Middleware and Distributed Systems

Time

Martin v. Löwis

Time In Distributed Systems

- Time is important
  - Timestamp of electronic commerce transactions
  - Some distributed algorithms depend on global time (consistency, authenticity checking, duplicate elimination)
- Time is problematic in distributed systems
  - Absence of global physical time
    - Local physical clock deviation
  - Order of event pairs from different nodes
    - E.g. distributed garbage collection

Time In Distributed Systems

- Collection of N processes $p_i = p_1, p_2, ..., p_N$
  - Each *process* on a single processor, no shared memory, own state s_i
  - Only message communication - send and receive *actions* change state
- *Event* as communication or state change action
  - Single total ordering of events in one process: history(p_i) = $e_i^0, e_i^1, e_i^2, ...$
- Timestamp for event
  - Physical computer clock: Counting of crystal oscillations, divide count and storage in register as hardware clock value $H_i(t)$
  - *Clock drift*: time is counted in different rates in the hardware clock
  - *Drift rate*: Offset to perfect clock value (10^{-6} to 10^{-8}, atom clock 10^{-13})

Drift Rates

Time	Quartz Clock
1second	0,000001 seconds
1 minute	0,00006 seconds
1 hour	0,0036 seconds
1 day	0,0864 seconds
1 week	0,6048 seconds
~11.6 days	1 second

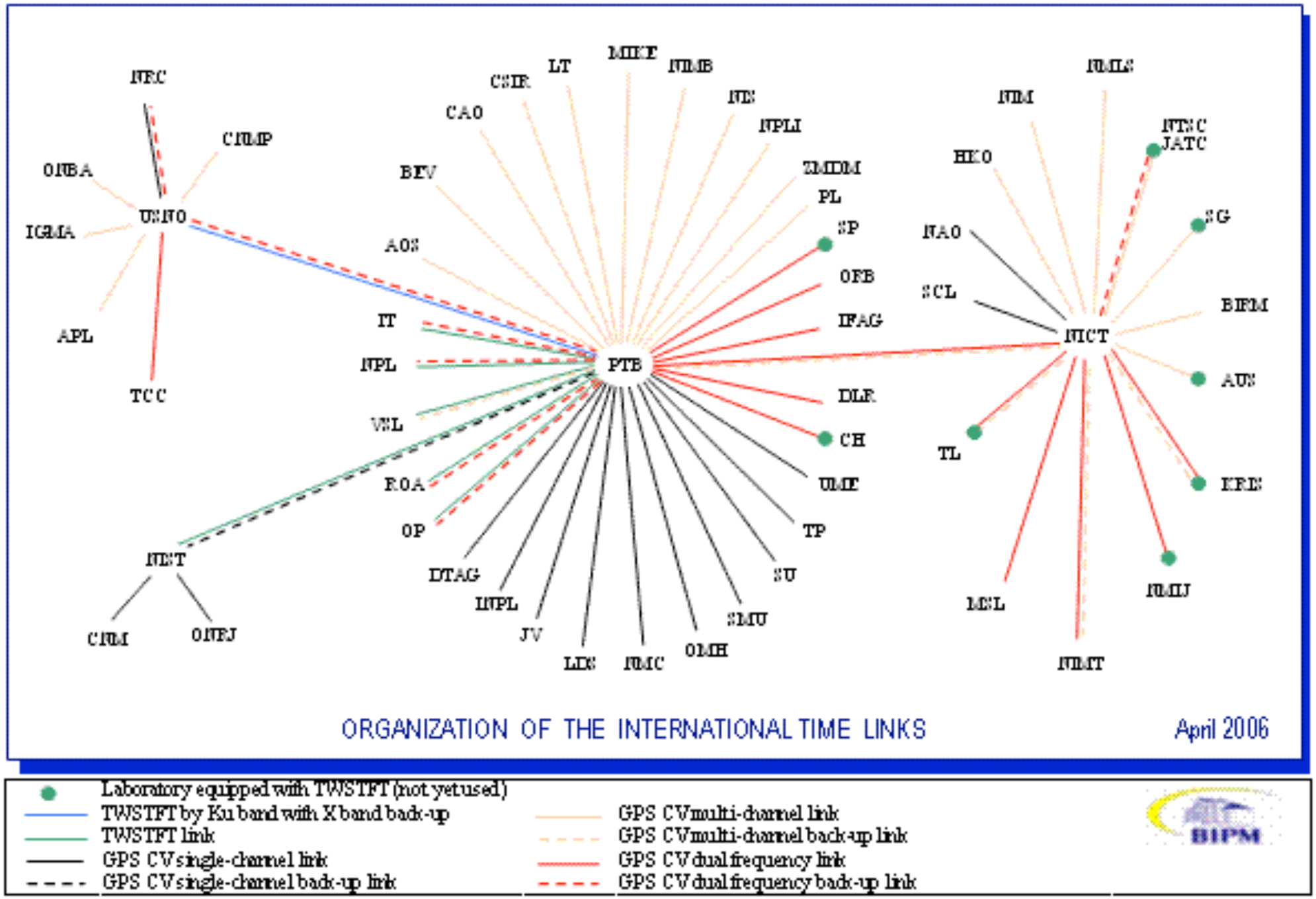
Clock Correctness

- *Correct* clock: Error in measured time interval is bounded
  - No jumps in time larger than pre-defined drift rate bound
- *Monotonous* clock: Clock only ever advances
  - Typical assumption for software clocks (make utility)
- Hybrid correctness approach: Monotonicity condition, but bounded drift rate only between synchronization points
- Faulty clock: crash fault (no more ticking) or arbitrary fault
  - Y2K clock: monotonicity condition broken
  - Large drift rate

Global Time

- A mean solar second is 1/86400 of a mean solar day
  - Solar day is interval between two consecutive transits (max height) of the sun
  - Solar day has gradually become 1.7 ms longer every century (moon acceleration)
  - 300 million years ago, a year had 400 days
- Invention of atomic clock in 1948
  - Counting of transitions of cesium 133 atom, since 1.1.1958
  - Increasing delay between mean solar time (UT1) and measured atomic time (TAI)
- International atomic time (TAI - Temps Atomique International)
  - Weighted average of the ticks of around 300 atomic clocks worldwide, e.g. TA(PTB)
    - Publication of latest TA(n) drift to TAI in Circular T publication
  - Managed by Bureau International des Poids et Mesures (BIPM)

TAI Computation by Pair-Wise Atomic Clock Drift



(C) BIPM

TAI Computation

- Algorithm from 1973, weighted average calculation every month
  - Pair-wise measurements every 10 days
  - Assumes stable clock frequency over 30 days
  - Linear equations with given difference values
- Weights of the participating clocks are computed iteratively
  - Take weights from last cycle at start, then always from the last iteration
  - Compute TAI, compute standards deviation of each clock for the last year
    - If smaller than 3.16ns/day, limit weight
    - Zero weight for incorrect clocks

TAI Correction Factors

- Ionospheric Maps
- Operational satellite positioning from International GPS Service (IGS)
- 1970's: Different tick rates due to gravitational time dilation
  - Corrections to correspond to proper time at mean sea level
  - TAI slowed down by about 10^{-12}
- 1990's: Periodic variations due to blackbody radiation, which varies with ambient temperature
  - Corrections factors to correspond to caesium atom at rest and at absolute zero temperature
  - TAI speed up by $10^{-14.3}$
- Mobile GPS receiver for GPS hardware callibration factor

UTC

- Coordinated Universal Time (CUT) or Temps Universel Coordonné (TUC)
  - Compromise abbreviation UTC by ITU organization
  - Replaced ‘greenwich mean time’ (astronomical time) with TAI time
  - Zero-point reference for time zones: ‘zulu time’
  - Occasionally insertion of leap second (diff > 0.6s), to keep in step with solar time
    - End of June 30 or December 31, decided by IERS
- Most power companies rely 60Hz / 50Hz timing on UTC clock
  - With leap second, frequency raised to 61Hz or 51Hz for 60 or 50 sec
  - Meanwhile 33 leap seconds in UTC
- UTC time by shortwave radio pulse very second (±10ms) or satellite (±0.5 ms)

Synchronizing physical clocks

- Synchronized clocks needed for global total ordering of events
- *External synchronization*
  - Process clocks must be synchronized with authoritative external source
  - Clocks C_i are accurate to within a given bound D
- *Internal synchronization*
  - Synchronization of process clocks with each other, without external reference
  - Clocks C_i agree within a given bound D
- Externally synchronized system with bound D is internally synchronized with bound 2D

Interaction Models

- Synchronous distributed system
  - Lower and upper bound for each process execution step
  - Upper bound for message transmission time
  - Local clock for each process, upper bound for clock drift rate
  - Mostly for theoretical models, hard to build in reality
- Asynchronous distributed systems
  - Different process execution step times, each with arbitrary length
  - Different message transmission delays
  - Different clock drift rates on each node

Clock Synchronization In Synchronous System

- Known bounds for drift rate and transmission delay
- P_1 sends t_1 of local clock to P_2, which theoretically sets $t_2 = t_1 + T_{trans}$
  - T_{trans} is unknown
  - Uncertainty $U(T_{trans}) = T_{max} - T_{min}$
  - Receiver can set his clock: $t_2 = t_1 + T_{max}$ or $t_2 = t_1 + T_{min}$
    -> clock skew may be as much as U in both cases
  - Receiver can set $t_2 = t_1 + (T_{max} - T_{min})/2$ -> clock skew is at most U/2
  - [Lundelius and Lynch 84] Optimum bound on clock skew for N clocks in a synchronous system is $U*(1-1/N)$
- Asynchronous system:
  - Unbounded message delays and execution step time

Task: Server time synchronization algorithm

Old Approaches

- First protocols in 1983
- DAYTIME protocol (RFC 867), port 13
  - Unspecified ASCII string with date and time
  - Example: time.nist.gov
- TIME protocol (RFC 868), port 37
  - 32-bit number, number of seconds since 00:00 1 January, 1900 GMT.

Clock Synchronization (Cristian 89)

- Central time server with time t_s, synchronized with UTC source
- Estimate roundtrip time between two nodes, since it remains constant
  - Exact local measurement of roundtrip time t_{round} is possible, because of relatively small clock drift rate (1 - 10 ms LAN, 10^{-5} ms drift rate)
  - $t_p = t_s + t_{round}/2$ (same time for request / reply)
- With minimum transmission time t_{min}
  - t_s was obtained earliest t_{min} after sending, and t_{min} before receiving
  - Accuracy of computed server time is $\pm(t_{round}/2 - t_{min})$
    - Time when reply arrives is in range width $t_{round} - 2t_{min}$
  - Minimal roundtrip time (as known value) brings most accuracy

Problems with Cristian Algorithm

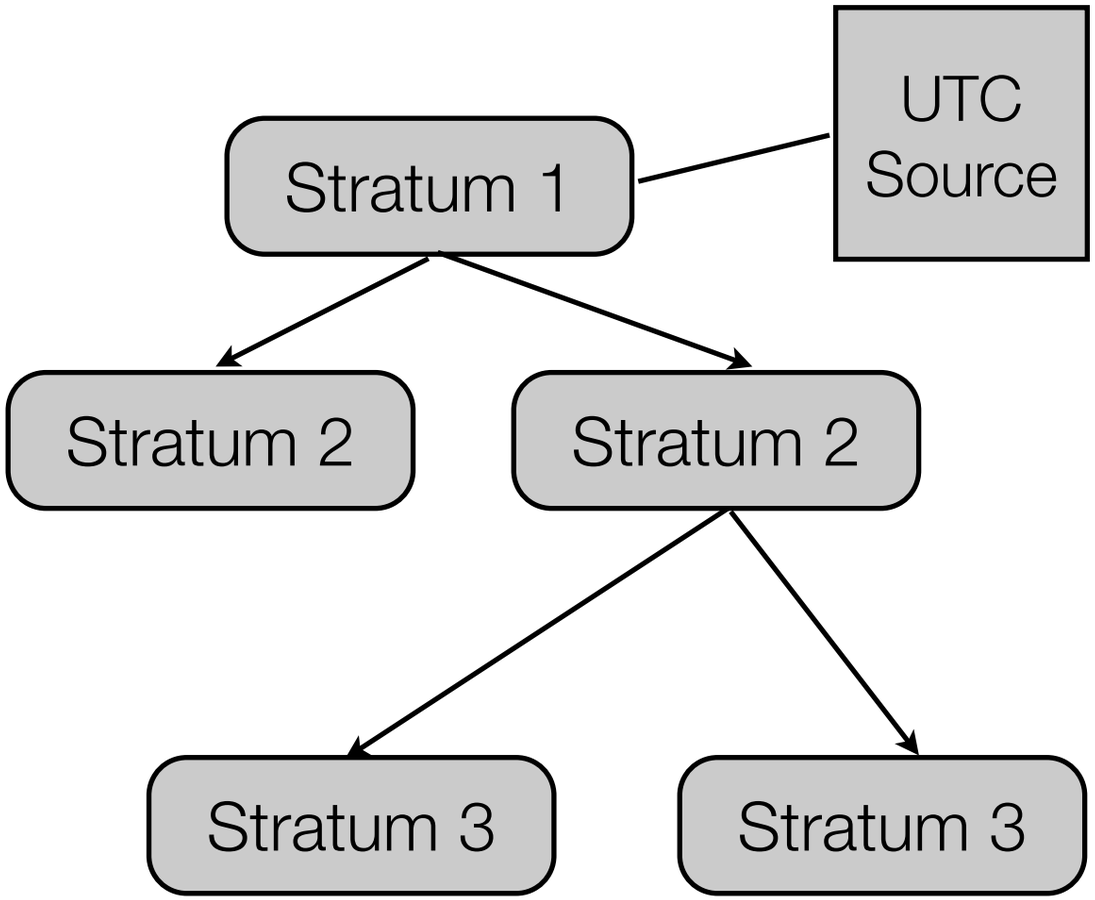
- No consideration of faulty clocks
  - Cristian suggested group of synchronized time servers
  - [Dolev 86]: f faulty clocks out of N demands $N > 3f$ for agreement
- Possibly time jumps on large skews
- Single roundtrip is possibly inexact
  - multiple measurements with average forming
  - discard runaways

Clock Synchronization (Gusella and Zatti 89)

- Berkeley (Unix) algorithm for internal synchronization
  - Master node polls slave nodes, slaves send clock values back
    - Master estimates each roundtrip time, and averages all values
    - Accuracy depends on maximum roundtrip time - longer reads are not considered in the average computation
  - Master sends back adjustment for each slave clock
    - Sending master time would introduce another transmission uncertainty
  - Failed master demands election protocol
  - Faulty slave nodes detected by fault-tolerant average
    - Average for sub-group should not differ too much from other sub-group

Network Time Protocol (Mills 95)

- Time synchronization over the Internet
  - Large and variable message delays - statistical techniques for data filtering, different time quality for different servers
  - Redundant servers and paths
  - Scalability for large numbers of periodically synchronizing clients
  - Protection with authentication techniques
  - Leaf servers executed in users workstation
  - External synchronization



NTP Terminology

- Clock stability: How well it can maintain a constant frequency
- Clock accuracy: How well frequency and time compare to national standard
- Clock precision: Precision of time quantities
- Clock offset: Time difference between two clocks
- Clock skew: Frequency difference between two clocks
- Clock skew variation: clock drift (assumed zero in NTP)
- NTP produces clock offset, roundtrip delay and dispersion
  - Clock offset is amount to adjust local clock to reference clock
  - Roundtrip delay allows to launch a message to arrive at specific time
  - Dispersion means the maximum clock offset to reference time

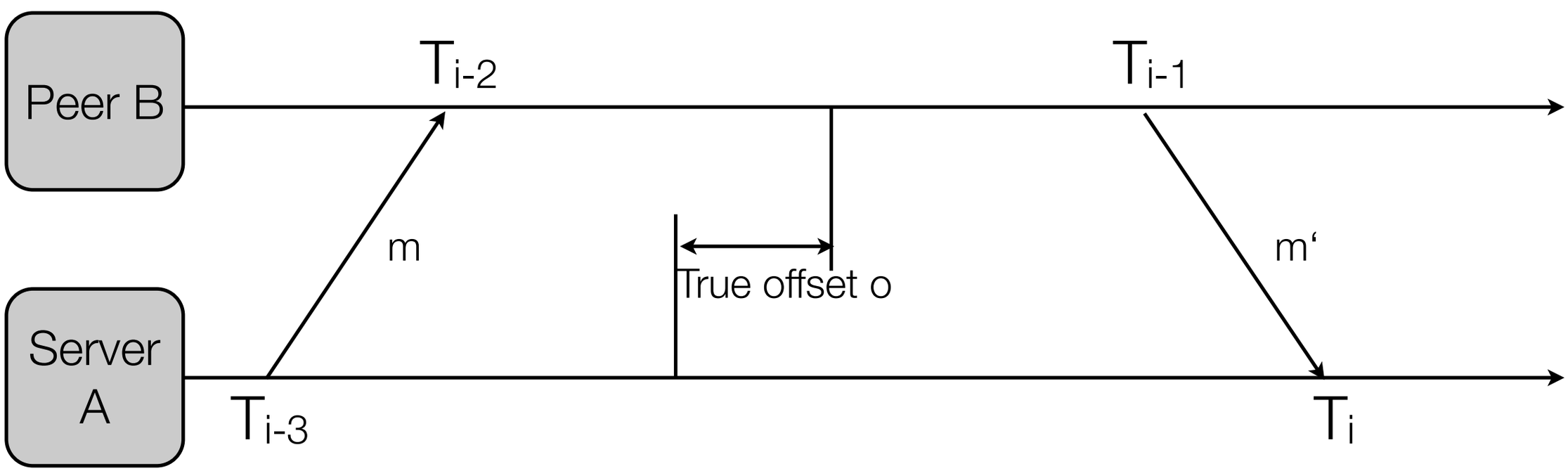
Demo

Network Time Protocol

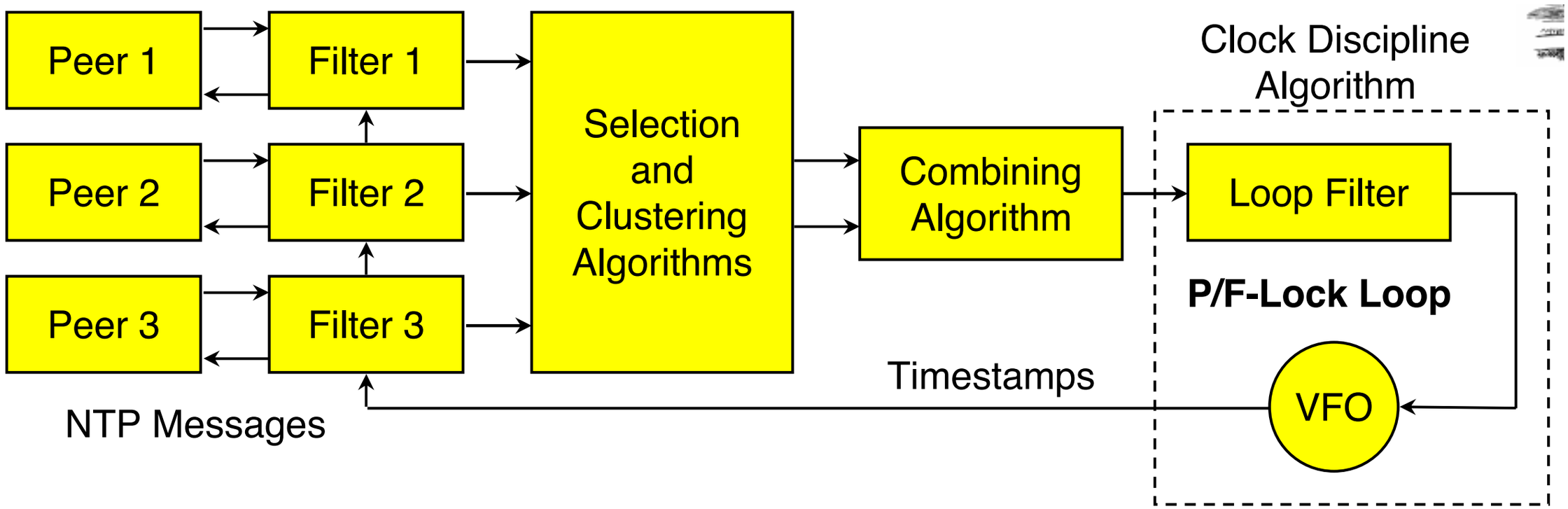
- UDP transport, port 123; oldest protocol in todays‘s internet
  - Own duplicate detection, single message format
- Over 230 primary servers, estimated 10-20 million NTP servers
- NTPv3 in RFC 1305, NTPv4 finished
- Long periods and multiple comparisons (with low network load) needed
- Accuracy in order of tens of milliseconds (Internet) or one millisecond (LAN)
- Operating system kernel adjusts microseconds / clock tick, based on measured clock drift
  - Consideration of *time_adj* value in each call of update_wall_time_one_tick() (Linux *timer.c*)

NTP Algorithm

- Exchange of timestamps to estimate offset o_i between the clocks at T_i and roundtrip delay $d_i=m+m'$; Errors due network delays dominate
- $T_{i-2}=T_{i-3}+m+o_i$ and $T_i=T_{i-1}+m'-o_i$, therefore $d_i=T_{i-2}-T_{i-3} + T_i-T_{i-1}$
- Estimated offset $o_i=(T_{i-2}-T_{i-3} + T_{i-1}-T_i)/2$
- $o <= T_{i-2}-T_{i-3}$ and $o >= T_{i-1}-T_i$ since m, m' > 0 ,leads to $\mathbf{o_i-d_i/2 <= o <= o_i+d_i/2}$



NTP Algorithm



- Computed delay and offset pairs $<o_i, d_i>$ are filtered
- Statistical quantity expresses the measurement quality (filter dispersion)
- Similar to Cristian, the o_j for the smallest d_j is taken (from 8 samples)
- Clock frequency adjustment according to drift rate, runs in interval of a second

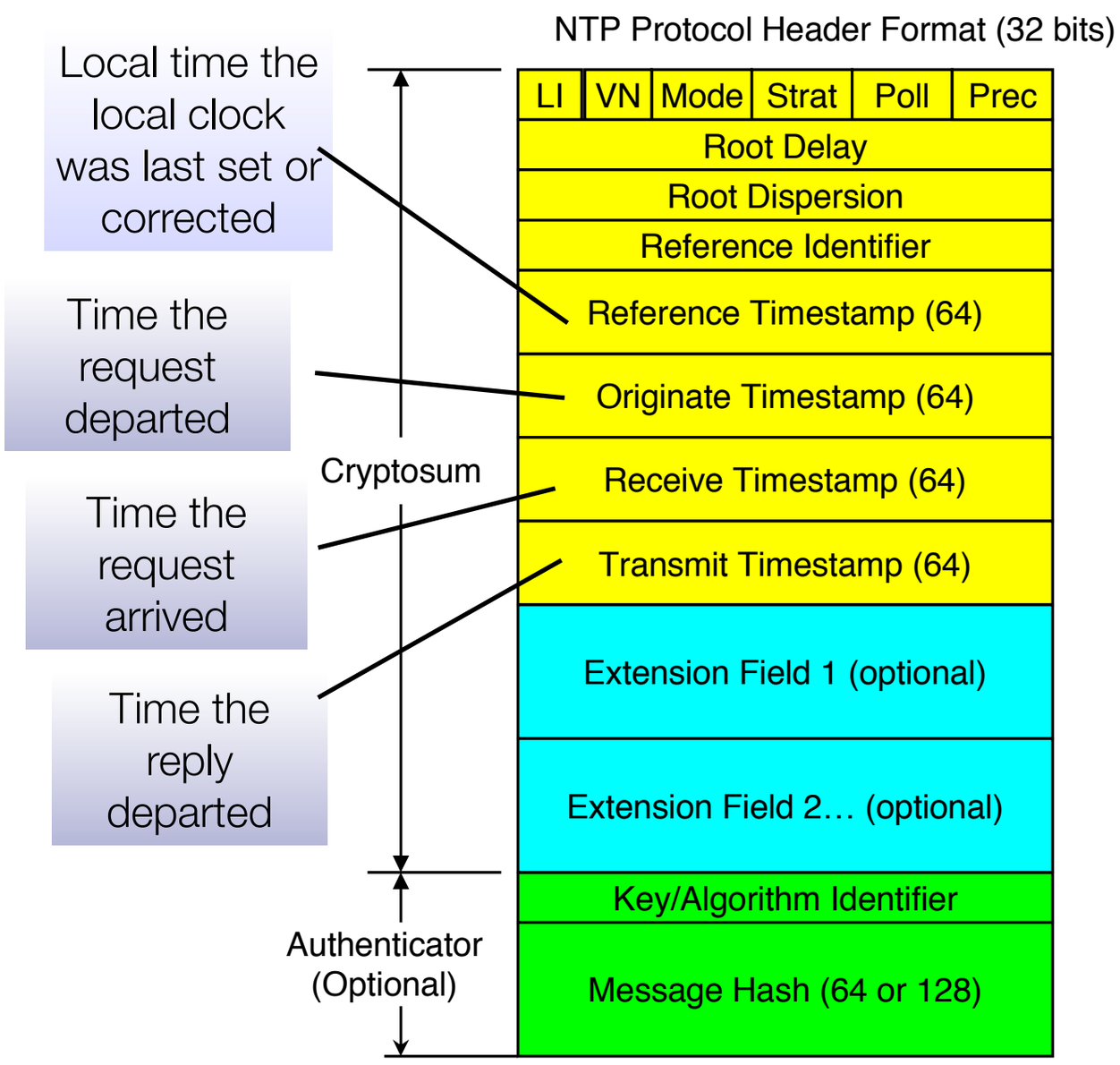
Synchronization Modes

- Multicast
  - For high-speed LAN, receiving servers assume small delay of a few milliseconds
  - Server announces time, but does not accept NTP messages
- Procedure call
  - Server in client mode is willing to be synchronized, but offers no service
  - For LAN‘s without multicast or demand for higher accuracy
- Symmetric
  - Active mode for high-stratum servers, with pre-configured peers
  - Passive mode for low-stratum servers and large number of peers

Data Formats in NTP

- NTP timestamp: 64bit unsigned fixed-point number
  - Number of standard seconds since 1.1.1900, relative to UTC
  - Precision of 232 picoseconds
  - 1968, the most significant bit has been set - overflow in 2036
- Leap indicator: 00 no warning, 01 last minute has 61 seconds, 10 last minute has 59 seconds
- Precision of the local clock, roundtrip delay to primary reference source, maximum error to primary reference source
- Reference clock identifier
  - Stratum 2 or greater - IP address of reference clock; otherwise special tag (e.g. ATOM or GPS)

NTP Protocol



Authenticator uses DES-CBC or MD5 cryptosum of NTP header plus extension fields (NTPv4)

LI	leap warning indicator
VN	version number (4)
Strat	stratum (0-15)
Poll	poll interval (log2)
Prec	precision (log2)

NTP Timestamp Format (64 bits)

Seconds (32)	Fraction (32)

Value is in seconds and fraction since 0^h 1 January 1900

NTPv4 Extension Field

Field Length	Field Type
Extension Field (padded to 32-bit boundary)	

Last field padded to 64-bit boundary

NTP v3 and v4
NTP v4 only
authentication only

NTP Robustness

- Peer is considered unreachable if 8 poll intervalls fail
- Originate timestamp as one-time pad
  - Should somehow match to transmit timestamp of last message
- Longest poll intervals must be consistent with required accuracy
  - Poll interval is increased from minute to about 17 minutes, as long as filter dispersion is below an experimentally determined threshold
- Consistency algorithm for NTP peer selection (to find most reliable clock from a population)
  - Highest reliability = lowest stratum and lowest synchronization distance (sum of total roundtrip delays to the synchronization root)