NTPv5 Timestamp Extension Fields

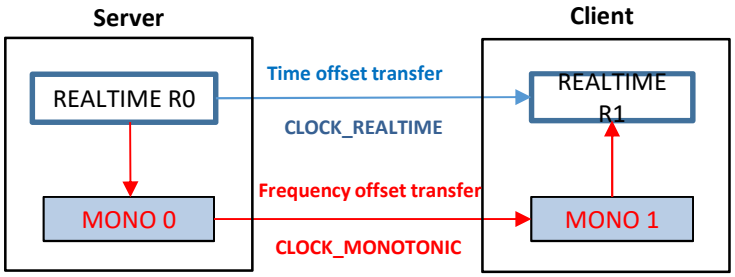
Siyu Tang, Guanhua Zhuang

draft-tang-ntp-ntpv5-extension-field-00

IETF 124, Montréal, 4th November 2025

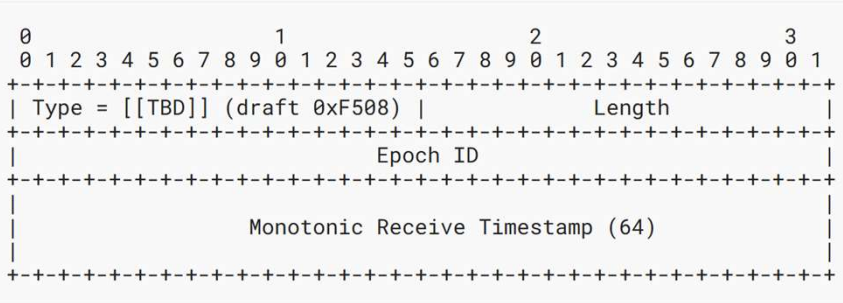
NTPV5: FREQUENCY TRANSFER AND MONOTONOIC TIMESTAMP EXTENSION FIELD

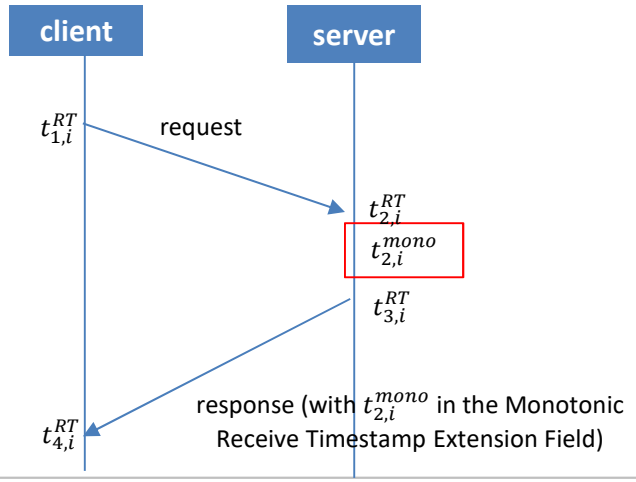
- **NTP Version 5 (draft-ietf-ntp-ntpv5-07) proposes to separate the time and frequency offset transfer with different clock sources**



Remove the error caused by intertwined clock and frequency adjustment with a single clock source.

- **A Monotonic Timestamp Extension Field is included**





NTPv5 explicitly does not define algorithms (e.g., clock control, filtering, security) in conjunction with the wire protocol

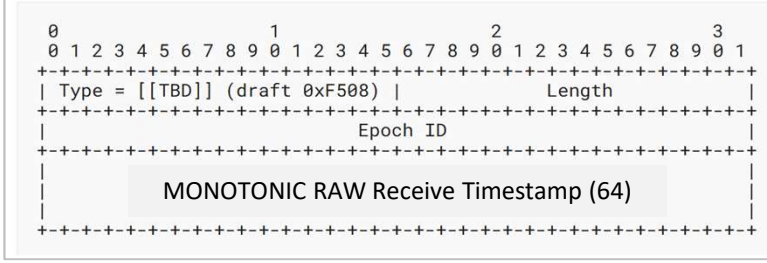
THE MONOTONIC RAW TIMESTAMP EXTENSION FIELDS

❑ Linux supports multiple clock sources

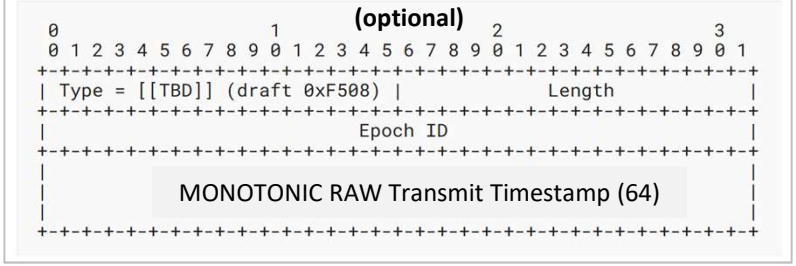
Linux ClockID	Impact of NTP adjustment on Frequency	Impact of NTP adjustment on Time
CLOCK_REALTIME	Y	Y
CLOCK_MONOTONIC	Y	N
CLOCK_MONOTONIC_RAW	N	N

- CLOCK_MONOTONIC is not affect by NTP adjustment if the clock is stepped.
- The frequency of CLOCK_MONOTONIC is changed if the clock is slewed.
- CLOCK_MONOTONIC_RAW is a clock that is not affected by STEP or SLEW, offering a more stable clock source
- Carrying timestamps from CLOCK_MONOTONIC_RAW enables simpler algorithmic implementation and enhances sync accuracy

MONOTONIC RAW Receive Timestamp



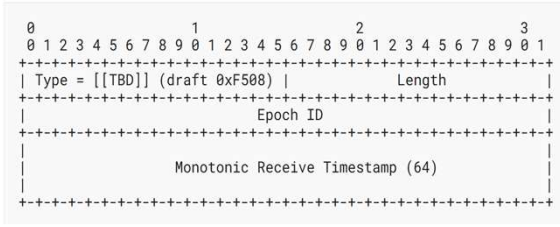
MONOTONIC RAW Transmit Timestamp (optional)



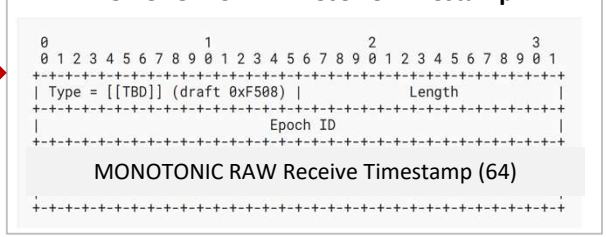
THE MONOTONIC RAW TIMESTAMP EXTENSION FIELDS

NTPv5 explicitly does not define algorithms (e.g., clock control, filtering, security) in conjunction with the wire protocol

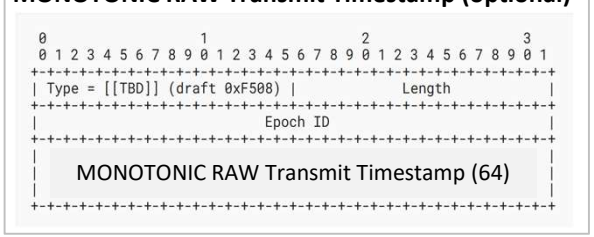
MONOTONIC Receive Timestamp



MONOTONIC RAW Receive Timestamp



MONOTONIC RAW Transmit Timestamp (optional)



Symmetric mode

Network asymmetry

❑ **Timestamp used for offset/time correction**

1) **Implementation 1**:
- Use the Monotonic receive timestamp to correct clock offset measured with CLOCK_REALTIME
- Estimate frequency offset from the corrected clock offsets

1) **Implementation 2**: use the Monotonic receive timestamp to correct the time keeping of CLOCK_MONOTONIC; estimate frequency offset from the corrected times

❑ **Timestamp used to directly estimate the frequency offset, without any correction**

1) **Implementation 1**: use the Monotonic RAW receive timestamp to estimate frequency offset (one direction)

2) **Implementation 2**: use the Monotonic RAW receive & Transmit timestamps to estimate frequency offset (bi-direction)

Simpler implementation, enhanced accuracy

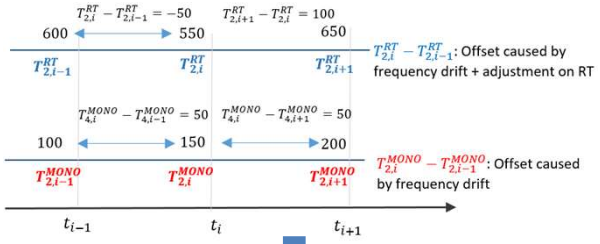
IMPLEMENTING FREQUENCY TRANSFER WITH MONOTONOIC TIMESTAMP EXTENSION FIELD

Implementation 1: use $t_{2,i}^{mono}$ to correct the offset measured by CLOCK_REALTIME, estimate frequency from the corrected clock offset

Step 1: Receive $t_{2,i}^{mono}$ from server

Step 2: Calculate offset caused by CLOCK_REALTIME adjustment

$$\delta_i^{RT} = T_{2,i}^{RT} - T_{2,i-1}^{RT} - \left(T_{2,i}^{MONO} - T_{2,i-1}^{MONO}\right)$$



Example:

$\delta_i^{RT} = -50 - 50 = -100$

$\delta_{i+1}^{RT} = 100 - 50 = 50$

Step 3: Correct CLOCK_REALTIME clock offset

- Clock offset computed with CLOCK_REALTIME timestamps

$$\theta_i^{RT} = \frac{(T_{2,i}^{RT} - T_{1,i}^{RT}) + (T_{3,i}^{RT} - T_{4,i}^{RT})}{2}$$

- Corrected clock offset by taking off the amount offset caused by CLOCK_REALTIME adjustment

$$\theta_{i,cor}^{RT} = \theta_i^{RT} - \delta_i^{RT}$$

Step 4: Correct RT clock offset if time is stepped or slewed locally

❑ If local clock is stepped by Δ at time t_n

This can be read locally, thus, easily included to the corrected clock offset

$$\theta_{i,cor}^{RT,loc} = \theta_{i,cor}^{RT} + \Delta$$

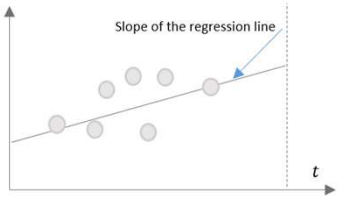
❑ If local clock is slewed at time t_n, let f_{old} and f_{new} be the frequency before and after the slew

Derive the corre $\theta_{i,cor}^{RT,loc}$ cted clock offset as

$$\theta_{i,cor}^{RT,loc} = \theta_{i,cor}^{RT} + \frac{f_{new} - f_{old}}{1 - f_{old}} \cdot (t - t_n)$$

Step 5: Derive the frequency offset from a number of $\theta_{i,cor}^{RT,loc}$ samples

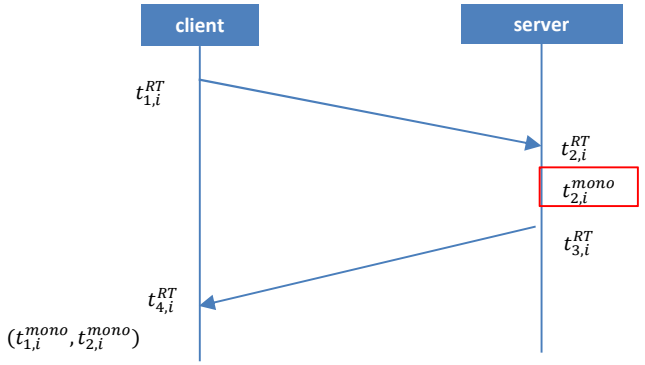
$\gamma \approx \frac{\theta_{i,correct}^{RT} - \theta_{i,correct}^{RT}}{t_{i+1} - t_i}$ Or with linear regression



Note: CLOCK_MONOTONIC moves at the same rate as the system clock. $T_{2,i}^{MONO}$ will be driven by different rate(s) if clock is slewed.

IMPLEMENTING FREQUENCY TRANSFER WITH MONOTONOIC TIMESTAMP EXTENSION FIELD

Implementation 2: correct the timestamps sent and received by CLOCK_MONOTONIC



CLOCK_REALTIME timestamps ($t_{1,i}^{RT}$, $t_{2,i}^{RT}$, $t_{3,i}^{RT}$, $t_{4,i}^{RT}$)

↓

Compute clock offset

$$\theta_i^{RT} = \frac{(T_{2,i}^{RT} - T_{1,i}^{RT}) + (T_{3,i}^{RT} - T_{4,i}^{RT})}{2}$$

↓

Adjust time
(STEP or SLEW)

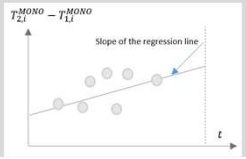
CLOCK_MONOTONIC timestamps ($t_{1,i}^{MONO}$, $t_{2,i}^{MONO}$)

↓

Correct ($t_{1,i}^{MONO}$, $t_{2,i}^{MONO}$) if time is SLEWED,
keep a list of virtual CLOCK_MONO times
E.g., $t_{1,i}^{MONO} = t_{1,i}^{MONO} + \frac{f_{new} - f_{old}}{1 - f_{old}} \cdot (t - t_n)$

↓

Estimate frequency offset, e.g., linear regression



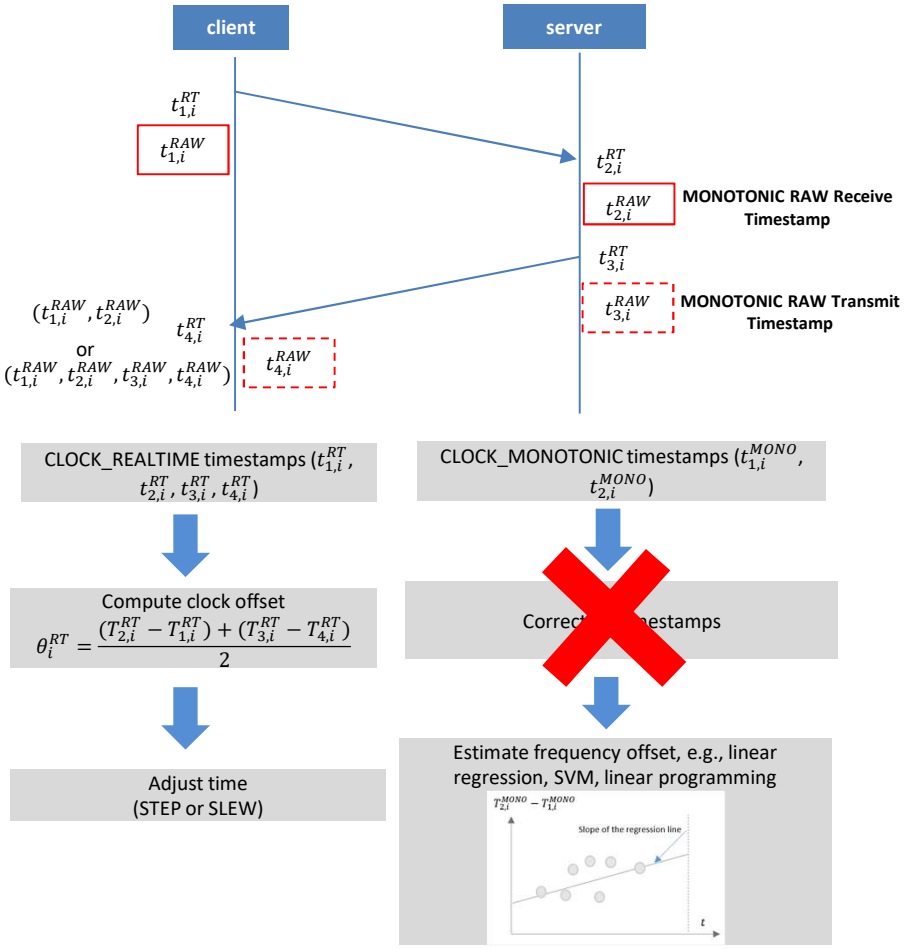
Linux ClockID	Impact of NTP adjustment on Frequency	Impact of NTP adjustment on Time
CLOCK_REALTIME	Y	Y
CLOCK_MONOTONIC	Y	N
CLOCK_MONOTONIC_RAW	N	N

- Use timestamp $t_{2,i}^{mono} - t_{1,i}^{mono}$ to estimate the frequency offset
  - No θ_i^{RT} (i.e., offset measured with CLOCK_REALTIME) is involved
- If the system clock is stepped, no impact on the monotonic timestamps, $t_{2,i}^{mono} - t_{1,i}^{mono}$ can be directly applied
- If the system clock is slewed, use the same principle in the first implementation, i.e., correct MONOTONIC timestamps with $\frac{f_{new} - f_{old}}{1 - f_{old}} \cdot (t - t_n)$

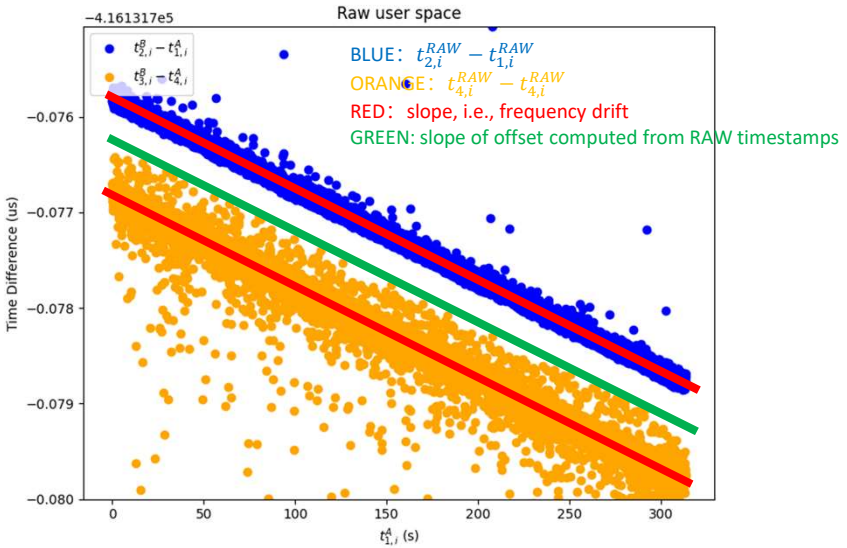
↓

This can be done in a simpler way, with **CLOCK_MONOTONIC_RAW** timestamps, no correction is needed after all.

THE MONOTONIC RAW RECEIVE TIMESTAMP EXTENSION FIELDS



❑ **An example with dataset obtained from a real production network**



❑ Various ways to compute the frequency offset

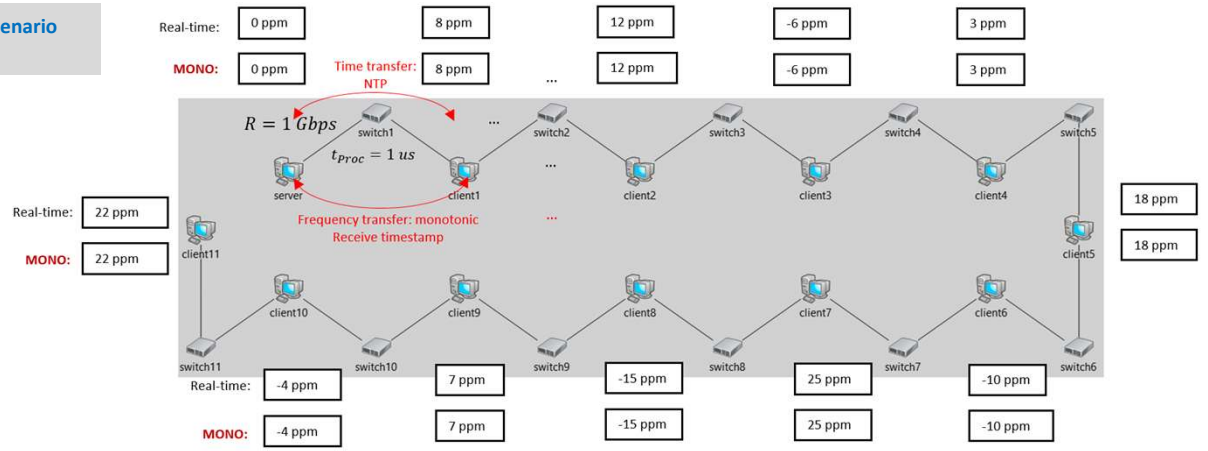
1) Linear regression directly on $(t_2 - t_1)$ and/or $(t_3 - t_4)$, take the average
2) Compute offset from the MONOTONIC RAW timestamps, estimate the frequency offset

❑ No need to maintain list(s) of corrected timestamps

IMPLEMENTING FREQUENCY TRANSFER WITH MONOTONOIC TIMESTAMP EXTENSION FIELD

Implementation 1: Omnest simulation, NTPv4 for realtime clock adjustment, monotonic timestamp extension field for frequency transfer

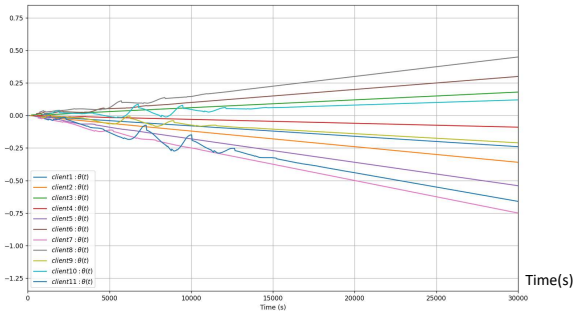
Noiseless Multi-hop Network Scenario (Omnest simulation)



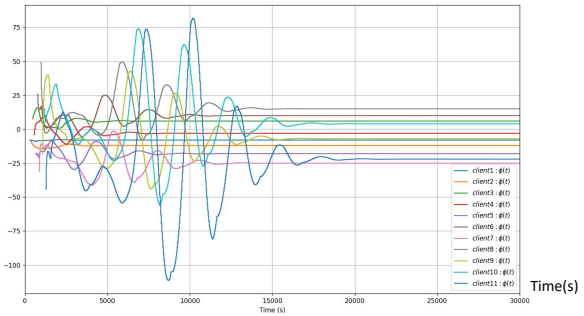
Simulation Time = 10000 s

Path Delay: Symmetric

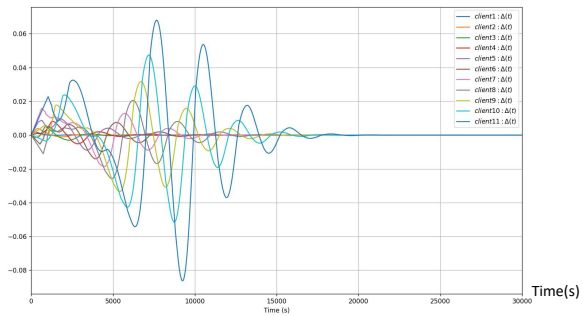
Corrected Offset (s)



Estimated Frequency(ppm) [winSize=50]



Time Error (s)

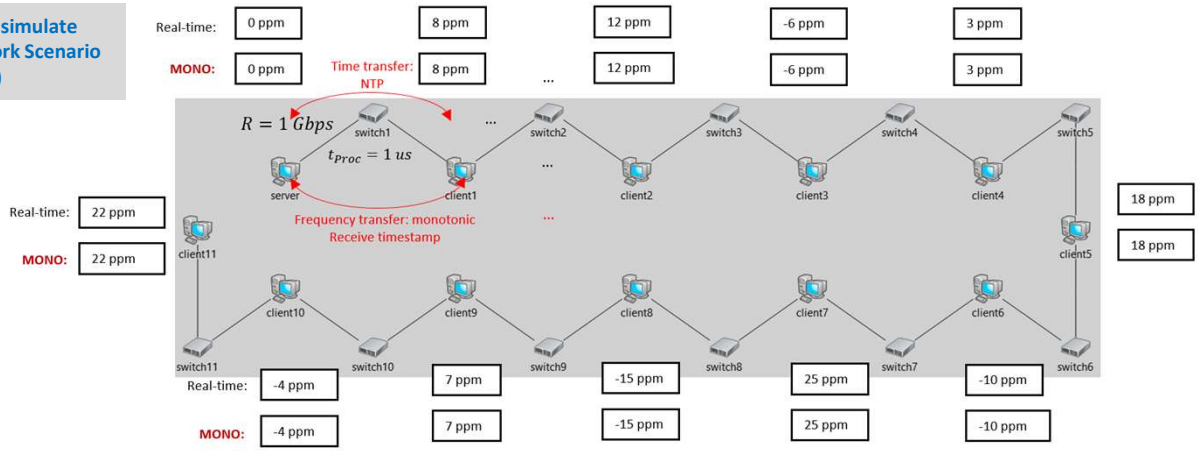


- If no error is introduced, implementation 1 is able to correct the offset and estimate the frequency offset correctly.

IMPLEMENTING FREQUENCY TRANSFER WITH MONOTONOIC TIMESTAMP EXTENSION FIELD

Implementation 1: Omnest simulation, NTPv4 for realtime clock adjustment, monotonic timestamp extension field for frequency transfer

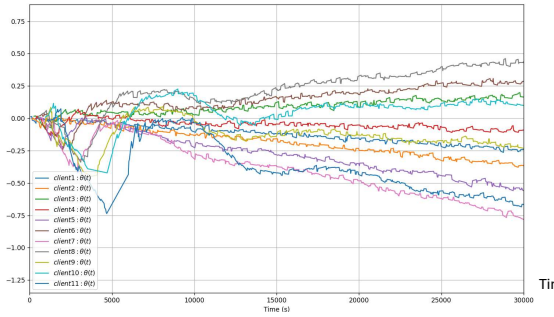
Gaussian noise (0, 100μs) to simulate network jitter Multi-hop Network Scenario (Omnest simulation)



Simulation Time = 10000 s

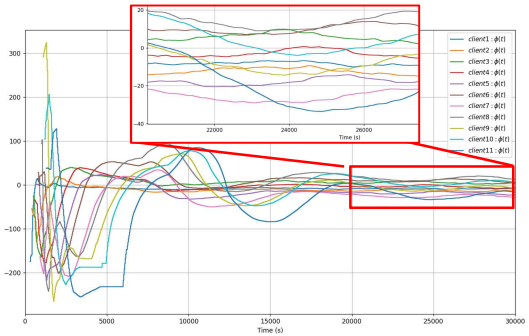
Path Delay: Symmetric

Corrected Offset (s)



Time(s)

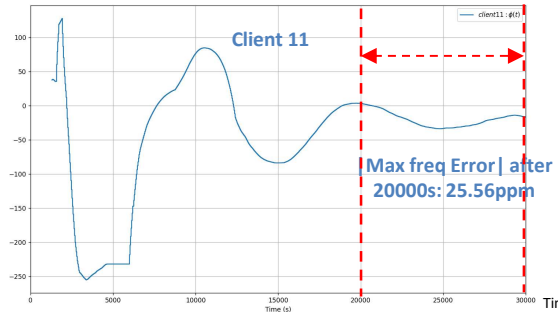
Estimated Frequency(ppm) [winSize=50]



Time(s)

Estimated Frequency(ppm) [winSize=50]

Client 11



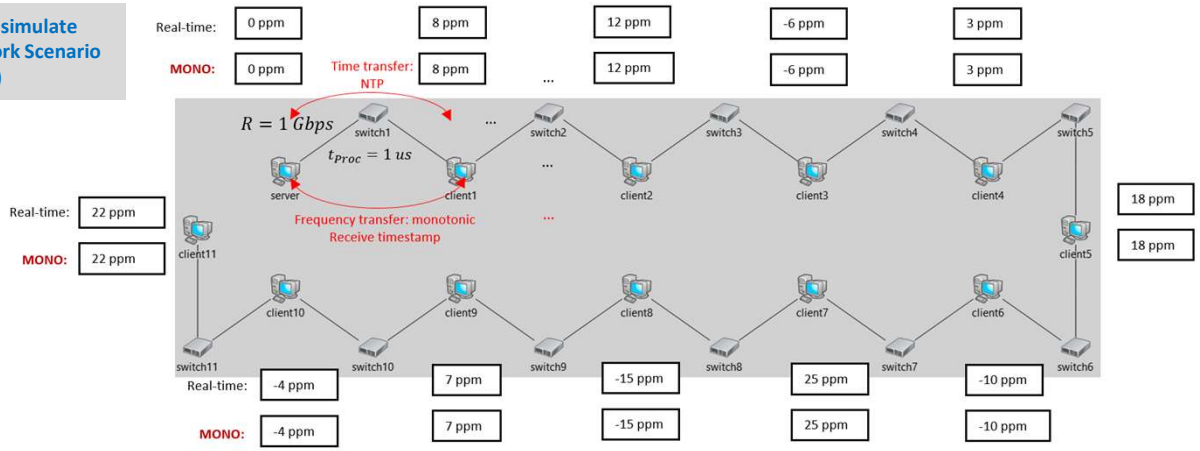
Time(s)

- With the introduction of Gaussian noise, it talks approx. 20000s for frequency estimation to converge; The maximum error after 20000s is 25.56ppm.

IMPLEMENTING FREQUENCY TRANSFER WITH MONOTONOIC RAW TIMESTAMP EXTENSION FIELDS

Implementation: estimate frequency offset from $(t_{1,i}^{RAW}, t_{2,i}^{RAW}, t_{3,i}^{RAW}, t_{4,i}^{RAW})$

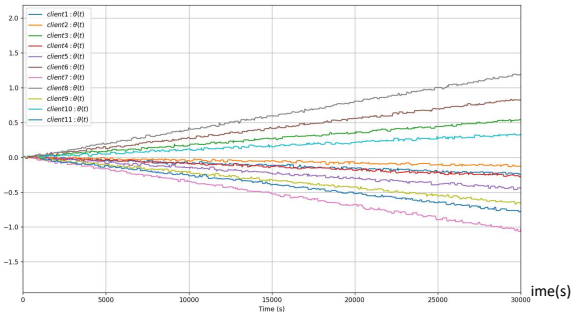
Gaussian noise (0, 100μs) to simulate network jitter Multi-hop Network Scenario (Omnest simulation)



Simulation Time = 10000 s

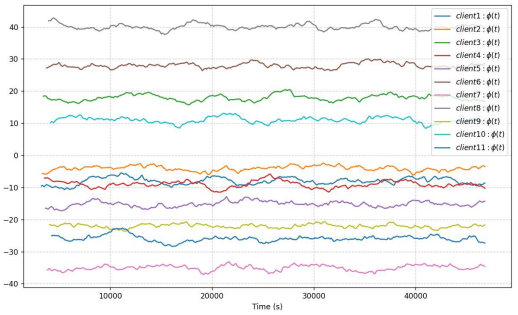
Path Delay: Symmetric

Corrected Offset (s)



ime(s)

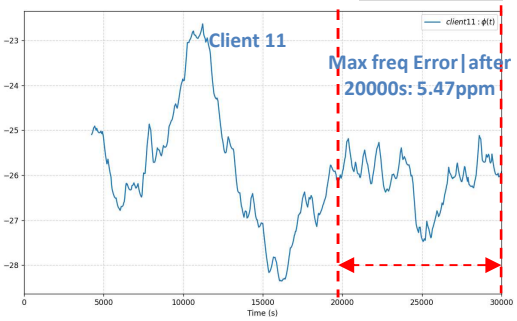
Estimated Frequency(ppm) [winSize=50]



Time(s)

Estimated Frequency(ppm) [winSize=50]

Client 11

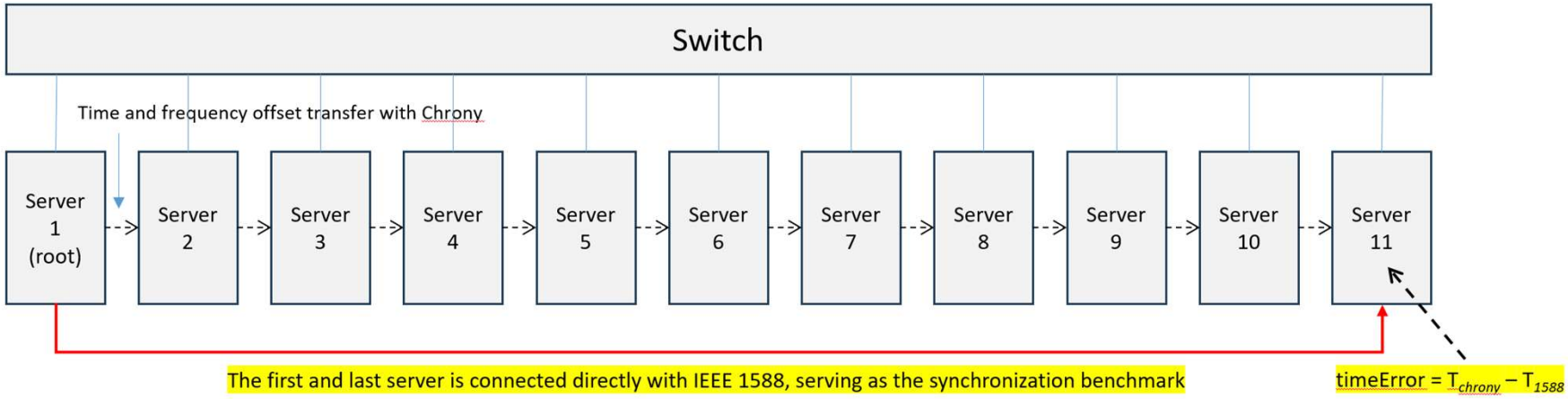


Time(s)

- With CLOCK_MONOTONIC_RAW timestamps, frequency offset can be estimated with an maximum error of 5.47ppm on client 11.

COMPARE FREQUENCY TRANSFER BETWEEN MONOTONOIC AND RAW TIMESTAMP EXTENSION FIELD

- **Experimental evaluation of Chrony 4.8 in a multi-hop testbed**



- 11 servers connecting with Chrony 4.8 clock synchronization scheme
- Poll interval 8 seconds
- Background traffic between each node pair to emulate network delay and jitter

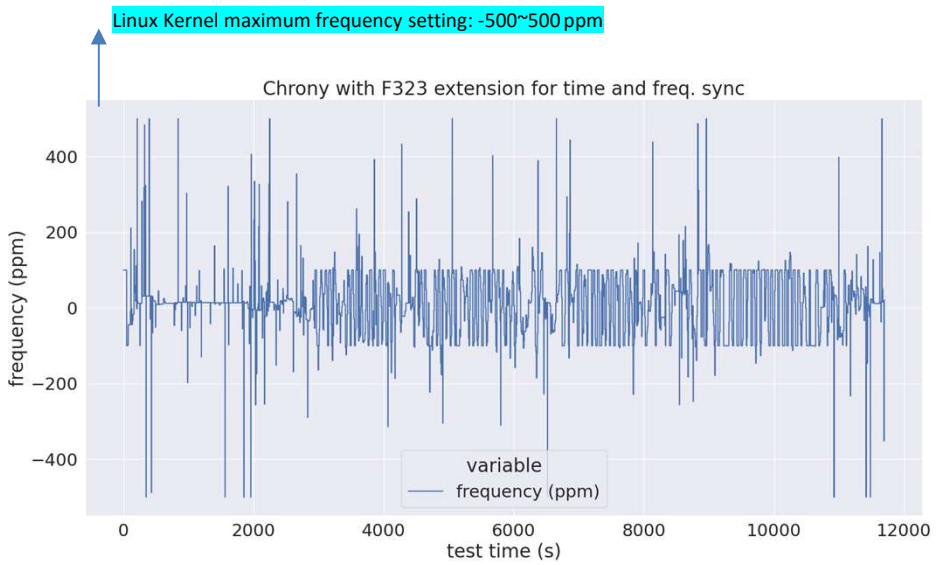
  while true; do
  background traffic@700Mbps
  duration of 60s
  randomly choose forward, backward or bi-direction
  sleep 10second
  Done

COMPARE FREQUENCY TRANSFER BETWEEN MONOTONOIC AND RAW TIMESTAMP EXTENSION FIELD

- **Experimental evaluation of Chrony 4.6 in a multi-hop testbed**

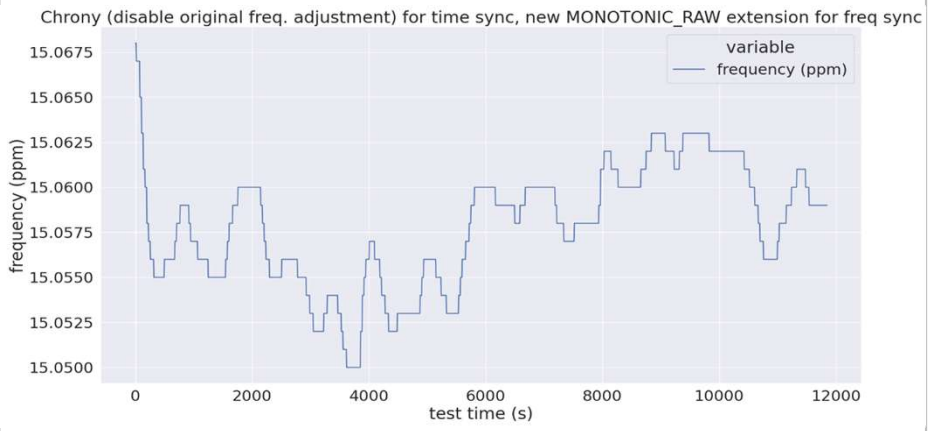
chrony.conf
server <PRE_HOP_IP> minpoll 3 maxpoll 3 extfield F323 xleave
maxdrift 100
makestep 0.001 -1
local stratum <PRE_HOP_STRATUM>
allow

Chrony: Frequency at server 11



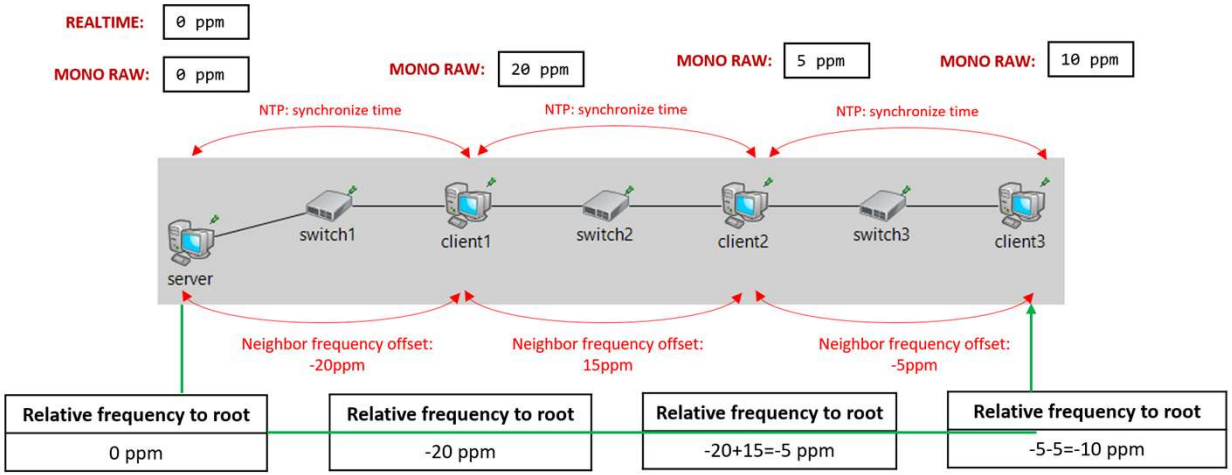
The frequency offset at the last server fails to converge.

MONOTONIC RAW: Frequency at server 11



Maximum error the frequency offset estimation: < 0.014 ppm

FREQUENCY TO THE ROOT SERVER EXTENSION FIELD



- In NTPv5, the accuracy of frequency transfer at a node i (with respect to the root) depends on if its previous node $i-1$ has been synchronized with the root
  - Once the frequency is adapted in previous node, collect new samples to estimated the current frequency offset between a node pair
- The convergence speed can be expedited if the protocol carries the frequency offset relative to the root

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|s|              Frequency To The Root Server (31)              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- **Sign bit (1-bit)**: a binary number indicates if the frequency of a client is faster (1) or slower (0) relative to the root.
- The absolute **frequency-transfer offset relative to the primary server** (i.e., root) is carried by the remaining 31-bit.

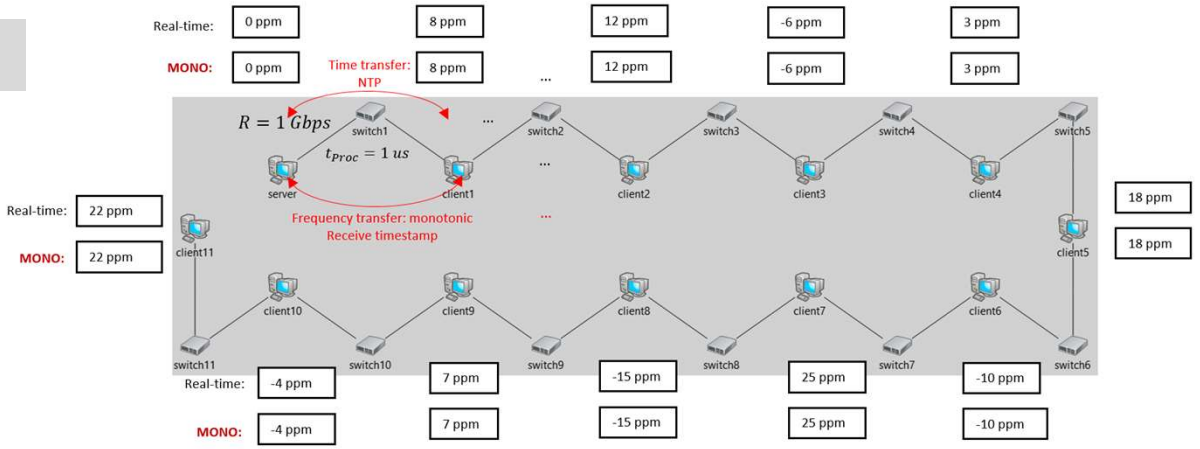
$$\boldsymbol{freq2root_{i+1} = freq2root_i + freq2neighbor_{i+1}} \text{ for } i \geq 1$$

where index of the root is $i = 0$. $freq2neighbor_{i+1}$ is measured with MONO RAW timestamps

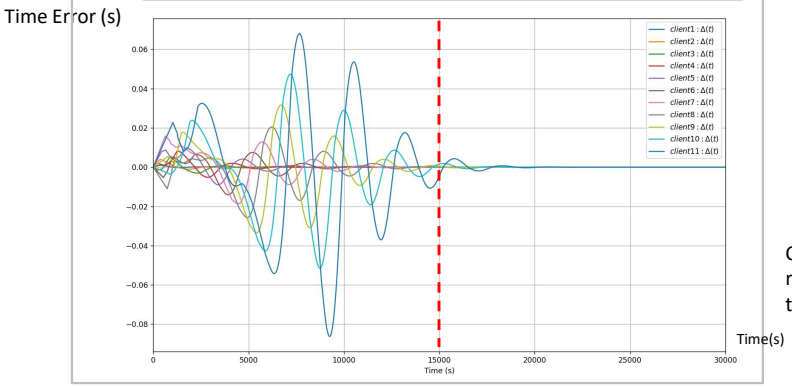
FREQUENCY TO THE ROOT SERVER EXTENSION FIELD

Performance comparison in a multi-hop network with and without Frequency To The Root Server Extension Field

Noiseless Multi-hop Network Scenario

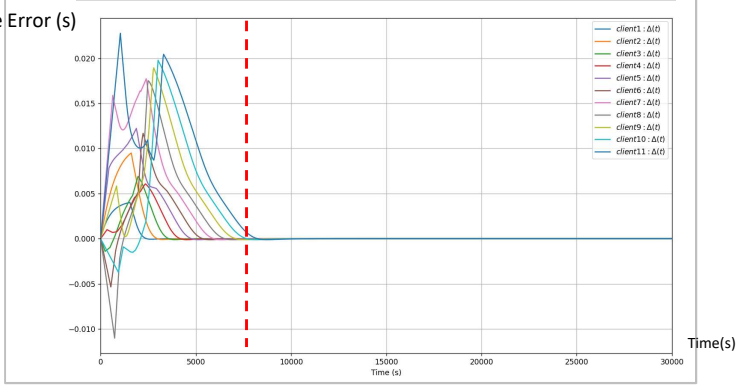


Without Frequency To The Root Server Extension Field



Convergence speed reduces to half at the last client.

With Frequency To The Root Server Extension Field



QEUSTIONS?