A New Approach to IEEE1588

Applying a Client/Server Model to Unicast PTP as an Alternative to NTP

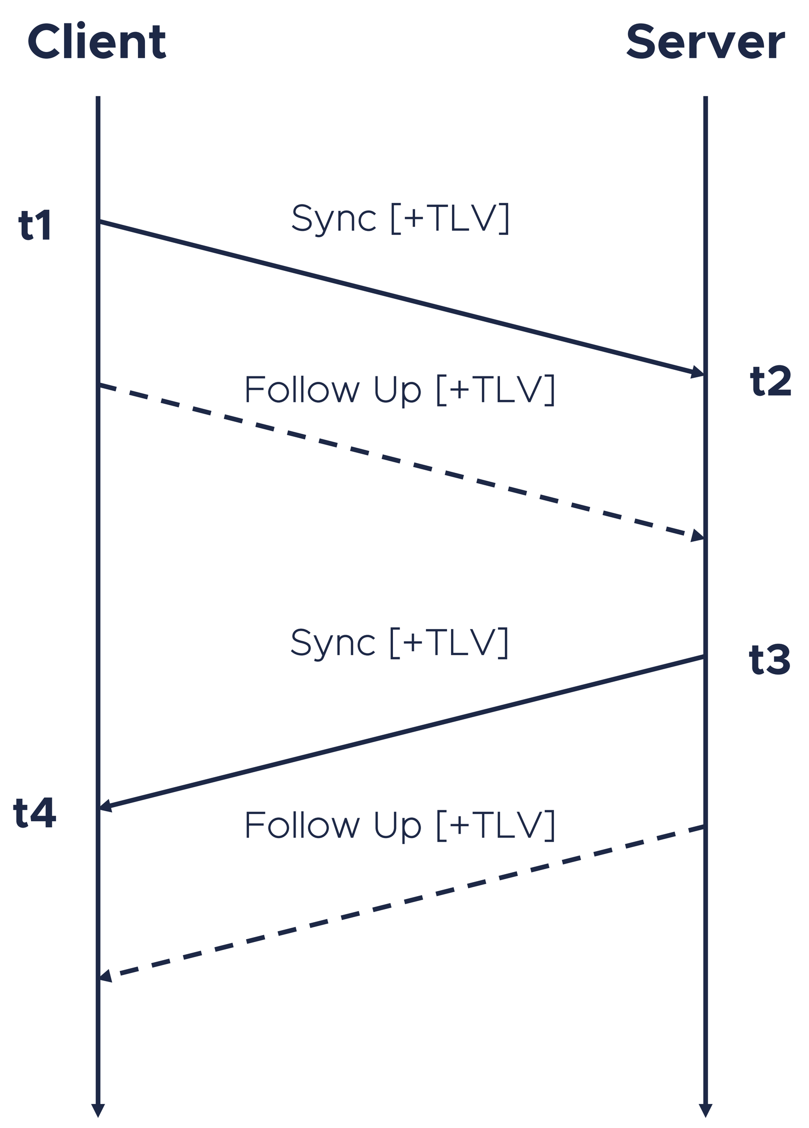
MEINBERG
The Synchronization Experts.

❖ **Improve Efficiency** ❖ **Reduce Complexity** ❖ **Enhance Security** ❖ **Maintain Accuracy**

Designed to improve efficiency while reducing complexity, this new approach to IEEE1588 modifies unicast PTP to enable it to adopt a client/server model, while maintaining—or even improving on—the achievable synchronization accuracy by combining the hardware timestamping capabilities of PTP with certain aspects of the clock filtering, selection, and combining algorithms provided by NTP.

The client/server-based communication flow of this new concept allows it to adopt established security technologies such as NTS (Network Time Security) ordinarily used for NTP for the purpose of authenticating PTP messages exchanged between clients and servers—potentially a game changer when it comes to PTP traffic security.

How It Works



Sync Request

The client sends Sync and Follow Up Messages (assuming that Two-Step mode is running) with a Request TLV attached to either of them.

t1 *Sync Message egress timestamp (Client)*
t2 *Sync Message ingress timestamp (Server)*

Sync Response

Upon receipt of a complete Sync Request sequence, the server responds with Sync and Follow Up Messages (again, assuming Two-Step mode) with a Response TLV attached to the same message as in the Request sequence.

t3 *Sync Message egress timestamp (Server)*
t4 *Sync Message ingress timestamp (Client)*

Request TLV

Field	Octets	TLV Offset	Value
tlvType	2	0	3 (ORG_EXT)
lengthField	2	2	36+N
organizationId	3	4	0xEC4670
organizationSubType	3	7	0x526571
flags	4	10	...
pad	22+N	14	0

Response TLV

Field	Octets	TLV Offset	Value
tlvType	2	0	3 (ORG_EXT)
lengthField	2	2	36+N
organizationId	3	4	0xEC4670
organizationSubType	3	7	0x526573
flags	4	10	...
error	2	14	...
reqIngressTimestamp	10	16	...
reqCorrectionField	8	26	...
utcOffset	2	34	...
bmcaComparisonDS	0+N	36	...

PTP Message Header

Field		Value	
majorSdoId	messageType	0	see IEEE1588
minorVersionPTP	versionPTP	1	2
messageLength		see IEEE1588	
domainNumber		0	
minorSdoId		0	
flagField		see IEEE1588	
correctionField		0	
messageTypeSpecific		0	
sourcePortIdentity		see IEEE1588	
sequenceId		see IEEE1588	
controlField		see IEEE1588	
logMessageInterval		Req.: Interval (log_2), Resp.: 0x7f	

Mean Path Delay

$$(t2 - t1 + t4 - t3) / 2$$

Offset

$$(t2 + t3 - t1 - t4) / 2$$

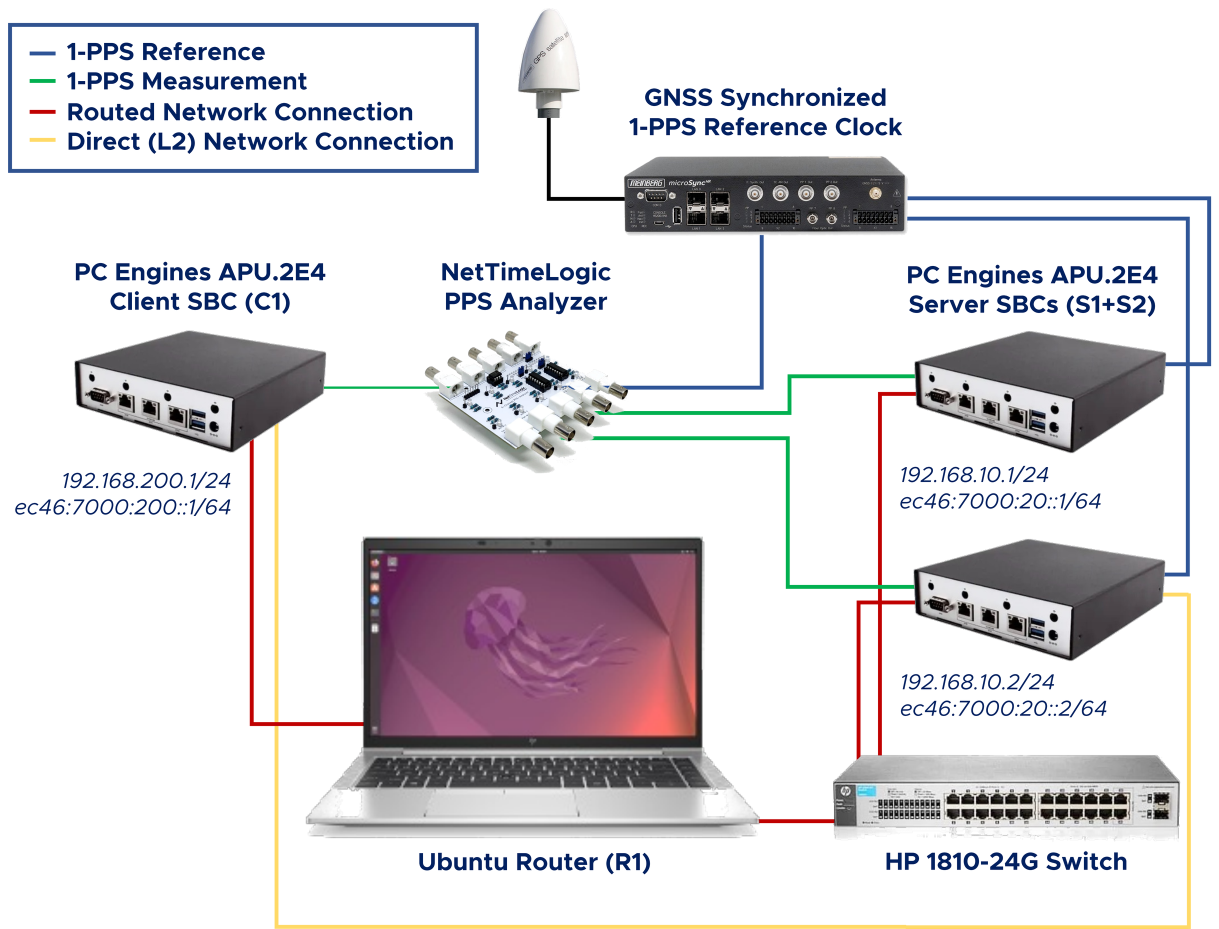
Try it out!

Our reference implementation **flashptpd** used for the presented test scenarios and measurements is free and open-source (MIT License).

Please feel free to download, test, or contribute:

https://github.com/thomas-behn/flashptpd

How It Performs



Test Setup

In the test setup, we used three **PC Engines APU.2E4 SBCs** as DUTs: one as a client (C1) and two as servers (S1+S2). These SBCs have three Intel i210-AT NICs, each with its own PTP hardware clock (PHC).

A laptop running Ubuntu Linux acted as a software router (R1) between the client and servers.

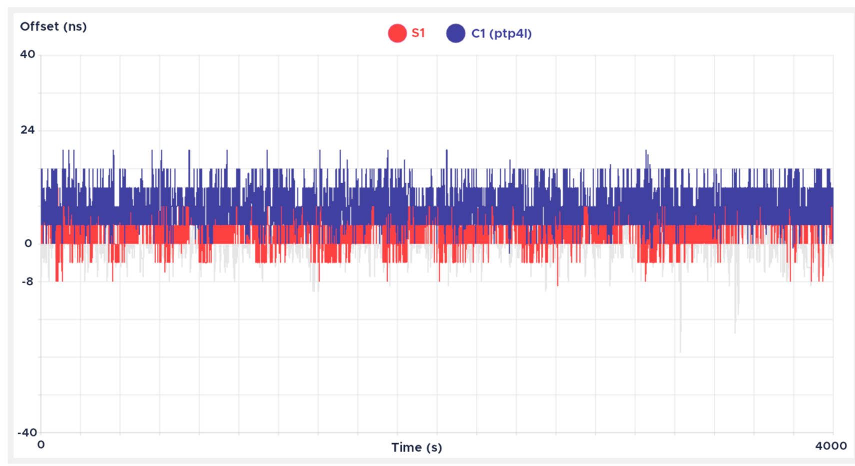
The servers and the PPS analyzer board provided by NetTimeLogic GmbH shared a GNSS synchronized 1-PPS reference clock.

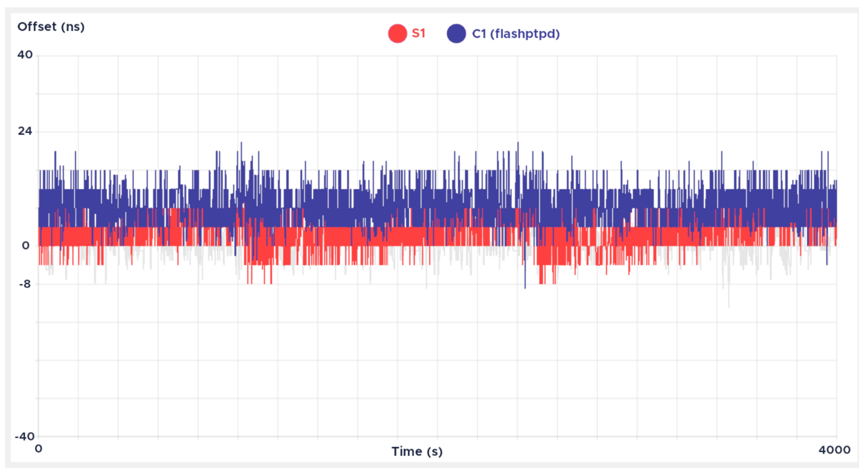
In each of our test scenarios, we ran both ptp4l and flashptpd for at least 3600 seconds (1 hour), one after the other, before comparing the measurements, see results below.

ptp4l | **flashptpd**

1) Direct (L2) Connection – Small Delay, No Jitter

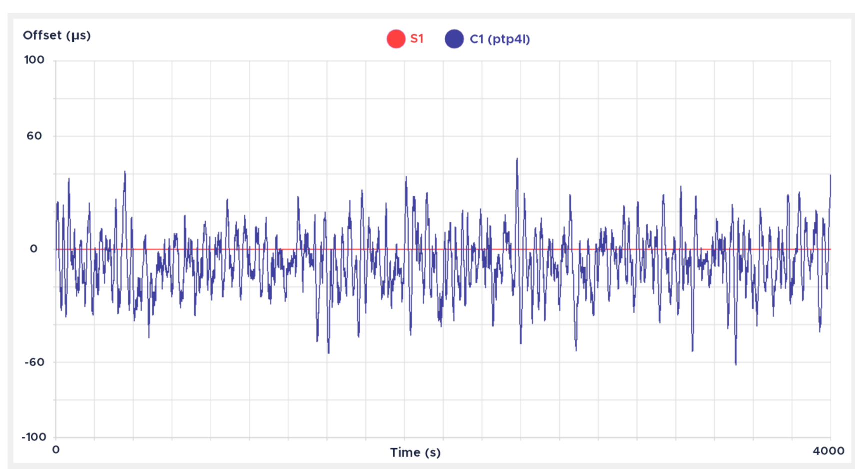
Both ptp4l as well as flashptpd adjusted the PHC to a very stable and small offset of **-8 to +24ns**.

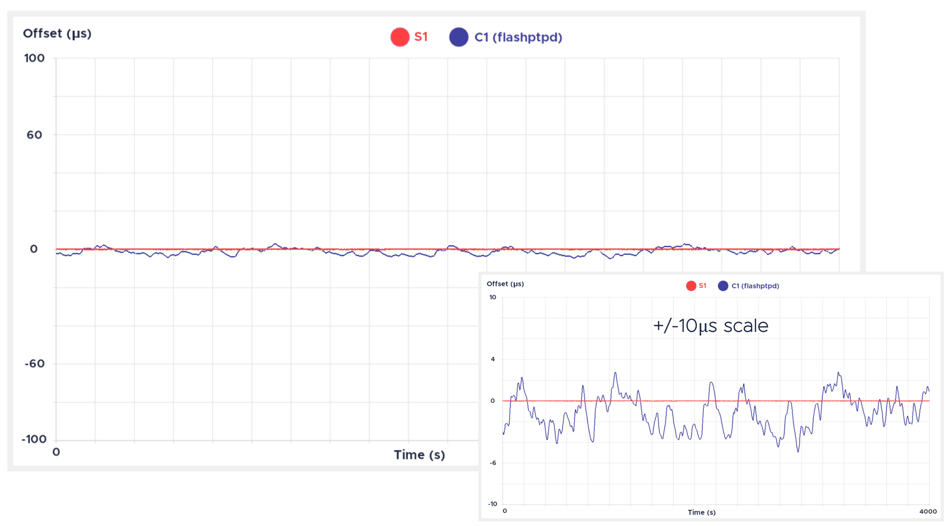




2) Routed Connection - 30 to 750µs Delay, up to 300µs Jitter

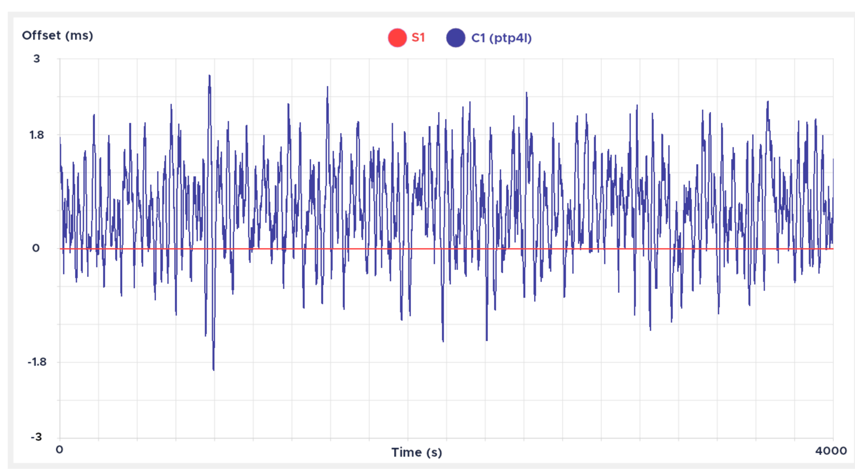
While ptp4l kept the PHC within around +/-60µs (even with high message rates and small PI proportional and integral constants), flashptpd adjusted the PHC to a pretty stable offset of **-5 to +3µs**.

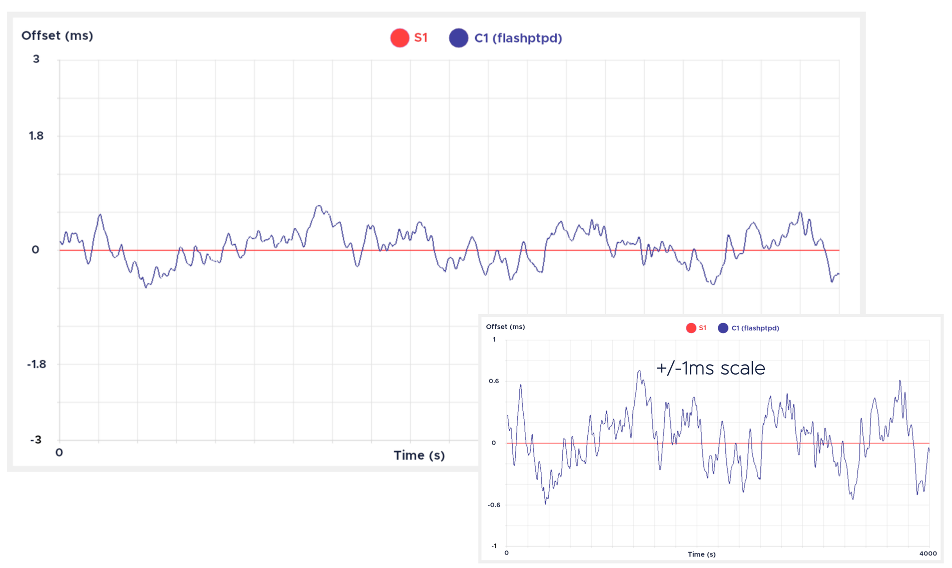




3) Routed Connection – 40 to 75ms Delay, up to 20ms Jitter

When emulating the properties of a WAN by adding delay and jitter, ptp4l kept the PHC roughly within -2 and +2.8ms, while flashptpd was able to stay well within **+/-1ms**.





MEINBERG

Thomas Behn
Senior Software Engineer
Phone: *+49 (0)5281 9039-438* ***Email:*** *thomas.behn@meinberg.de*

Meinberg Funkuhren GmbH & Co. KG
Lange Wand 9
31812 Bad Pyrmont
Germany