Time Synchronization, Localization and Positioning

Lecture 13
EE 493/593
Wireless Sensor Networks

Outline
- Goals and Tasks
- Time Synchronization
- Sensor Nodes Localization & Positioning
- Typical Strategies
Goals and Tasks

- **Time Synchronization Among Sensor Nodes**
  - Transmitter/Receiver Synchronization
  - Receiver/Receiver Synchronization
- **Localization & Positioning of Sensor Nodes**
  - Anchor nodes that know their position
  - Directly adjacent
  - Over multiple hops
The Role of Time in WSNs

- Time synchronization algorithms can be used to better synchronize clocks of sensor nodes.
- Time synchronization is needed for WSN applications and protocols:
  - Applications: AOA estimation, beamforming
  - Protocols: TDMA, protocols with coordinated wakeup, ...
  - Distributed debugging: timestamping of distributed events is needed to figure out their correct order of appearance
- WSN have a direct coupling to the physical world, hence their notion of time should be related to physical time:
  - physical time = wall clock time, real-time, i.e. one second of a WSN clock should be close to one second of real time
  - Commonly agreed time scale for real time is UTC, generated from atomic clocks and modified by insertion of leap seconds to keep in synch with astronomical timescales (one rotation of earth)
  - Other concept: logical time (Lamport), where only the relative ordering of events counts but not their relation to real time

Clocks in WSN Nodes

- **Hardware clock**
  - An oscillator generates pulses at a fixed nominal frequency
  - A counter register is incremented after a fixed number of pulses
    - Only register content is available to software
    - Register change rate gives achievable time resolution
  - Node i's register value at real time t is $H_i(t)$
    - Convention: small letters (like $t, t'$) denote real physical times, capital letters denote timestamps or anything else visible to nodes

- **Software clock**
  - $L_i(t) = \theta_i H_i(t) + \phi_i$ (not considering overruns of the counter-register)
  - $\theta_i$ is the (drift) rate, $\phi_i$ the phase shift
  - Time synchronization algorithms modify $\theta_i$ and $\phi_i$, but not the counter register
Synchronization Accuracy / Agreement

- **External synchronization:**
  - Synchronization with external real time scale like UTC
  - Nodes $i=1, ..., n$ are accurate at time $t$ within bound $\delta$ when $|L_i(t) - t| < \delta$ for all $i$
  - Hence, at least one node must have access to the external time scale

- **Internal synchronization**
  - No external timescale, nodes must agree on common time
  - Nodes $i=1, ..., n$ agree on time within bound $\delta$ when $|L_i(t) - L_j(t)| < \delta$ for all $i,j$

Sources of Inaccuracies

- Nodes are switched on at random times, phases $\theta_i$ hence can be random
- Actual oscillators have random deviations from nominal frequency (drift, skew)
  - Deviations are specified in ppm (pulses per million), the ppm value counts the additional pulses or lost pulses over the time of one million pulses at nominal rate
  - The cheaper the oscillators, the larger the average deviation
    - For sensor nodes values between 1 ppm (one second every 11 days) and 100 ppm (one second every 2.8 hours) are assumed, Berkeley motes have an average drift of 40 ppm
- Oscillator frequency depends on time (oscillator aging) and environment (temperature, pressure, supply voltage, ...)
  - Especially the time-dependent drift rates call for frequent re-synchronization, as one-time synchronization is not sufficient
  - However, stability over tens of minutes is often a reasonable assumption
General Properties of Time Synchronization Algorithms

- Physical time vs. logical time
- External vs. internal synchronization
- Global vs. local algorithms
- Absolute vs. relative time
- Hardware vs. software-based mechanisms
  - A GPS receiver would be a hardware solution, but often too heavyweight/costly/energy-consuming in WSN nodes, and in addition a line-of-sight to at least four satellites is required
- A-priori vs. a-posteriori synchronization
  - Is time synchronization achieved before or after an interesting event?
    - Post-facto synchronization
- Deterministic vs. stochastic precision bounds
- Local clock update discipline
  - Avoid backward jumps of local clocks?
  - Avoid sudden jumps?

Performance Metrics and Fundamental Structure

- Metrics:
  - Precision: maximum synchronization error for deterministic algorithms, error mean / stddev / quantiles for stochastic ones
  - Energy costs, e.g. # of exchanged packets, computational costs
  - Memory requirements
  - Fault tolerance: what happens when nodes die?
- Fundamental building blocks of time synchronization algorithms:
  - Resynchronization event detection block: when to trigger a time synchronization round? Periodically? After external event?
  - Remote clock estimation block: figuring out the other nodes clocks with the help of exchanging packets
  - Clock correction block: compute adjustments for own local clock based on estimated clocks of other nodes
  - Synchronization mesh setup block: figure out which node synchronizes with which other nodes
Constraints for Time Synchronization in WSNs

- An algorithm should scale to large networks of unreliable nodes
- Quite diverse precision requirements, from ms to tens of seconds
- Use of extra hardware (like GPS receivers) is mostly not an option
- Low mobility
- Often there are no fixed upper bounds on packet delivery times (due to MAC delays, buffering, ...)
- Negligible propagation delay between neighboring nodes
- Manual node configuration is not an option

Post-Facto Synchronization

- Basic idea:
  - Keeping nodes synchronized all the time incurs substantial energy costs due to need for frequent resynchronization
    - Especially true for networks which become active only rarely
  - When a node observes an external event at time t, it stores its local timestamp $L_i(t)$, achieves synchronization with neighbor node / sink node and converts $L_i(t)$ accordingly
  - Can be implemented in different ways, to be discussed later
Protocols based on Sender/Receiver Synchronization

- In this kind of protocols, a receiver synchronizes to the clock of a sender
- We have to consider two steps:
  - Pairwise synchronization: how does a single receiver synchronize to a single sender?
  - Networkwide synchronization: how to figure out who synchronizes with whom to keep the whole network / parts of it synchronized?

LTS – Pairwise Synchronization

- Trigger resynchronization
- Format synch packet
- Timestamp packet with Hand over packet for transmission
- Operating System, channel access
- Start packet transmission

- Propagation delay
- Packet transmission time
- Packet reception interrupt
- Timestamp with Hand over packet for transmission
- Operating System, channel access
- Start packet transmission

- t0
- t1
- t2
- t3
- t4
LTS – Pairwise Synchronization

- Under the assumption that the remaining uncertainty is allocated equally to both i and j, node i can estimate $L_j(t_5)$ as

$$L_i(t_5) = \frac{L_i(t_1) + \tau + t_p + L_i(t_8) - \tau - t_p - (L_j(t_6) - L_j(t_5))}{2}$$

This means:

$$0 = \Delta(t_5) = L_i(t_5) - L_j(t_5) = \frac{L_i(t_8) + L_i(t_1) - L_j(t_6) - L_j(t_5)}{2}$$

LTS – Networkwide Synchronization

- We are given one reference node R, to which all other nodes / a subset of nodes want to synchronize
  - R’s direct neighbors (level-1 neighbors) synchronize with R
  - Two-hop (level-2) neighbors synchronize with level-1 neighbors
  - ....
- This way a spanning tree is created
- But one should not allow arbitrary spanning trees:
  - Consider a node i with hop distance $h_i$ to the root node
  - Assume that:
    - all synchronization errors are independent
    - all synch errors are identically normally distributed with zero mean and variance $4\sigma^2$
  - Then node i’s synchronization error is a zero-mean normal rv with variance $h_i \cdot 4\sigma^2$
  - Hence, a minimum spanning tree minimizes synchronization errors
LTS – Centralized Multihop LTS

- Reference node R triggers construction of a spanning tree, it first synchronizes its neighbors, then the first-level neighbors synchronize second-level neighbors and so on.
- Different distributed algorithms for construction of spanning tree can be used, e.g. DDFS, Echo algorithm.
- Communication costs:
  - Costs for construction of spanning tree
  - Synchronizing two nodes costs 3 packets, synchronizing n nodes costs 3n packets

LTS – Distributed Multihop LTS

- No explicit construction of spanning tree needed, but each node knows identity of reference node(s) and routes to them.
- When node 1 wants to synchronize with R, an appropriate request travels to R - following this, 4 synchronizes to R, 3 synchronizes to 4, 2 synchronizes to 3, 1 synchronizes to 2.
  - By-product: nodes 2, 3, and 4 are synchronized with R.
- Minimum spanning tree constructed implicitly: node 1 should know shortest route to the closest reference node.
LTS – Distributed Multihop

LTS -- Variations

- When node 5 wants to synchronize with R, it can:
  - issue its own synchronization request using route over 3, 4 and put additional synchronization burden on them
  - ask in its local neighborhood whether someone is synchronized or has an ongoing synchronization request and benefit from that later on
  - Enforce usage of path over 7, 8 (path diversification) to also synchronize these nodes
- Discussion:
  - Simulation shows that distributed multihop LTS needs more packets (between 40% and 100%) when all nodes have to be synchronized, even with optimizations
  - Distributed multihop LTS allows to synchronize only the minimally required set of nodes → post-facto synchronization

Protocols Based on Receiver/Receiver Synchronization

- In this class of schemes the receivers of packets synchronize among each other, not with the transmitter of the packet
- RBS = Reference Broadcast Synchronization (Elson et al)
- RBS has two components:
  - Synchronize receivers within a single broadcast domain
  - A scheme for relating timestamps between nodes in different domains
- RBS does not modify the local clocks of nodes, but computes a table of conversion parameters for each peer in a broadcast domain
- RBS allows for post-facto synchronization
Summary

- Time synchronization is important for both WSN applications and protocols
- Using hardware like GPS receivers is typically not an option, so extra protocols are needed
- Post-facto synchronization allows for time-synchronization on demand, otherwise clock drifts would require frequent re-synchronization and thus a constant energy drain
- Some of the presented protocols take significant advantage of WSN peculiarities like:
  - small propagation delays
  - the ability to influence the node firmware to timestamp outgoing packets late, incoming packets early
- Of course, there are many, many more schemes ....

Localization & Positioning

- Determine **physical position** or **logical location**
  - Coordinate system or symbolic reference
  - Absolute or relative coordinates
- Options
  - Centralized or distributed computation
  - Scale (indoors, outdoors, global, ...)
  - Sources of information
- Metrics
  - Accuracy (how close is an estimated position to the real position?)
  - Precision (for repeated position determinations, how often is a given accuracy achieved?)
  - Costs, energy consumption, ...
Main Approaches (information sources)

- Proximity
  - Exploit finite range of wireless communication
  - E.g.: easy to determine location in a room with infrared room number announcements
- (Tri-/Multi-) lateration and angulation
  - Use distance or angle estimates, simple geometry to compute position estimates
- Scene analysis
  - Radio environment has characteristic “signatures”
  - Can be measured beforehand, stored, compared with current situation

Estimating Distances – RSSI

- Received Signal Strength Indicator
  - Send out signal of known strength, use received signal strength and path loss coefficient to estimate distance

\[
P_{\text{recv}} = c \frac{P_{\text{tx}}}{d^\alpha} \Leftrightarrow d = \sqrt[\alpha]{\frac{cP_{\text{tx}}}{P_{\text{recv}}}}
\]
Estimating Distances – Other Means

- Time of arrival (ToA)
  - Use time of transmission, propagation speed, time of arrival to compute distance
  - Problem: Exact time synchronization

- Time Difference of Arrival (TDoA)
  - Use two different signals with different propagation speeds
  - Example: ultrasound and radio signal
    - Propagation time of radio negligible compared to ultrasound
  - Compute difference between arrival times to compute distance
  - Problem: Calibration, expensive/energy-intensive hardware

Determining Angles

- Directional antennas
  - On the node
  - Mechanically rotating or electrically “steerable”
  - On several access points
    - Rotating at different offsets
    - Time between beacons allows to compute angles
Some Range-free, Single-hop Localization Techniques

- **Overlapping connectivity**: Position is estimated in the center of area where circles from which signal is heard/not heard overlap.

- **Approximate point in triangle**
  - Determine triangles of anchor nodes where node is inside, overlap them.
  - Check whether inside a given triangle - move node or simulate movement by asking neighbors.
  - Only approximately correct.

Trilateration

- Assuming distances to three points with known location are exactly given.
- Solve system of equations:
  - $(x_i, y_i)$: coordinates of anchor point $i$, $r_i$ distance to anchor $i$.
  - $(x_u, y_u)$: unknown coordinates of node.
  
  $$(x_i - x_u)^2 + (y_i - y_u)^2 = r_i^2$$ for $i = 1, \ldots, 3$.

  $$2(x_3 - x_1)x_u + 2(y_3 - y_1)y_u = (r_1^2 - r_3^2) - (x_1^2 - x_3^2) - (y_1^2 - y_3^2)$$

  $$2(x_3 - x_2)x_u + 2(y_3 - y_2)y_u = (r_2^2 - r_3^2) - (x_2^2 - x_3^2) - (y_2^2 - y_3^2)$$
**Multihop Range Estimation**

- How to estimate range to a node to which no direct radio communication exists?
  - No RSSI, TDoA, ...
  - But: Multihop communication is possible

- Idea 1: Count number of hops, assume length of one hop is known (*DV-Hop*).
  - Start by counting hops between anchors, divide known distance

- Idea 2: If range estimates between neighbors exist, use them to improve total length of route estimation in previous method (*DV-Distance*).

---

**Iterative Multilateration**

- Assume some nodes can hear at least three anchors (to perform triangulation), but not all.
- Idea: let more and more nodes compute position estimates, spread position knowledge in the network.
- Problem: Errors accumulate.
Conclusions

- Determining location or position is a vitally important function in WSN, but fraught with many errors and shortcomings
  - Range estimates often not sufficiently accurate
  - Many anchors are needed for acceptable results
  - Anchors might need external position sources (GPS)
  - Multilateration problematic (convergence, accuracy)