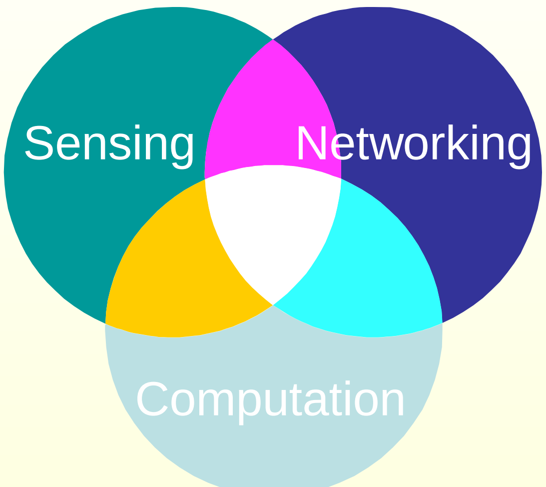
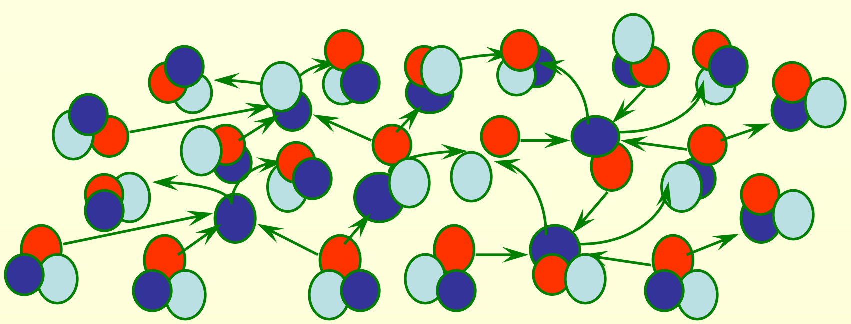
CS321: Time Synchronization



Ian Downes

Slides from Brano Kusy



Why Do We Need Time Coordination?

- What is the time now?
- Do we all agree? Why/why not?
- Where are we sourcing our time from?
- How accurate do we need time?

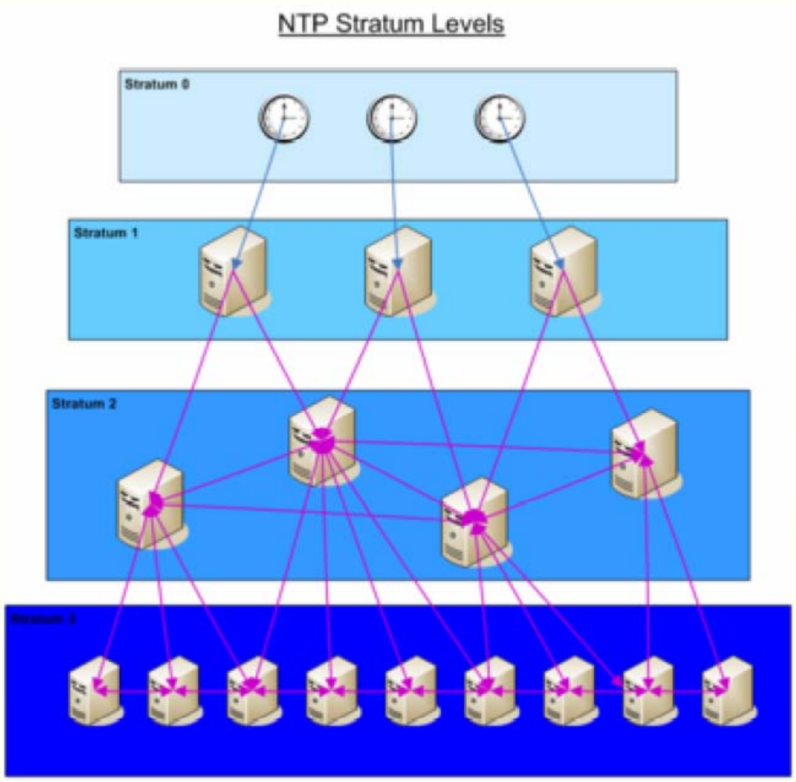
Why Do We Need Time Coordination?

- Major reasons for timesync:
  - Determine temporal relationships of observations between sensors
    - Events with timestamps
      - e.g. Acoustic source localization: Accuracy 30us ~ 1cm error
    - Data with timestamps
      - e.g. Golden Gate bridge monitoring: Accuracy 10us
    - Delay measurements for distance estimation/location
      - e.g. Audio/sound/noise propagation: Accuracy 1ms - 10ms
      - e.g. Radio propagation: Accuracy 10ns - 1us
  - Coordinated actuation of sensors, reacting to sensed events in real time
    - e.g. TDMA: a few us
- In sensor networks, each node has its own clock
  - Clocks drift apart from each other
    - different startup times
    - manufacturing differences, environmental effects (battery power, temperature, humidity)

Time Synchronization Challenges

- **Time synchronization**: a system service maintaining a common notion of time across multiple nodes over possibly multi-hop links
- **Challenges**:
  - large variety of timesync needs
    - no method is optimal for all applications
  - heterogeneous and rapidly evolving hardware platforms
  - high accuracy application requirements but resource constrained hardware
- **Would traditional solutions work?**
  - NTP (Network time protocol)
  - Lamport's logical clocks
  - GPS at each node

Network Time Protocol



NTP [Mills 1995]

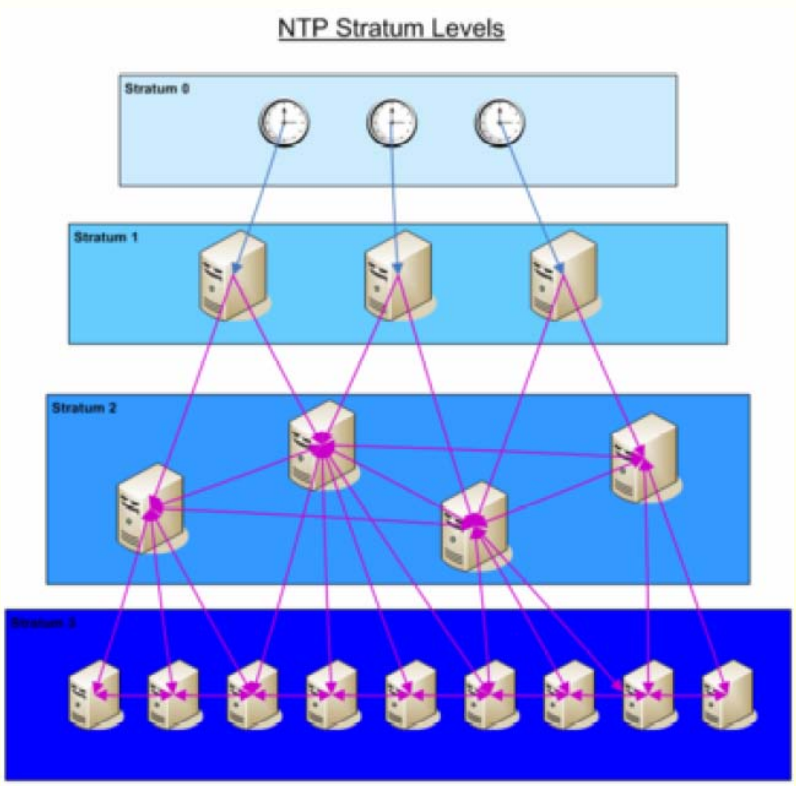
- the Internet timekeeper
- uses a unique “leader” clock
- advanced and tested in large scale

Overview

- Designed for static networks
- Global reference time is injected to the network by time servers (Stratum 1)

  - synced out of band by GPS
- Nodes participating in NTP form hierarchy
- Timesync information is frequently obtained from parents (RTT time)
- Statistical techniques overcome RTT non-deterministic delays in Internet

Network Time Protocol



NTP [Mills 1995]

- the Internet timekeeper
- uses a unique “leader” clock
- advanced and tested in large scale

Problems

- No accuracy guarantee: 2—100ms is typical
- Scarce resources may preclude out-of-band synchronization
- Not energy optimized – e.g. requires all nodes to be synced with max accuracy
- NTP servers must accept timesync requests at any time (no radio off)
- End-to-end delay is unpredictable, routes may be long (hop-wise)
- Statistical techniques require significant computation and memory

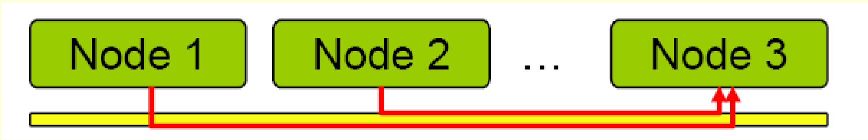
Lamport's Logical Clocks

Logical Clock

- Assign relative time to events, so that their causality is not violated
- Time may deviate from absolute local time
- For distributed „make", only order of events is important!

Happens before relation (→):

- On the same node:
  a → b, if time(a) < time(b)
- If n1 sends m to n2:
  send(m) → receive(m)
- Transitivity:
  If a → b and b → c then a → c



Leslie Lamport [1978]

- All nodes use a counter (clock) with initial value of zero
- A node increments its counter when it sends a message or detects an event
- Messages carry timestamps
- On message receipt, the receiver's counter is updated

Node 1	Node 2	Node 3
0	0	10
2	3	13
4	6	16
6	9	19
8	12	22
10	23	25
24	26	28
26	29	31
28	**31**	**34**

Correction

Frequent message exchange reduces clock deviation

Lamport's Logical Clocks

Logical Clock

- Assign relative time to events, so that their causality is not violated
- Time may deviate from absolute local time
- For distributed „make", only order of events is important!

Happens before relation (⟶):

- On the same process:
  a → b, if time(a) < time(b)
- If p1 sends m to p2:
  send(m) → receive(m)
- Transitivity:
  If a → b and b → c then a → c

Leslie Lamport [1978]

- All nodes use a counter (clock) with initial value of zero
- A node increments its counter when it sends a message or detects an event
- Messages carry timestamps
- On message receipt, the receiver's counter is updated

Problems:

- Delivery order of messages in WSNs does not imply causality of events due to MAC, routing delays
- Partial order only

Traditional Approaches Do Not Always Work

GPS at every node

- Accurate: some GPSs provide 1 pps @ O(10ns) accuracy
- But doesn't work everywhere and has cost, size, and energy issues

NTP

- potentially long and varying paths to time-servers due to multi-hopping and short-lived links
- delay and jitter due to MAC and store-and-forward relaying
- discovery of time servers
- Perfectly acceptable in many cases (coarse grain synchronization), but inefficient when fine-grain sync is required

Logical clocks

- Delivery order of messages in WSNs does not ensure causality of events

Traditional Approaches Do Not Always Work

NTP

- potentially long and varying paths to time-servers due to multi-hopping and short-lived links
- dela
- disc
- Perfe ineffi

- Improve accuracy of time-synchronization
- Enable resource efficient implementation with low computation and memory requirements
- Allow for dynamic changes in topology and ad-hoc deployments

Logica

- Delivery order of messages in WSNs does not assure causality of events

GPS at every node

- Accurate: some GPSs provide 1 pps @ O(10ns) accuracy
- But doesn't work everywhere and has cost, size, and energy issues

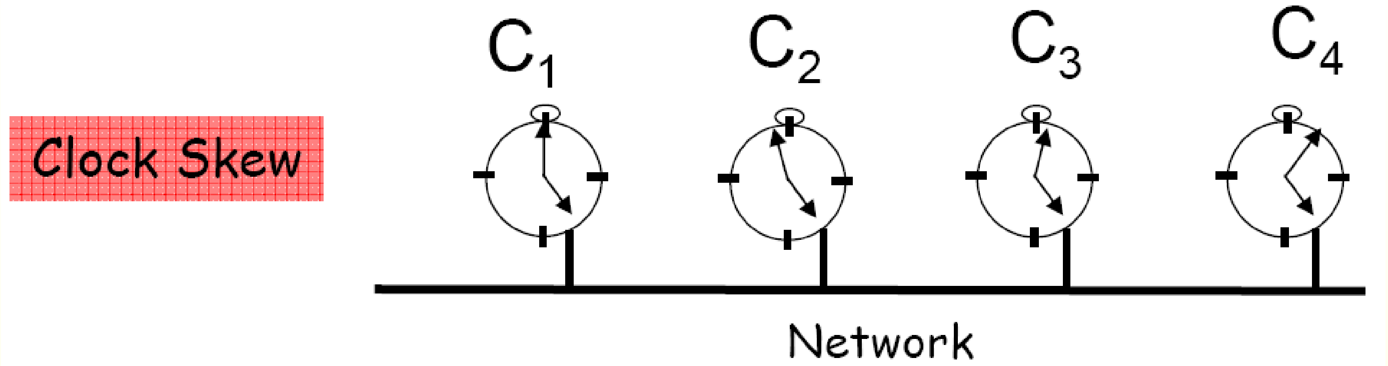
Computer Clocks

- Sensors do not have clocks (due to cost) !
  - Typical sensor CPU has counters that increment by each cycle, generating interrupt upon overflow (oscillations of a quartz)
  - External oscillators (with HW counter) can keep time when CPU is off
  - A counter represents the passing of time:

    $H_i(t)$
  - The OS can maintain SW Clock by scaling and adding an offset to a counter:

    $C_i(t) = \alpha H_i(t) + \beta$
  - $C_i(t)$ is typically implemented by a 32-bit word, representing microseconds that have elapsed at time t
  - Successive events can be distinguished if the clock resolutions is smaller that the time interval between the two events

Drift and Skew



- Computer clocks, like any other clocks tend not to be in perfect agreement !!
- **Clock skew**: the difference between the times on two clocks

$$|C_i(t) – C_j(t)|$$

- **Clock drift**: clocks count time at different rates

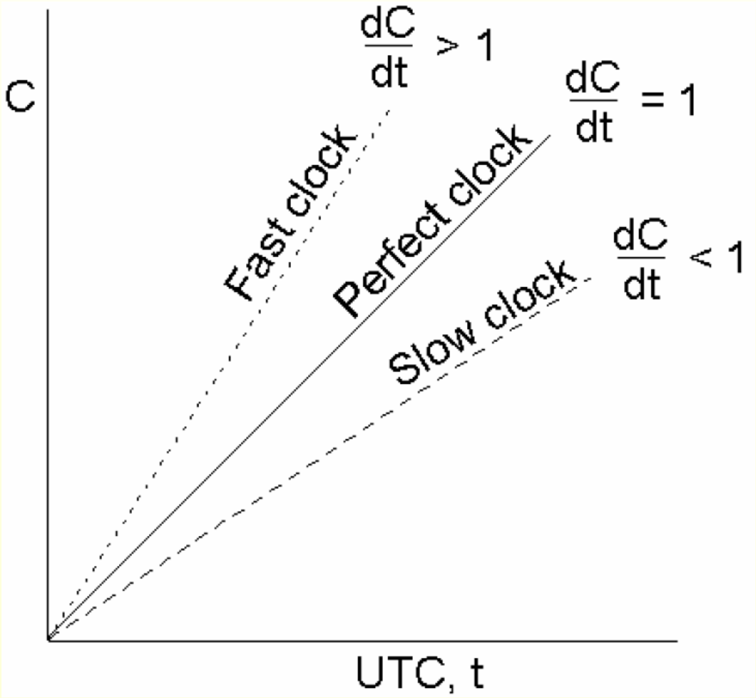
$$dC_i/dt \mathrel{!=} dC_j/dt$$

Clock Drift

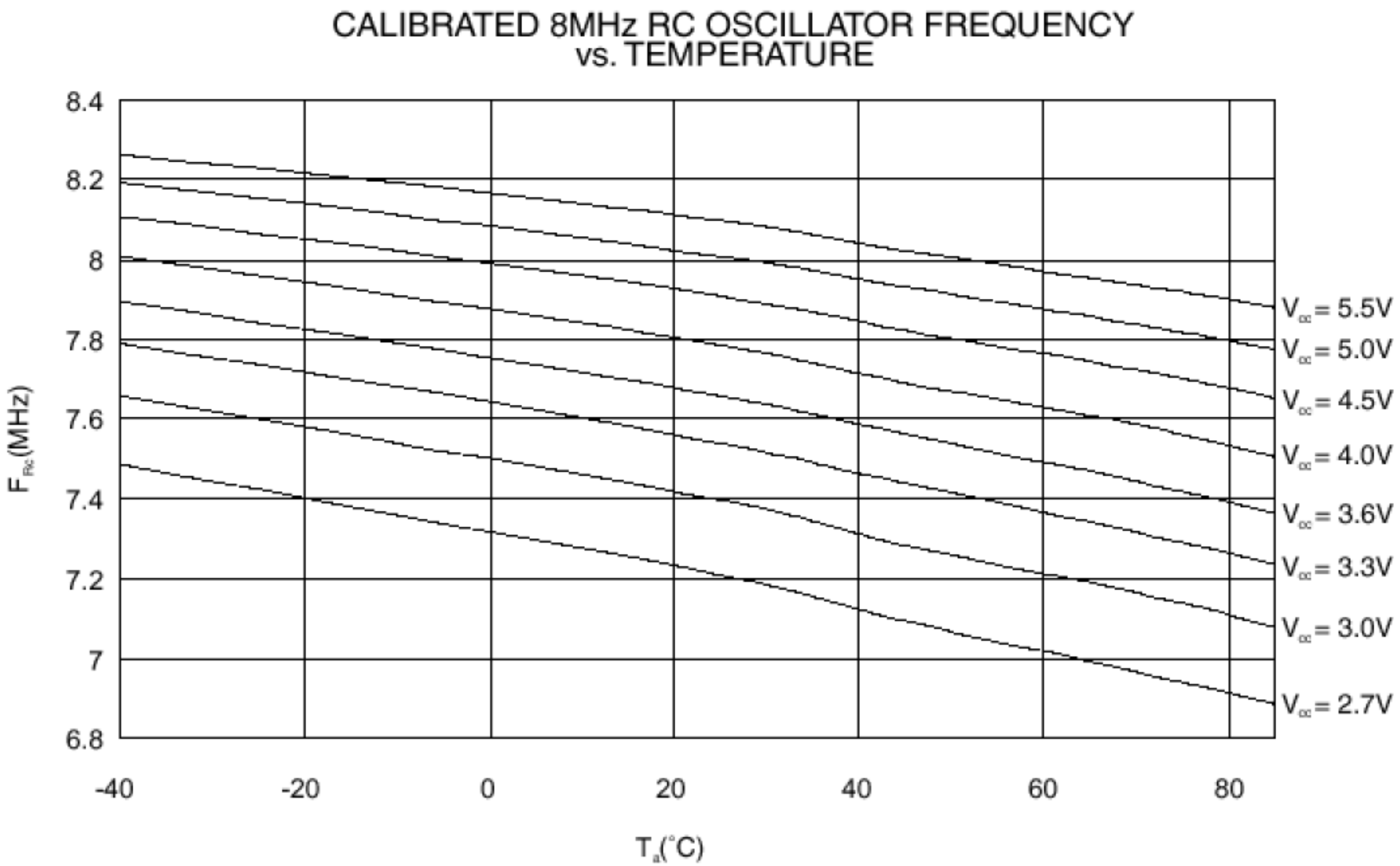
- Clock makers specify a maximum drift rate ρ ppm
  - Ordinary cheap quartz crystals drift by ~ 1sec in 2 days (10^{-5} secs/sec)
  - Clock drift is often given in parts-per-million (e.g. 10 ppm)
- By definition

$$1-\rho \leq dC/dt \leq 1+\rho$$

- Clock drifts depend on
  - manufacturing defects,
  - temperature, and
  - power supply variation



Typical Oscillator Data

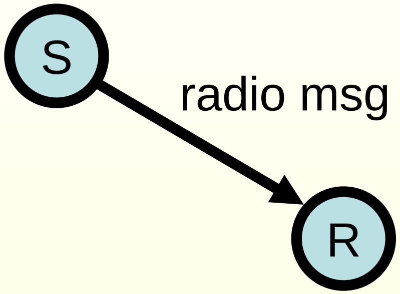


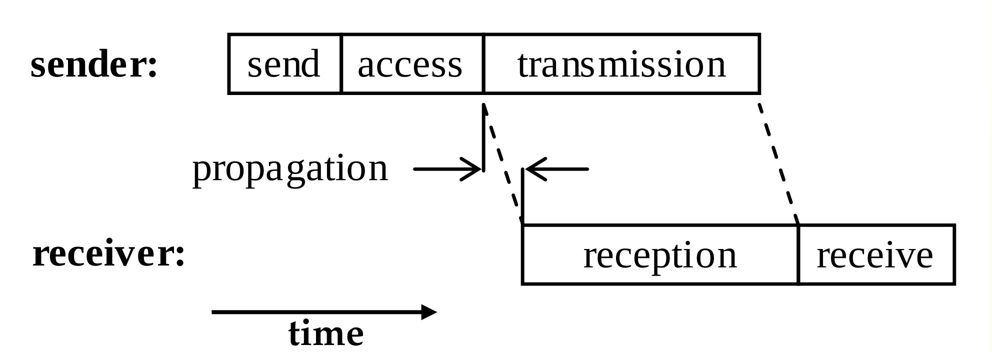
Time Synchronization

- The main objective is to determine relative drifts of the clocks of different sensor nodes
- This is a multi-step process
  - Node-to-node instantaneous synchronization
  - Node-to-node continuous synchronization
  - Multi-hop synchronization

Node to Node Instantaneous Synchronization

- determine difference between local clocks of 2 nodes
- most popular method is timestamping radio messages





Delays incurred in the process of timestamping:

send time: the time used to assemble the msg and issue the send request to MAC

access time: the delay incurred waiting for access to the transmit channel up to the point when transmission begins

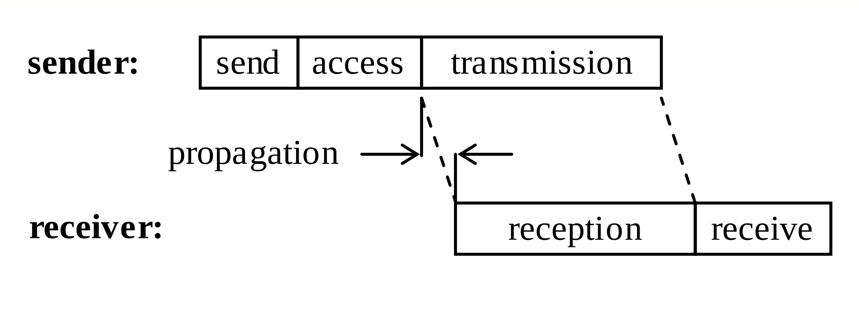
transmission time: the time required for the sender to transmit the message.

propagation time: required for message to propagate from sender to the receiver

reception time: the time required for the receiver to receive the message.

receive time: time to process the incoming message and to notify the receiver application.

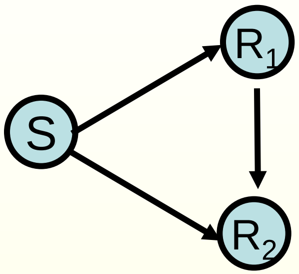
Node to Node Instantaneous Synchronization



Time	Magnitude	Distribution
Send and Receive	0 – 100 ms	nondeterministic, depends on the processor load
Access	10 – 500 ms	nondeterministic, depends on the channel contention
Transmission / Reception	10 – 20 ms	deterministic, depends on message length
Propagation	< 1µs for distances up to 300 meters	deterministic, depends on the distance between sender and receiver

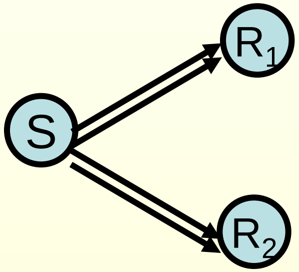
RBS ['02]

- Eliminates send and access times
- Requires additional radio communication



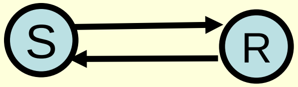
FTSP ['05]

- Timestamps after MAC granted
- Single broadcast syncs multiple receivers

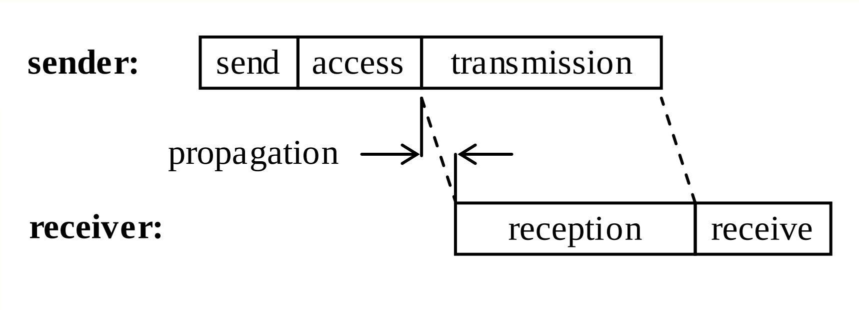


TPSN ['04]

- Two way (unicast) communication
- Determines both offset and drift



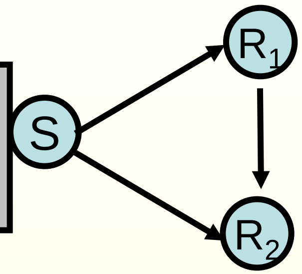
Node to Node Instantaneous Synchronization



Time	Magnitude	Distribution
Send and Receive	0 – 100 ms	nondeterministic, depends on the processor load
Access	10 – 500 ms	nondeterministic, depends on the channel contention
Transmission / Reception	10 – 20 ms	deterministic, depends on message length
Propagation	< 1µs for distances up to 300 meters	deterministic, depends on the distance between sender and receiver

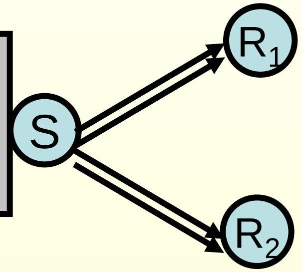
RBS ['02]

Implementation:
no special access to radio is required



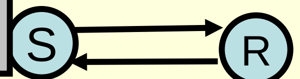
FTSP ['05]

Efficiency:
single message multiple receivers



TPSN ['04]

Value:
Both offset and drift are found

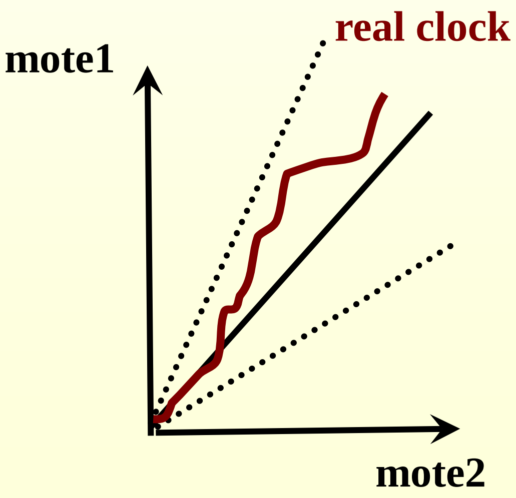


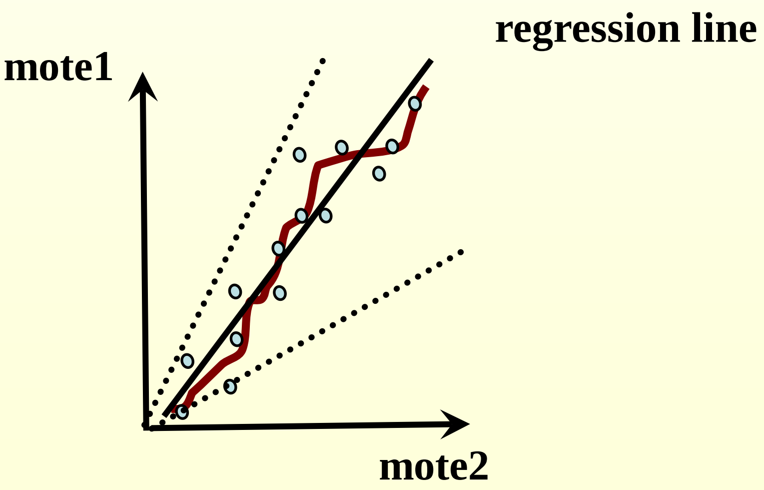
All three algorithms achieve a few µs accuracy.

Node to node continuous synchronization

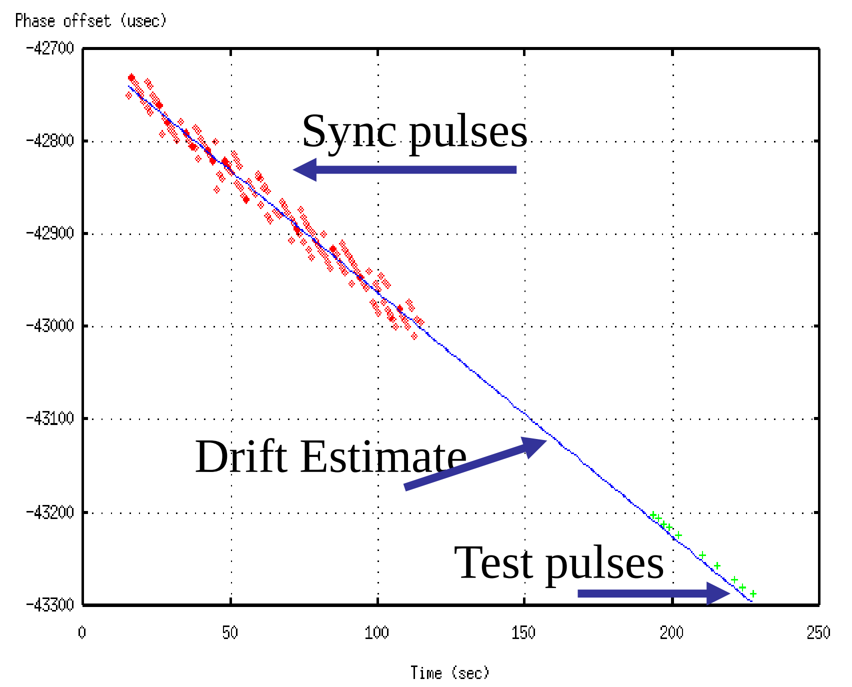
Relative drift synchronization:

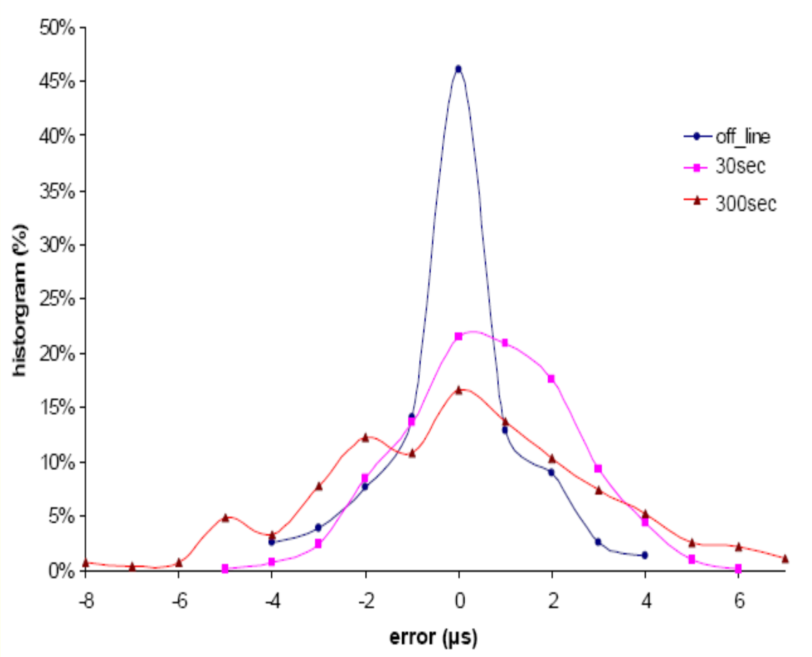
- most commonly, we continuously estimate both rate and offset of the local clocks of 2 nodes
- synchronization in rounds: a popular method is **linear regression**
- For nodes n_i, n_j a linear relation $C_i(t) = \alpha C_j(t) + \beta$ is postulated
- α, β are determined by minimizing square differences of the times





Linear Regression





RBS:
7usec error after 60 seconds of silence

FTSP:
The distribution of the errors of linear-regression

Continuous accuracy of a few μs is possible using LR.

Multi-hop synchronization

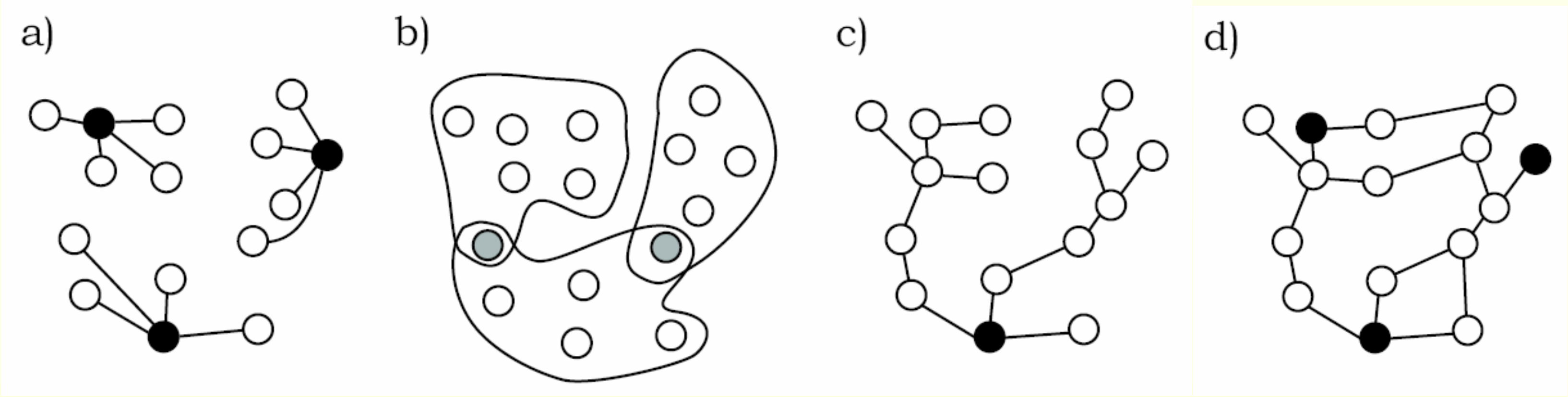
Multi-hop needs to be dealt with explicitly – overlaying could introduce large errors, techniques to organize multi-hop synchronization:

a) Single-hop synchronization: with a set of master nodes which are synced out of band. (e.g., using GPS)

b) Single-hop synchronization in overlapping clusters, gateway nodes translate time stamps. (RBS)

c) Tree hierarchy with a single master node at the root. (TPSN)

d) Unstructured, master node is elected. (FTSP)



Continuous multi-hop accuracy of a few μs is possible.

Specific Problems in WSNs

- Certain WSN scenarios may prevent us from deploying sensor nodes at precise locations, or to provide more reliable, GPS equipped leader nodes

Ad-hoc operation is required

- Power supply is limited and continuous synchronization is a resource demanding service

Power efficient methods are required

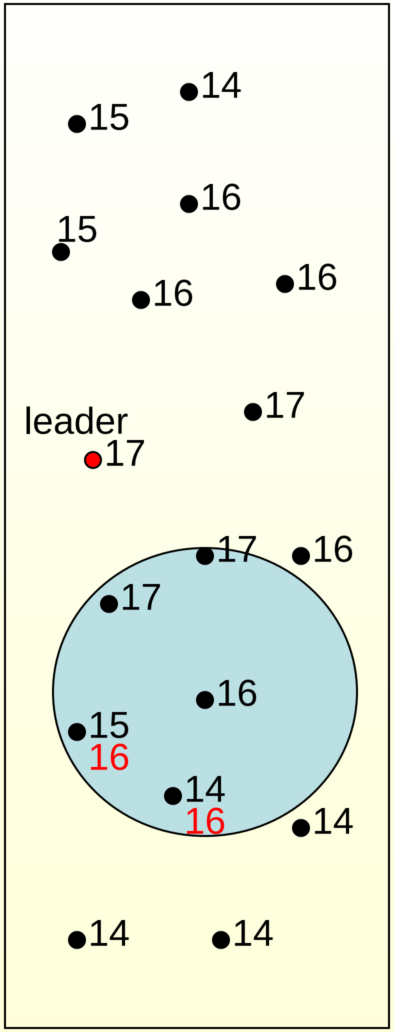
Ad-hoc Mode of Operation - FTSP

Overview

- Global time is synchronized to the local time of an elected leader
- No hierarchy is maintained, instead asynchronous diffusion is utilized: each node sends one synchronization msg per 30 seconds, constant network load
- Sequence number, incremented only by the elected leader
  - to determine when the leader fails
  - to distinguish old and new timestamps

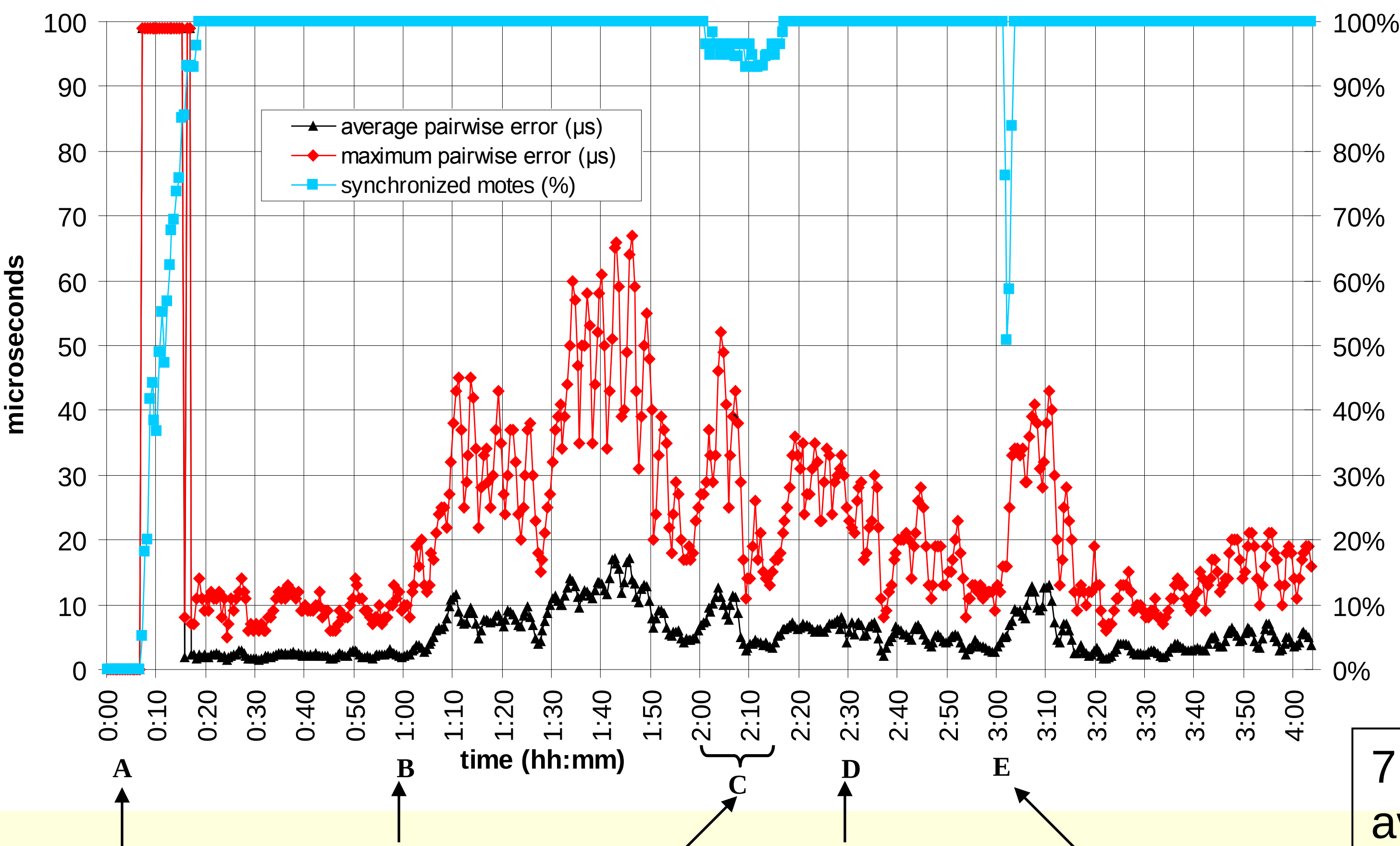
Robustness

- If leader fails, new leader is elected automatically. The new leader keeps the offset and skew of the old global time
- When leader failure is detected, all nodes become leaders; election algorithm rapidly resolves this anarchy
- Fault tolerant: nodes can enter and leave the network, links can fail, nodes can be mobile, topology can change

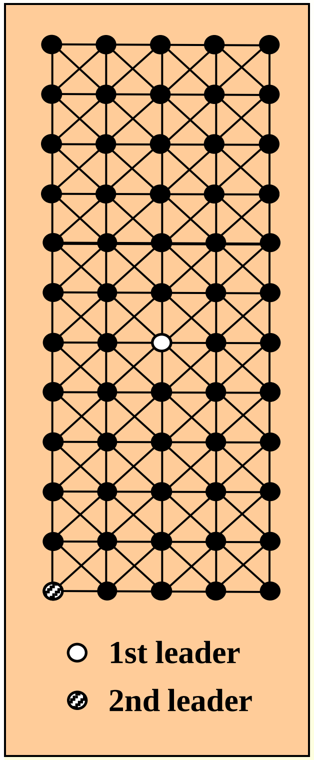


sequence numbers

FTSP experimental evaluation



topology:



all turned on

1st leader turned off

random nodes turned off/on

50% turned off

all turned back on

7.38 MHz CPU
avg. error: 1.6 μs
max. error: 6.1 μs
per hop

Continuous vs Post-facto Synchronization

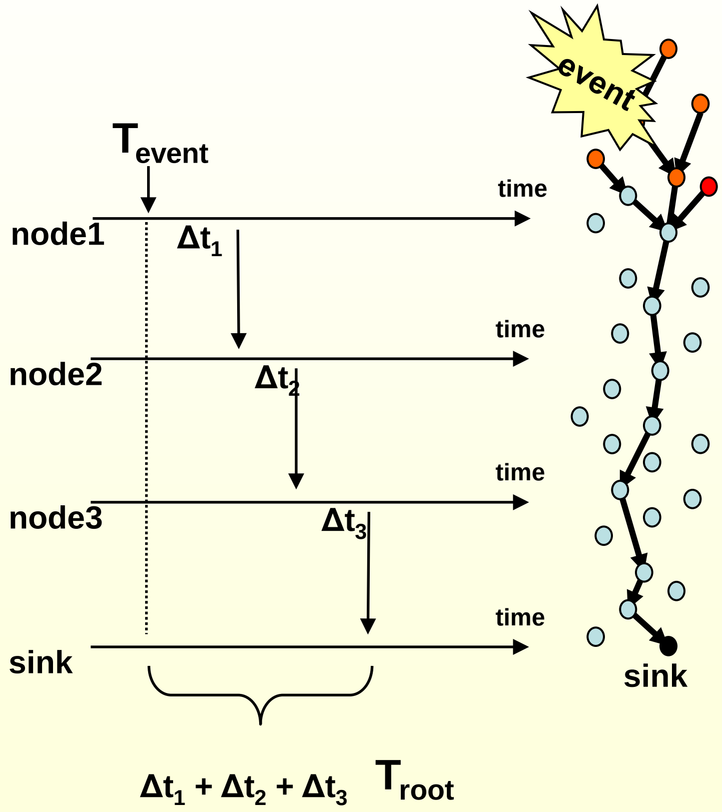
So far we have only seen continuous mode of operation – **virtual global time service**

Post-fact techniques

- Synchronize after an event was detected
- Enable power saving mode
- However, timestamps are not available immediately (wait for synchronization)

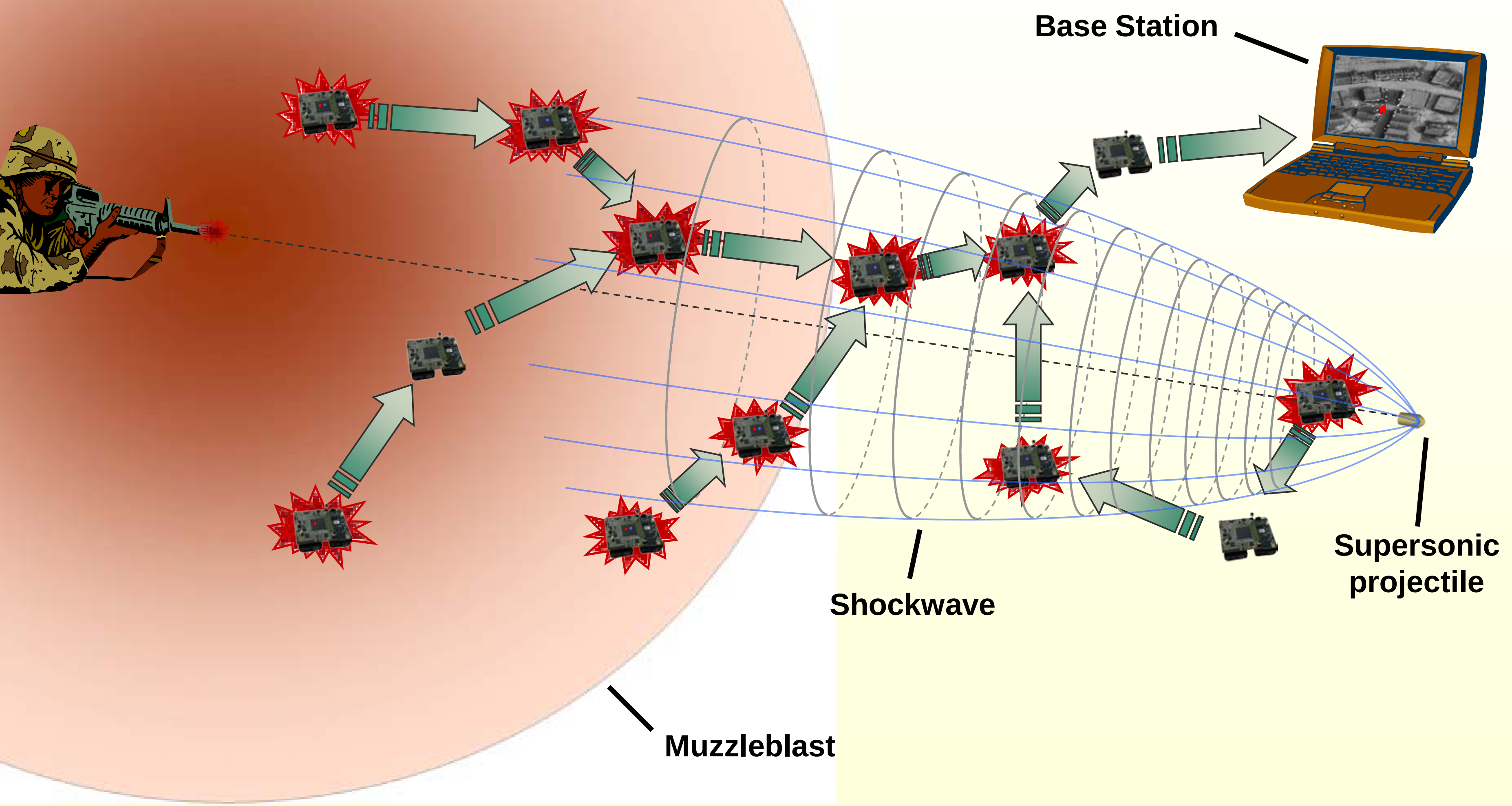
TDOA (time-difference-of-arrival) apps:

- Special post-facto case
- Only differences of event detection times are important
- Transmit age of events, rather than event times, root calculates time differences per its local clock
- Can be piggybacked to existing radio traffic
- 5.7 μs average, 80 μs maximum error were achieved in a 10-hop, 45-node network

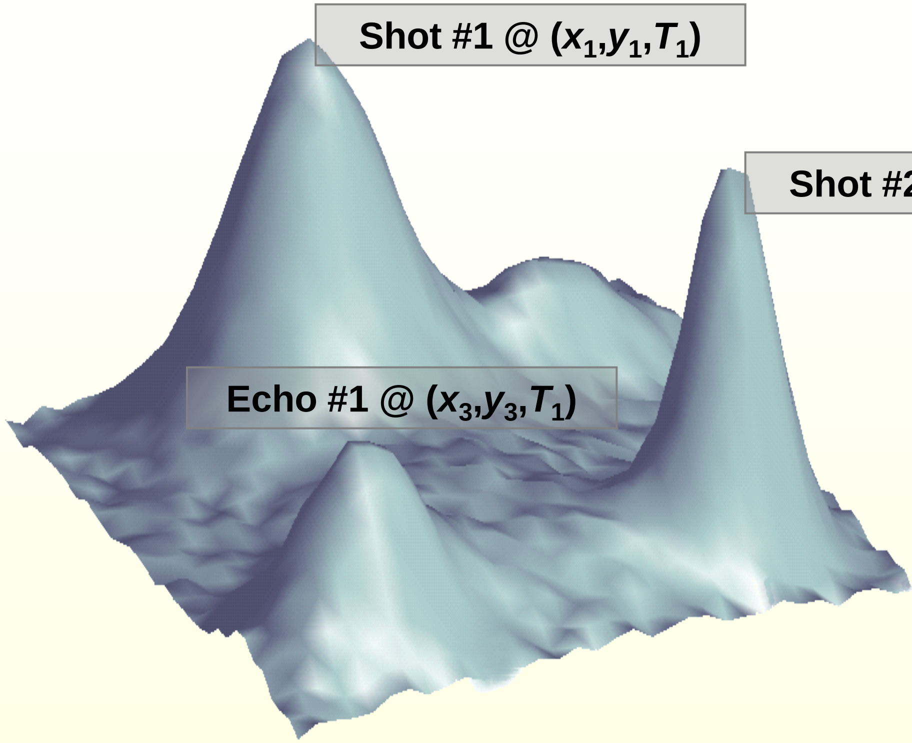


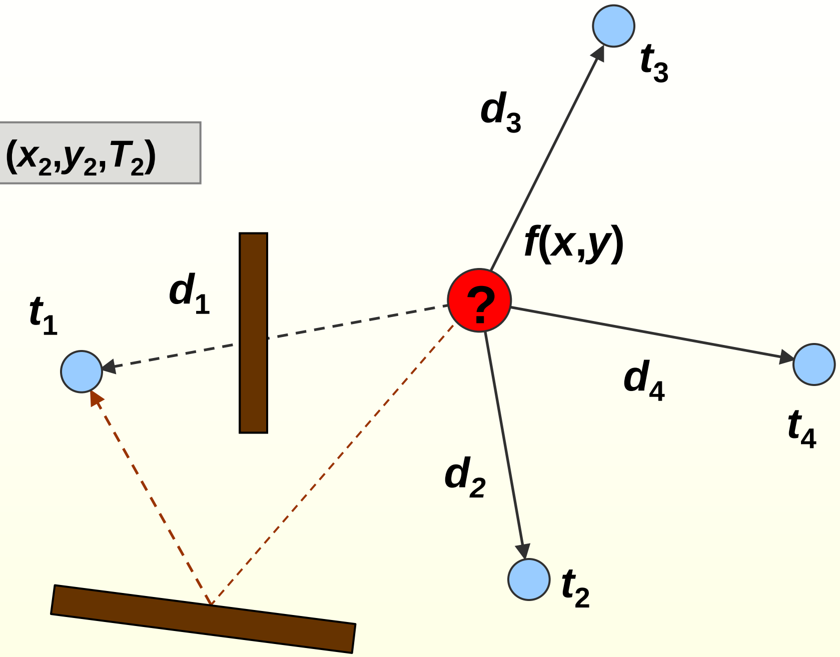
$$T_{event} = T_{root} - \Delta t_1 - \Delta t_2 - \Delta t_3$$

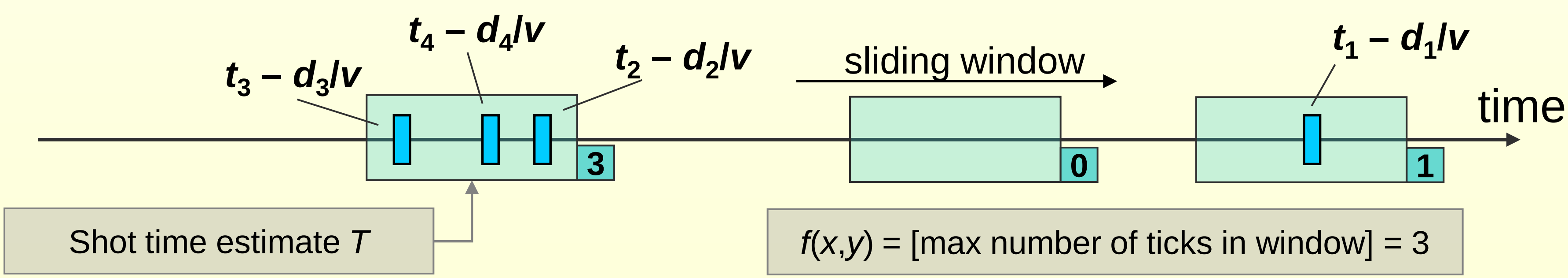
Timesync in Practice: A Countersniper System



Sensor Fusion Requires Timesync







Proactive or Reactive Timesync?

Proactive timesync

- All nodes continuously synchronize
- Shot events are timestamped with global time
- Base station (BS) combines global times to find the sniper location

Cons:

- Active synchronization may reveal the countersniper system
- Active synchroinzation is power demanding

Reactive timesync

- Nodes are turned on only when a shot is detected
- Shot events are timestamped with local times and rapidly sent to BS
- Base station combines local times to find the sniper location

Pros:

- Power efficient, stealthy mode
- As long as a collection tree exists, we do not worry about nodes entering/leaving, or mobility
- Timestamps can be embedded in the routing messages

Conclusion

- we saw how time sync has different needs & opportunities in wireless sensor networks than for traditional LAN/WAN/Internet
- propagation delay often insignificant
- special techniques to deal with radio/MAC/system delays

- there are quite varied alternatives for how to synchronize in multihop networks
  - single-hop beacon (like GPS) good for some situations
  - time sync strategies can be similar to routing protocol structures (trees, zones)
  - extra care may be required for ad-hoc and power efficient operation
- virtual global time service is expensive, consider post-facto techniques for energy efficiency