8. Time synchronization

Protocols of time synchronization in wireless sensor networks

Wireless sensor networks

Martin Úbl
ublm@kiv.zcu.cz

2023/24

<DCSE> DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Time

Initial remarks

- time is maintained by an oscillator
  - crystal-based
  - RLC-based
  - ...
- every oscillator oscillates a bit differently
  - construction differences
  - material properties
  - temperature
  - other influences
- *every node has a different perception of time*

Time

Initial remarks

- every oscillator ticks correctly from the perspective of its owner
- potentially, any node is a time-reference node
- every oscillator maintains an internal **clock**
  - clock is an increasing counter, incremented with frequency proportional to oscillator frequency
- errors:
  - *offset* – absolute difference against reference clock
  - *skew* – absolute difference against reference frequency
  - *drift* – speed of *skew* change (second derivation)
- we would like to eliminate all the errors, or at least minimize their influence

Time

Initial remarks

- why do we need time (clock) synchronization?
  - scheduling – mainly TDMA
    - synchronous TDMA is much more energy efficient, if done correctly
  - event scheduling
  - tracking (time, location, ...)
  - detection of duplicate messages
  - event and message ordering

Time

Initial remarks

- how do we synchronize in "big world"?
- NTP
  - Network Time Protocol
  - precise reference clock
  - network delay must be symmetrical and deterministic
  - delays must be reasonably small
  - unsuitable for WSN
- GPS
  - Global Positioning System
  - dependent on satellite visibility
  - energy consuming, inefficient
  - may take a long time (even minutes)
  - cannot be used indoors
  - unsuitable for WSN
- we must consider a different approach for WSN

Time

Initial remarks

- do we really need to synchronize clocks?
- we often don’t need the absolute time to be synchronized
- in most cases, we can rely on relative timestamps
  - when scheduling e.g., the next wakeup time for TDMA between two adjacent nodes
- this may eliminate the *offset* error
- however... some applications need absolute time synchronization
  - for example, when all nodes in network needs to wake up at the same time (e.g., to listen to a broadcast slot in TDMA)
  - they basically synchronize their offsets

Time

Initial remarks

- we always need to synchronize frequencies
  - i.e., eliminate skew and drift
- possible only to a certain degree
- to get rid of skew, we can e.g., use:
  - phase-locked loop (PLL)
  - tick skipping
  - secondary fractional timer
- drift is a difficult error to get rid of
- $\rightarrow$ we need to synchronize time continuously (periodically)

Time

Synchronization

- we may synchronize:
  - absolute time
    - globally
    - within a cluster (master-slave)
    - between two adjacent nodes (peer-to-peer)
  - relative time
    - within a cluster (master-slaves)
    - between two adjacent nodes (peer-to-peer)
  - external absolute time
    - beacon, "time authority" broadcasts the time
  - we may choose not to synchronize time
    - sometimes it is not possible or required
    - e.g., asynchronous TDMA or other CSMA methods

Time

Synchronization

- *event-based synchronization*
- continuous synchronization is expensive
- the core idea is to synchronize only if we require so
- used mainly for logical timestamp
- user for event ordering inside the WSN

Time

Synchronization

- when synchronizing time, even the most precise algorithm must account for the following delays:
  - message serialization delay
  - medium access delay – time to obtain permission to transmit
  - transmission delay – time of the actual transmission (can be calculated from data length and transmission rate)
  - travel time – time between transmission and reception
  - reception delay – time of the actual reception (same as transmission delay)
  - processing time

Time

Synchronization

- to synchronize time, we need a *time synchronization protocol*
- base requirements:
  - accuracy
  - energy efficiency
  - scalability (low dependency on the number of nodes)
  - low complexity
  - robustness
  - spread (local, cluster-local, global)
  - delay for synchronization

Time synchronization

Protocols

- Reference Broadcast Synchronization (RBS)
- Time-Diffusion Synchronization Protocol (TDSP)
- Timing-sync Protocol for Sensor Networks (TPSN)
- Lightweight Time Synchronization (LTS)
- Flooding Time Synchronization Protocol (FTSP)
- etc.

Time synchronization

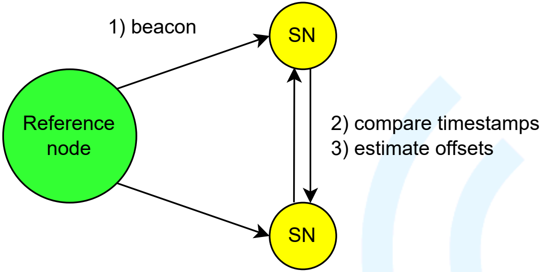
RBS

- **Reference Broadcast Synchronization** (RBS)
- there is a beacon node, that holds reference time for the whole network
- it periodically broadcasts the reference time
- all nodes store their reception timestamps
- then, nodes exchange their timestamps
  - for first few turns, they synchronize offsets
  - then, they try to eliminate skew and drift
  - *receiver-to-receiver synchronization*
- the only delay, that cannot be predicted – travel time
  - it must be estimated based on known topology or localization

Time synchronization

RBS

- example flow



Time synchronization

RBS

- advantages
  - simple
  - usable in wide variety of topologies
  - can be used in wired and wireless networks
  - most delays are eliminated implicitly
  - optimal for single-hop networks (reference node sees all nodes)

Time synchronization

RBS

- disadvantages
  - not so straight-forward modification to support multi-hop networks
  - nondeterministic transfer delay
  - in some topologies exhibits too large errors
  - reference node must always function as time source
    - it must either have a precise time source
    - or use NTP/GPS to obtain it
  - large synchronization delay (lots of messages before the times are reasonably close)

Time synchronization

TDSP

- **Time-Diffusion Synchronization Protocol** (TDSP)
- automatic organization to a tree-like structure
- works in two modes:
  - with precise time source – i.e., a time source synchronized to a NTP/GPS
    - goal: global synchronization
  - without precise time source – no connection to outer world
    - goal: global synchronization or local clustered synchronization
- relies on a presence of "time authorities"
  - e.g., the sink node may serve as one
- basic idea from RBS, but supports multi-hop networks

Time synchronization

TDSP

- there is a *master node*
- nodes randomly select *leader nodes*
- network is organized to a tree – each tree node represents a cluster of network nodes
- master node is a root of the tree
- leader nodes are responsible for synchronization with their parent cluster master nodes or root node
- algorithm:
    1. distributed leader node election
    2. faulty nodes identification (to avoid parasitic info)
    3. load balancing
    4. peer node evaluation
    5. time diffusion
    6. clock synchronization

Time synchronization

TDSP

- advantages:
  - multi-hop networks support
  - supports dynamic topologies (leaders are re-elected each time)
  - works well even without NTP/GPS precise time
  - packet loss-tolerant

Time synchronization

TDSP

- disadvantages:
  - energy consuming
  - highly complex
  - tends to drain much more energy on fully-connected mesh topologies
    - for which the RBS performs slightly better

Time synchronization

TPSN

- **Timing-sync Protocol for Sensor Networks** (TPSN)
- hierarchical protocol
- any node can trigger synchronization
  - usually it is a node with precise time (NTP/GPS)
- two-phase
  1. spanning tree calculation – based on number of hops from root to each node, a tree is created
  2. pair-wise synchronization

Time synchronization

TPSN

- *spanning tree calculation phase*
- determine topology by e.g., flooding
- each node is assigned to a level *n* based on the number of hops from root
- algorithm is repeated until all nodes have its level assigned
- this phase may be very energy-consuming
  - we benefit from static topologies by remembering the assignment
  - next time the algorithm runs, we may skip this phase

Time synchronization

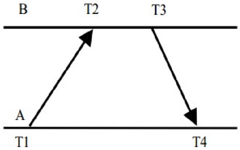
TPSN

- *pair-wise synchronization phase*
- nodes of level n are synchronized by nodes of level $n - 1$
- at first, root node ($n = 0$) synchronizes next-level nodes ($n = 1$)
- then, nodes of level $n = 1$ synchronize nodes of level $n = 2$, etc.
- all nodes use standard Cristian's algorithm

Time synchronization

TPSN

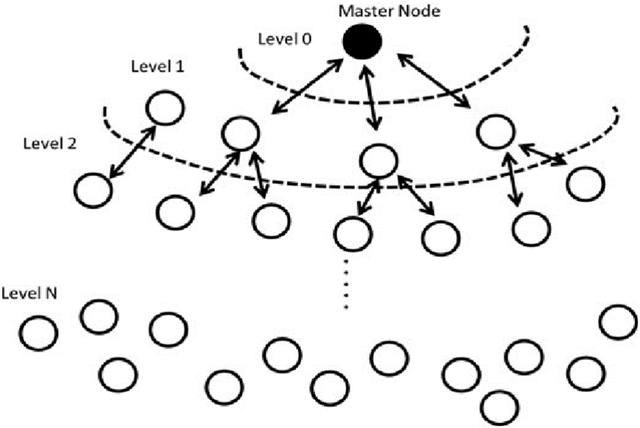
- *Cristian's algorithm*
- A – node being synchronized, B – time source
- T1 – A requests time from B
- T2 – B receives request and processes
- T3 – B sends its timestamp to A
- T4 – A synchronizes by B
- $T2 = T1 + \Delta + d$
  - Δ – clock offset
  - d – communication delay
- $T4 = T3 - \Delta + d$
- therefore, $\Delta = \frac{(T2-T1)-(T4-T3)}{2}$ and $d = \frac{(T2-T1)+(T4-T3)}{2}$



Time synchronization

TPSN

- Depiction of network structure



Time synchronization

TPSN

- advantages
  - built for multi-hop large-scale networks
  - relatively fast
  - accurate (up to tens of μs between levels)
- disadvantage
  - cannot effectively deal with dynamic topologies
  - unbalanced energy consumption – some nodes are depleted much faster

Time synchronization

LTS

- **Lightweight Time Synchronization** (LTS)
- focused not on accuracy, but to simplicity and speed
- assumes existence of time sources with precise time (NTP/GPS)
- nodes may synchronize:
  - with adjacent nodes (pair-wise)
  - with network nodes (multi-hop)

Time synchronization

LTS

- *pair-wise synchronization*
- nodes synchronize with adjacent nodes
- they use Cristian's algorithm
- potentially very accurate

Time synchronization

LTS

- *multi-hop synchronization*
- at first, a minimum spanning tree is constructed
  - using breadth-first search (BFS)
  - or distributed depth-first search (DFS)
  - or "Echo" algorithm
- minimum spanning tree – has the lowest depth possible
- lowest depth = shortest run-time of the algorithm = most precise synchronization
- after the spanning tree is constructed, pair-wise synchronization between different levels occurs (as in TPSN)

Time synchronization

FTSP

- **Flooding Time Synchronization Protocol** (FTSP)
- local synchronization protocol
- multi-hop protocol
- nodes form an ad-hoc structure (not a tree)
- network elects a root node, which holds the time source (must not be NTP/GPS sync'd)

Time synchronization

FTSP

- as the structure does not form a tree, the algorithm is robust
- supports dynamic topologies
- however, it requires a few things:
  - each node has an ID
  - root node has the lowest ID
  - no two nodes share the same ID
- nodes synchronize by propagating the message containing:
  - timestamp
  - root ID
  - sequence number

Time synchronization

FTSP

- phases:
  - *root election*
  - *reconfiguration*
  - *synchronization*

Time synchronization

FTSP

- *root election*
- *1 of N* algorithm
- distributed election based on the lowest ID assigned
- the ID may change between runs

Time synchronization

FTSP

- *reconfiguration*
- if the node does not obtain any SYNC message, it sets itself into the root role
- eventually, new root is elected if the original one failed

Time synchronization

FTSP

- *synchronization phase*
- all nodes that are synchronized broadcasts the SYNC message (Cristian's algorithm)
  - initially, only root considers himself synchronized
  - over time, more and more nodes become synchronized
- nodes receive multiple SYNC messages
- after some time (and after multiple messages), they are able to calculate the correct time
  - e.g., using a simple form of linear regression

Time synchronization

FTSP

- advantages
  - fairly simple
  - very accurate
  - multi-hop synchronization
  - robust and scalable
- disadvantages
  - requires initialization phase
  - significant traffic (basically flooding)

Time synchronization

Final remarks

- time synchronization is important
- there are lots of delay sources
- we usually want to implement time synchronization on the **data link layer** to minimize delays
- essential for synchronous TDMA

Time synchronization

Final remarks

- how to use synchronous TDMA, when no synchronization has occurred yet?
- split operation to two phases:
  1. asynchronous TDMA (preamble sampling, ...) phase
     - newly connected node searches for adjacent nodes to synchronize
     - upon synchronization, it switches to phase 2
  2. synchronous TDMA
     - node has synchronized successfully
     - future operation uses synchronous mode
- if the node fails or loses synchronization, it may always revert back to phase 1 and resynchronize when needed