Ubiquitous Networks

Clock Synchronization

Lynn Choi
Korea University

Why do we need to synchronize clocks?

- **Data fusion**
  - Elimination/integration of redundant data from multiple sensors
- **Synchronization for networking protocols**
  - Wakeup scheduling: for (2-partie) communication / for the latency
  - Slot time, interframe spacing, timeouts
  - TDMA scheduling
- **Event ordering**
  - The relative ordering (or time interval) between two events that happened on different machines in the network
- **Localization (ToA, TDoA)**
- **Cooperative operation by multiple sensors**
  - Velocity estimate of a moving object
  - Measure the time-of-flight of sound

Requirements for Wireless Sensor Networks

- **Energy efficiency**
  - Need to consider energy efficiency without external energy source
- **Scalability**
  - Must be scalable to a large number of nodes (should be distributed)
- **Robustness**
  - Fault-tolerant, without human involvement
- **Cost and size**
  - Must be applicable to low-cost sensors
  - Limited bandwidth, limited computation power and storage space
- **Accuracy and precision**
  - Depend on the objectives and the applications
- **Scope**
  - Local or global

Crystal Oscillator

Characteristics of crystal oscillators

Accuracy

- The difference between the expected frequency and actual frequencies. This difference is called the *frequency error*, whose maximum is specified by the manufacturer.
  - The maximum error is in the range of one part in 10^4 to 10^6, which translates to 1 ~ 100 **μ**s/s.
  - Two Berkeley Motes may have 4.75 **μ**s/s of skew at the maximum, which leads to 17.1ms after 1 hour and 1 second after 58 hours

Stability

- An oscillator's tendency to stay at the same frequency.
- Short-term instability is caused by environmental factors such as temperature, supply voltage, and shock
- Long-term instability is caused by oscillator aging.

Clock Model

- **Clock can be modeled by drift and offset**
  - *Drift* (*skew*) denotes the rate (frequency) of the clock
  - *Offset* (or *phase offset*) denotes the difference in value from the *real time t*
  - For a node i in the network, its local clock can be represented as

$$C_i(t) = a_i t + b_i$$

where $a_i(t)$ denotes the clock skew and

$b_i(t)$ is the offset of node i's clock.

- **Using the equation, we can compare the local clocks of two nodes as**

$$C_1(t) = a_{12} C_2(t) + b_{12}$$

  - Where a_{12} denotes the relative drift and b_{12} denotes the relative offset.
  - If two clocks are perfectly synchronized, then their relative drift is 1 and the relative offset is zero

Distributed Time Synchronization

- **All network time synchronization schemes rely on some message exchanges between nodes**
  - Nondeterminism in the network makes the synchronization task challenging

- **Sources of time synchronization errors**
  - Send time
    - Time required to transfer the message from the host to its network interface
  - Access time
    - Time waiting for access to transmit the message
  - Propagation time
    - This time is very small (1ns/foot) and can be ignored
  - Receive time
    - The time required for the network interface to generate a message reception signal

Existing Algorithms

- **They vary primarily in their methods for estimating and correcting for these sources of errors**

- **Most share a basic design**
  - A server periodically sends a message containing its current clock value to a client
    - If the typical latency from a server to a client is small compared to the desired accuracy, a simple one-way message is enough
  - A common extension is to use a client request followed by a server's response.
    - By measuring the round-trip time of two packets, the client can estimate the one-way latency

- **Example: NTP**
  - NTP performs a large number of request/response messages to filter random delays (i.e. shortest round-trip time)

Remote Clock Reading (Cristian, 1989)

- A client sends a message to the server requesting a timestamp. Let this message be initiated at time T_0 local to the client.
- The server then returns a message holding the timestamp (S_{time}). S_{time} is the local time at the server.
- The client receives this message at its local time, say, T_1.
- The client then sets its time to S_{time} (accurate time from the server) + (T_1-T_0)/2 (time required to transmit the message).
- To ensure accuracy, several round-trips are made and the average is used or the shortest round-trip is used.

Figure 4: Cristian's synchronization protocol.

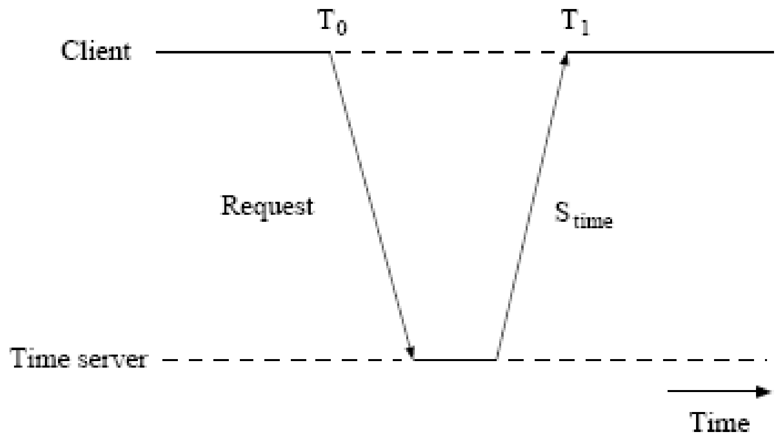


Figure 5: Remote Clock Reading [11].

Offset Delay Estimation (used by NTP)

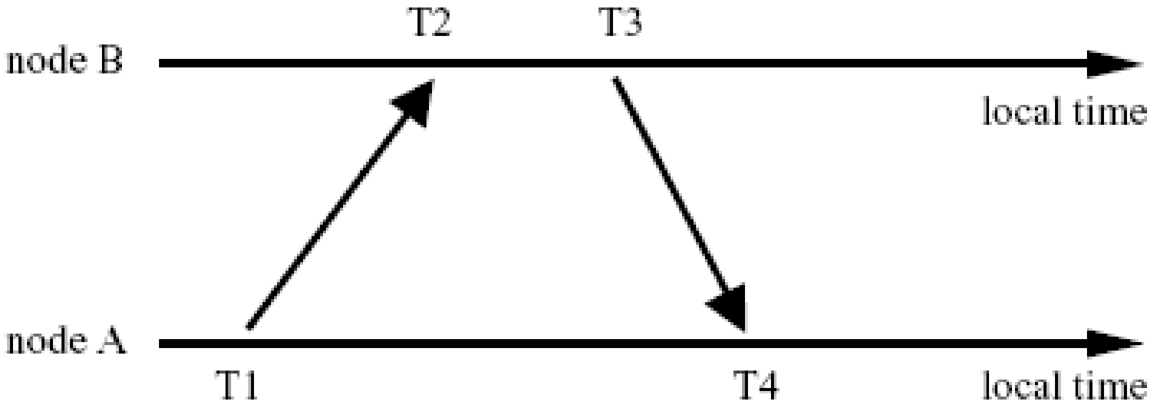
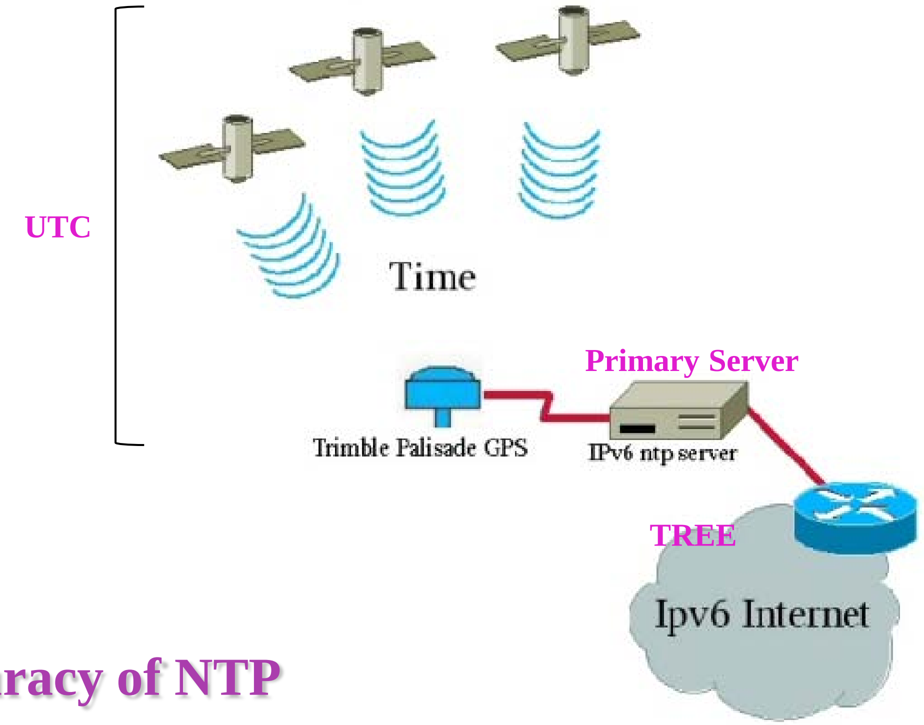


Figure 2: Two way message exchange between a pair of nodes.

$$\Delta = \frac{(T2 - T1) - (T4 - T3)}{2}; \quad d = \frac{(T2 - T1) + (T4 - T3)}{2}$$

NTP (Network Time Protocol)

- **Hierarchy of NTP servers**
  - Primary server at the root synchronizes with the UTC
  - A node synchronizes with its parent by performing several trials of offset delay estimation and choose the offset with the minimum delay (to compensate for the delay variance)



- **The reported accuracy of NTP**
  - 1 ~ 50ms (1ms for LAN, 28.7ms for WAN)

UTC (Coordinated Universal Time)

UTC

- Primary *time standard* by which the world regulates clocks and time
  - Similar to GMT (Greenwich Mean Time), which is no longer precisely defined
- Based on *International Atomic Time*, a time standard calculated using a weighted average of signals from atomic clocks of nearly 70 national labs around the world
  - Occasionally adjusted by adding a *leap second* in order to keep up with Earth's rotation.
- *The* Standard
  - Used for Internet (NTP), aviation, traffic control, weather forecast, etc.

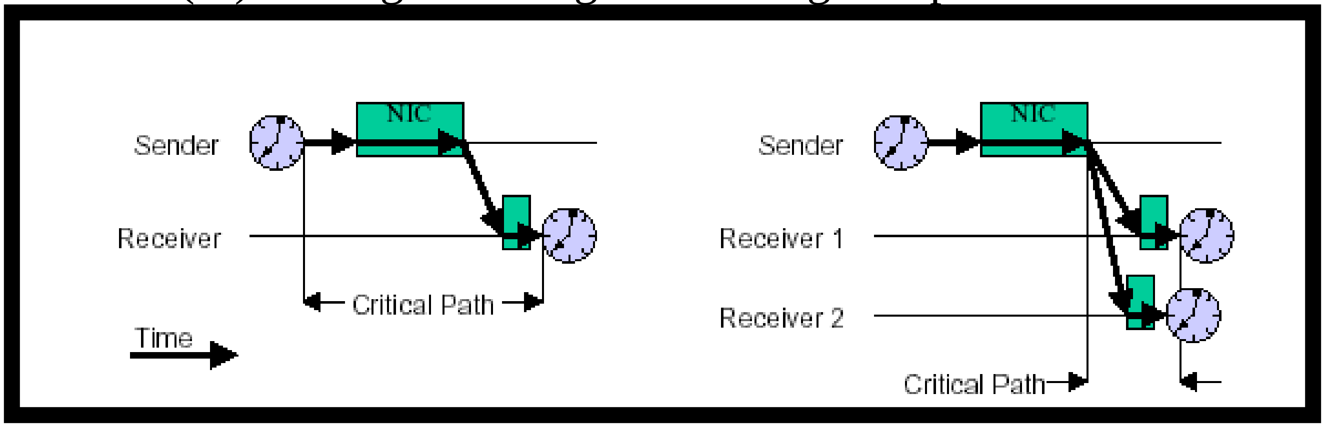
NTP (Network Time Protocol)

Variations of NTP

- SNTP (Simple NTP)
  - Less accurate, but simpler
- IEEE 1588
  - For measurement and control on small networks
    - Only for synchronization within subnet (no router)
  - Accuracy of several hundred nanoseconds
- GPS
  - Accuracy of 10ns

Synchronization Protocols for WSN

- **RBS (Reference Broadcast Synchronization), OSDI,2002**
  - Receiver to receiver synchronization (no timestamp)
    - A message broadcast at the physical layer will arrive at a set of receivers with very little variability in its delay
      - Transmitter broadcasts a reference packet to two receivers
      - Each receiver records the reception time according to its local clock
      - The receivers exchange their observations
    - Eliminate the largest sources of error (send time and access time) from the critical path
    - Issues: $O(n^2)$ message exchanges for a single-hop network of n nodes



NTP RBS

Source: USENIX

Synchronization Protocols for WSN

- **TPSN (Timing -Sync Protocol for Sensor Networks), SenSys 2003**
  - Two-phase protocols
    - Level discovery phase – create a tree
    - Synchronization phase
      - Starting from the root, each node synchronizes with its parent by using offset delay estimation
  - Implemented on Berkeley's Mica architecture
  - Use MAC layer time-stamping
    - Eliminate timestamp uncertainty due to send and access time
  - Claims that uncertainty at the sender contributes little to the total synchronization error and they can outperform RBS

- **Lightweight synchronization schemes for WSN**
  - Tiny-Sync & Mini-Sync, WCNC 2003
  - LTS (Lightweight Tree-based Synchronization), WSNA 2003
  - FTSP (Flooding Time Synchronization Protocol), SenSys 2004

Lightweight Schemes

Tiny/Mini-Sync

- Use only a subset of data points for simplification but compensate with
  - Multiple round-trip measurement and a line-fitting technology to obtain offset and rate difference of two nodes
- The scheme considers not only clock offset but also *clock skew*
  - Use a history of clock information used in the past synchronization processes
- Mini-sync is an extension of Tiny-Sync
  - Find a better solution but with an increase in complexity

LTS

- Only for nodes that need synchronization
- Propose two variations: on-demand and proactive algorithms

Lightweight Schemes

- **FTSP**
  - The broadcast-based approach
    - The flooding-based scheme causes unexpected collisions and useless packet transmissions.
  - FTSP reduces the number of message per node to one.
  - Do not adjust error due to the message delay.