SEIKO

コンピューター・システムの時刻とネットワーク時刻同期

～うるう秒はOS内部でどのように扱われるか～

セイコーソリューションズ株式会社
倉田　陽介
岩本　哲也

目次

SEIKO

第1部

コンピュータークロックとうるう秒

コンピューターの時刻って? (1)

SEIKO

問:

そもそもコンピューターにおける時間って何でしょう?

えっ! そこから??

でも、必要な抽象化なんです! (POSIX.1 A.4.16, ITU-T G.810, IEEE1588-2008 etc.)

答:

連続的に真に単調増加する値であって、

ジャンプしたり、逆戻りしないもの。

SEIKO

Epochからの経過時間で計る

- Epoch
  - ✓ 経過時間を計る**起点**。
  - ✓ UNIXでは**1970年1月1日 0:00:00 (UTC)**
- 経過時間
  - ✓ Epochからの経過時間は**SI秒**で計り取る。
- SI秒
  - ✓ セシウム原子のある変化時間を元に決めた**物理量**

だから、コンピューター(OS)は本当は暦を認識していない

問：　暦ってなんでしょう？

答：　我々が2019年1月24日ってな感じで普段使ってるヤツ

問：　我々が普段使ってる暦って？

答：　**グレゴリオ暦**

コンピューターの時刻って？(4)

SEIKO

でも、でも、

```
kurata@ubuntu1604:~$ date
2019年　1月 24日 木曜日 14:40:35 JST
```

って、ほらちゃんと暦 認識してるよ！

答：

OSの時間を暦（カレンダー時間）に変換して見せてくれる賢い奴(**tz database**)のおかげ。

コンピューターの時刻って? (5)

SEIKO

OSが認識している時間は、

```
kurata@ubuntu1604:~$ date +%s
1548308435
```

というわけで、

2019年 1月 24日 木曜日 14:40:35 JST

＝ **1548308435(秒)**

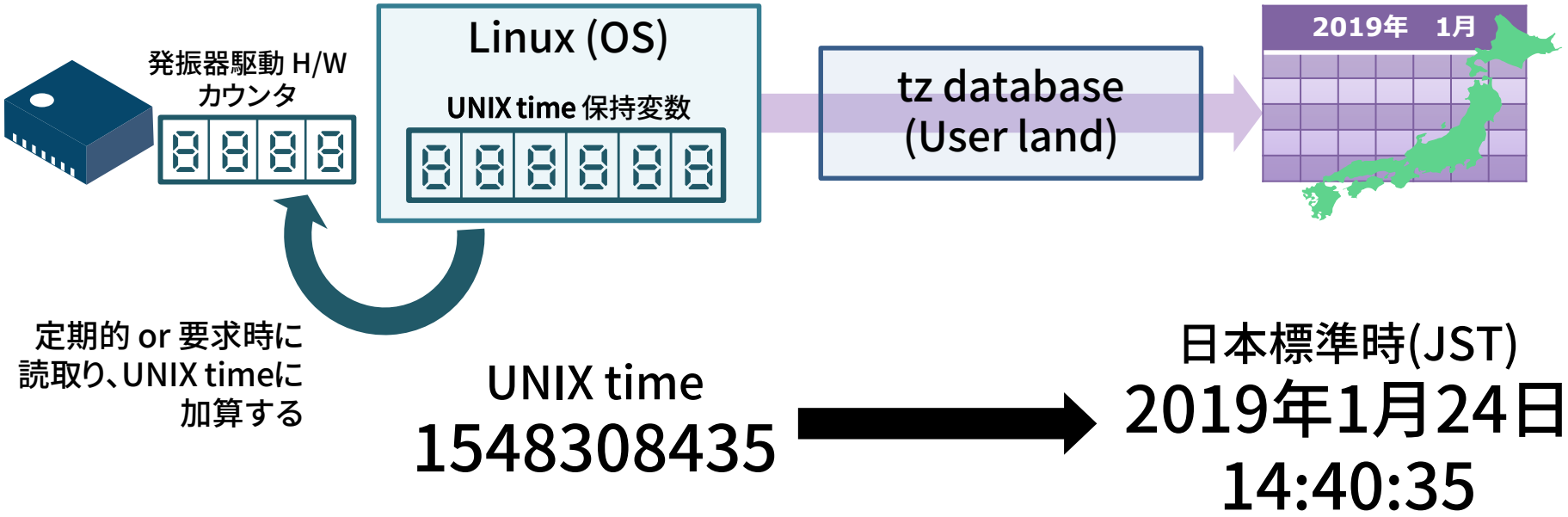
(1970 年1月1日 0:00:00(UTC) から 1548308435秒 経過したところである)

これを**UNIX time(POSIX time)**と呼ぶ

Epochからの経過時間が基本!

TZ Database (1)

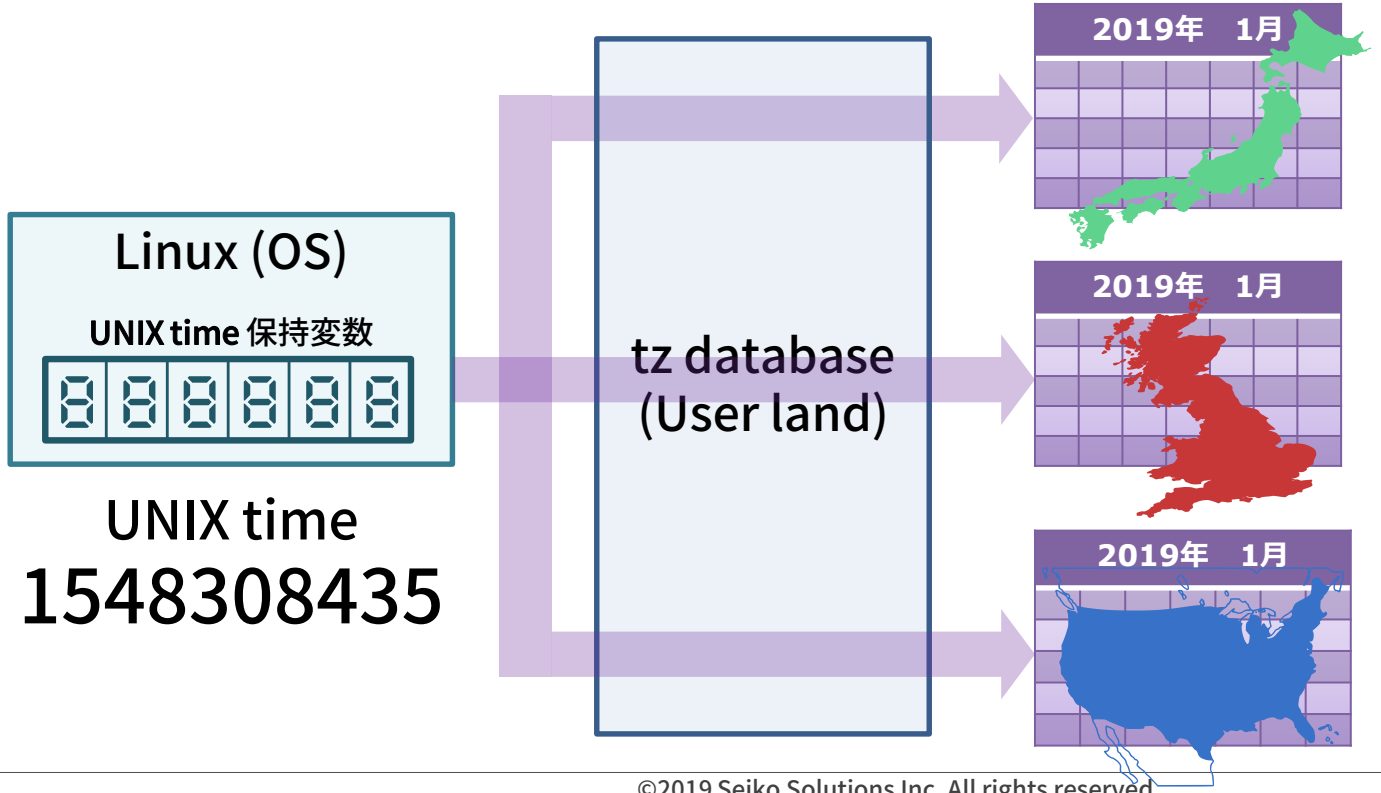
- 多くのUNIX系OSの暦解釈は**tz database**による
- UNIX time を **カレンダー時間**に変換する



TZ Database (2)

SEIKO

- 各地域の標準時に解釈する(これがtz databaseの本来の仕事)



TZ Database (3)

SEIKO

- 夏時間（サマータイム）を正しく扱える

UNIX time	東部標準時
1520751598	2018年2月11日(Sun) 01:59:58 (EST)
1520751599	2018年2月11日(Sun) 01:59:59 (EST)
1520751600	2018年2月11日(Sun) 03:00:00 (EDT)
1520751601	2018年2月11日(Sun) 03:00:01 (EDT)
:	:

:	:
1541311198	2018年11月4日(Sun) 01:59:58 (EDT)
1541311199	2018年11月4日(Sun) 01:59:59 (EDT)
1541311200	2018年11月4日(Sun) 01:00:00 (EST)
1541311201	2018年11月4日(Sun) 01:00:01 (EST)
:	:

重要：UNIX timeにはジャンプがない！（連続している）

うるう秒とUNIX time(1)

SEIKO

問：

うるう秒もサマータイムと同じじゃないんですか？

答：

うるう秒は基本的にはtz databaseでは扱わない

うるう秒はOSが直接取り扱う

UNIX timeの連続性に影響がある

うるう秒とUNIX time(2)

SEIKO

問：

1972年7月1日 0:00:00 (UTC)

はUNIX timeでいくらでしょうか？

答：(計算で求まる) **78796800**

- ✓ 1970年はうるう年でない ⇒ 31536000秒
  (86400秒×365日)
- ✓ 1971年はうるう年でない ⇒ 31536000秒
- ✓ 年始から1972年7月1日まで ⇒ 15724800秒
  (86400秒×(31日×3+30日×2+29日))
- ✓ 31536000×2+15724800＝**78796800**

うるう秒とUNIX time(3)

SEIKO

1972年6月30日 24時は世界初のうるう秒（＋1秒）実施日

UNIX time	UTC
78796798	1972年6月30日 23:59:58
78796799	1972年6月30日 23:59:59
????	1972年6月30日 23:59:60
78796800	1972年7月1日 00:00:00
78796801	1972年7月1日 00:00:01

「????」に“適切な”整数がない

A) 78796799 を**もう1回やる**
B) 事前に値の増加速度（歩度）を変えて**ごまかす**

のどちらかしか（現実的な）方法がない
特に A) は**時間の連続性が崩れる**

SEIKO

うるう秒のキーポイント：

- UTCに合わせて生活している上では**避けられない**
- 実施日が不定期
  - ✓ 将来の実施日が不明
  - ✓ UNIX timeの計算から除外されている
- コンピューター時間の連続性定義に反する
  - ✓ 不連続が時間差の計算を直撃する
  - ✓ 未対策のコンピュータープログラムは影響を受ける

うるう秒の事前テストが重要です！

第2部

ネットワーク時刻同期

UTCの取得と設定

SEIKO

時刻(UTC)をOSに設定する方法

- 手で設定する
  - ✓ dateコマンド
  - ✓ 誤差が大きい
- 時刻源(GNSS、標準電波、テレホンJJY、etc)から直接情報をもらう
  - ✓ 専用の装置がないと無理
- IP上のネットワークプロトコルでもらう
  - ✓ ほとんどの装置で可能
  - ✓ 誤差も小さい

時刻同期プロトコル・概要

SEIKO

- Network Time Protocol (NTP)
  - ✓ 皆さんご存知 **RFC 5905**
- Precision Time Protocol (PTP)
  - ✓ 適用する業界ごとにプロトコル(Profile)が異なる
  - ✓ 最も標準的な規格が **IEEE Std. 1588-2008**

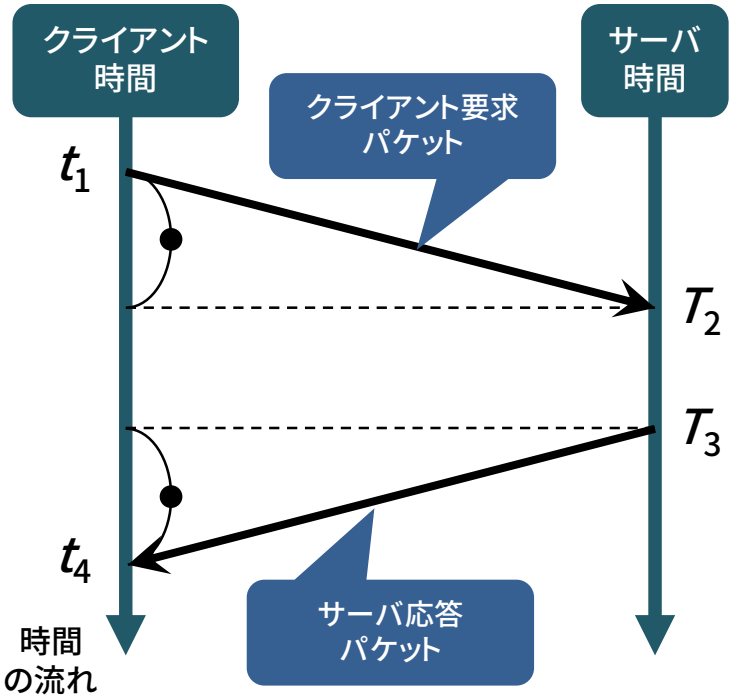
問：　ネットワーク的に遠いと同期誤差は大きくなる？

答：　理論的には距離は**誤差に影響しない**

時刻同期プロトコル・原理（T_1-T_4方式）

SEIKO

NTPもPTPも考え方は同じ（T_1-T_4方式）
サーバとクライアントで**時刻が違っても**伝送遅延時間は分かる



往復遅延時間

$$= (T_2 - t_1) + (t_4 - T_3)$$
$$= (t_4 - t_1) - (T_3 - T_2)$$

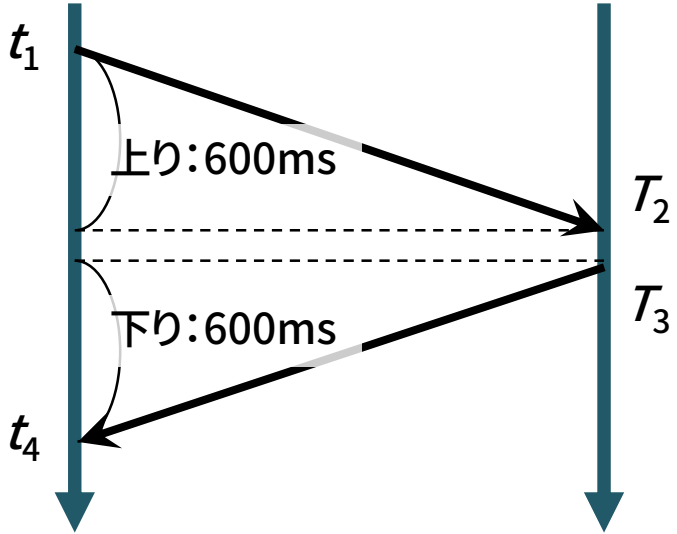
片方向遅延時間推定

$$= \frac{\text{往復遅延時間}}{2}$$
$$= \frac{(t_4 - t_1) - (T_3 - T_2)}{2}$$

$T_4 = T_3 +$ 片方向遅延時間推定
が t_4 の（あるべき）正しい時間である

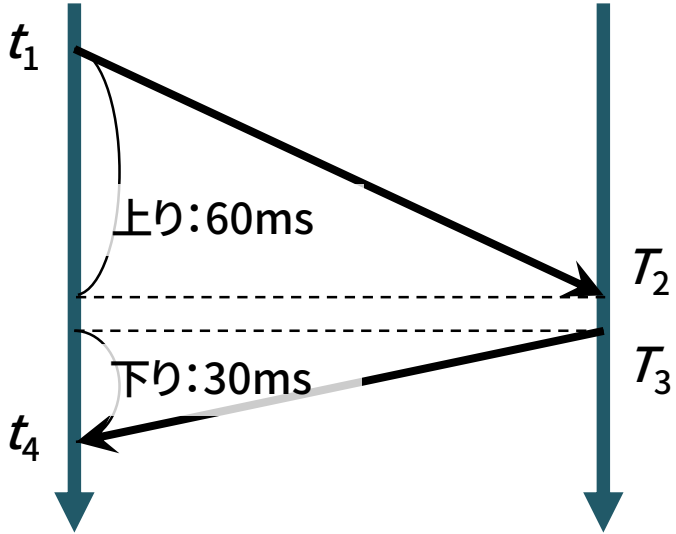
T_1-T_4方式の問題点(1)

伝送距離は問題にならないが、上り・下りの伝送遅延時間の非対称性には弱い。



片方向遅延推定=600ms
実際の下り時間と一致

遠くとも正しく同期



片方向遅延推定=45ms
実際の下り時間と15msもの誤差

近いのに15msもずれる

T_1- T_4方式の問題点(2)

SEIKO

上り下りの伝送遅延時間差を**小さくすれば**高精度に同期する

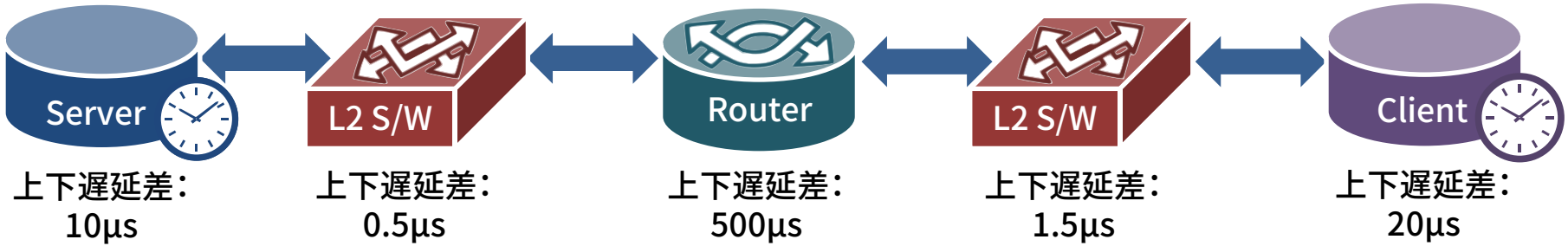
非対称の原因は？

- 装置内部
  - ✓ タイムスタンプの打刻位置ずれ
  - ✓ 中継装置滞留時間の非対称性
- 物理リンク
  - ✓ 上りと下りのケーブル長の違い
  - ✓ 同一光ファイバー内の波長の非対称性
- 通信経路
  - ✓ そもそもパケットの通過経路が上り下りで異なる

T_1- T_4方式の問題点(3)

問：　そうはいっても非対称って無視できるほど小さくないですか？

答：　それは必要とされる精度によります…



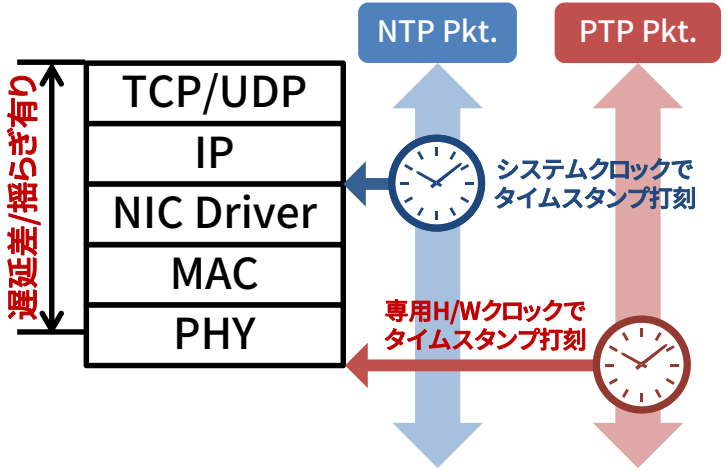
伝送遅延差合計：532μs ⇒ **推定誤差：266μs**

この266μsを大きいと見るか小さいと見るか？

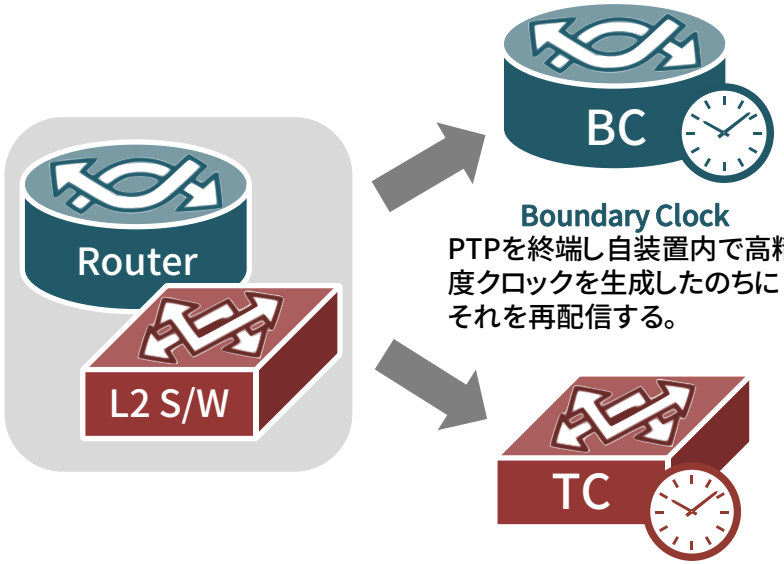
PTPはNTPに比べてなぜ高精度なのか？

PTPは伝送遅延の誤差要因を可能な限り排除する仕組みになっている。

対策(1)：装置内部の遅延・揺らぎ対策



対策(2)：中継装置の遅延・揺らぎ対策



Boundary Clock
PTPを終端し自装置内で高精度クロックを生成したのちにそれを再配信する。

Transparent Clock
PTPパケットの装置滞留時間を当該パケットに書込み転送する。

これらの仕組みを全く利用しないと、NTPと同程度の精度
専用装置がなくとも精度向上の工夫ポイントはありそうですね…

その他の高精度化のための工夫

NTPには**インターネット上**での時刻同期精度を高めるための種々の工夫が盛り込まれている

- Clock Filter Algorithm
- Select Algorithm
- Cluster Algorithm
- Combine Algorithm

PTPにはこういった工夫は**規定されていない**。
（各機器メーカーの独自実装に委ねられている）

SEIKO

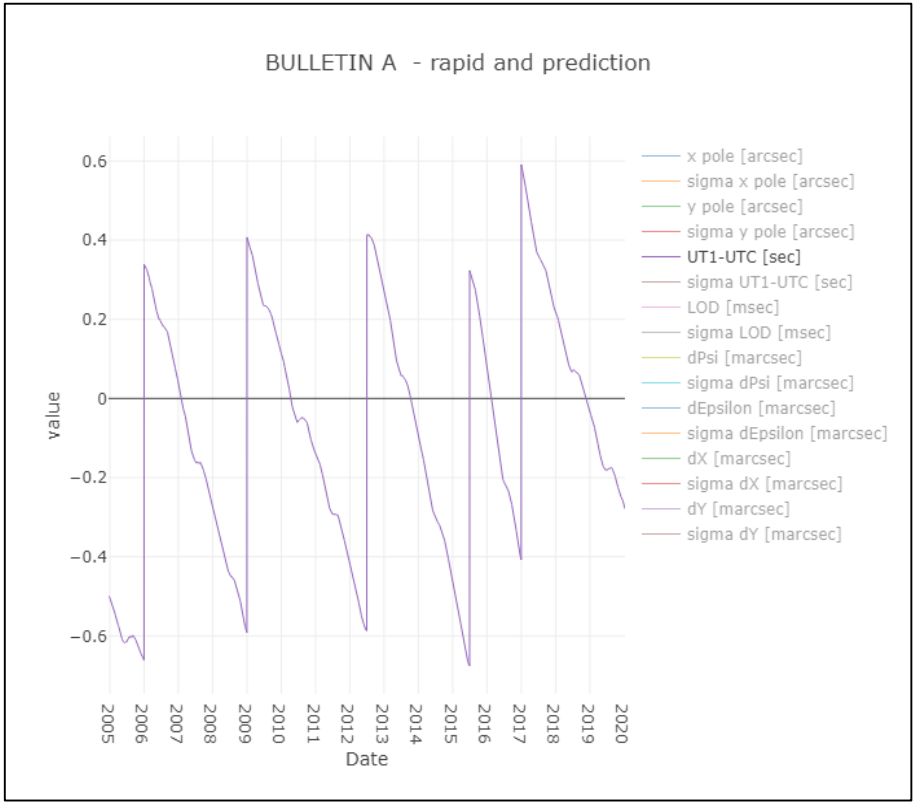
NTPとPTPプロトコルのタイムスケール

- NTP
  - ✓ UTC時系
  - ✓ Epoch: 1900年1月1日 0:00:00(UTC)からの経過時間
  - ✓ うるう秒処理がプロトコル上も必要
- PTP
  - ✓ TAI（国際原子時）時系
  - ✓ Epoch: 1970年1月1日 0:00:00(UTC)からの経過時間
  - ✓ うるう秒がない
    （うるう秒情報なしではカレンダー時間が分からない）
  - ✓ UTCとの時差情報を管理し配信する

第3部

うるう秒対応 プラクティス

第28回うるう秒実施予測



IERS, "Earth orientation data",
https://www.iers.org/IERS/EN/DataProducts/EarthOrientationData/eop.html

2019年7月
未実施が決定済み

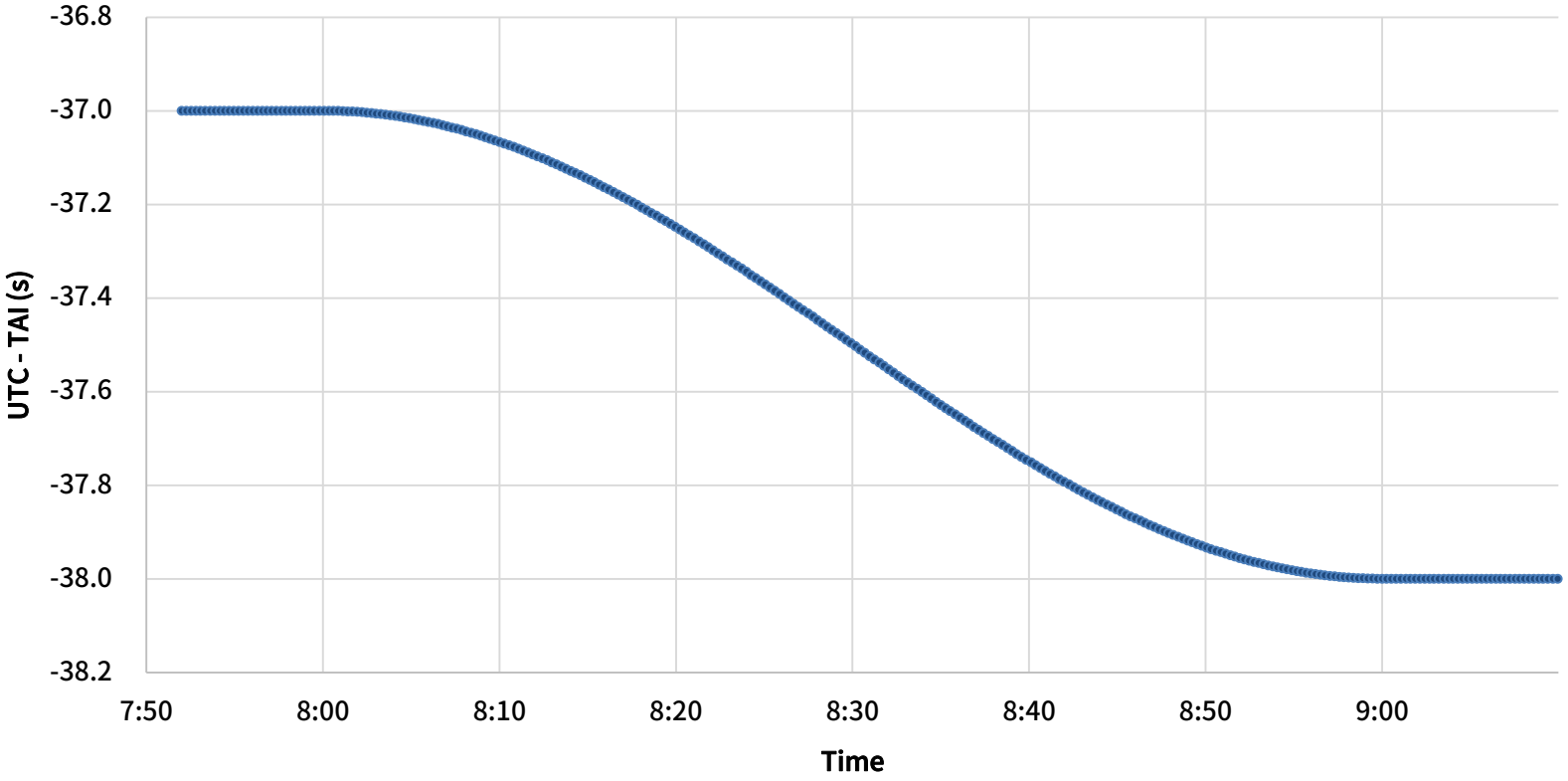
2020年1月
IERS予測では差異約-0.3秒
実施可能性あり

2020年7月
前回実施時と同等差異と予想
実施最有力!?

うるう秒テスト(1)

SEIKO

OSS機能の利用で「うるう秒テスト」が可能です



うるう秒テスト(2)

SEIKO

- ✓ NTP Project サーバソフト
  - ➢ Undisciplined Local Clock
    - ✓ システムクロックをそのまま時刻配信するReference Clock
  - ➢ Leap Smear
    - ✓ うるう秒実施時に配信時刻を漸次調整する機能
      - --enable-leap-smear指定でconfigureで有効化
- ✓ Leap Seconds List
  - ➢ うるう秒実施日とUTCオフセットが列挙されたファイル
    - ✓ テストしたい実施日/UTCオフセットを追加

うるう秒テスト(3)

SEIKO

Leap Seconds List

```
#$            3676924800
#@            3802291200
2272060800    10            # 1 Jan 1972
2287785600    11            # 1 Jul 1972
2335219200    13            # 1 Jan 1974
2366755200    14            # 1 Jan 1975
2398291200    15            # 1 Jan 1976
2429913600    16            # 1 Jan 1977
2461449600    17            # 1 Jan 1978
2492985600    18            # 1 Jan 1979
2524521600    19            # 1 Jan 1980
2571782400    20            # 1 Jul 1981
2603318400    21            # 1 Jul 1982
2634854400    22            # 1 Jul 1983
2698012800    23            # 1 Jul 1985
2776982400    24            # 1 Jan 1988
...
3550089600    35            # 1 Jul 2012
3644697600    36            # 1 Jul 2015
3692217600    37            # 1 Jan 2017
3786825600    38            # 1 Jan 2020
#h            a602db50 ea9d0c66 28242d25 bf45adc2 b0ac5eb5
```

ntp.conf

```
server 127.127.1.0 minpoll 4 maxpoll 4   # Local Clock指定
fudge 127.127.1.0 stratum 5
leapfile /tmp/leap-seconds.list  # Leap Seconds Listファイル
leapsmearinterval 3600           # うるう秒調整期間(秒)
...
```

SEIKO

セイコーソリューションズ株式会社