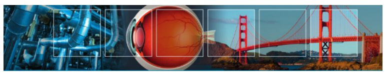
Chapter 9: Time Synchronization



Chapter 9: Roadmap

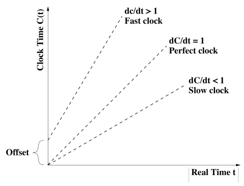
- The time synchronization problem
- Time synchronization in wireless sensor networks
- Basic techniques for time synchronization
- Time synchronization protocols

Clocks and the Synchronization Problem

- Common time scale among sensor nodes is important for a variety of reasons
  - identify causal relationships between events in the physical world
  - support the elimination of redundant data
  - facilitate sensor network operation and protocols
- Typical clocks consist of quartz-stabilized oscillator and a counter that is decremented with every oscillation of the quartz crystal
- When counter reaches 0, it is reset to original value and interrupt is generated
- Each interrupt (clock tick) increments software clock (another counter)
- Software clock can be read by applications using API
- Software clock provides local time with $C(t)$ being the clock reading at real time t
- Time resolution is the distance between two increments (ticks) of software clock

Clock Parameters

- Clock offset: difference between the local times of two nodes
- Synchronization is required to adjust clock readings such that they match
- Clock rate: frequency at which a clock progresses
- Clock skew: difference in frequencies of two clocks
- Clock rate *dC/dt* depends on temperature, humidity, supply voltage, age of quartz, etc., resulting in drift rate (*dC/dt-1*)



Clock Parameters

- Maximum drift rate ρ given by manufacturer (typical 1ppm to 100ppm)
- Guarantees that: $$1-\rho \le \frac{dC}{dt} \le 1+\rho$$
- Drift rate causes clocks to differ even after synchronization
- Two synchronized identical clocks can drift from each other at rate of at most $2\rho_{max}$
- To limit relative offset to δ seconds, the resynchronization interval τ_{sync} must meet the requirement: $$\tau_{sync} \le \frac{\delta}{2\rho_{\max}}$$

Clock Parameters

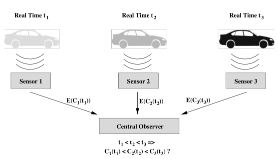
- C(t) must be piecewise continuous (strictly monotone function of time)
  - clock adjustments should occur gradually, e.g., using a linear compensation function that changes the slope of the local time
  - simply jumping forward/backward in time can have unintended consequences
    - time-triggered events may be repeated or skipped

Time Synchronization

- External synchronization
  - clocks are synchronized with external source of time (reference clock)
  - reference clock is accurate real-time standard (e.g., UTC)
- Internal synchronization
  - clocks are synchronized with each other (no support of reference clock)
  - goal is to obtain consistent view of time across all nodes in network
  - network-wide time may differ from external real-time standards
- External synchronization also provides internal synchronization
- Accuracy: maximum offset of a clock with respect to reference clock
- Precision: maximum offset between any two clocks
- If two nodes synchronized externally with accuracy of Δ, also synchronized internally with precision 2Δ

Why Time Synchronization in WSNs?

- Sensors in WSNs monitor objects and events in the physical world
- Accurate temporal correlation is crucial to answer questions such as
  - how many moving objects have been detected?
  - what is the direction of the moving object?
  - what is the speed of the moving object?
- If real-time ordering of events is $t_1<t_2<t_3$, then sensor times should reflect this ordering: $C_1(t_1)<C_2(t_2)<C_3(t_3)$



Why Time Synchronization in WSNs?

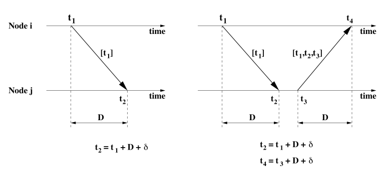
- Time difference between sensor time stamps should correspond to real-time differences: $\Delta=C_2(t_2)-C_1(t_1)=t_2-t_1$
  - important for data fusion (aggregation of data from multiple sensors)
- Synchronization needed by variety of applications and algorithms
  - communication protocols (at-most-once message delivery)
  - security (limit use of keys, detect replay attacks)
  - data consistency (caches, replicated data)
  - concurrency control (atomicity and mutual exclusion)
  - medium access control (accurate timing of channel access)
  - duty cycling (know when to sleep or wake up)
  - localization (based on techniques such as time-of-flight measurements)

Challenges for Time Synchronization

- Traditional protocols (e.g., NTP) are designed for wired networks
- WSNs pose a variety of additional challenges
- Environmental effects
  - sensors often placed in harsh environments
  - fluctuations in temperature, pressure, humidity
- Energy constraints
  - finite power sources (batteries)
  - time synchronization solutions should be energy-efficient
- Wireless medium and mobility
  - throughput variations, error rates, radio interferences, asymmetric links
  - topology changes, density changes
  - node failure (battery depletion)
- Other challenges
  - limited processor speeds or memory (lightweight algorithms)
  - cost and size of synchronization hardware (GPS)

Synchronization Messages

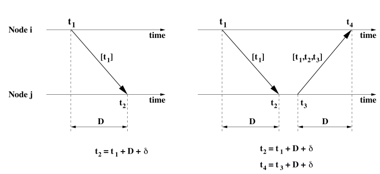
- Pairwise synchronization: two nodes synchronize using at least one message
- Network-wide synchronization: repeat pairwise synchronization throughout network
- One-way message exchange:
  - single message containing a time stamp
  - difference can be obtained from $(t_2-t_1)=D+\delta$ (D=propagation delay)



Synchronization Messages

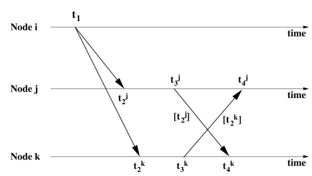
- Two-way message exchange:
  - receiver node responds with message containing three time stamps
  - assumption: propagation delay is identical in both directions and clock drift does not change between measurements

$$D = \frac{(t_2 - t_1) + (t_4 - t_3)}{2} \qquad offset = \frac{(t_2 - t_1) - (t_4 - t_3)}{2}$$



Receiver-Receiver Synchronization

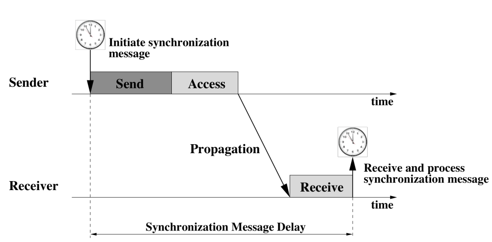
- So far: sender-receiver approaches
- Receiver-receiver: multiple receivers of broadcast messages exchange their message arrival times to compute offsets among them
- Example: 2 receivers; 3 messages (1 broadcast, 2 exchange messages)
- No time stamp in broadcast message required



Nondeterminism of Communication Latency

- Several components contribute to total communication latency
- Send delay:
  - generation of synchronization message
  - passing message to network interface
  - includes delays caused by OS, network protocol stack, device driver
- Access delay:
  - accessing the physical channel
  - mostly determined by medium access control (MAC) protocol
- Propagation delay:
  - actual time for message to travel to sender (typically small)
- Receive delay:
  - receiving and processing the message
  - notifying the host (e.g., interrupt)

Nondeterminism of Communication Latency

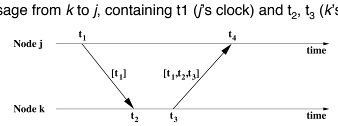


Reference Broadcasts

- Global Positioning System (GPS) is a well-known global source of time
  - time measured from epoch started at 0h January 6, 1980 UTC
  - unlike UTC, GPS not perturbed by leap seconds
    - GPS is ahead by 15 seconds (and increasing)
- Terrestrial radio stations
  - WWV/WWVH & WWVB (National Institute of Standards & Technology)
  - continuously broadcast time based on atomic clocks
- Problems with these techniques:
  - not universally available (underwater, indoors, outer space)
  - need for high-power receivers
  - size
  - cost

Lightweight Tree-Based Synchronization

- Goal of LTS is to provide specified precision with little overhead
- Based on pairwise synchronization:
  - message from *j* to *k*, containing time stamp t_1 (*j*'s clock)
  - message from *k* to *j*, containing t1 (*j*'s clock) and t_2, t_3 (*k*'s clock)



  - assuming message delay D

$$offset = \frac{t_2 - t_4 - t_1 + t_3}{2}$$

Lightweight Tree-Based Synchronization

- Centralized multi-hop version of LTS
  - reference node is root of spanning tree containing all nodes
  - breadth first search used to construct tree
  - once tree established, reference nodes synchronizes with children
  - errors from pairwise synchronization are additive
    - keep depth of tree small
  - overhead of pairwise synchronization: 3 messages
  - overhead of network-wide synchronization: 3n-3 messages (n edges)

Lightweight Tree-Based Synchronization

- Distributed multi-hop version of LTS
  - one or more reference nodes contacted by sensors whenever synchronization is required
  - nodes determine resynchronization period based on desired clock accuracy, distance to reference node, clock drift ρ, time of last synchronization
  - node can query neighbors for pending synchronization requests, i.e., node synchronizes with neighbor instead of reference node

Timing-sync Protocol for Sensor Networks

- TPSN is another sender-receiver technique
- Uses a tree to organize network
- Uses two phases for synchronization
  - discovery phase
  - synchronization phase

Timing-sync Protocol for Sensor Networks

- Level discovery phase
  - establish hierarchical topology
    - root resides at level 0
  - root initiates phase by broadcasting *level_discovery* message (contains level and identity of sender)
  - receiver can determine own level (level of sender plus one)
  - receiver re-broadcasts message with its own identity and level
  - receiver discards multiple received broadcasts
  - if node does not know its level (corrupted messages, etc.), it can issue *level_request* message to neighbors (which reply with their levels)
    - node's level is then one greater than the smallest level received
    - node failures can be handled similarly (i.e., if all neighbors at level i-1 disappear, node issues level_request message
    - if root node dies, nodes in level 1 execute leader election algorithm

Timing-sync Protocol for Sensor Networks

- Synchronization phase
  - pairwise synchronization along the edges of hierarchical structure
  - each node on level *i* synchronizes with nodes on level *i-1*
    - approach similar to LTS:
      - node *j* issues synchronization pulse at t1 (containing level and time stamp)
      - node *k* receives message at t_2 and responds with an ACK at t_3 (containing t_1, t_2, t_3, and level)
      - node *j* receives ACK at t_4
    - node *j* calculates drift and propagation delay

$$D = \frac{(t_2 - t_1) + (t_4 - t_3)}{2}$$

$$offset = \frac{(t_2 - t_1) - (t_4 - t_3)}{2}$$

Timing-sync Protocol for Sensor Networks

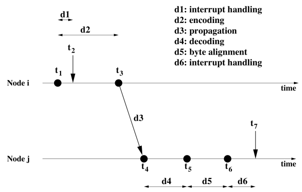
- Synchronization phase (contd.)
  - phase initiate by root node issuing *time_sync* packet
  - after waiting for random interval (to reduce contention), nodes in level 1 initiate two-way message exchange with root node
  - nodes on level 2 will overhear synchronization pulse and initiate two-way message exchange with level 1 nodes after random delay
  - process continues throughout network
- Synchronization error of TPSN
  - depth of hierarchical structure
  - end-to-end latencies

Flooding Time Synchronization Protocol

- Goals of FTSP include:
  - network-wide synchronization with errors in microsecond range
  - scalability up to hundreds of nodes
  - robustness to topology changes
- FTSP uses single broadcast message to establish synchronization points
- Decomposes end-to-end delay into different components

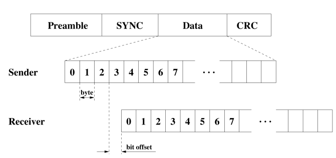
Flooding Time Synchronization Protocol

- t_1: wireless radio informs CPU that it is ready for next message
- d1: interrupt handling time (few microseconds)
- t_2: CPU generates time stamp
- d2: encoding time (transform message into electromagnetic waves; deterministic, low hundreds of microseconds)
- d3: propagation delay (from t_3 on node *i* to t_4 on node *j*; typically very small and deterministic)
- d4: decoding time (deterministic, low hundreds of microseconds)
- d5: byte alignment time (delay caused by different byte alignments (bit offsets), i.e., receiving radio has to determine the offset from a known synchronization byte and then shift incoming message accordingly); can reach several hundreds of microseconds
- t_7: interrupt, CPU obtains time stamp



Flooding Time Synchronization Protocol

- Time-stamping in FTSP
  - sender sends single broadcast containing time stamp (estimated global time)
  - receiver extracts time stamp from message and time-stamps arrival (leads to global-local time pair, providing a synchronization point)
  - synchronization message begins with preamble followed by SYNC bytes, data field, and CRC
  - preamble bytes are used to synchronize receiver radio to carrier frequency
  - SYNC bytes are used to calculate bit offset



Flooding Time Synchronization Protocol

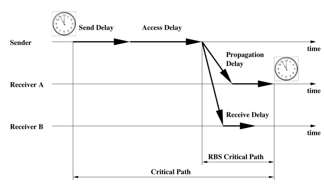
- Time-stamping in FTSP (contd.)
  - multiple time stamps are used at both sender and receiver to reduce jitter of interrupt handling and encoding/decoding times
  - time stamps are recorded at each byte boundary after the SYNC bytes as they are transmitted or received
  - time stamps are normalized by subtracting appropriate integer multiple of nominal byte transmission time (e.g., approx. 417µs on Mica2)
  - jitter in interrupt handling can be reduced by taking the minimum of normalized time stamps
  - jitter in encoding/decoding can be reduced by averaging these corrected normalized time stamps
  - final (error-corrected) time stamp is added into data part of message
  - at receiver side, time stamp must further be corrected by the byte alignment time (can be determined from transmission speed and bit offset)

Flooding Time Synchronization Protocol

- Multi-hop synchronization
  - root node is elected based on unique node IDs
  - root node maintains global time and all other nodes synchronize to root
  - synchronization is triggered by broadcast message by the root node
    - whenever node does not receive synchronization message for certain amount of time, it declares itself to be the new root
    - whenever root receives a message from node with lower node ID, it gives up root status
  - all receiver nodes within range establish synchronization points
  - other nodes establish synchronization points from broadcasts of synchronized nodes that are closer to the root
  - a new node joining the network with lowest node ID will first collect synchronization messages to adjust its own clock before claiming root status

Reference-Broadcast Synchronization

- Key idea of RBS: in the wireless medium, broadcast messages will arrive at receivers at approximately the same time
  - set of receivers synchronize with each other using a broadcast message
  - variability in message delay dominated by propagation delay and time needed to receive and process incoming message (send delay and access delay are identical)
  - RBS critical path is short than critical path of traditional techniques



Reference-Broadcast Synchronization

- Example with 2 receivers:
  - receivers record arrival of synchronization message
  - receivers exchange recorded information
  - receivers calculate offset (difference of arrival times)
- More than 2 receivers:
  - maximum phase error between all receiver pairs is expressed as group dispersion
  - likelihood that a receiver is poorly synchronized increases with the number of receivers (larger group dispersion)
  - increasing the number of broadcasts can reduce group dispersion
- Offsets between two nodes can be computed as the average phase offsets for all m packets received by receivers i and j:

$$offset[i,j] = \frac{1}{m}\sum_{k=1}^{m}(T_{j,k} - T_{i,k})$$

Reference-Broadcast Synchronization

- Multi-hop scenarios possible by establishing multiple reference beacons, each with its own broadcast domain
- Domains can overlap and nodes within overlapping regions serve as bridges to allow synchronization across domains

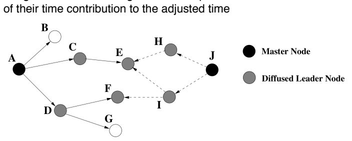
- RBS uses large amount of message exchanges
- However, RBS is a good candidate for post-facto synchronization
  - nodes synchronize *after* event of interest has occurred to reconcile their clocks

Time-Diffusion Synchronization Protocol

- In TDP, nodes agree on network-wide equilibrium time and maintain clocks within a small bounded deviation from this time
- Nodes structure themselves into tree-like configuration with two roles:
  - master nodes
  - diffused leader nodes
- TDP's Time Diffusion Procedure (TP) diffuses time information from master nodes to neighbors, some of which become diffused leader nodes responsible for propagating the master node's messages
- During the active phase of TDP, master nodes are elected every τ seconds using an Election/ Reelection Procedure (ERP)
  - balances workload in the network
  - τ further divided into intervals of δ seconds, each beginning with the election of diffused leader nodes
  - ERP eliminates leaf nodes and nodes with clocks that deviate from neighboring clocks by more than a certain threshold (achieved through message exchanges to compare clocks)
  - ERP also considers energy status in election process
- During the inactive phase of TDP, no time synchronization takes place

Time-Diffusion Synchronization Protocol

- Elected master node broadcasts timing information to neighbors
- Diffused leader nodes respond with ACK message
- Master nodes determine round-trip delay Δ_j for each neighbor *j*, an estimate of one-way delay for all neighbors ($\Delta_{avg}/2$), and standard deviation of the round-trip delays
- Standard deviation is sent in another time-stamped message to each neighboring diffused leader node
- Diffused leader nodes adjust their clocks using the time-stamp, the one-way delay estimation, and the standard deviation
- Diffused leader nodes repeat process with their neighbors (*n* times, where *n* is the distance from the master node in hops)
- Nodes receiving timing information messages from multiple masters use the standard deviations as weighted ratio of their time contribution to the adjusted time



Mini-Sync and Tiny-Sync

- Pairwise synchronization approaches with low bandwidth, storage, and processing requirements
- Two clocks $C_1(t)$ and $C_2(t)$ can be represented as:

$$C_1(t) = a_{12}C_2(t) + b_{12}$$
$$a_{12} = \text{relative drift}$$
$$b_{12} = \text{relative offset}$$

- Nodes can use two-way messaging scheme to determine these parameters:
  - node 1 sends a time-stamped probe at t_0
  - node 2 responds with time-stamped replay at t_1
  - node 1 records arrival time of reply at t_2
  - results in data point (t_0,t_1,t_2), which must satisfy:

$$t_0 < a_{12}t_1 + b_{12}$$
$$t_2 > a_{12}t_1 + b_{12}$$

Mini-Sync and Tiny-Sync

- Procedure is repeated several times to obtain series of data points and new constraints on admissible values of a_{12} and b_{12}
  - increases the precision of the algorithms
- Not all data points are useful; every data point results in two constraints for the relative drift and offset
- Tiny-sync:
  - maintains only four of these constraints
  - whenever new data point available, the current 4 and the 2 new constraints are compared and only the 4 that result in best estimates are kept
  - disadvantage: constraints that may provide better estimates if combined with other data points that have yet to occur may be eliminated
- Mini-sync:
  - only discards a data point if it is certain that this point will be useless
  - larger computational and storage costs, but increased precision