ネットワークコンピューティング
第11回

中山　雅哉 (m.nakayama@m.cnl.t.u-tokyo.ac.jp)
関谷　勇司 (sekiya@nc.u-tokyo.ac.jp)

時刻

計算機で用いられる時刻について

- リアルタイムクロック（RTC）
- システム時刻（system time）

← 水晶発振器の精度に伴い誤差が生じる

各計算機はバラバラな時刻を刻むことになる

- （ある程度）正確な時刻を刻むようにするためには
- 標準時刻に同期することが必要

リアルタイムクロック

- リアルタイムクロック（RTC）は、現在時刻を刻み続けるコンピュータの時計のこと
  - RTC は、リチウムバッテリーなどの別電源により、コンピュータの主電源が切れていたり使用できない時でも時刻を刻みつづけることができる
  - ほとんどの RTC は、クォーツ時計や腕時計で用いられているのと同じ、32,768kHz の周波数を持つ水晶発信器から分周して 1Hz を作っていることが多い。
- システム時刻（system time）は、カーネルが boot 時に、RTC に同期して生成される。

システム時刻

- システム時刻とは、システム・クロック（clock）によって生成される一定のテンポ（tick）を用いて算出される時刻のこと
  - 代表的な実装方法としては、”epoch” と呼ばれるある開始日を起点として “ticks” を単純積算して生成される。
  - Epoch:
    - UNIX and POSIX-compliant:　　1 Jan. 1970 00:00:00 UT
    - UNIX and POSIX-compliant 32bits OS では、“19 Jan. 2038 03:14:07 UT” までしか表現できない。
  - time(), clock_gettime() および gettimeofday() 関数を用いてシステム時刻を得ることができる（UNIX, POSIX）

time()

#include <time.h>

time_t time(time_t *tloc);

- tloc: the value of time in seconds since midnight, Jan. 1, 1970.
- If this function fails, it returns (time_t) -1.

char * ctime(const time_t *clock);

- ctime() function transforms binary date and time values to string.

clock_gettime()

#include <time.h>

int clock_gettime(clockid_t clock_id, struct timespec *tp);

- clock_id: CLOCK_REALTIME, CLOCK_MONOTONIC, …
- tp: value of time

```
struct timespec {
        time_t  tv_sec;         /* seconds */
        long    tv_nsec;        /* and nanoseconds */
};
```

gettimeofday()

#include <sys/time.h>

int gettimeofday(struct timeval *tp, struct timezone *tzp);

- tp: the time expressed in secondes and microseconds since midnight, Jan. 1, 1970.
- tzp: timezone

```
struct timeval {
        time_t          tv_sec;         /* seconds since Jan. 1, 1970 */
        suseconds_t     tv_usec;        /* and microseconds */
};
struct timezone {
        int     tz_minutewest;  /* of Greenwich */
        int     tz_dsttime;     /* type of dst correction to apply */
};
```

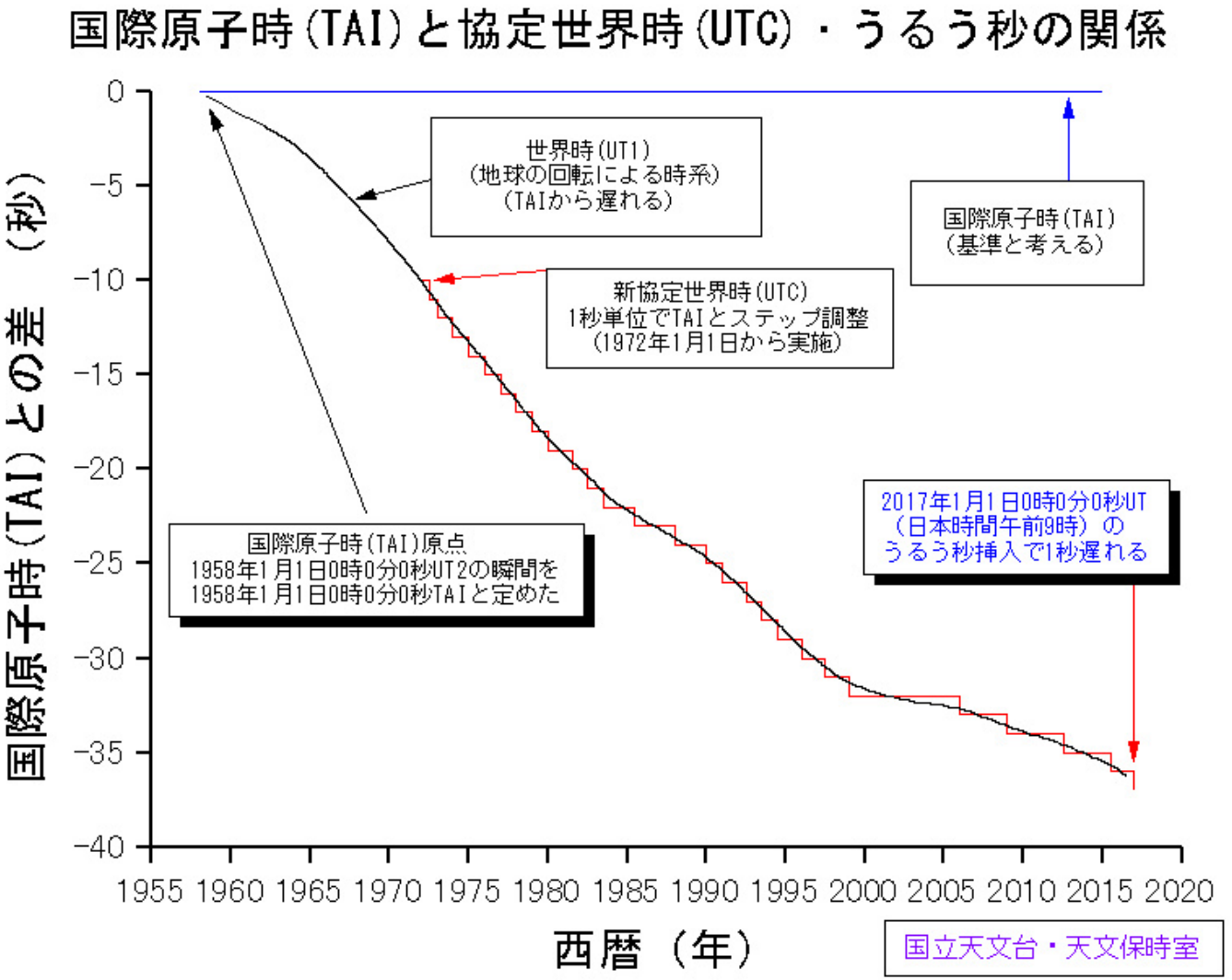
「１日」や「１秒」の決め方

- 古くは、地球の自転を基準にして「１日」の長さが決められ、その24分の1を1時間、さらに60分の1を1分、その60分の1を1秒としていた。（地球の自転に基づく時系：世界時 UT1）
- その後、（時間を測定する技術が進歩して）原子時計で時刻が測定できるようになり、原子時計によって決まる時刻が用いられる様になってきた。（原子の放射に基づく時系：原子時 TAI）
- この他にも、太陽系天体の運動に基づく地球重心における時系：（力学時 TCG）などがある。

「うるう秒」とは？

- 地球の自転が遅い状態が続いたり、自転の速い状態が続いたりして、地球の自転によって決まる時刻（UT1）と原子時計によって決まる時刻（TAI）のずれが大きくなった時、両者の時刻のずれを修正するために行う調整のこと
- うるう秒の調整は、12月か6月の末日の最後の秒(世界時)で行われ、地球の自転が遅い場合は、59分59秒のあとに 59分60秒を 1秒挿入し、地球の自転が速い場合は、59分58秒の次を 0分0秒として 1秒削除することになる。将来のうるう秒の実施時期は、地球の自転速度の長期変化が予測できないため、（予め）知ることができない。

TAI と UTC の関係



http://www.miz.nao.ac.jp/vlbi/leapsec.html

計算機の時刻同期

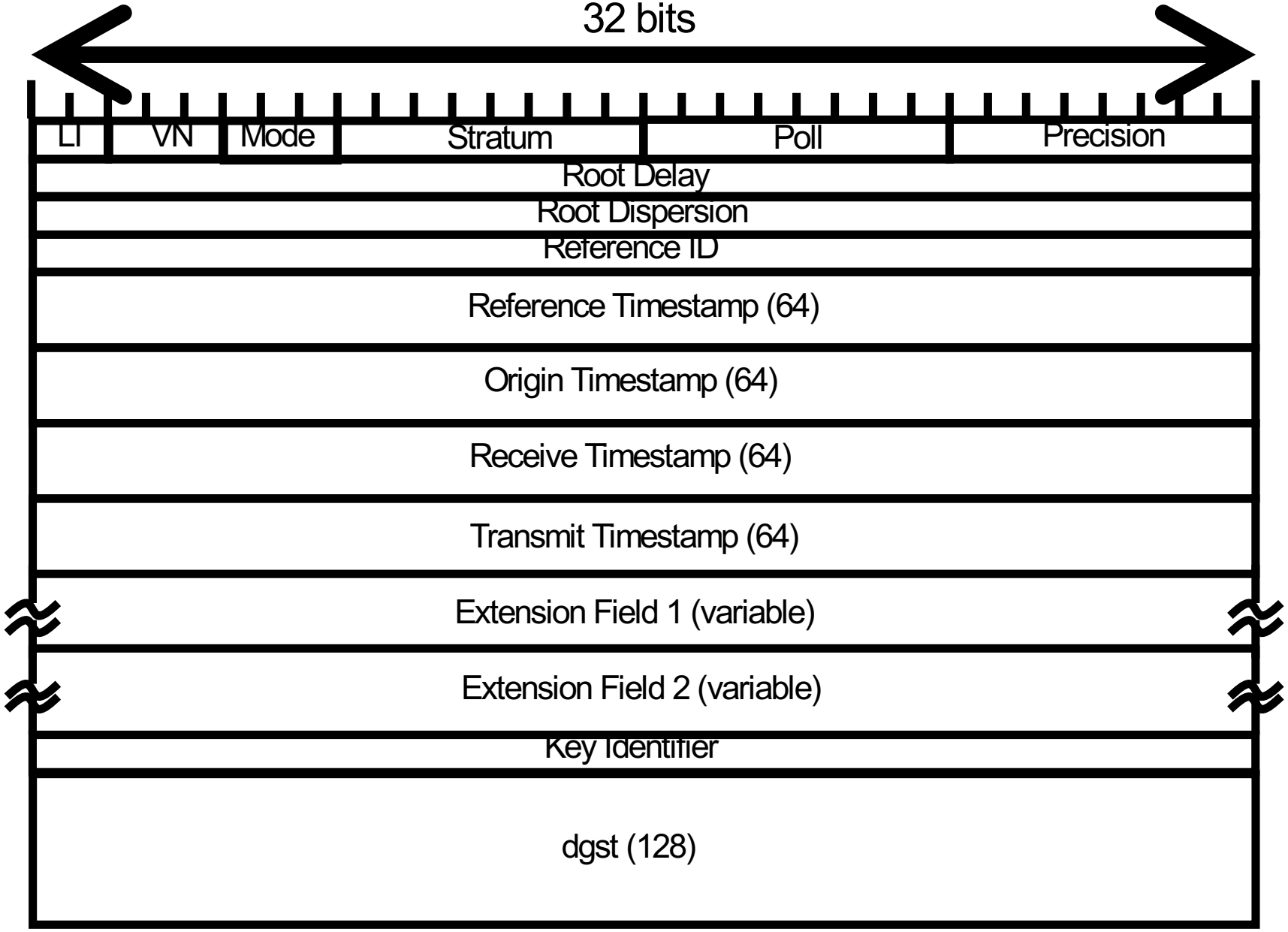
- 各計算機は OS 起動時に RTC を用いて時刻を定め、システム時刻を自律的に刻んでいるが、必ず誤差が含まれることになる。また「うるう秒」の調整を行う必要がある。
- （何か）基準となる時刻源があれば、それに同期することで計算機の時刻を自動調整することができる。

- 同期時刻源
  - 原子時計
  - GPS, WWV, WWVH, WWVB
  - JJY, テレホンJJY, CDMA携帯基地局, FM

NTP (Network Time Protocol)

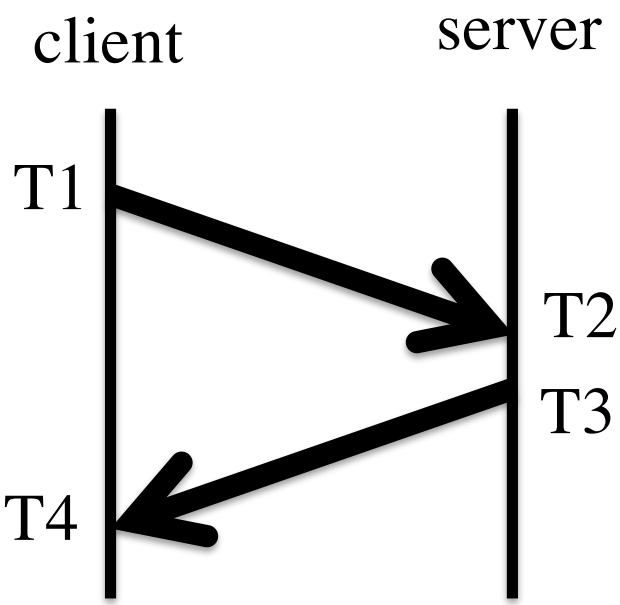
- NTP は、ネットワークに接続されている機器どうしの時刻を同期するための UDP/123 を用いた通信プロトコル
- 正確な時刻源を直接の同期時刻源にしている NTP サーバを stratum 1 サーバと呼び、stratum 1 サーバを同期時刻源にして NTP サーバを stratum 2 サーバと呼ぶ。以下、stratum 15 サーバまで定義されている。
- 最新の規格
  - RFC5905: Network Time Protocol Version 4: Protocol and Algorithms Specification, June 2010.

NTP packet header format



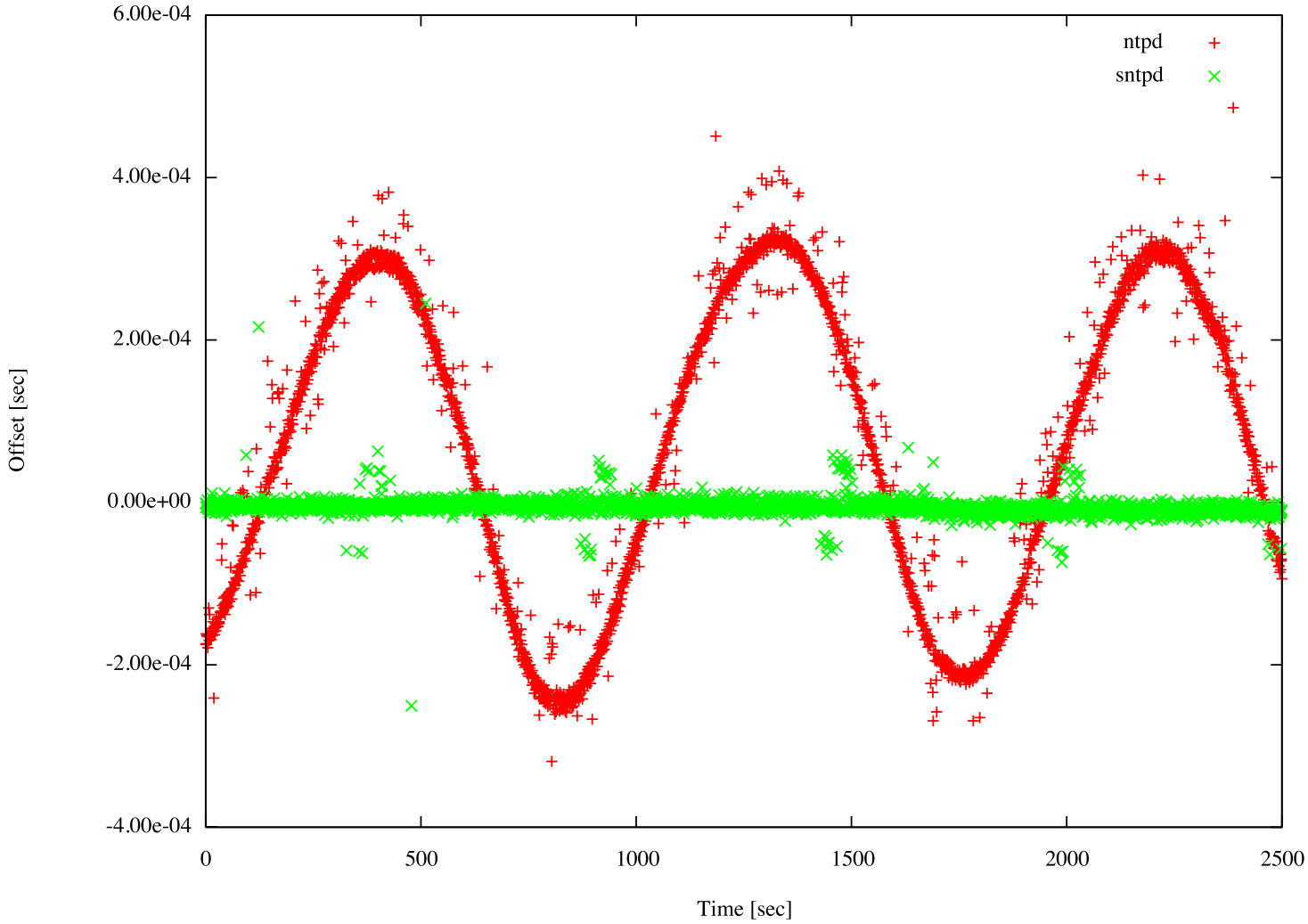
NTP による時刻同期方法

- NTP query & reply sequence
  - T1: 要求送信
  - T2: 要求受信
  - T3: 回答送信
  - T4: 回答受信



- 時刻同期方法
  - offset (client と server の時刻のずれ)を用いて補正
- offset の算出方法
  - (前提)“行き”と“帰り”の遅延時間が等しいこと
  - 片道通信時間 = ((T4−T1) − (T3−T2)) / 2
  - offset = T2 − T1 − 片道通信時間 = ((T2−T1) + (T3−T4))/2

NTP サーバの offset 変動（例）



ソフトウェア PLL に起因する周期的なゆらぎが発生

ntpd の特徴

- 複数サーバから同期するサーバを選択
  - Stratum 優先
    - 不安定で精度が悪い Stratum 1 サーバでも、安定した精度の高い Stratum 2 サーバより優先して選択される
  - 同じ Stratum からの選択
    - サーバに参照する毎に精度(offset) と分散(dispersion) を計算し、もっとも安定していると思われるサーバを Reference に選択する

- /etc/ntp.conf で設定
  - server 0.jp.pool.ntp.org
  - server 0.ubuntu.pool.ntp.org
  - server 0.freebsd.pool.ntp.org