CLOCK SYNCHRONIZATION

INTRODUCTION

In wireless sensor networks, each node has its own clock and its own notion of time. However, a common time scale or reference among sensor nodes is important to identify causal relationships between events in the physical world, to support the elimination of redundant sensor data, and to generally facilitate sensor network operation.

Since each node in a sensor network operates independently and relies on its own clock, the clock readings of different sensor nodes will also differ.

In addition to these random differences (phase shifts), the gap between clocks of different sensors will further increase due to the varying drift rates of oscillators.

Therefore, time (or clock) synchronization is required to ensure that sensing times can be compared in a meaningful way.

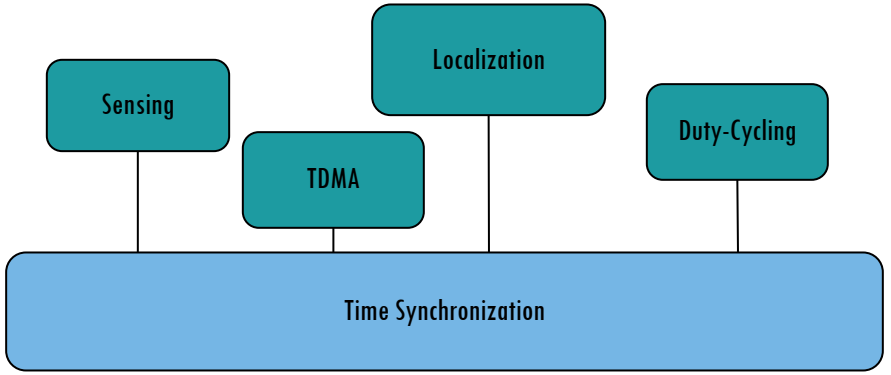
While time synchronization techniques for wired networks have received a significant amount of attention, these techniques are unsuitable for wireless sensors because of the unique challenges posed by wireless sensing environments.

These challenges include:

- ✓ the potentially large scale of wireless sensor networks,
- ✓ the necessity for self-configuration and robustness,
- ✓ the potential for sensor mobility, and
- ✓ the need for energy conservation.

Motivation

- Synchronizing time is essential for many applications
  - Coordination of wake-up and sleeping times (energy efficiency)
  - TDMA schedules
  - Ordering of collected data/events
  - Co-operation of multiple sensor nodes
  - Estimation of position information (e.g. shooter detection)
  - Data Diffusion: Merging individual sensor readings into a high level sensing result
  - Coordinated Actuation
  - Object Tracking
  - Many of the applications of sensor need the event with time stamp.
  - Events are reported by multiple nodes.
  - When sensor is energy save enabled, it need all nodes to be in sync in order to be in Idle or Active mode.
  - Medium Access Scheduling
  - Secure cryptographic schemes
  - Coordination of future actions



- Goals of clock synchronization
  - Compensate *offset** between clocks
  - Compensate *drift** between clocks

*terms are explained on following slides

Properties of Clock Synchronization Algorithms

- External versus internal synchronization
  - External sync: Nodes synchronize with an external clock source (UTC)
  - Internal sync: Nodes synchronize to a common time
    - to a leader, to an averaged time, or to anything else
- One-shot versus continuous synchronization
  - Periodic synchronization required to compensate clock drift
- A-priori versus a-posteriori
  - A-posteriori clock synchronization triggered by an event
- Global versus local synchronization
- Accuracy versus convergence time, Byzantine nodes, …

CLOCK SOURCES

Radio Clock Signal:

- Clock signal from a reference source (atomic clock) is transmitted over a long wave radio signal
- DCF77 station near Frankfurt, Germany transmits at 77.5 kHz with a transmission range of up to 2000 km
- Accuracy limited by the distance to the sender, Frankfurt–Zurich is about 1ms.
- Special antenna/receiver hardware required

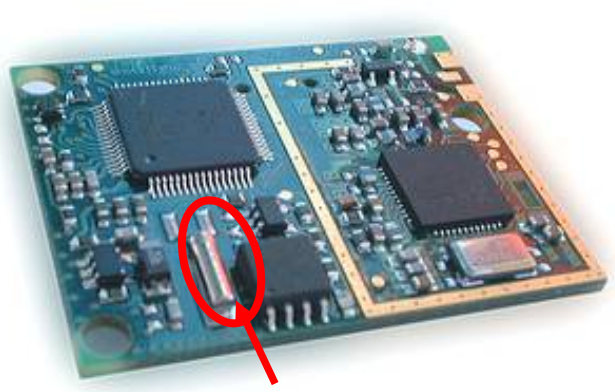
Global Positioning System (GPS):

- Satellites continuously transmit own position and time code
- Line of sight between satellite and receiver required
- Special antenna/receiver hardware required

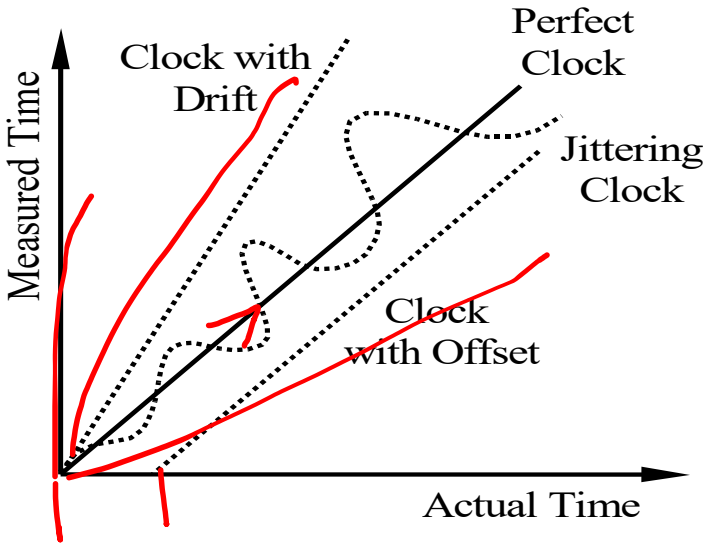
Clock Devices in sensor Nodes

- Structure
  - External oscillator with a nominal frequency (e.g. 32 kHz)
  - Counter register which is incremented with oscillator pulses
  - Works also when CPU is in sleep state
- Accuracy
  - Clock drift: random deviation from the nominal rate dependent on power supply, temperature, etc.
  - E.g. TinyNodes have a maximum drift of 30-50 ppm at room temperature

Platform	System clock	Crystal oscillator
Mica2	7.37 MHz	32 kHz, 7.37 MHz
TinyNode 584	8 MHz	32 kHz
Tmote Sky	8 MHz	32 kHz

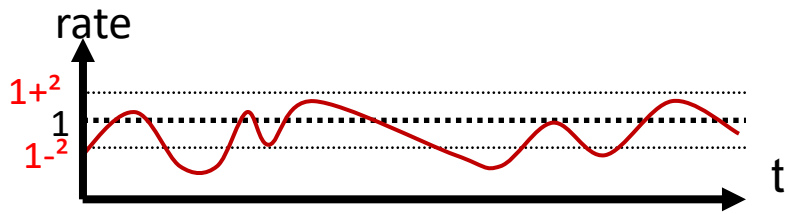


Oscillator



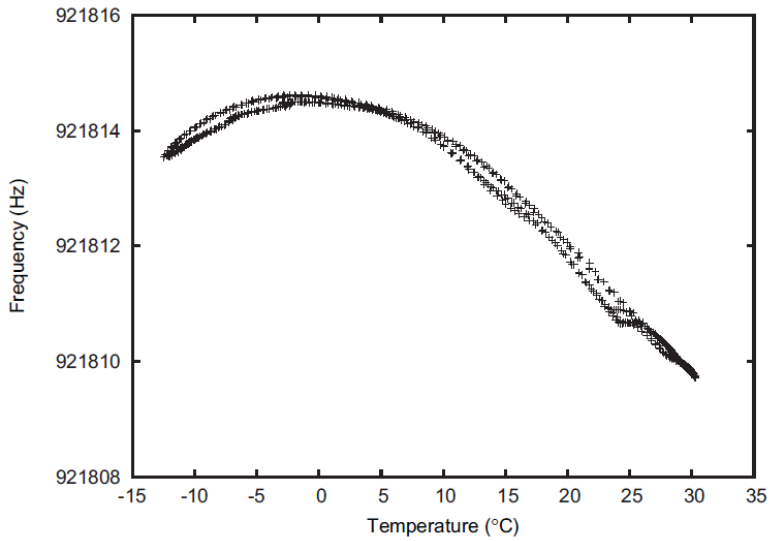
Clock Drift

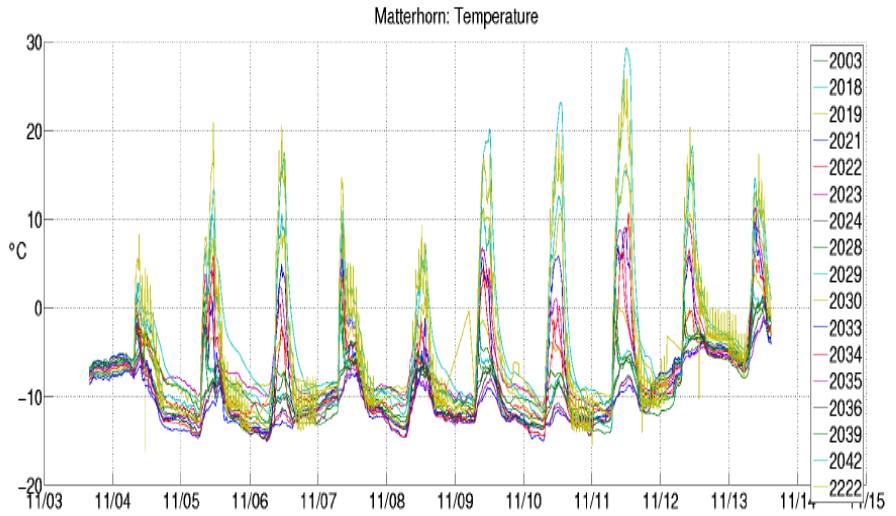
- Accuracy
  - Clock drift: random deviation from the nominal rate dependent on power supply, temperature, etc.



This is a drift of up to 50 µs per second or 0.18s per hour

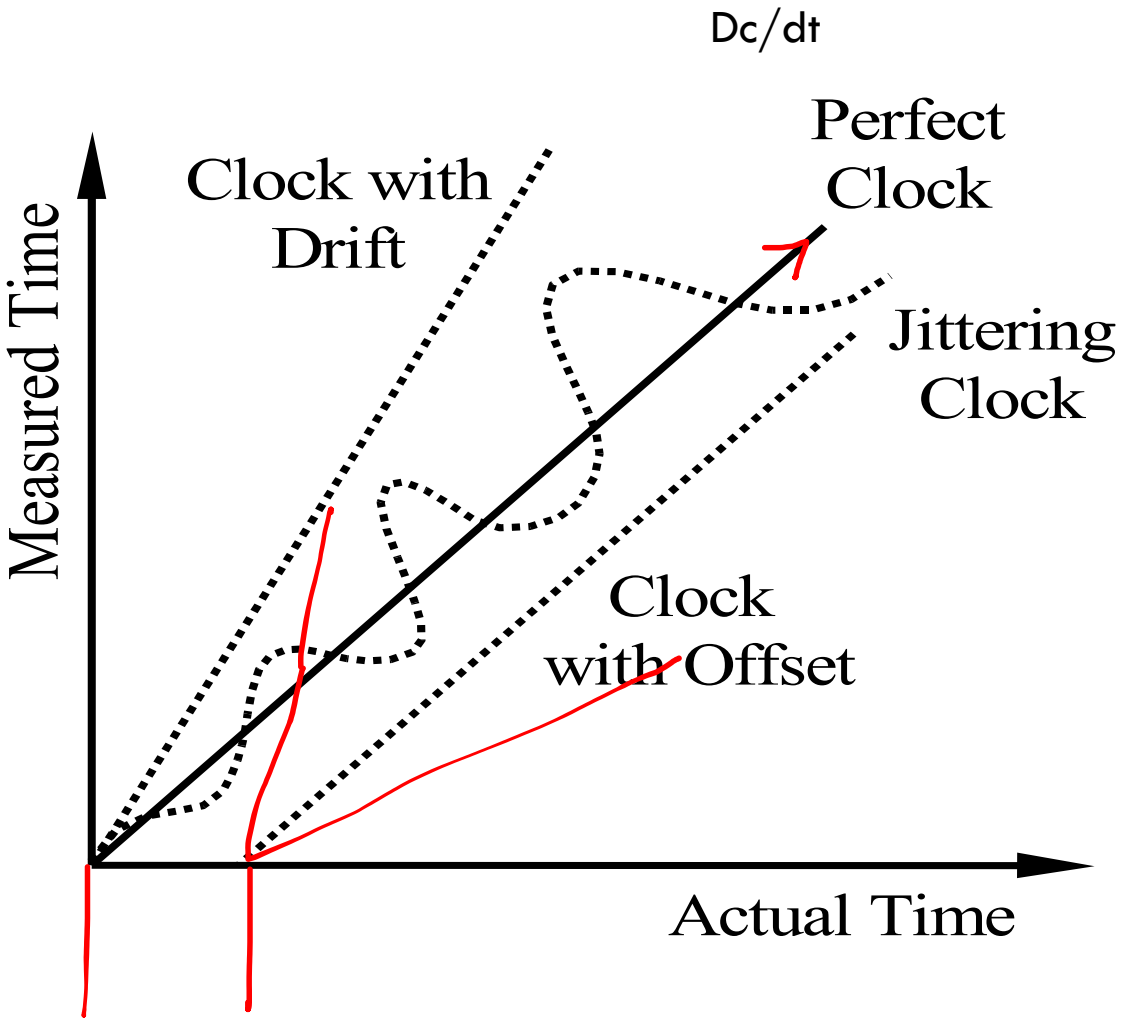
  - E.g. TinyNodes have a maximum drift of $30-50$ ppm at room temperature





Motivation for time synchronization

- Goals of clock synchronization
  - Compensate *offset** between clocks
  - Compensate *drift** between clocks



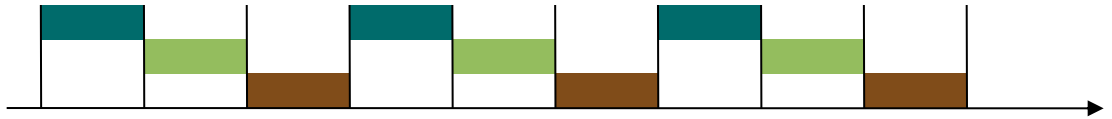
GLOBAL VS. LOCAL TIME SYNCHRONIZATION

Common time is essential for many applications:

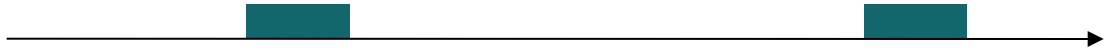
Global
- Assigning a timestamp to a globally sensed event (e.g. earthquake)

Local
- Precise event localization (e.g. shooter detection, multiplayer games)

Local
- TDMA-based MAC layer in wireless networks

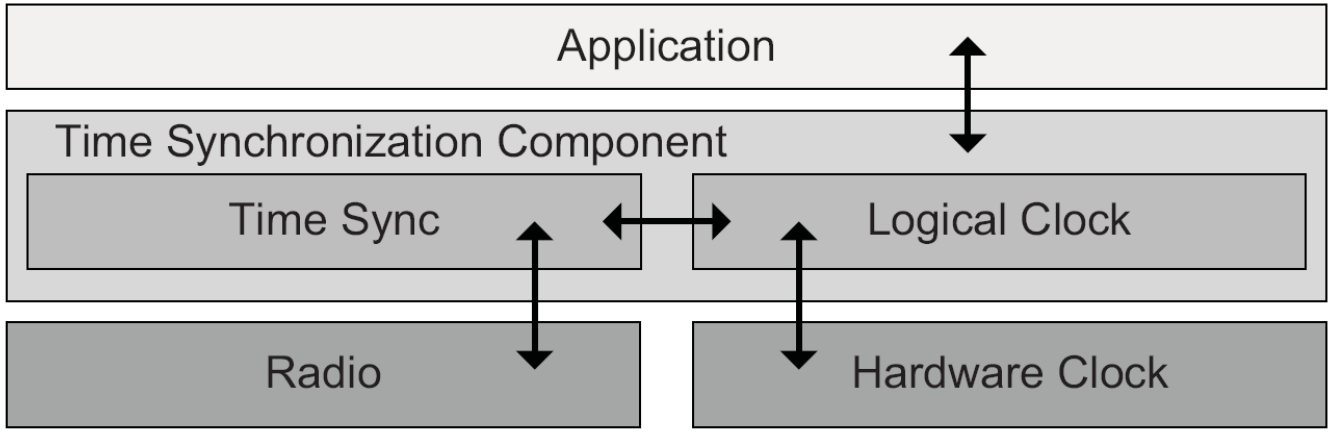


Local
- Coordination of wake-up and sleeping times (energy efficiency)



GENERAL FRAMEWORK

The clock synchronization framework must provide the abstraction of a correct logical time to the application. This logical time is based on the (inaccurate) hardware clock, and calibrated by exchanging messages with other nodes in the network.



AFFECTING FACTORS ON TS SCHEMES

- Communication Overhead
- Implicit Messages v/s Explicit Messages
- Available Bandwidth
- Accuracy Requirements
- Scalability
- Infrastructure Requirement

Source of Synchronization Error

Clock Drift

This deviation results in a *drift rate*, which expresses the rate by which two clocks can drift apart, that is, $dC/dt - 1$.

The maximum drift rate of a clock is expressed as ρ with typical values for quartz-based clocks being $1-100$ ppm (1 ppm = 10^{-6}) per second.

This number is given by the manufacturer of the oscillator and guarantees that

$$1 - \rho \leq \frac{dC}{dt} \leq 1 + \rho$$

Major influence in sensor networks where communication is infrequent.

Communication Message Delay

- Send Time
- Access Time
- Propagation Time

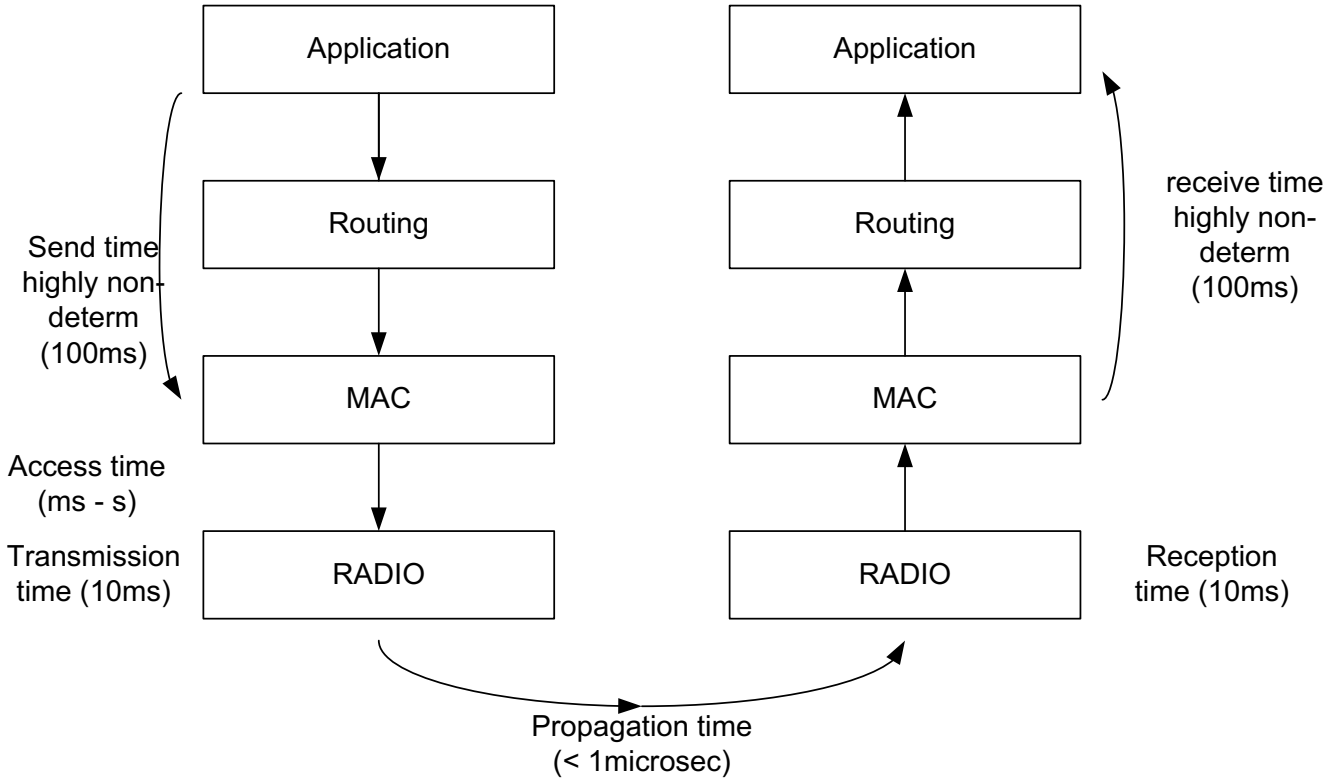
  (Negligible for most of the cases)
- Receive Time

Assuming identical clocks, any two clocks that are synchronized can drift from each other at a rate of at most $2\rho_{max}$. To limit the relative offset to δ seconds, the resynchronization interval τ_{sync} must meet the requirement:

$$\tau_{sync} \leq \frac{\delta}{2\rho_{max}}$$

CLOCK SYNC DELAYS

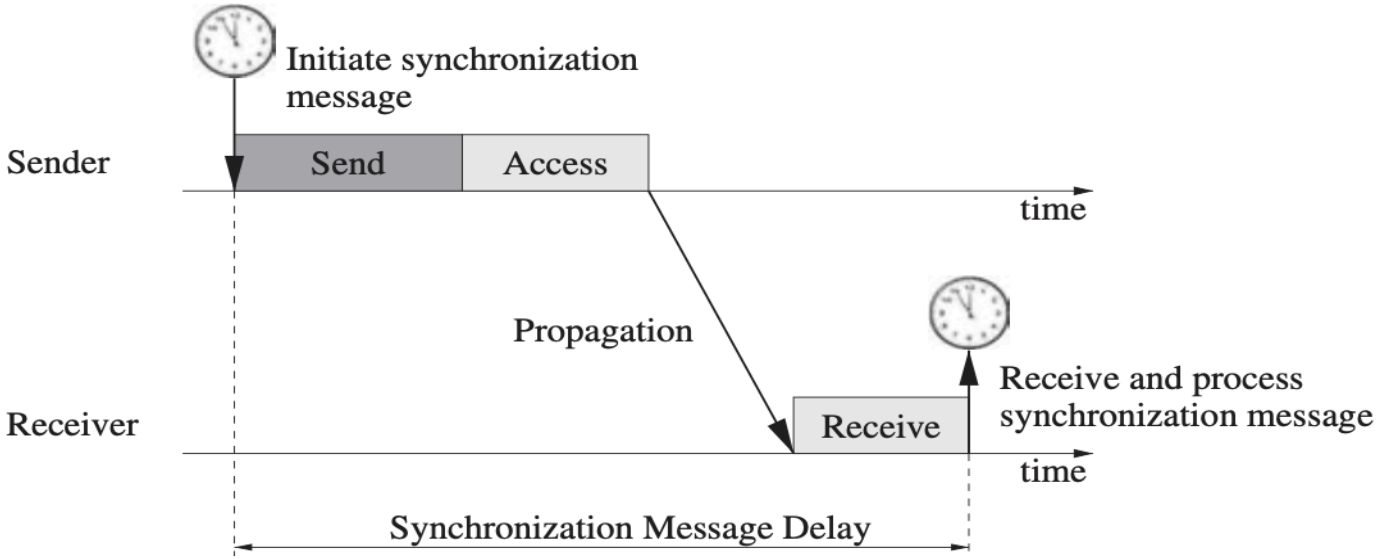
Uncertainties of radio message delivery



Nondeterminism of Communication Latency

The nondeterminism of the communication latency significantly contributes to the precision that can be achieved. In general, this latency experienced by synchronization messages is the sum of several components (Kopetz and Ochsenreiter 1987), as illustrated in Figure 9.5:

1. *Send delay:* This is the time spent by the sender to generate the synchronization message and pass the message to the network interface. This includes delays caused by operating system behavior (system call interface, context switches), the network protocol stack, and the network device driver.
2. *Access delay:* This is the time spent by the sender to access the physical channel and is mostly determined by the medium access control (MAC) protocol in use. Contention-based protocols such as IEEE 802.11's CSMA/CA must wait for an idle channel before



End-to-end delay experienced by a synchronization message.

access is allowed. When multiple devices access the channel at the same time, collisions occur that cause further delays (e.g., through the exponential backoff mechanism used in many MAC protocols). More predictable delays are experienced by protocols based on time-division (TDMA), where a device must wait for its periodic slot before transmission can occur.

3. *Propagation delay:* The actual time needed for the message to travel from the sender to the receiver is called propagation delay. When the nodes share the same physical medium, propagation delays are very small and are often negligible in critical path analysis.
4. *Receive delay:* This is the time spent by the receiver device to receive the message from the medium, to process the message, and to notify the host of its arrival. Host notification typically occurs via interrupts, at which the local time (i.e., the message arrival time) can be read. As a consequence, the receive time tends to be much smaller than the send time.

Many synchronization schemes for WSNs apply low-level techniques aimed at reducing the amount or variation of some of these components. For example, MAC-layer time stamping can reduce the send and receive delays on the sender and receiver, respectively.

TYPES OF CLOCK

Digital Clock

- Changes in Supply Voltage, Temperature can drift the rate
- Constant, Bounded, or Bounded Variation Drifts

Software Clock

- Value of counter converted in Time
- Offset time value can be added/subtracted to achieve time synchronization

COMPUTER CLOCKS

Clocks in computers

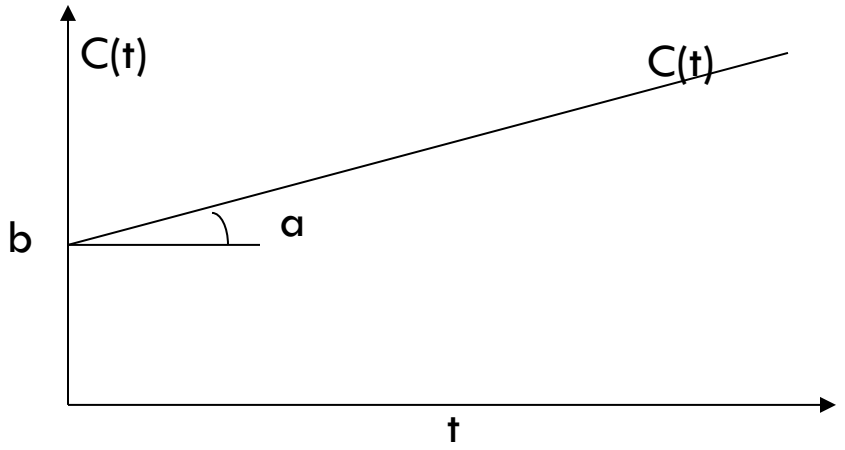
- $C(t)=k\int_0^t \omega(\tau)d\,\tau + C(t_0)$
- ω is frequency of oscillator, $C(t_0)$ clock offset.
- Time of the computer click implemented based on a hardware oscillator

Computer clock is an approximation of a real time t

- $C(t)=a*t+b$
  - a is a clock drift (rate)
  - b is an offset of the clock

Perfect clock

- Rate = 1
- Offset = 0



DEFINITION OF TIME SYNCHRONIZATION

Let $C(t)$ be a perfect clock

- A clock $C_i(t)$ is called correct at time t
  - If $C_i(t)=C(t)$
- A clock $C_i(t)$ is called accurate at time t
  - If $dC_i(t)/dt = dC(t)/dt = 1$
- Two clocks $C_i(t)$ and $C_k(t)$ are synchronized at time t
  - if $C_i(t)=C_k(t)$

Time synchronization

- Requires knowing both offset and drift

DIFFICULTIES IN SENSOR NETWORKS

No periodic message exchange is guaranteed
- There may be no links between two nodes at all

Transmission delay between two nodes is hard to estimate
- The link distance changes all the time

Energy is very limited
- Nodes sleep most of the time to conserve power

Node need to be small and cheap
- No expensive clock circuitry

EXISTING SOLUTIONS

NTP: Network Time Protocol

- Local clocks sync to NTP time servers; external time sources

RBS: Reference Broadcast

- Receivers record receiving time and exchange with other node

TPSN: Time–Sync Protocol for sensor Networks

- Pair–wise synchronization along edges

FTSP: Flooding Time Synchronization Protocol

- MAC layer time stamping

CONCEPT OF TTS- TRADITIONAL TIME SYNCHRONIZATION

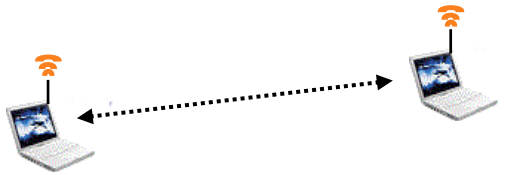
The sender periodically sends a message with its current clock as a timestamp to the receiver

Receiver then synchronizes with the sender by changing its clock to the timestamp of the message it has received from the sender (if the latency is small compared to the desired accuracy)

Sender calculates the phase error by measuring the total round trip−time by sending and receiving the respective response from the receiver (if the latency is large compared to the desired accuracy)

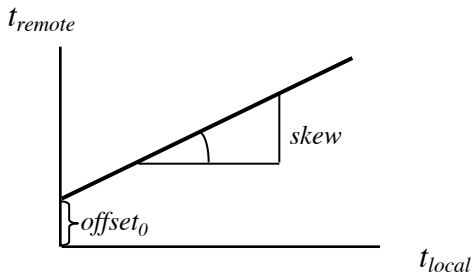
BILATERAL CLOCK SYNCHRONIZATION

Synchronization between two neighbors

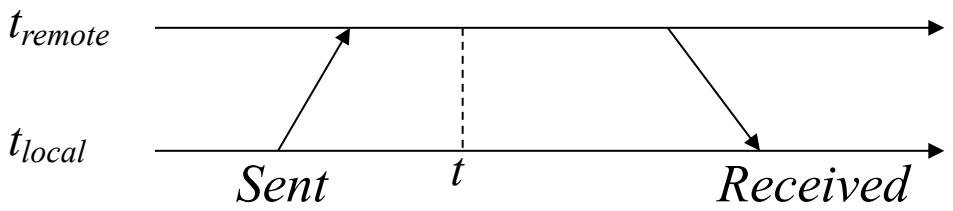


- Linear clock model

$$t_{remote}(t) = \alpha t_{local}(t) + offset_0$$



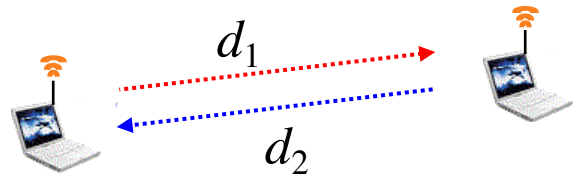
- Propagation delay from i to $j = d_i$



AN IMPOSSIBILITY TASK

Theorem:

- It is impossible to determine through any packet exchanges the four unknown parameters
  - d_1, d_2
  - Offset_0
  - Skew α

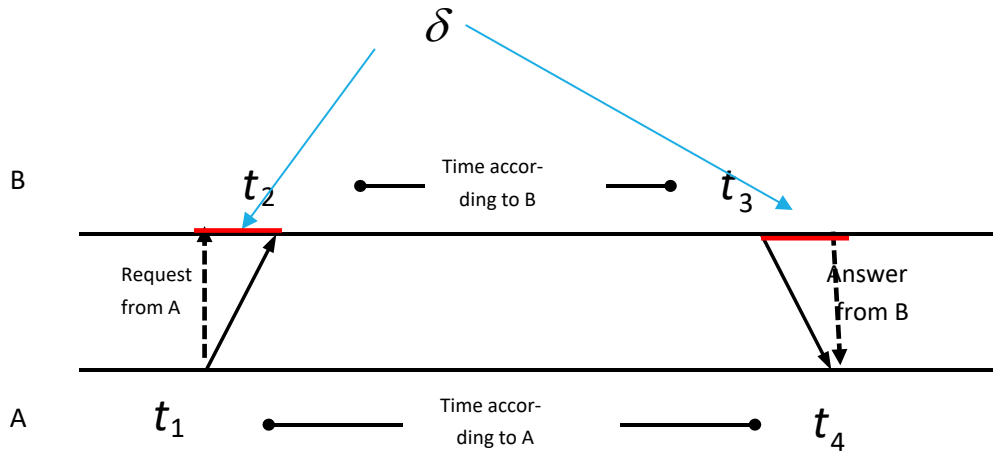


Consequence

- Cannot estimate all four parameters if delays in two directions are asymmetric
- Solution: Assume that delays in both directions are symmetric

Sender/Receiver Synchronization

- Round-Trip Time (RTT) based synchronization

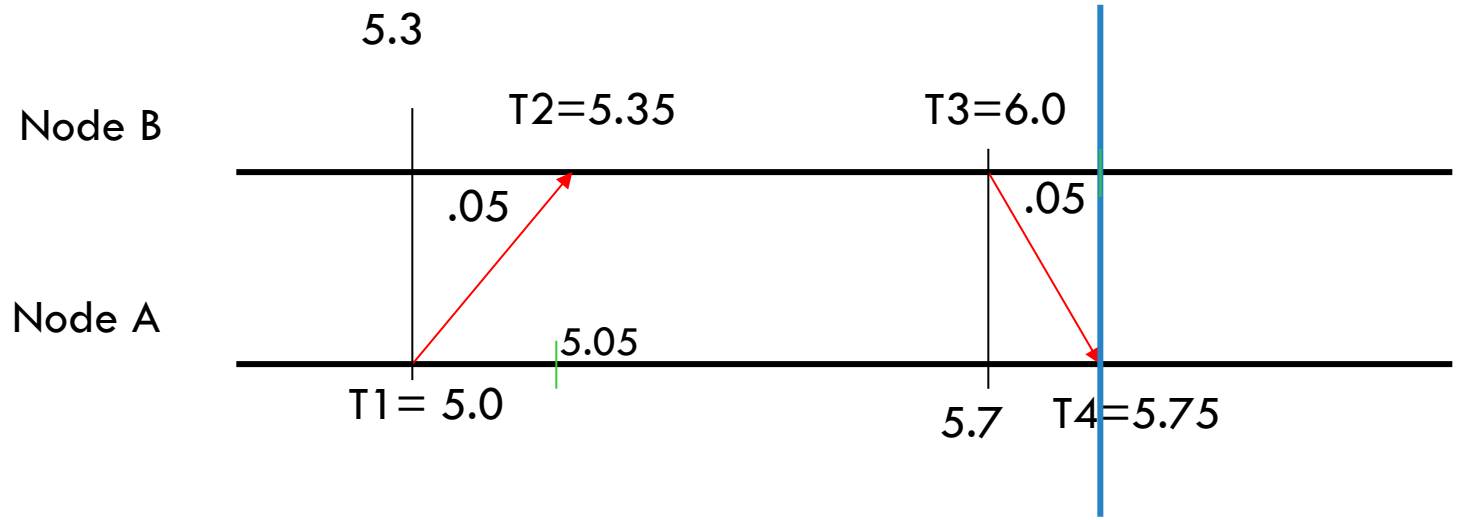


- Receiver synchronizes to the sender‘s clock
- Propagation delay δ and clock offset θ can be calculated

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

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

EXAMPLE



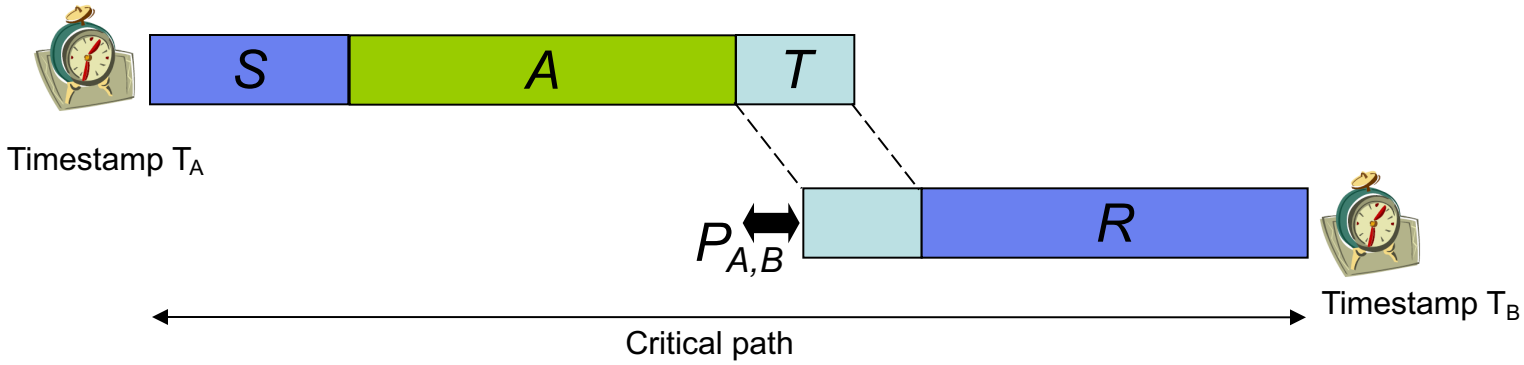
δ = ((T2-T1)+(T4-T3))/2
δ = ((5.35-5.0)+(5.75-6))/2
δ =((.35)+(-.25))/2
δ = .1/2=.05

θ = ((T2-T1)-(T4-T3))/2
θ = ((.35)-(-.25))/2
θ = 0.6/2 =.3

So A adds .3 to 5.75 to get 6.05
Only need Delta to adjust clocks

Disturbing Influences on Packet Latency

- Influences
  - Sending Time *S* (up to 100ms)
  - Medium Access Time *A* (up to 500ms)
  - Transmission Time *T* (tens of milliseconds, depending on size)
  - Propagation Time $P_{A,B}$ (microseconds, depending on distance)
  - Reception Time *R* (up to 100ms)

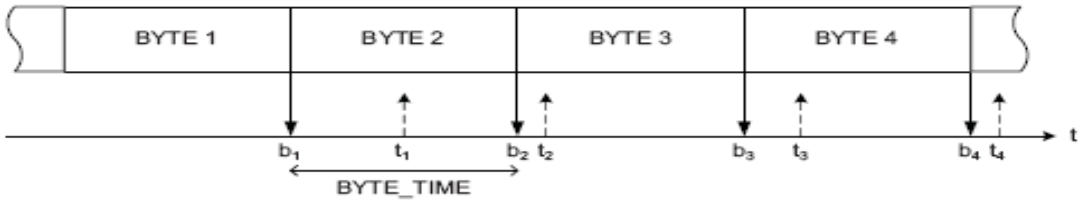


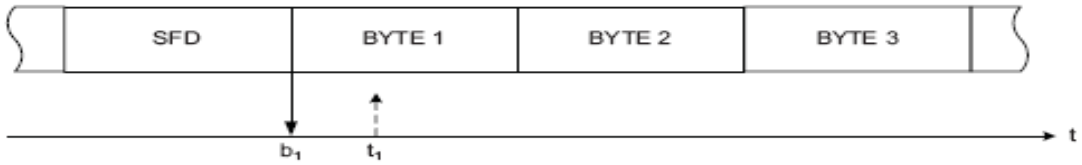
- Asymmetric packet delays due to *non-determinism*
- Solution: timestamp packets at MAC Layer

SOME DETAILS

Different radio chips use different paradigms:

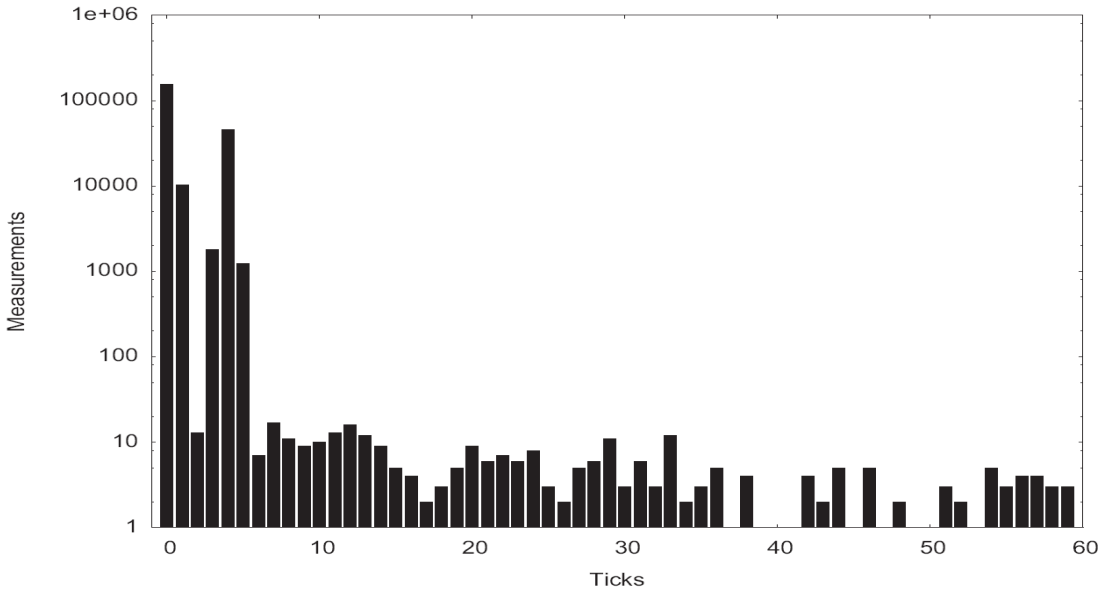
- Left is a CC1000 radio chip which generates an interrupt with each byte.
- Right is a CC2420 radio chip that generates a single interrupt for the packet after the start frame delimiter is received.





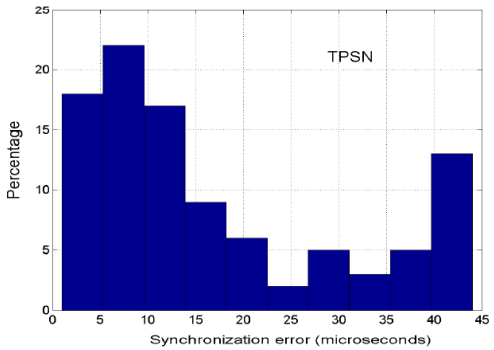
in sensor networks propagation can be ignored (<1¹s for 300m).

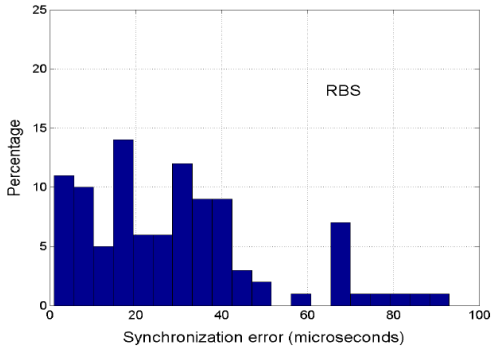
Still there is quite some variance in transmission delay because of latencies in interrupt handling (picture right).

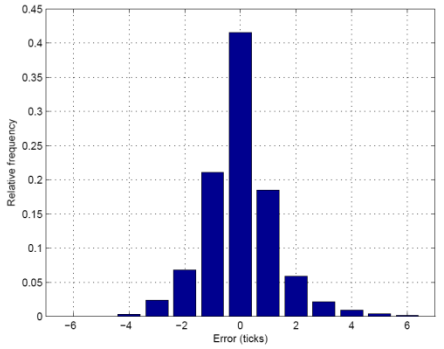


SYMMETRIC ERRORS

Many protocols don't even handle single-hop clock synchronization well. On the left figures we see the absolute synchronization errors of TPSN and RBS, respectively. The figure on the right presents a single-hop synchronization protocol minimizing systematic errors.







Even perfectly symmetric errors will sum up over multiple hops.

- In a chain of *n* nodes with a standard deviation ¾ on each hop, the expected error between head and tail of the chain is in the order of $¾\sqrt{n}$.

NETWORK TIME PROTOCOL (NTP)

- NTP was designed for large−scale networks preferably static topology
- Nodes are externally synchronized to a global reference time
- These are injected into the network at many places through a set of master nodes
- These master nodes are synchronized out of band for instance via GPS (which provides global time with a precision at a great extent below 1μ sec)
- Nodes participating in NTP, leaf nodes are called clients, inner nodes are called Stratum L servers
- L is considered as level of node in the hierarchy
- Parents of each node must be specified in configuration files due to which nodes frequently exchange synchronization messages with their parents and then use achieved information to adjust their clocks by regularly incrementing them

WHY NOT NTP?

Unpredictable network topology in sensor.

In NTP, Master–Client can be multi–hop apart.

Periodic maintenance of H/W & S/W is not likely.

Higher Communication Overheads due to continuous synchronization

Servers need to be active all the time

It does not adapt to the requirement of sensor

Need external source of time as Stratum 0.

NETWORK DYNAMICS

sensor nodes can be mobile, may be dead due to the weakness of batteries or due to influence of environments & new sensor nodes can be added at any point at any time

This operation happens in frequent manner and any unpredictable changes in the network topology can take place, even network partitions

Mobile nodes make the transportation of messages across partition boundaries by storing received messages and further transporting it as soon as a new partition is entered

End-to-end delay of such type of message path is unstable and hard to predict as well

REFERENCE BROADCAST SYNCHRONIZATION (RBS)

RBS is based on the receiver–receiver synchronization RRS. node P represents the parent or reference node, which initiates the synchronization protocol by transmitting beacons to node A and node B, which are assumed to lie within the communication
range of node P.

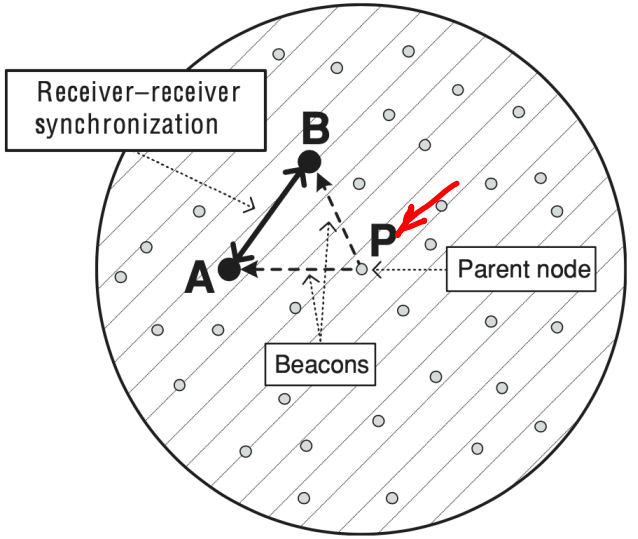


Figure 3.3: RBS signaling mechanism.

Let the timestamps recorded at node A and node B for receiving the ith common packet be denoted as $T_{2,i}^{(A)}$ and $T_{2,i}^{(B)}$, respectively. The estimate of the clock offset between node A and node B is proposed in [22] as

$$\hat{\phi}^{(BA)} = \frac{1}{N}\sum_{i=1}^{N}\left[T_{2,i}^{(A)} - T_{2,i}^{(B)}\right], \tag{3.9}$$

Nodes exchange their recorded times with each other

29.1 microsec accuracy for 1 hop case

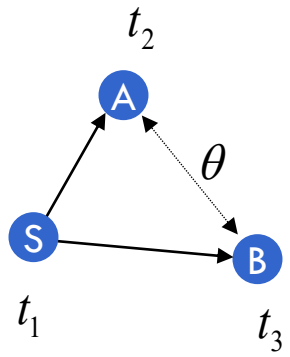
No transmitter side non–determinism

Not extended to large multi–hop networks

Reference-Broadcast Synchronization (RBS)

- A sender synchronizes a set of receivers with one another
- Point of reference: beacon's arrival time

$$t_2 = t_1 + S_S + A_S + P_{S,A} + R_A$$
$$t_3 = t_1 + S_S + A_S + P_{S,B} + R_B$$
$$\theta = t_2 - t_3 = (P_{S,A} - P_{S,B}) + (R_A - R_B)$$



- Only sensitive to the difference in propagation and reception time
- Time stamping at the interrupt time when a beacon is received
- After a beacon is sent, all receivers exchange their reception times to calculate their clock offset

- Post-synchronization possible
- E.g., least-square linear regression to tackle clock drifts
- Multi-hop?

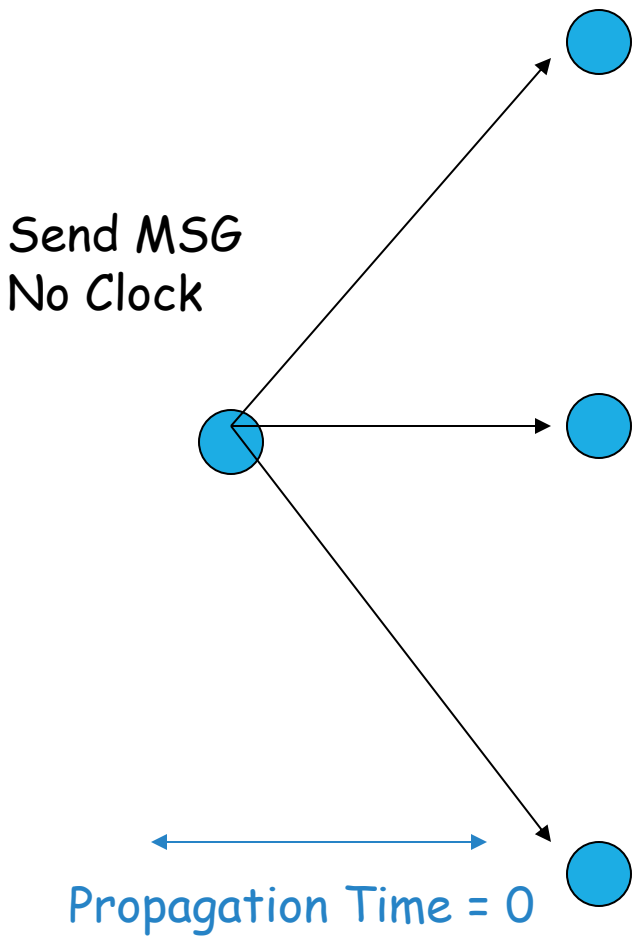
Send Time Latency
-time spent at the sender to construct the message

Access Time Latency
-time spent at the sender to wait for access to transmit the message

Prorogation Time Latency
-time spent by the message in traveling from the sender to the receiver

Receive Time Latency
-time spent at the receiver to receive the message from the channel and to notify the host

RBS



	Local Time	Exchange	Adjust
	5	6,7	Plus 2
	6	5,7	Plus 1
	7	5,6	OK

What can cause the “ideal” case of all three nodes getting the clock message at the “same” time to be false?

- Node turns off interrupts
  - Code explicitly turns off interrupts
- Node turns off radio (sleep mode)

REFERENCE BROADCAST SYNCHRONIZATION (RBS)

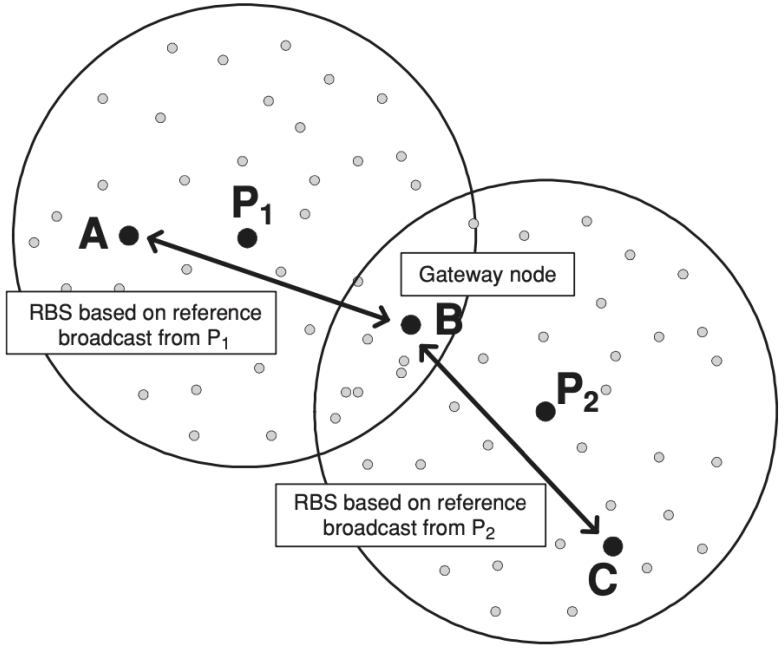
The RBS protocol discussed above can only synchronize a set of nodes that lie within a single broadcast domain.

In order to extend RBS to achieve global synchronization of a large-scale sensor network, Elson *et al*. [22] propose using *gateway* nodes to convert timestamps from one neighborhood's timebase to another.

This idea is illustrated in the figure. Nodes P1 and P2 send out synchronization beacons, and they create two overlapping neighborhood with node B lying in the overlapping area.

Since node A and node B lie within the same neighborhood, their clock relationship (i.e., clock offset and skew) can be established from node P1's reference broadcast.

Similarly the clock relationship between node B and node C can be established from node P2's reference broadcast. Therefore, the clock relationship between node A and node C can be computed with node B acting as a gateway.



REFERENCE BROADCAST SYNCHRONIZATION (RBS)

Application:

- It can be used in any application where nodes do not need absolute time (UTC).
- It can also be used in Energy saving networks.

TPSN

Creates a spanning tree

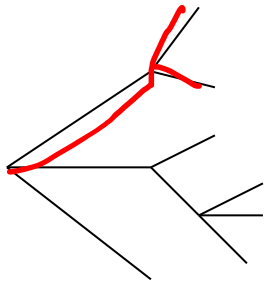
Perform pairwise sync along edges of the tree

Must be symmetric links

Considers “entire” system

- RBS looks at one hop

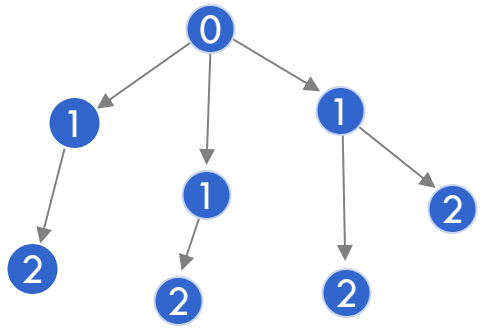
Exchange two sync messages with parent node



Time-sync Protocol for sensor Networks (TPSN)

Round-trip-time

- Traditional sender–receiver synchronization (RTT–based)
- *Initialization phase: Breadth–first–search flooding*
  - Root node at level 0 sends out a *level discovery* packet
  - Receiving nodes which have not yet an assigned level set their level to $+1$ and start a random timer
  - After the timer is expired, a new level discovery packet will be sent
  - When a new node is deployed, it sends out a *level request* packet after a random timeout

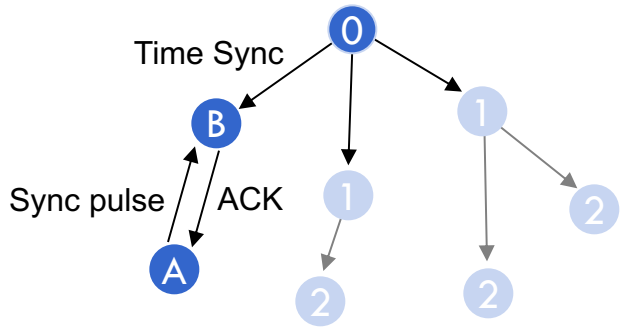


Why this random timer?

TIME-SYNC PROTOCOL FOR SENSOR NETWORKS (TPSN)

Synchronization phase

- Root node issues a *time sync* packet which triggers a random timer at all level 1 nodes
- After the timer is expired, the node asks its parent for synchronization using a *synchronization pulse*
- The parent node answers with an *acknowledgement*
- Thus, the requesting node knows the round trip time and can calculate its clock offset
- Child nodes receiving a synchronization pulse also start a random timer themselves to trigger their own synchronization

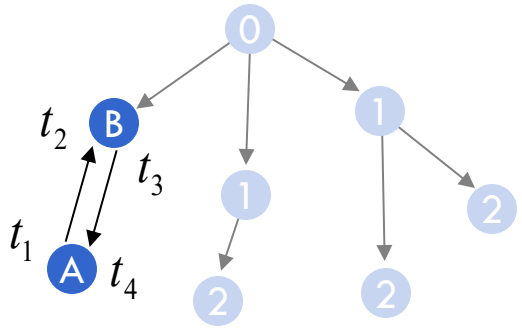


Time-sync Protocol for sensor Networks (TPSN)

$$t_2 = t_1 + S_A + A_A + P_{A,B} + R_B$$

$$t_4 = t_3 + S_B + A_B + P_{B,A} + R_A$$

$$\theta = \frac{(S_A - S_B) + (A_A - A_B) + (P_{A,B} - P_{B,A}) + (R_B - R_A)}{2}$$



- Time stamping packets at the MAC layer
- In contrast to RBS, the signal propagation time might be negligible
- Authors claim that it is “about two times” better than RBS
- Again, clock drifts are taken into account using periodical synchronization messages

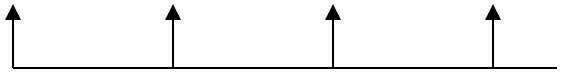
- Problem: What happens in a non-tree topology (e.g. grid)?
  - Two neighbors may have bad synchronization?

TPSN

No broadcasting so TPSN is expensive

Timestamp the message in the MAC layer multiple times and average those times (improves accuracy over RBS by 2 times)

(16.9 microsec accuracy for 1 hop)



TIME-SYNC PROTOCOL FOR SN (TPSN)

- Statistics of TPSN [3]:
  - Average error of 16.9 μs was achieved.
  - Worst case error of 44 μs
  - In multi-hop scenario the error 73.6 μs for 5 hops
  - Error increases as no of hops increases

TPSN

Assumes topology does not change

- Not robust to failure of a node
  - Node fails – must redo the spanning tree

REMOVE UNCERTAINTIES

Eliminate Send Uncertainty

- Get time in the MAC layer

Eliminate Access Time

- Get time after the message has access to the channel

Eliminate Receive Time

- Record local time message received at the MAC layer

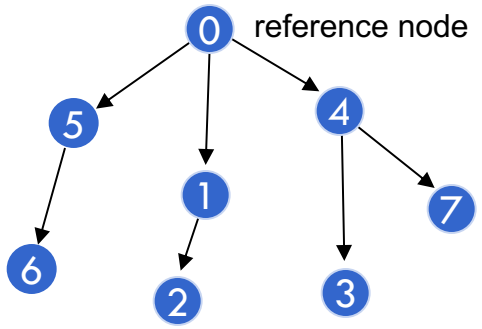
FLOODING TIME SYNCHRONIZATION PROTOCOL (FTSP)

Each node maintains both a local and a global time

Global time is synchronized to the local time of a reference node

- Node with the smallest id is elected as the reference node

Reference time is flooded through the network periodically



Timestamping at the MAC Layer is used to compensate for deterministic message delays

Compensation for clock drift between synchronization messages using a linear regression table

FTSP

Root maintains global time for system

All others sync to the root

Nodes form an ad hoc structure rather than a spanning tree

Root broadcasts a timestamp for the transmission time of a certain byte

Every receiver time stamps reception of that byte

Account for deterministic times

Differences are the clock offsets

SUMMARY OF FTSP SO FAR

MAC-layer timestamp

Correct sender timestamp to account for known delays

- Compute final offset error

Result: 1.48 microsec accuracy for 1 hop

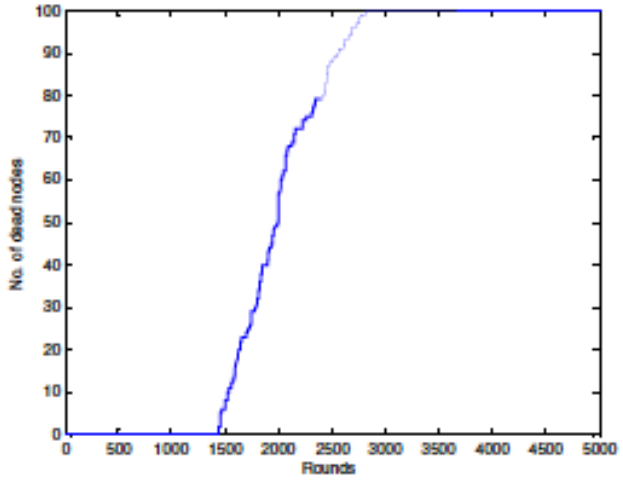


Figure 3: Dead nodes in RBS

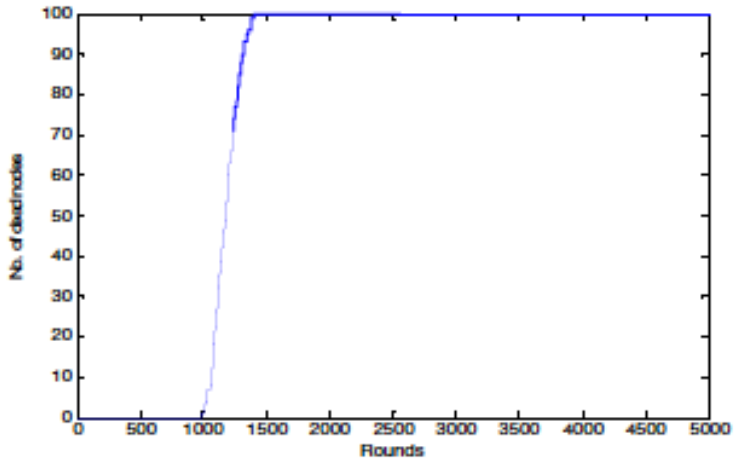
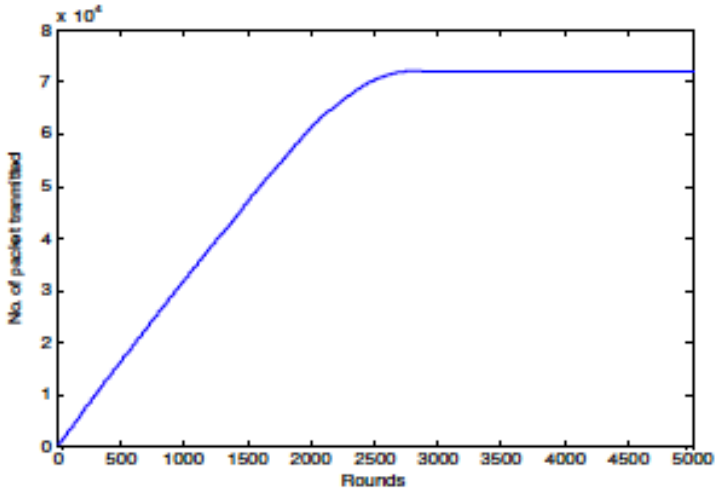
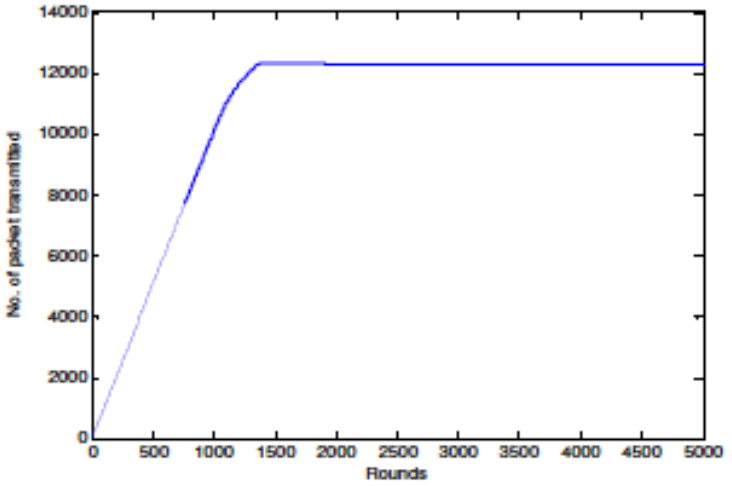


Figure 4: Dead nodes in TPSN





Summary

Class	RBS	TPSN	TS/TM	LTS	TSync	FTPS	AD
Internal vs. External	I	E	I	E	E	I	I
Cont. vs. On-demand	O	C	C	O	C	C	C
All nodes vs. Subset	S	A	S	A/S	A	A	A
Rate vs. Offset	RO	O	RO	O	O	RO	O
Assumption							
Broadcast	X	X			X	X	X
Uni vs. Bidirectional	U	B	B	B	B	U	B
Constant rate			X				
Bound Drift				X			
Multichannel					X		
MAC access		X				X	

Table 4.1: de facto Standards and how they perform

de facto Standard	Accuracy	Scalability	Energy efficiency
RBS	29.1 µs	Good	High
TPSN	16.9 µs	Poor	High
FTSP	1.48 µs	Average	High

CLOCK DRIFT

If local clocks all had *exact* same frequency then no clock drift corrections are needed and no re-syncs needed either

MICA2 motes

- Crystals used have drifts of up to 40 microseconds per second
- Implication: resync every sec to maintain 40 microsec accuracy
- Too costly – estimate clock drift and fix!

SUMMARY

Very accurate clock sync is possible (in microsec range)

Time to perform clock sync is important

- Initial delay at system init time may be important
  - Consider a system of $10{,}000$ nodes
  - Multiple base stations issues

Solve clock drift

- Frequent sync messages
- Infrequent sync messages but adjust locally

SUMMARY

Consider costs

- Energy
- Congestion
- Time

Consider accuracy requirement

REFERENCE

Sichitiu, Veerarittiphan, "Simple, Accurate Time Synchronization for Wireless Sensor Networks"

Ganeriwal, Kumar, Srivastava, "Timing-Sync Protocol for Sensor Networks"

Genunen, Rabaey, "Lightweight Time Synchronization for Wireless Sensor Networks"

Li, Rus, "Global Clock Synchronization in Sensor Networks"

Dai, Han, "TSync: A Lightweight Bidirectional Time Synchronization Service for Wireless Sensor Networks"

22. J. Elson, L. Girod, and D. Estrin, Fine-grained network time synchronization using reference broadcasts, in *Proceedings of the 5th Symposium on Operating Sys- tem Design and Implementation, Boston, MA, December 2002*, pp. 147–163. ACM, 2002.