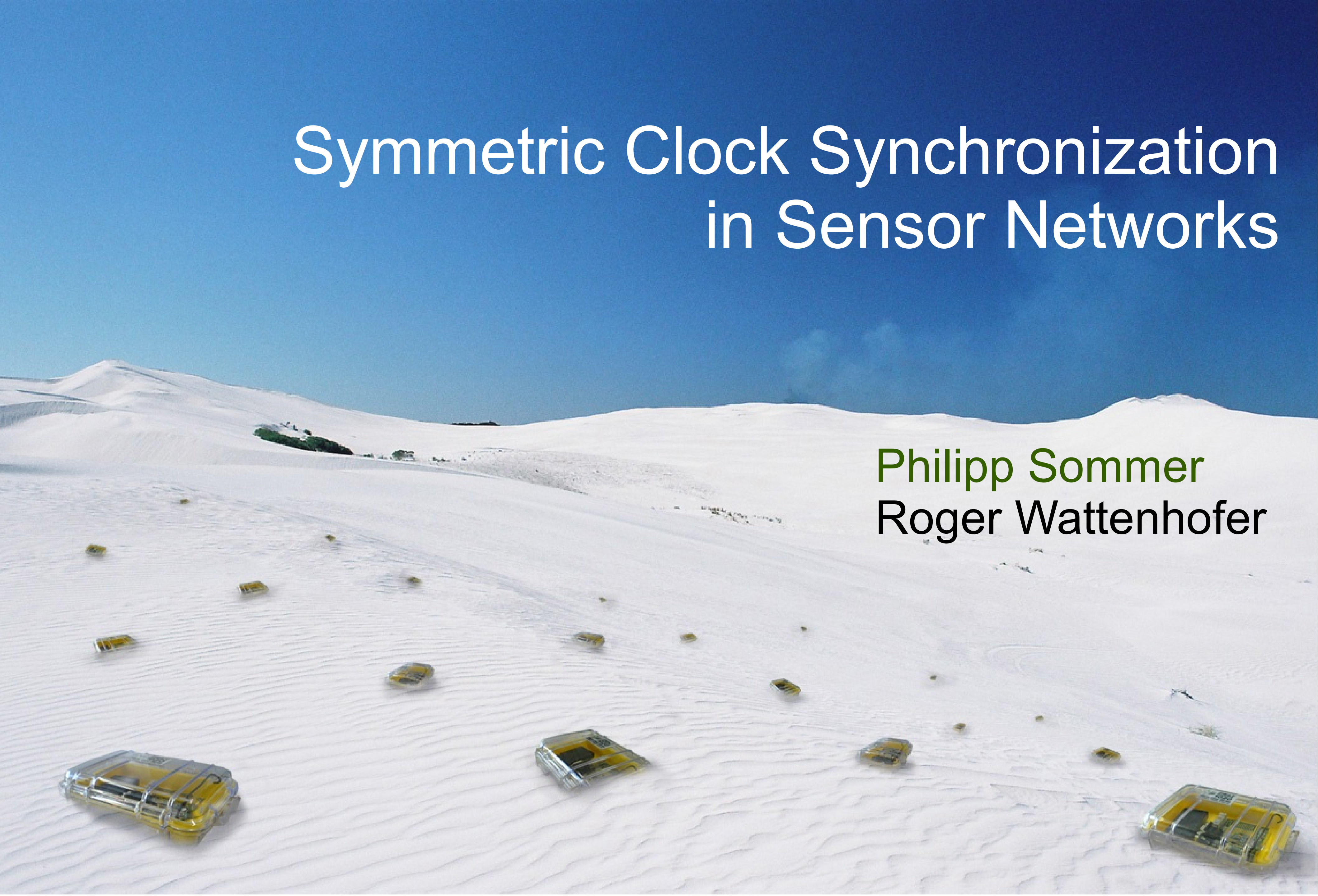
Symmetric Clock Synchronization in Sensor Networks

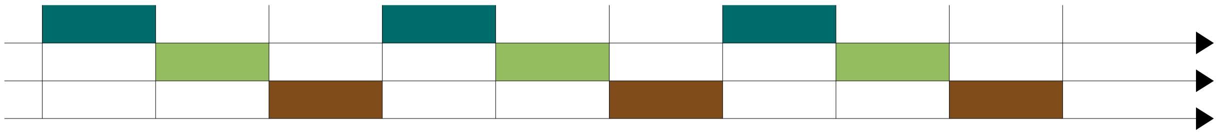


Philipp Sommer
Roger Wattenhofer

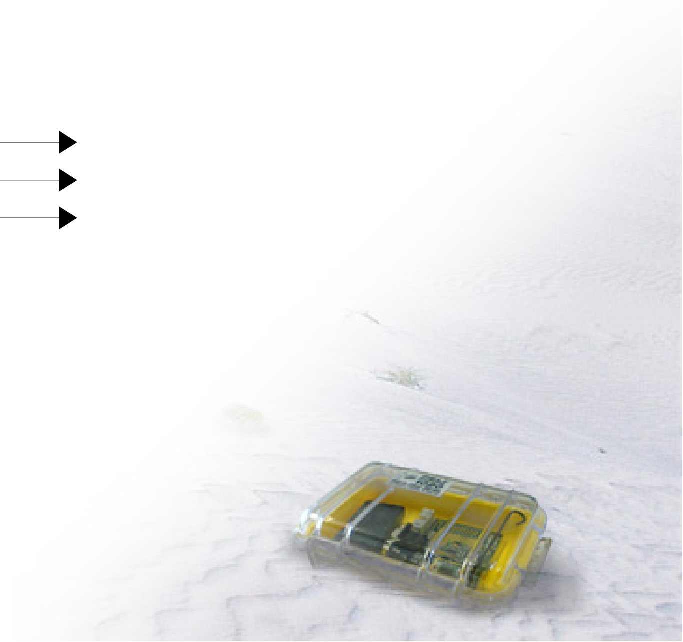
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Motivation

- Time synchronization is essential for many applications
  - Ordering of collected sensor data/events
  - Estimation of position information (e.g. shooter detection)
  - Coordination of wake-up and sleeping times (energy efficiency)
  - TDMA schedules



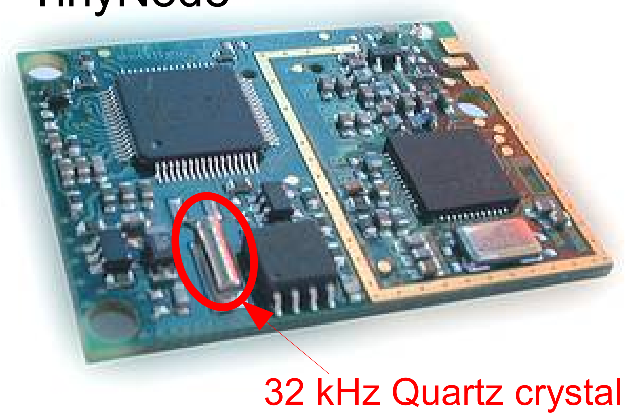
  - Co-operation of multiple sensor nodes

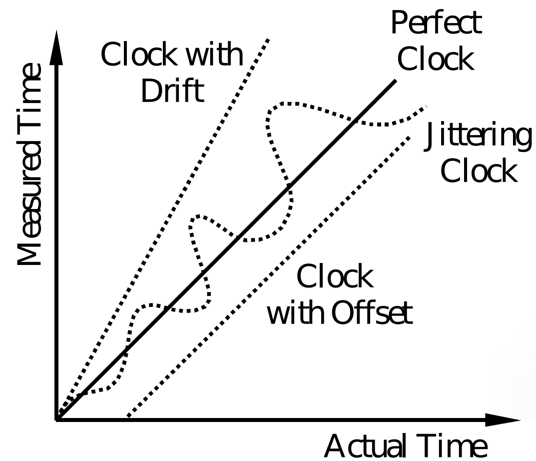


Hardware Clocks in Sensor Nodes

- Structure
  - External oscillator with a nominal frequency
  - Counter register is incremented with each oscillator pulse
  - Works also when the CPU is in sleep state
- Accuracy
  - Clock drift: deviation from the nominal rate dependent on temperature and other factors

TinyNode

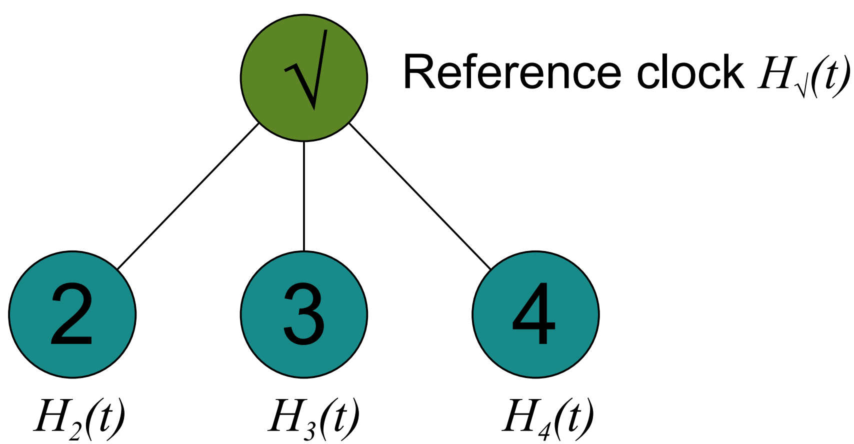


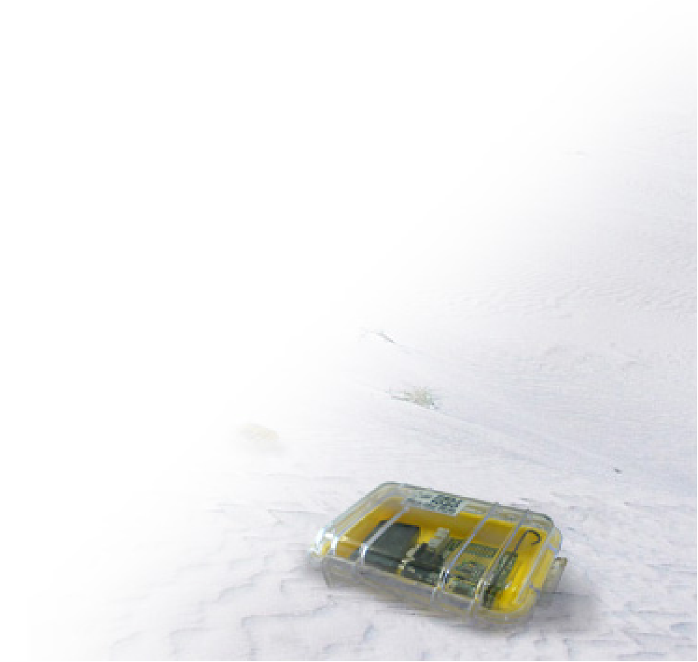


Single-Hop Clock Synchronization

- Each node maintains a hardware clock $H_i(t)$
  - Increases at the rate of the oscillator

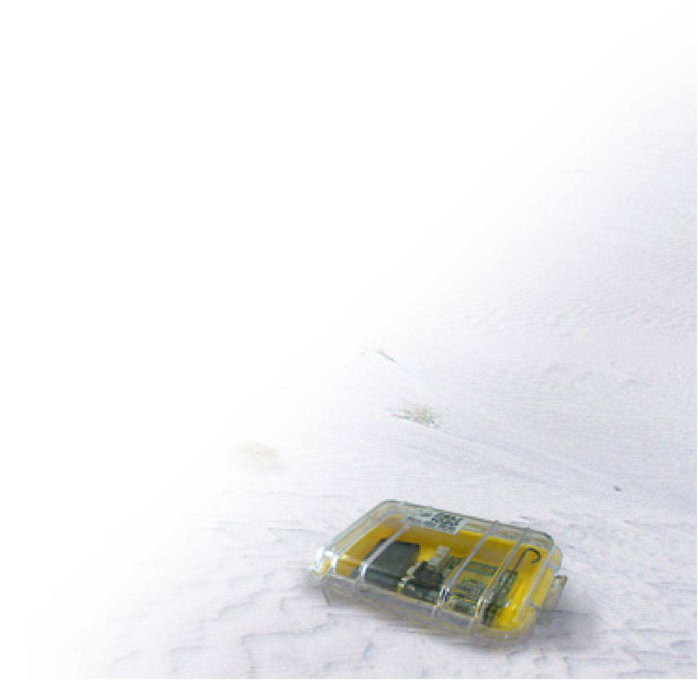
- Each node maintains a software clock $S_i(t) = \hat{H}_{\surd}(t)$
  - Estimation of the reference clock value based on $H_i(t)$





Time Synchronization

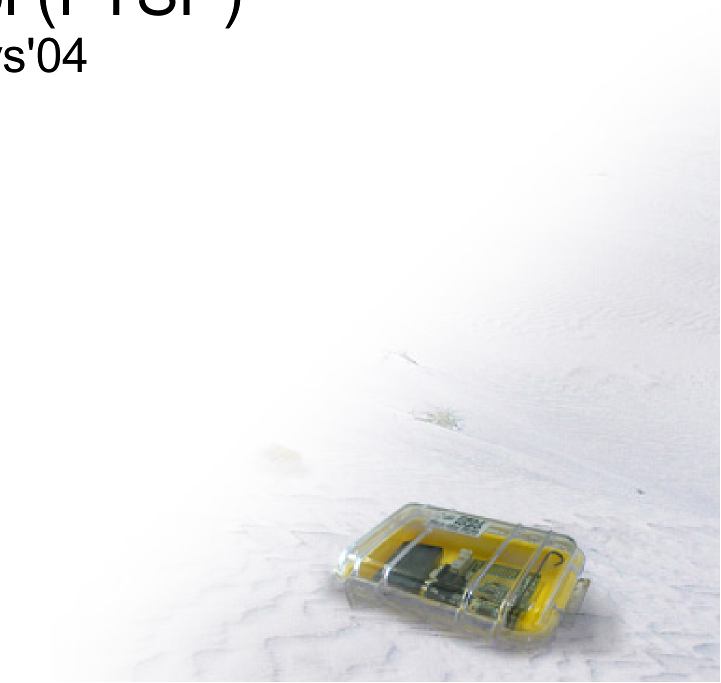
- Goals
  - Compensation of the offset between clocks
  - Compensation of relative drift
- Evaluation of the performance of an algorithm
  - Metrics
    - Average error, worst-case error, variance
  - Clock granularity
  - Distribution of the synchronization error
  - Message complexity



Related Work

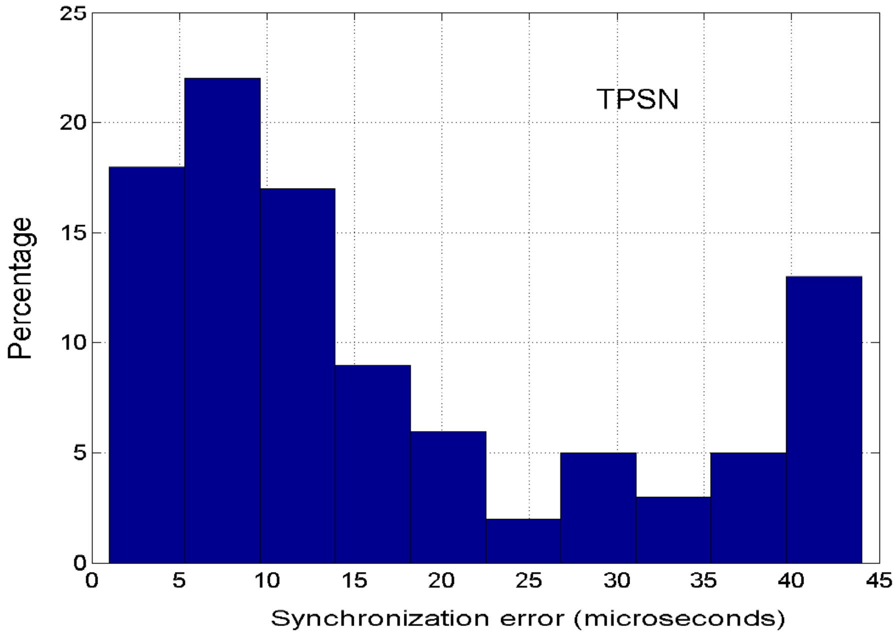
- Time synchronization for WSN is a well-studied problem:
  - Reference Broadcast Synchronization (RBS)
    J. Elson, L. Girod and D. Estrin, OSDI'02
  - Timing-sync Protocol for Sensor Networks (TPSN)
    S. Ganeriwal, R. Kumar and M. Srivastava, SenSys'03
  - Flooding Time Synchronization Protocol (FTSP)
    M. Maróti, B. Kusy, G. Simon and Á. Lédeczi, SenSys'04

  ...

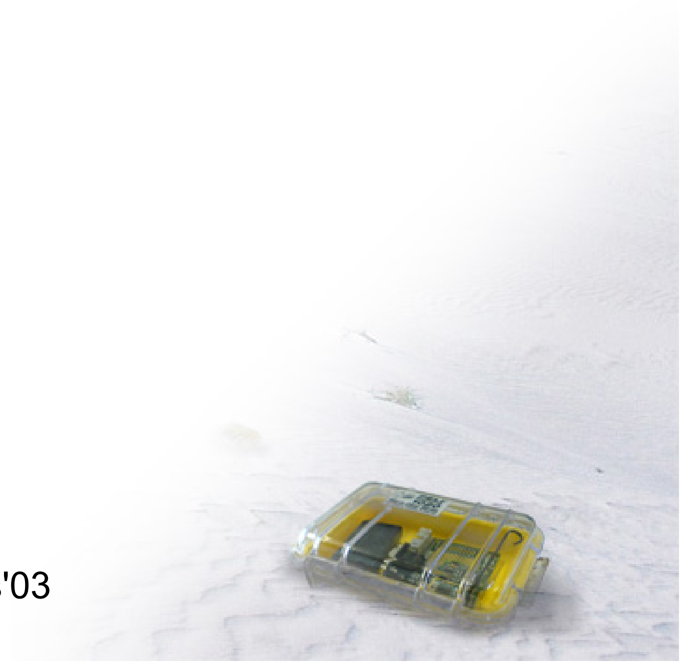


Synchronization Error

- Synchronization error with TPSN

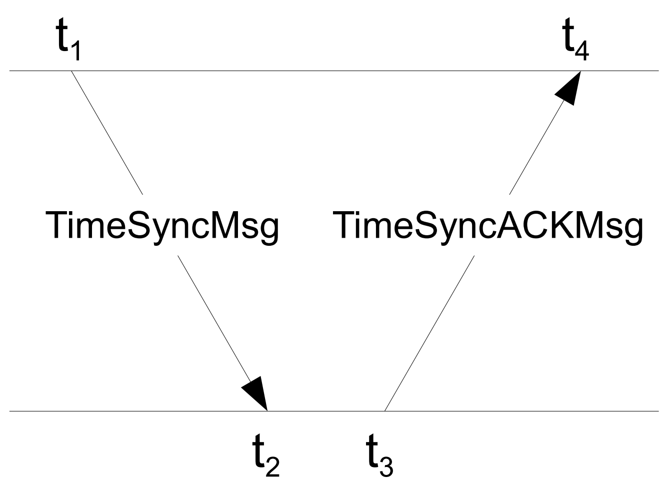


Source: Timing-sync Protocol for Sensor Networks, Ganeriwal et. al, SenSys'03



Synchronization Algorithm

- Sender-Receiver Synchronization (e.g. TPSN)
  - Two-way message exchange
  - Each message is timestamped at both the sending (t_1,t_3) and receiving node (t_2,t_4)
  - Transmission delay and clock offset can be calculated:

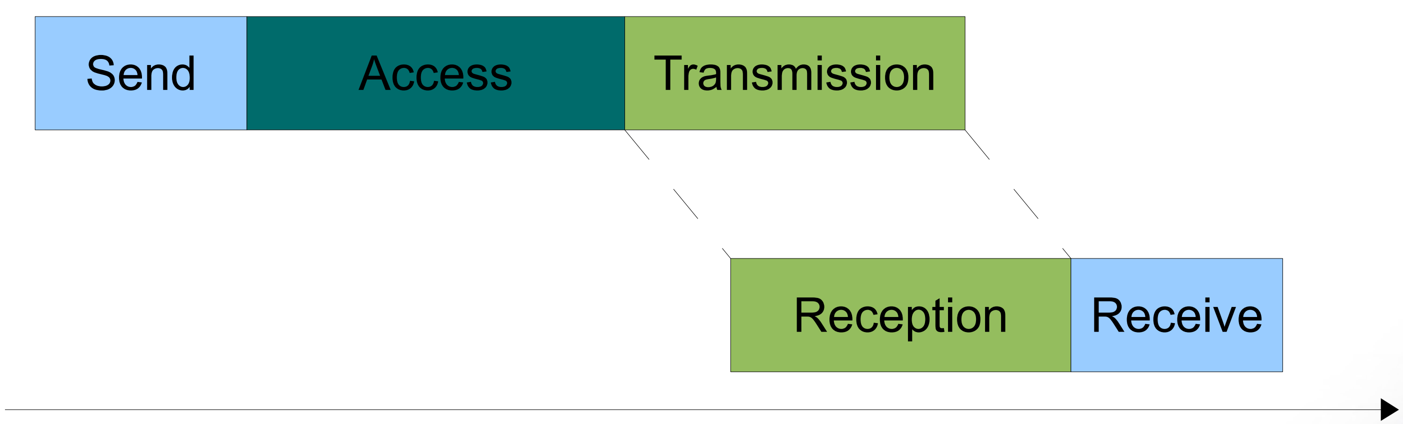


$$\delta = \frac{(t_4 - t_1) - (t_3 - t_2)}{2}$$

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

Sources of Errors

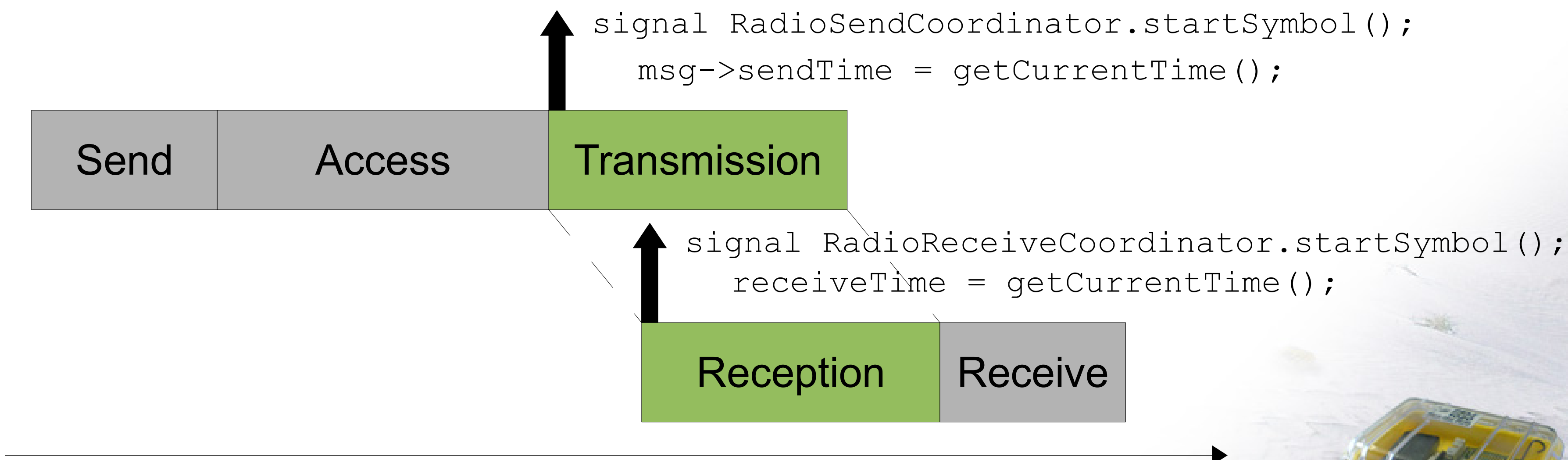
- Uncertainty in the message delay
  - Sending time (0-100 ms, non-deterministic)
  - Medium access time (10-500 ms, non-deterministic)
  - Propagation delay (<1 μs, deterministic)
  - Receive time (0-100 ms, non-deterministic)



MAC Layer Timestamping

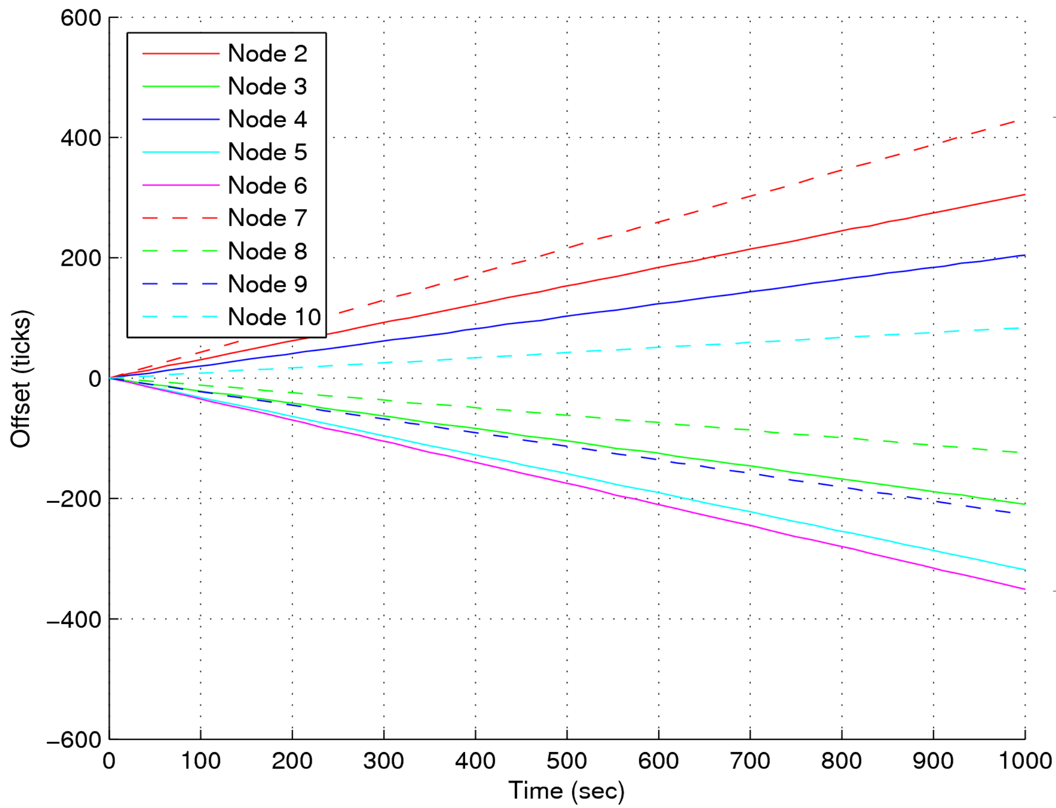
- Timestamping at the MAC layer eliminates *Send*, *Access* and *Receive* times
- Implementation in TinyOS 1.x
  - Interface `RadioCoordinator`

```
event void startSymbol(uint8_t bitsPerBlock, uint8_t offset, TOS_MsgPtr msg)
```



Clock Drift

- Crystal oscillators used in clocks have drift of 30-50 ppm
  - Measurement results with TinyNodes (32 kHz clock)

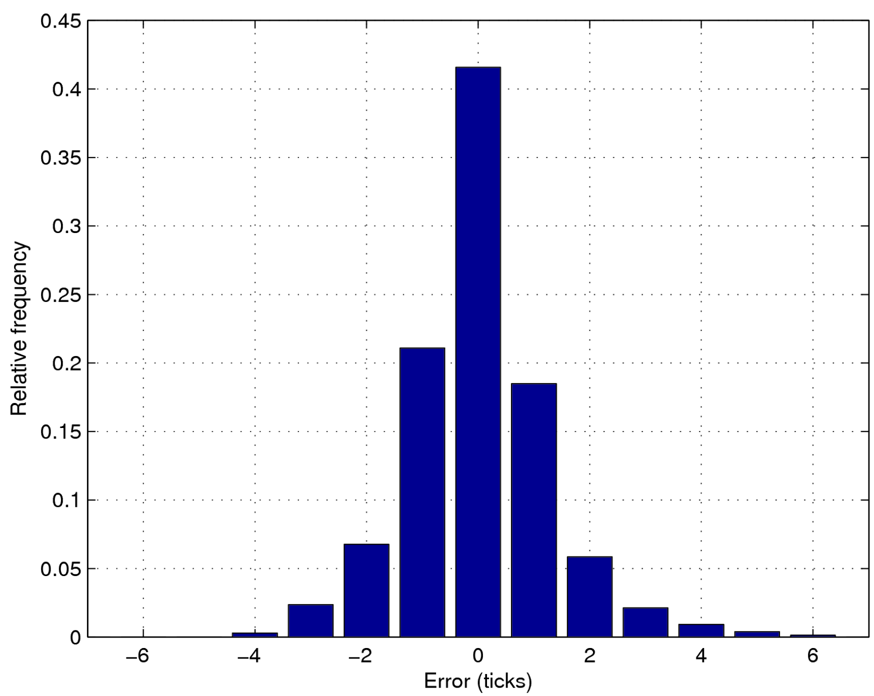


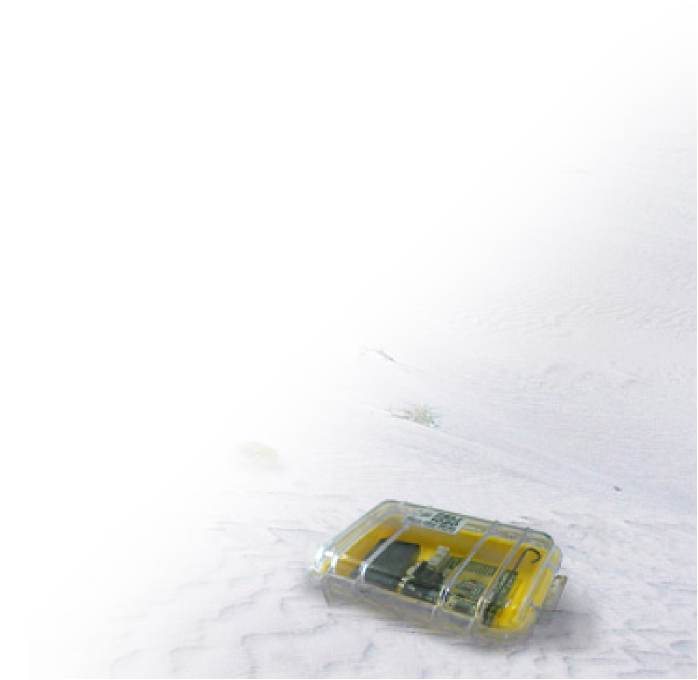
Offset of ~800 ticks (24 ms)
1000 s after the synchronization

  - Frequent re-synchronization is necessary

Drift Compensation

- Drift can be modelled as a linear function (plus noise)
  - Use linear regression to compensate drift (e.g. FTSP)
    - Problems: Rounding errors, memory consumption
  - Offline linear regression results:





Moving Average Filtering

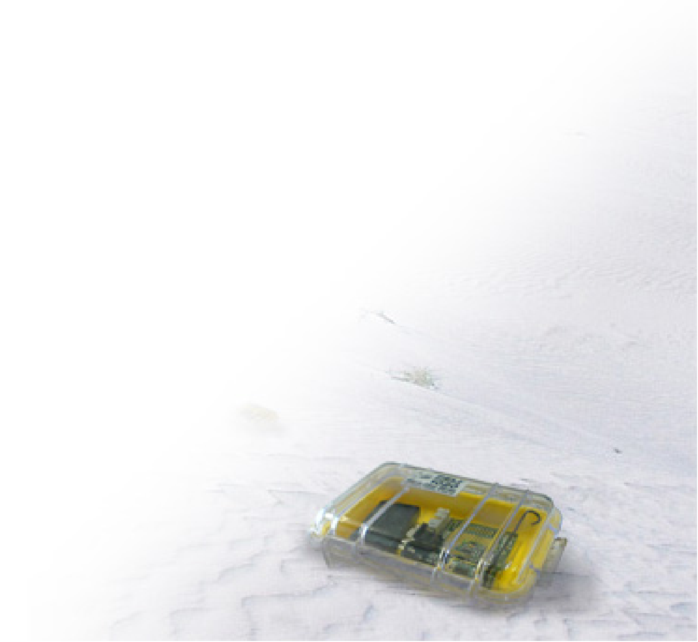
- Estimate the number of clock ticks after the offset needs to be corrected by a single tick
  - Example: Offset increases by 100 ticks during a synchronization interval of 30 s (~32768·30 ticks)

$$c_{ticks} = \frac{t_{sync}}{\Delta_{offset}} = \frac{32768 \cdot 30}{100} = 9830.4$$

$\rightarrow$ increase offset every 9830 ticks

- Moving average filter

$$c_{ticks_{avg}}(t) = \alpha \cdot c_{ticks} + (1 - \alpha) \cdot c_{ticks_{avg}}(t-1)$$



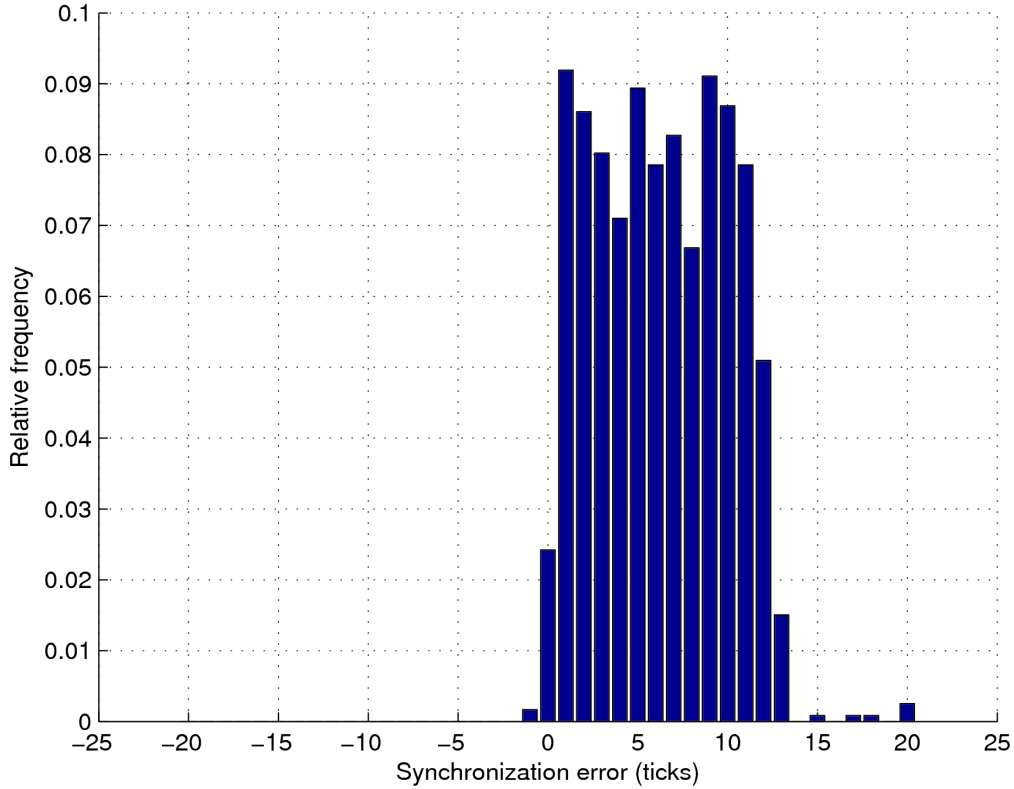
Evaluation

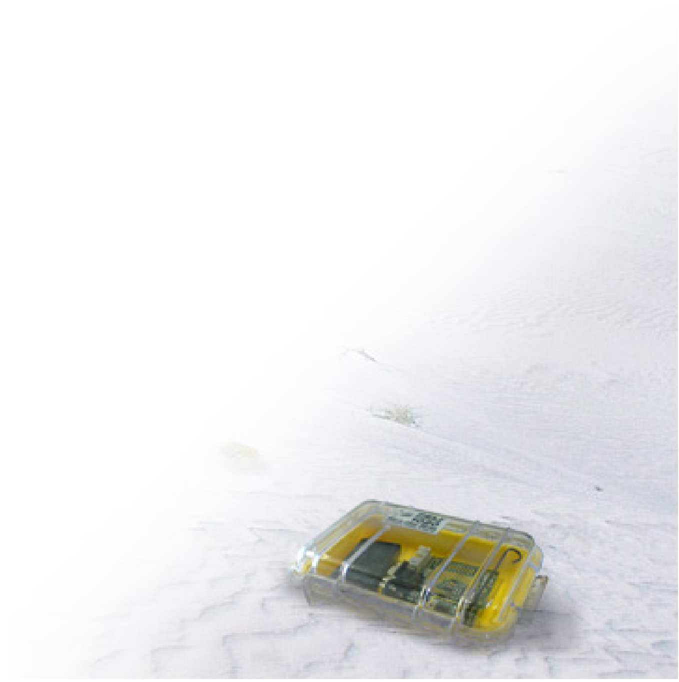
- Hardware Platform
  - TinyNode 584 (TI MSP430, Semtech XE1205 Radio)
- Indoor testbed consisting of 11 nodes
  - 1 reference node, 9 child nodes
  - 1 sniffer node (TOSBase)

- Time Probing
  - Sniffer node periodically broadcasts probe messages
  - Each node timestamps the reception of a probe message with the local time and the estimation of the reference time
  - Events are logged to the external flash memory

Measurement Results without Drift Compensation

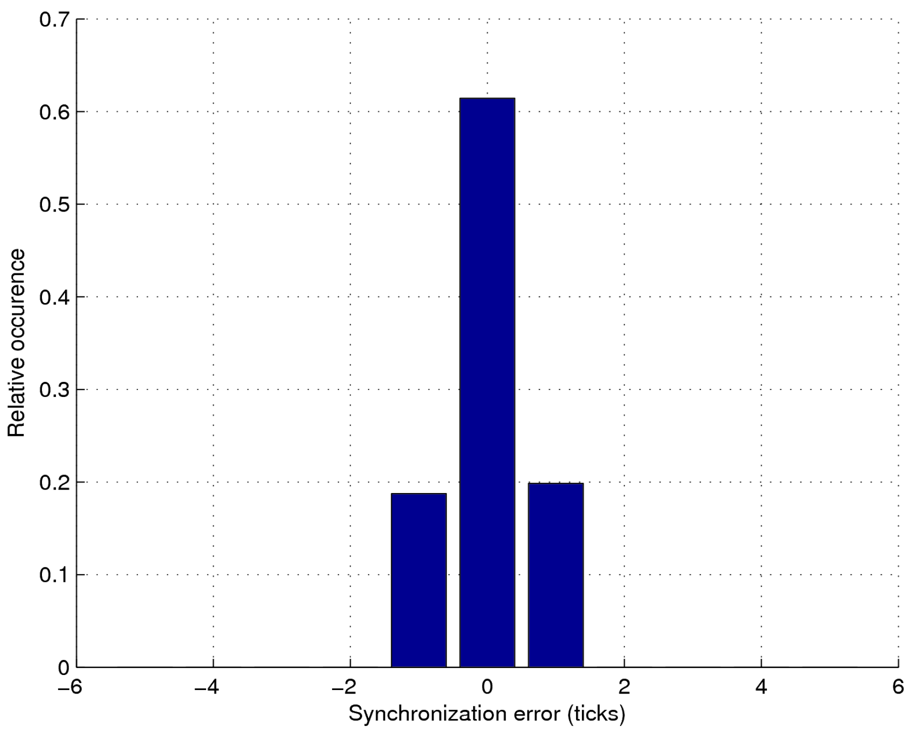
- Synchronization error (t_{sync} = 30 s)
  - Average error: 6.28 clock ticks (191.54 µs)
  - Worst-case error: 20 ticks (610 µs)





Measurement Results with Drift Compensation

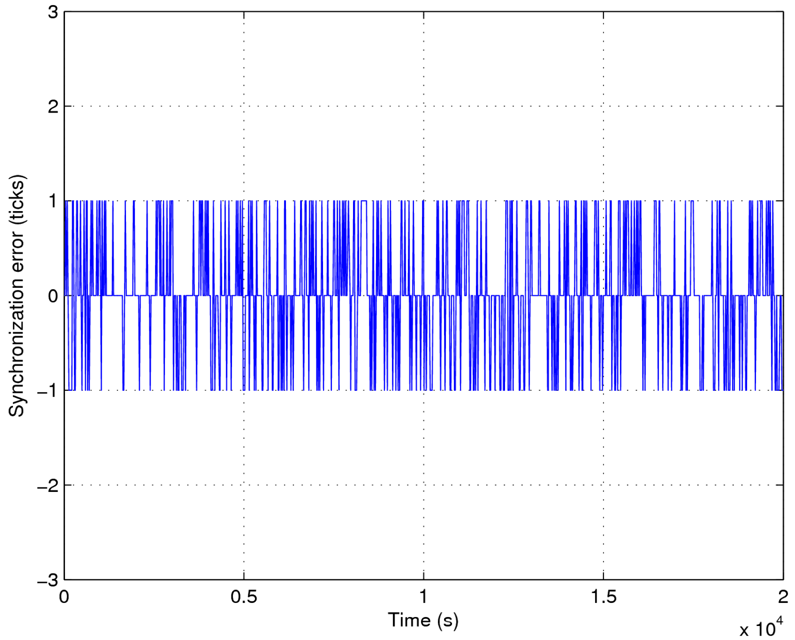
- Synchronization error (t_{sync} = 30 s)
  - Average error reduces to 0.37 ticks (11.32 μs)
  - Worst-case error is only a *single* clock tick
  - Symmetric distribution of remaining error (no bias)



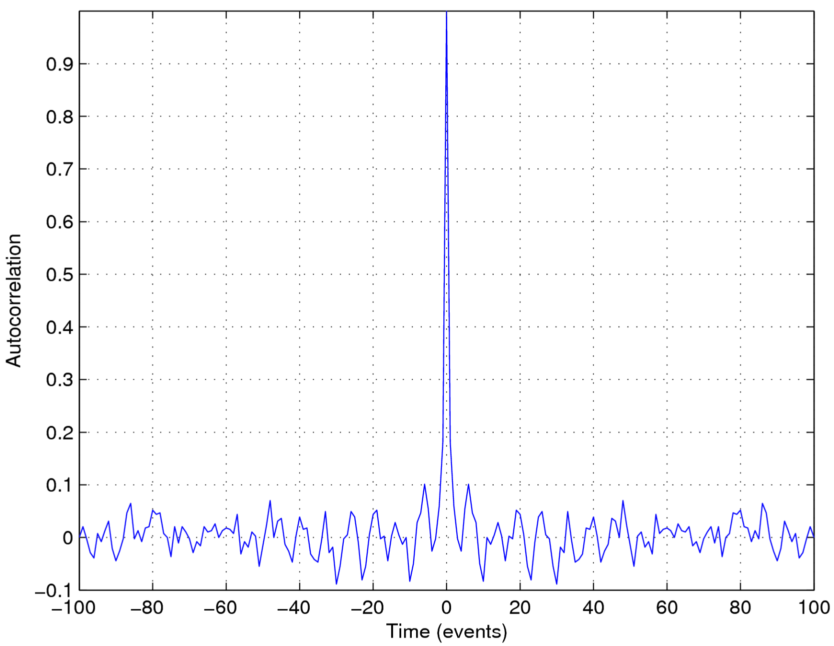
Distribution of the Synchronization Error

- Positive and negative errors are uncorrelated

Synchronization error

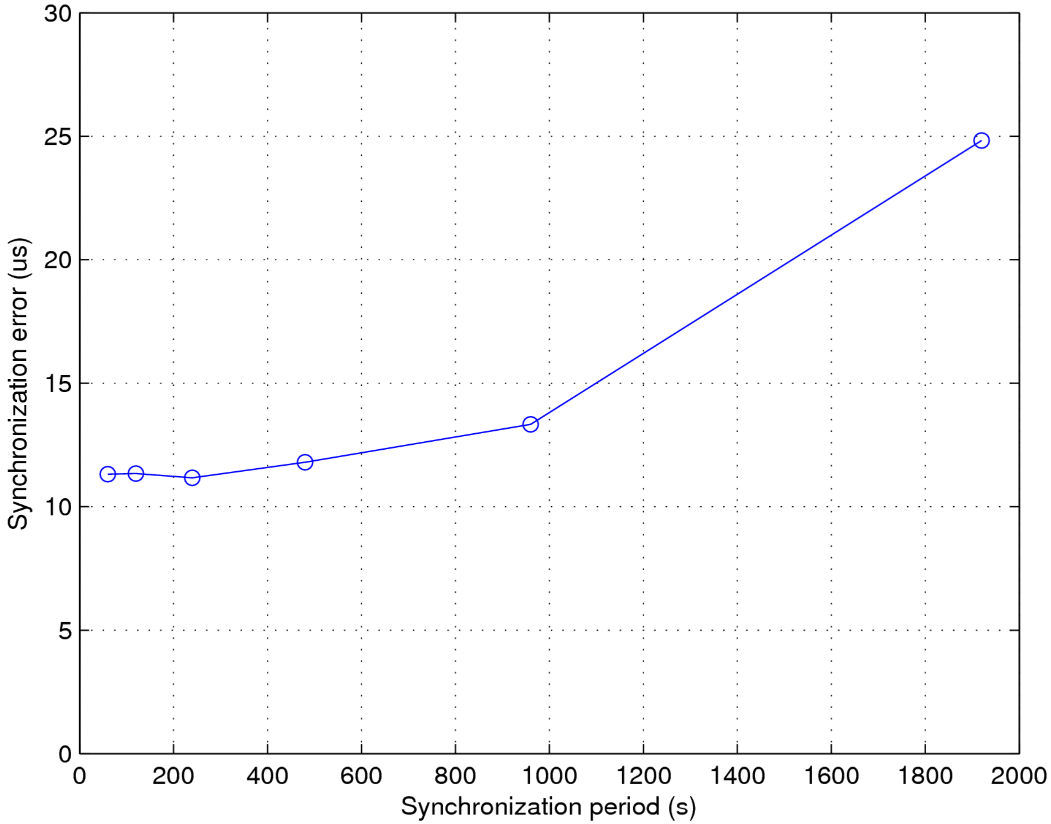


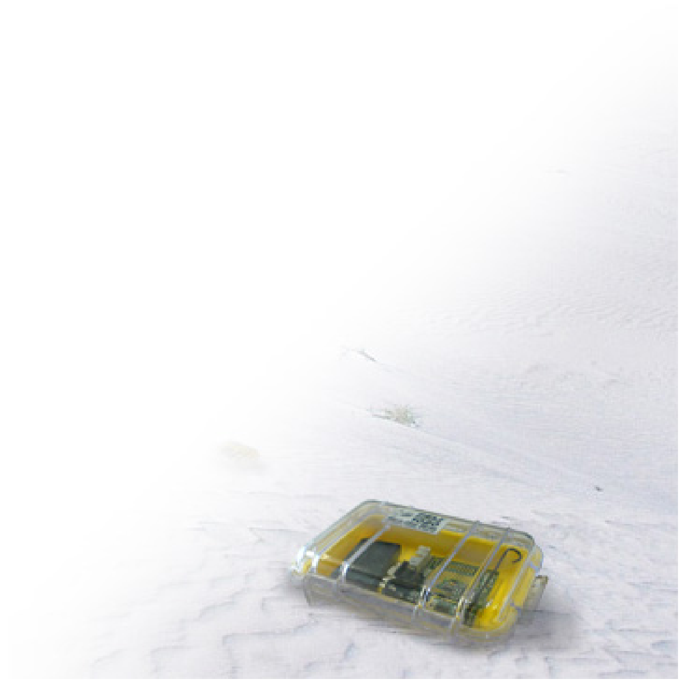
Auto-correlation of the error time series



Energy Efficiency

- Minimize the number of synchronization messages to reduce the energy consumption → increase sync interval
- Impact of the synchronization interval on the accuracy:





Clock Granularity of Sensor Hardware

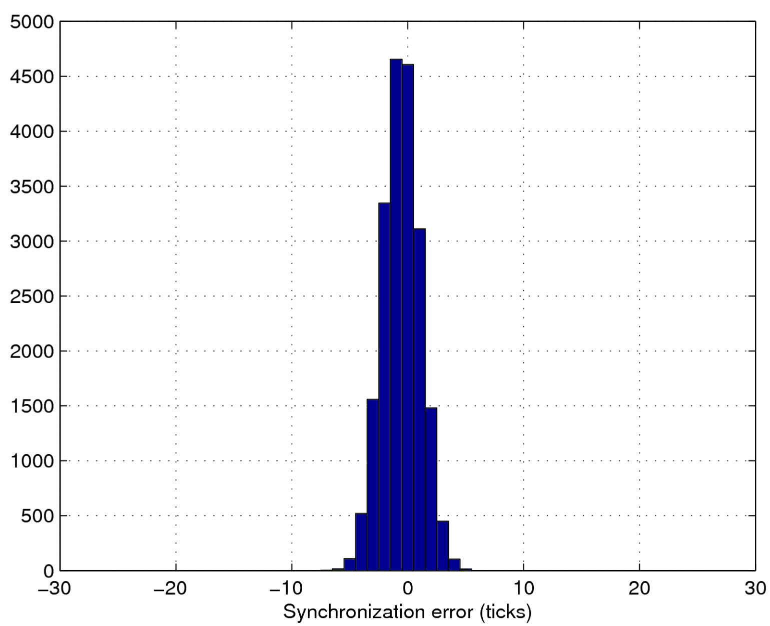
- Clock sources available in common sensor hardware

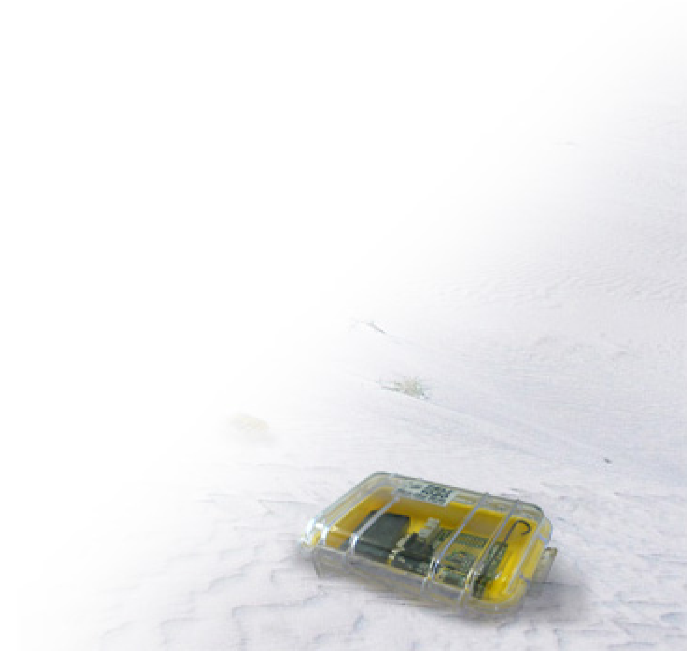
Product	System clock	Oscillators
TinyNode/Tmote Sky	8 MHz	32 kHz
Mica2/BTnode	7.37 MHz	7.37 MHz, 32 kHz

- External oscillators as hardware clock source
  - High stability
  - Continue to operate when CPU is in sleep mode
- Berkeley Mica2 motes
  - External 7.37 MHz quartz oscillator → 921.5 kHz clock
  - Clock granularity: 1 tick ~ 1 μs

Results on the Mica2 Platform

- Quick follow-up experiment with Mica2 nodes
  - Average synchronization error: 1.3 ticks (1.2 μs)
    - FTSP: Avg. error of 1.48 μs for single-hop
  - Worst-case error: 7 ticks (6.4 μs)
  - Open problem: Error has a small bias (-0.5 ticks)





Conclusion

- Implementation and evaluation of a sender-receiver based synchronization algorithm
- Drift compensation using a moving average filter
- Accuracy in the order of the clock granularity
- Distribution of the remaining error is symmetric and uncorrelated