresses	9

1. Introduction

In NTP [RFC5905] the autokey protocol [RFC5906] was introduced to provide authenticity to NTP servers and to ensure integrity of time synchronization. It is designed to meet the specific communication requirements of precise timekeeping. Its basic design is a combination of PKI and a pseudo-random sequence of symmetric keys, the so-called autokeys of which each are valid for one packet only. This design maintains the stateless nature of NTP and therefore does not compromise timekeeping precision.

This document focuses on a new definition of the autokey protocol for NTP, autokey version 2. The necessity to renew the autokey specification arises from various severe security vulnerabilities that have been found in a thorough analysis of the protocol [Roettger]. The new specification is based on the same assumptions as the original autokey specification. In particular, the prerequisite is that precise timekeeping can only be accomplished with stateless time synchronization communication, which excludes standard security protocols like IPsec or TLS. This prerequisite corresponds with the requirement that a security mechanism for timekeeping must be designed in such a way that it does not degrade the quality of the time transfer [I-D.ietf-tictoc-security-requirements].

1.1. Differences from the original autokey

Autokey version 2 is a major redraft of the original autokey specification. It is intended to mitigate security vulnerabilities of the original specification and it is based on the suggestions in the analysis of Roettger [Roettger]. The major changes are:

- o The bit length of server seed and cookie has been increased.
- o The utilized hash algorithms are negotiable.
- o The IP addresses of the synchronization partners in the calculation of the cookie have been replaced by the public key of the NTP client.
- o The identity schemes for the verification of the NTP server authenticity have been replaced by a hierarchical public key infrastructure (PKI) based on X.509 certificates.
- o Compatibility with the current autokey specification is not given.
- o The term proventication is not used, i.e., authorization and time synchronization are disentangled.

Discussion

The client verifies the authenticity of the server via PKI infrastructure. To this end, it has to verify the certification chain up to a trusted authority which, in the context of the PKI, is a certification authority (CA). Proventication may be established if the trusted authority is also the NTP stratum 1 server. See also the discussion in Section 6.2.

2. Security Threats

A profound analysis of security threats and requirements for NTP and Precision Time Protocol (PTP) can be found in the I-D [I-D.ietf-

tictoc-security-requirements].

3. Objectives

The objectives of the autokey specifications are as follows:

- o Authenticity: Autokey enables the client to authenticate its NTP server or peer.
- o Integrity: Autokey protects the integrity of time synchronization packets via a message authentication code (MAC) or a hash-based message authentication code (HMAC).
- o Confidentiality: Autokey does not provide confidentiality protection of the NTP packets.
- o Modes of operation: All operational modes of NTP are supported (Client-Server, symmetric, broadcast).
- o Hybrid mode: Both secure and insecure communication modes are possible for NTP servers and clients, respectively.
- o Compatibility: Interoperation with autokey version 1 and the symmetric key scheme described in [RFC1305] is not given. Insecure NTP associations are not affected.
- o Leap seconds are not in the scope of autokey.

4. Terms and abbreviations

- o Throughout this document the term "autokey" refers to autokey version 2.

5. Autokey Overview

In autokey, authenticity and integrity of NTP packets are ensured by an attached key ID and a message authentication code (MAC). The MAC is calculated with a so-called "autokey" which is a symmetric key that is valid for one packet only. The MAC is given by

$$\text{MAC} = \text{H}(\text{autokey} || \text{NTP packet}),$$

where `||` indicates concatenation and in which H is a hash algorithm on which client and server agree during the association message (ASSOC) exchange. The key ID uniquely identifies the autokey. The autokeys are calculated for each NTP packet according to:

$$\text{autokey} = \text{H}(\text{key ID} || \text{cookie}),$$

in which H is a hash function on which client and server have to agree (during ASSOC) and which is not necessarily identical to the one used for the MAC calculation. The cookie is a 128 bit secret

between client and server. It is exchanged during the cookie message protocol sequence (COOK). The cookie is calculated by the server via

```
cookie = MSB_128 (H(server seed || public key of client)).
```

The same hash algorithm H is utilized as in the calculation of the autokey. The function MSB_128 cuts off the 128 most significant bits of the result of the hash function. The server seed is a 128 bit random value of the server, which has to be kept secret. The cookie thus never changes. To comply with 4.5.3 in [I-D.ietf-tictoc-security-requirements] the server seed has to be changed periodically. The server does not keep a state of the client. Therefore it has to recalculate the cookie each time it receives a request from the client. To this end, the client has to attach its public key to each request (see Section 6.4).

Discussion

Alternative cookie calculation: Instead of using the client's public key for the cookie calculation, the hash value of the public key can be used. This has the advantage that during the time request message the client only needs to send the hash of its public key and not the whole public key itself.

6. Protocol Sequence

6.1. Association Message

The protocol sequence starts with the association message, in which the client sends an NTP packet with an extension field of type association. It contains the hostname of the client and a status word which contains the algorithms used for the signatures and the status of the connection. The response contains the hostname of the server and the algorithms for the signatures. Client and server MUST agree upon the employed MAC and hash algorithms.

6.2. Certificate Message

In this step, the client receives the certification chain up to the trusted authority (TA). To this end, the client requests the certificate for the subject name (hostname) of the NTP server. The response contains the certificate with the issuer name. If the issuer name is different from the subject name, the client requests the certificate for the issuer. This continues until it receives a certificate which is issued by a TA. The client recognizes the TA because it has a list of certificates which are accepted as TAs. The client has to prove that each issuer is authorized to issue new certificates. To this end, it has to prove that the X.509v3 extension contains the field "CA:TRUE". With the established certification chain the client is able to verify the server signatures and, hence, the authenticity of the server messages with extension fields is ensured.

Discussion

Note that this certification chain is a priori independent of the time synchronization chain, because the TA and the NTP root are not inevitably identical. This has consequences if proventication is required (Requirement 4.1.2 in [I-D.ietf-tictoc-security-requirements]). In this case, proventication can be ensured only if the NTP root server is also a recognized TA, hence a CA.

6.3. Cookie Message

The client requests a cookie from the server, which is used to calculate the autokeys. The request includes the public key of the client. The public key is used by the server to calculate the cookie. The response of the server contains the cookie encrypted with the public key.

6.4. Time request message

The client request includes a new extension field "time request" which contains its public key. The server needs the public key to recalculate the cookie for the client. The response is a normal NTP packet without extension field.

7. Hash and MAC algorithms

Hash algorithms are used for the calculation of cookie, autokey and MAC.

7.1. Hash Function for Cookie and Autokey

The hash algorithm utilized for the calculation of the cookie and the autokey is negotiated during the association message exchange (Section 6.1). The client MUST request SHA-1 or a stronger hash function. The server also MUST provide SHA-256.

7.2. Hash Function for the Message Authentication Code

The hash function for the MAC is negotiated during the association message exchange in Section 6.1. Client and server SHOULD negotiate a Keyed-Hash Message Authentication Code [RFC2104].

8. Server Seed Considerations

The server has to calculate a random seed which has to be kept secret and which has to be changed periodically.

8.1. Server Seed Function

8.2. Server Seed Live Time

9. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

10. Security Considerations

The client has to verify the validity of the certificates during the certification message exchange (Section 6.2). Since it generally has no reliable time during this initial communication phase, it is impossible to verify the period of validity of the certificates. Therefore, the client MUST use one of the following approaches:

- o The TA and the dependent certificates are trusted by default. Usually this will be the case in corporation networks.
- o The client ensures that the certificates are not revoked. To this end, the client uses the Online Certificate Status Protocol (OCSP) defined in [RFC6277].
- o The client requests a different service to get an initial time stamp in order to be able to verify the certificates' periods of validity. To this end, it can, e.g., use a secure shell connection to a reliable host. Another alternative is to request a time stamp from a Time Stamping Authority (TSA) by means of the Time-Stamp Protocol (TSP) defined in [RFC3161].

11. Acknowledgements

12. References

12.1. Normative References

- [I-D.ietf-tictoc-security-requirements]
Mizrahi, T. and K. O'Donoghue, "TICTOC Security Requirements", Internet-Draft draft-ietf-tictoc-security-requirements-02, June 2012.
- [RFC1305] Mills, D., "Network Time Protocol (Version 3) Specification, Implementation", RFC 1305, March 1992.
- [RFC2104] Krawczyk, H., Bellare, M. and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3161] Adams, C., Cain, P., Pinkas, D. and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, August 2001.
- [RFC5905] Mills, D., Martin, J., Burbank, J. and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

[RFC5906] Haberman, B. and D. Mills, "Network Time Protocol Version 4: Autokey Specification", RFC 5906, June 2010.

[RFC6277] Santesson, S. and P. Hallam-Baker, "Online Certificate Status Protocol Algorithm Agility", RFC 6277, June 2011.

12.2. Informative References

[Roettger] Roettger, S., "Analysis of the NTP Autokey Procedures", February 2012.

Appendix A. TICTOC Security Requirements

The following table compares the autokey specifications against the tictoc security requirements [I-D.ietf-tictoc-security-requirements].

Section	Requirement from I-D tictoc security-requirements-02	Type	Autokey V2
4.1	Authentication of sender.	MUST	OK
	Authentication of master.	MUST	OK
	Proventication	MUST	Open 1)
	Authentication of slaves.	SHOULD	OK
	PTP: Authentication of TCs.	SHOULD	N/A
	PTP: Authentication of Announce messages.	SHOULD	N/A
4.2	Integrity protection.	MUST	OK
	PTP: hop-by-hop integrity protection.	MUST	N/A
	PTP: end-to-end integrity protection.	SHOULD	N/A
4.3	Protection against DoS attacks.	MUST	NTP 2)
4.4	Replay protection.	MUST	NTP 2)
4.5	Security association.	MUST	OK
	Unicast and multicast associations.	MUST	OK
	Key freshness.	MUST	OK
4.6	Performance: no degradation in quality of time transfer.	MUST	OK
	Performance: lightweight.	SHOULD	YES
	Performance: storage, bandwidth.	MUST	OK
4.7	Confidentiality protection.	MAY	NO
	Protection against delay attacks.	MAY	NO
4.9	Secure mode.	MUST	NTP? 3)
	Hybrid mode.	MAY	YES

1) Refer to discussion in Section 6.2. 2) These requirements are fulfilled by the NTP on-wire protocol. 3) Has still to be checked.

Authors' Addresses

Dieter Sibold
Physikalisch-Technische Bundesanstalt
Bundesallee 100
Braunschweig, D-38116
Germany

Phone: +49-(0)531-592-8420
Email: dieter.sibold@ptb.de

Stephen Roettger
Technische Universitaet Braunschweig
Email: stephen.roettger@googlemail.com