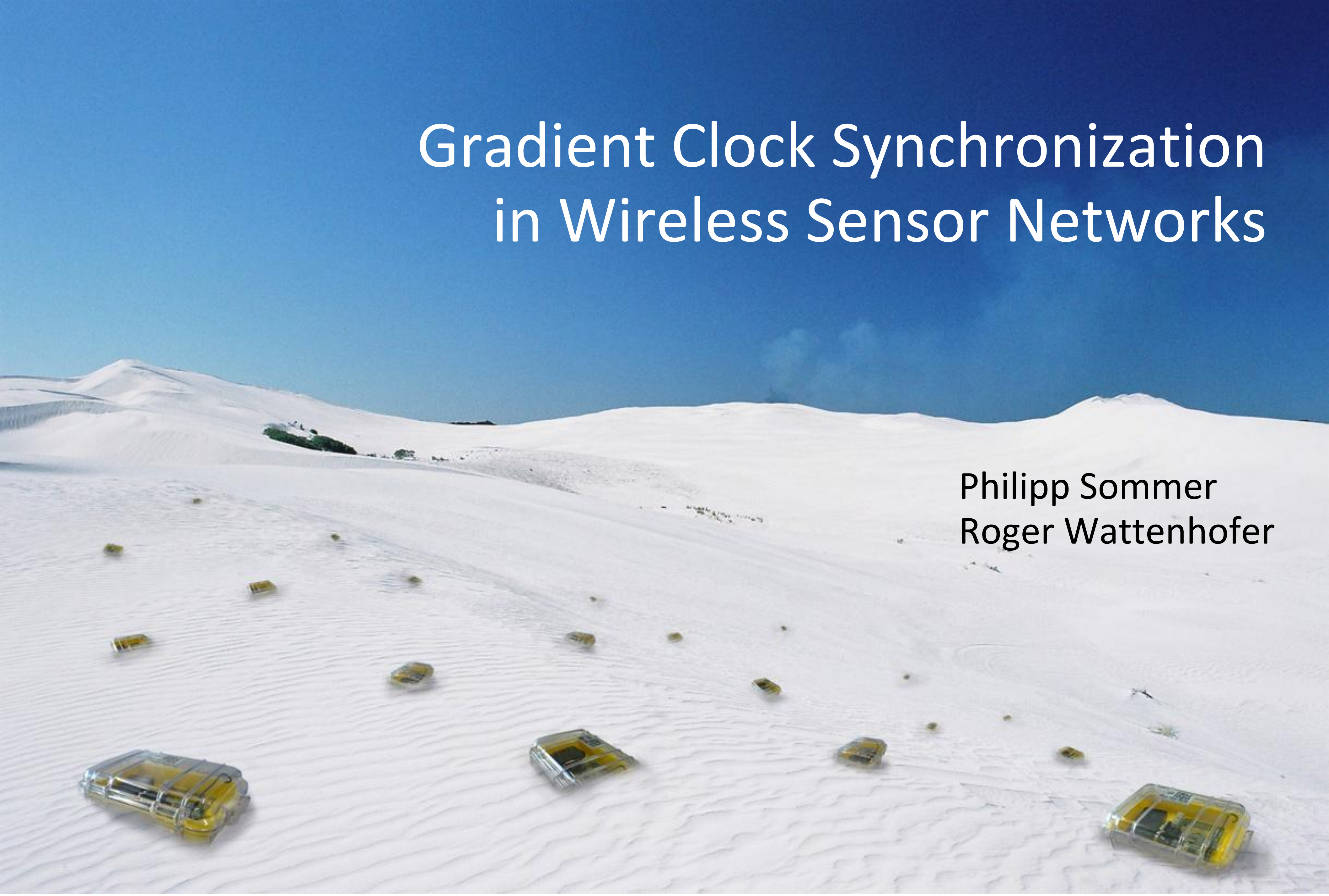


Gradient Clock Synchronization in Wireless Sensor Networks

Philipp Sommer
Roger Wattenhofer

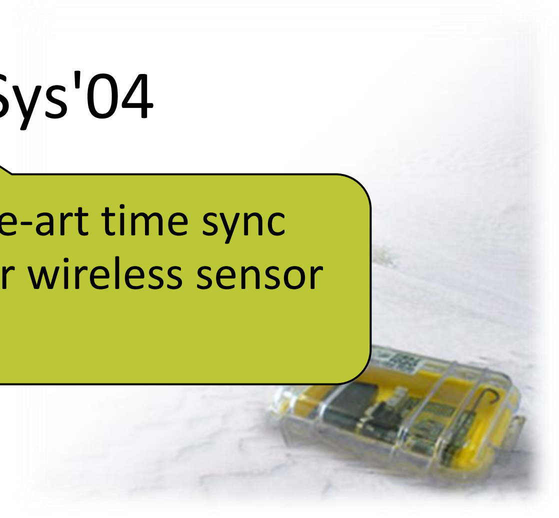
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing Group

Time Synchronization is a well-studied Problem

- *Time, Clocks, and the Ordering of Events in a Distributed System*
  L. Lamport, Communications of the ACM, 1978.
- *Internet Time Synchronization: The Network Time Protocol*
  D. Mills, IEEE Transactions on Communications, 1991
- *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
- and many more ...

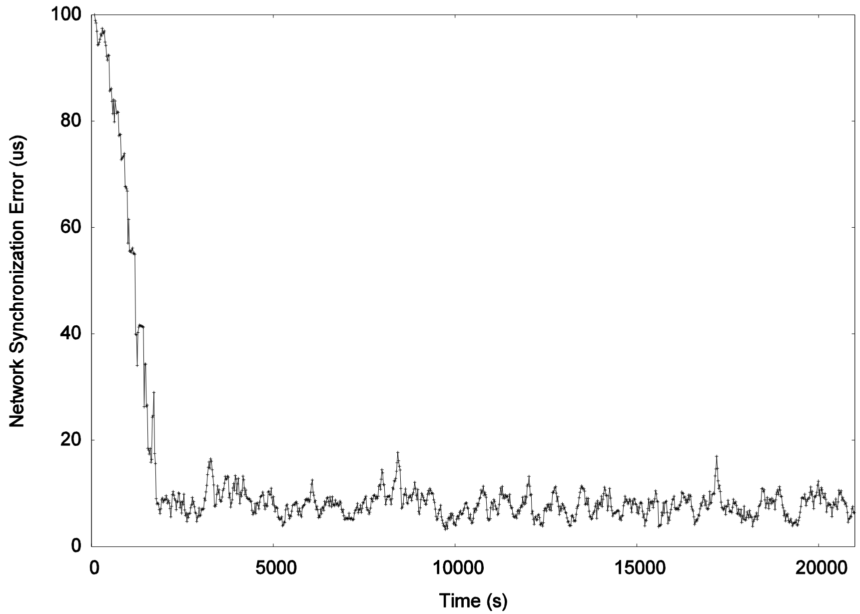
State-of-the-art time sync protocol for wireless sensor networks



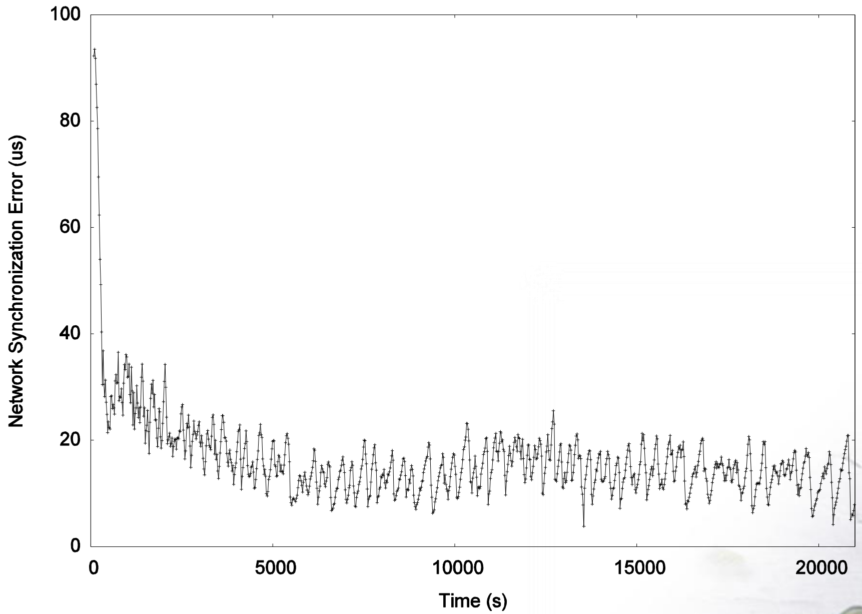
Preview: FTSP vs. GTSP

- Gradient Time Synchronization Protocol (GTSP) NEW
  - Details will follow soon
- Network synchronization error *(global skew)*
  - Pair-wise synchronization error between any nodes in the network

FTSP (avg: 7.7 µs)



GTSP (avg: 14.0 µs)



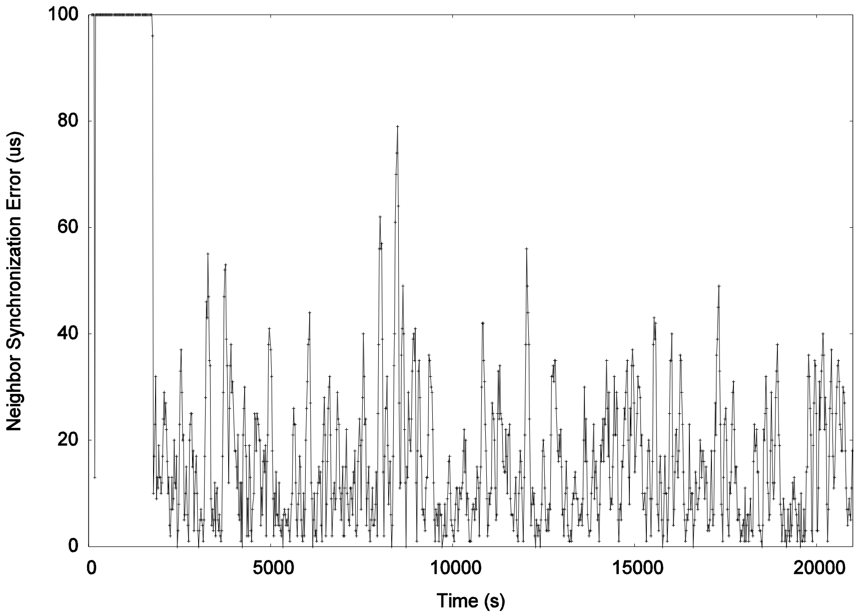
Preview: FTSP vs. GTSP (2)

- Neighbor Synchronization error *(local skew)*

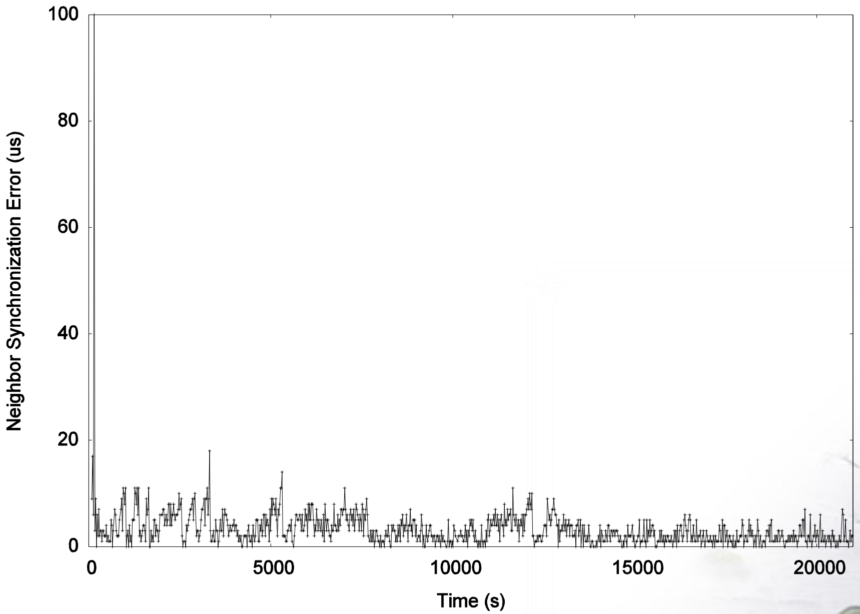
  Pair-wise synchronization error between neighboring nodes

- Synchronization error between two *direct* neighbors

FTSP (avg: 15.0 µs)



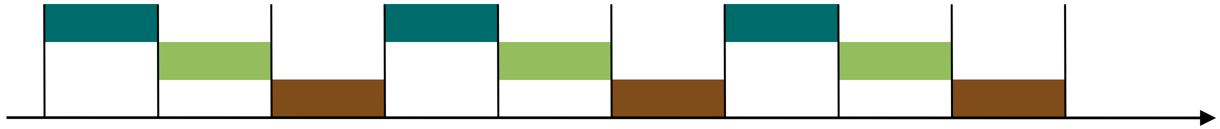
GTSP (avg: 2.8 µs)



Time in Sensor Networks

- Common time is essential for many applications:
  - Global — Assigning a global timestamp to sensed data/events
  - Global — Co-operation of multiple sensor nodes
  - Local — Precise event localization (e.g., shooter detection)
  - Local — Coordination of wake-up and sleeping times (energy efficiency)

  - Local — TDMA-based MAC layer

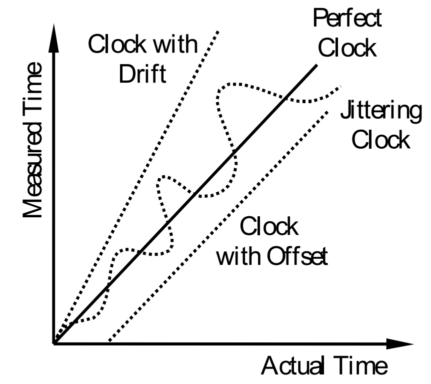


Outline

- ✓ Introduction
- Clock Synchronization Basics
- Gradient Time Synchronization Protocol (GTSP)
- Evaluation
- Conclusions

Sensor Node Clocks

- Each node has a hardware clock *H(t)*
  - Counter register of the microcontroller
  - Crystal quartz oscillator (e.g., 32kHz, 7.37 MHz)
  - Subject to clock drift (30 ppm)

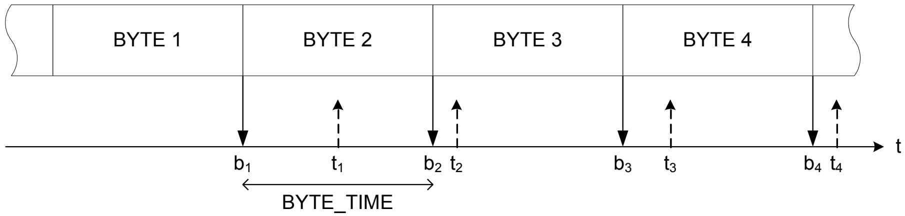


- Each node has a logical clock *L(t)*
  - Holds the estimation of the current global time
  - Computed as a function of the current hardware clock *H(t)*
  - Logical clock rate

Clock Synchronization Algorithm

- Exchange messages with current clock value L(t) with others
  - Adjust clock rates and offset
  - Repeat this process frequently

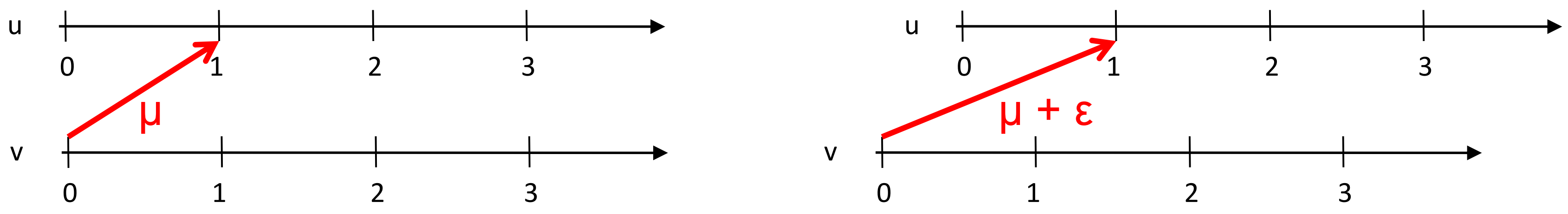
- Uncertainty (jitter) in the message delay
  - Various sources of errors (deterministic and undeterministic)
  - Can be reduced (but not eliminated) by timestamping at MAC layer



Theoretical Bounds on the Synchronization Accuracy

- Two nodes *u* and *v* cannot be synchronized perfectly

  Worst-case example:



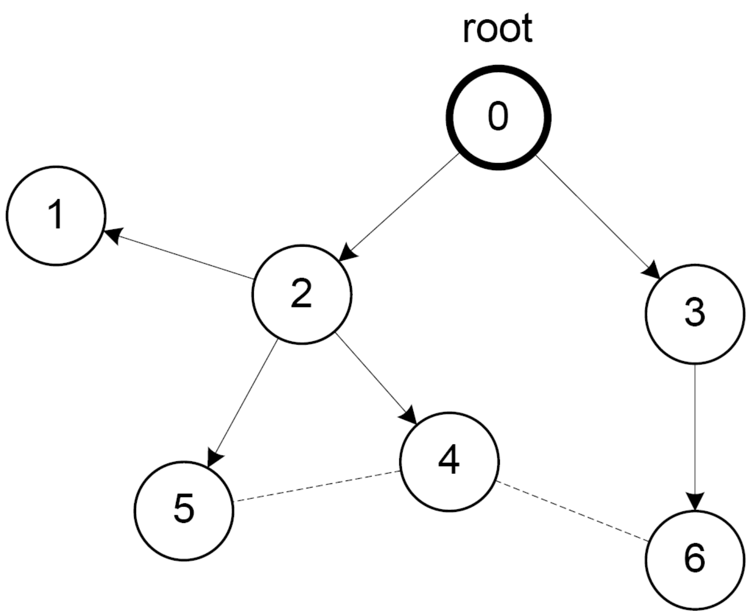
- Error increases with distance from the reference node
- Lower bound result from theoretical work

  Clock error between nodes distance *d* apart depends on the network diameter *D*: $\Omega(d + \frac{\log D}{\log \log D})$

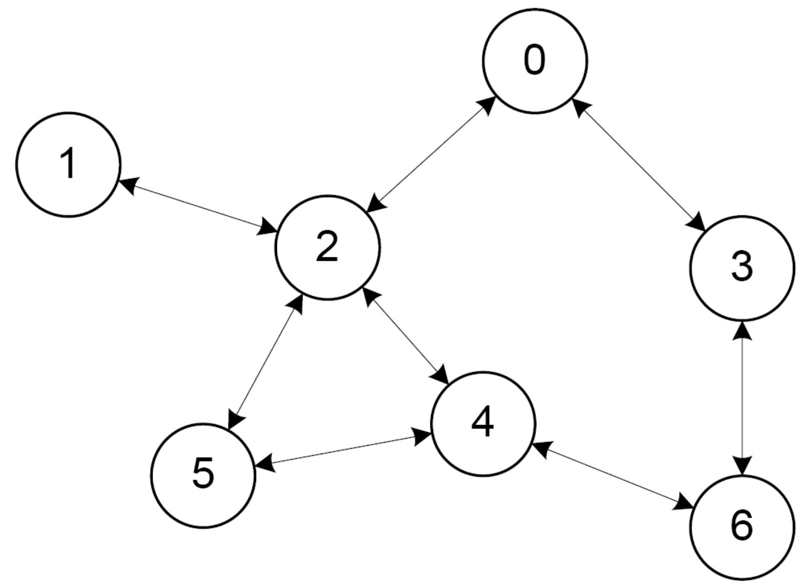
  R. Fan and N. Lynch. Gradient Clock Synchronization. In *PODC '04: Proceedings of the twenty-third annual ACM symposium on principles of distributed computing, 2004.*

Gradient Clock Synchronization

- Global property: Minimize clock error between any two nodes
- Local ("gradient") property: Small clock error between two nodes if the distance between the nodes is small.



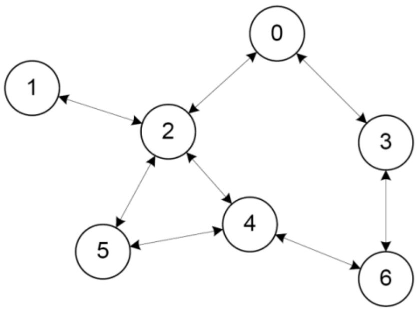
FTSP



GTSP

Gradient Time Synchronization Protocol (GTSP)

- Synchronize with **all** neighboring nodes
  - Broadcast periodic time beacons, e.g., every 30 s
  - No reference node necessary



- How to synchronize clocks without having a leader?
  - Follow the node with the fastest/slowest clock?
  - Idea: Go to the average clock value/rate of all neighbors (including node itself)

Drift and Offset Compensation in GTSP

- Update rule for the logical clock rate:

$$x_i(t_{k+1}) = \frac{\left(\sum_{j \in \mathcal{N}_i} x_j(t_k)\right) + x_i(t_k)}{|\mathcal{N}_i| + 1}$$

- Update rule for the logical clock offset:

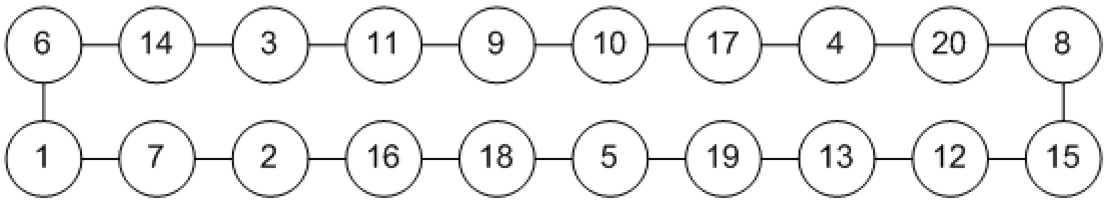
$$\theta_i(t_{k+1}) = \theta_i(t_k) + \frac{\sum_{j \in \mathcal{N}_i} L_j(t_k) - L_i(t_k)}{|\mathcal{N}_i| + 1}$$

Note: We will jump directly to a higher clock value if the offset exceeds a certain threshold, e.g., 20 µs.

Experimental Evaluation

- Mica2 platform using TinyOS 2.1
  - System clock: 7.37 MHz (crystal quartz)
  - Hardware clock: System clock divided by 8 = 921 kHz
  - Clock granularity of 1 microsecond (1 clock tick ≈ 1 µs)

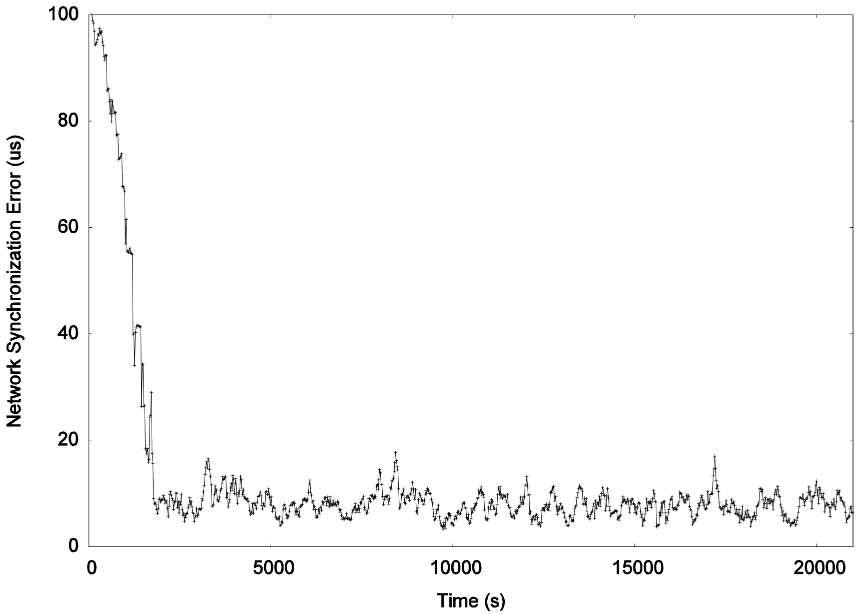
- Testbed of 20 Mica2 nodes
  - Base station triggers external events by sending time probe packets
  - Ring topology is enforced by software



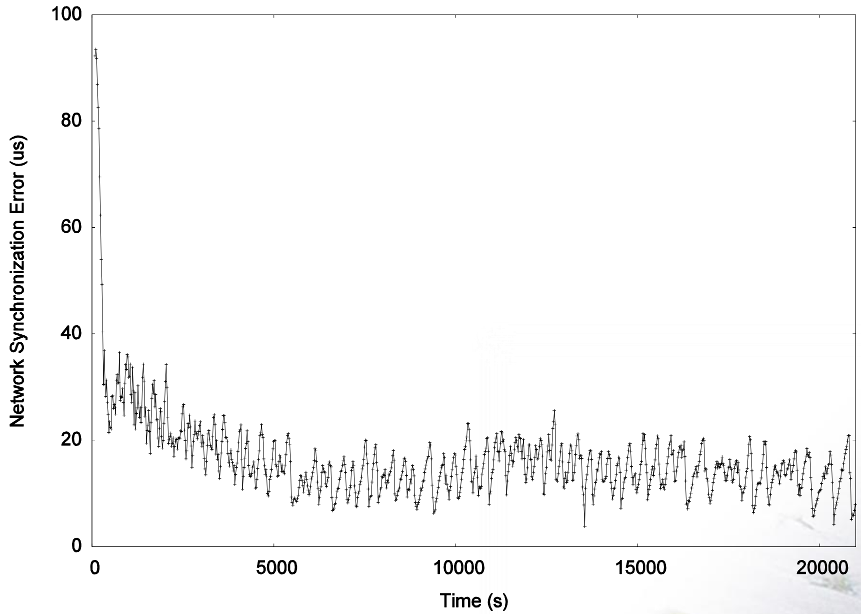
Experimental Results

- Network synchronization error *(global clock skew)*
  - 7.7 µs with FTSP, 14.0 µs with GTSP
- FTSP needs more time to synchronize all nodes after startup

FTSP



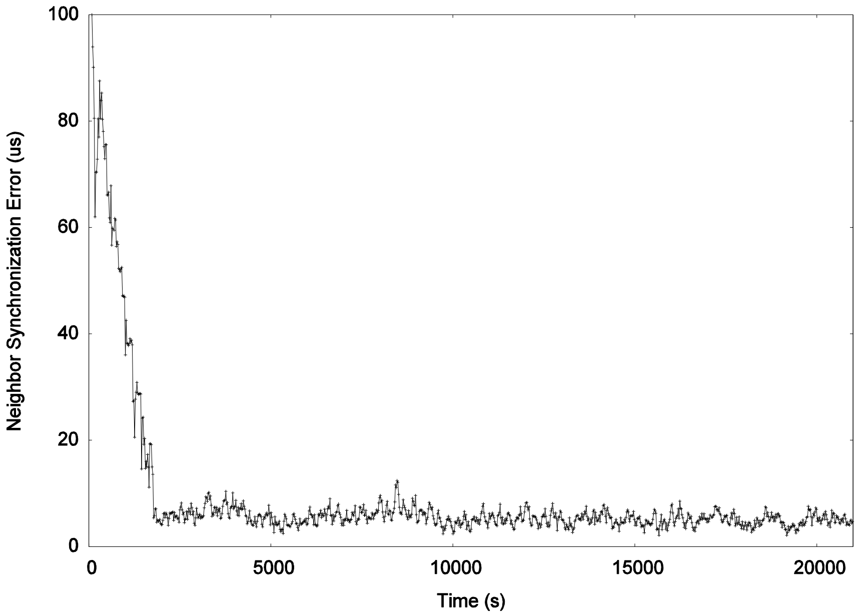
GTSP



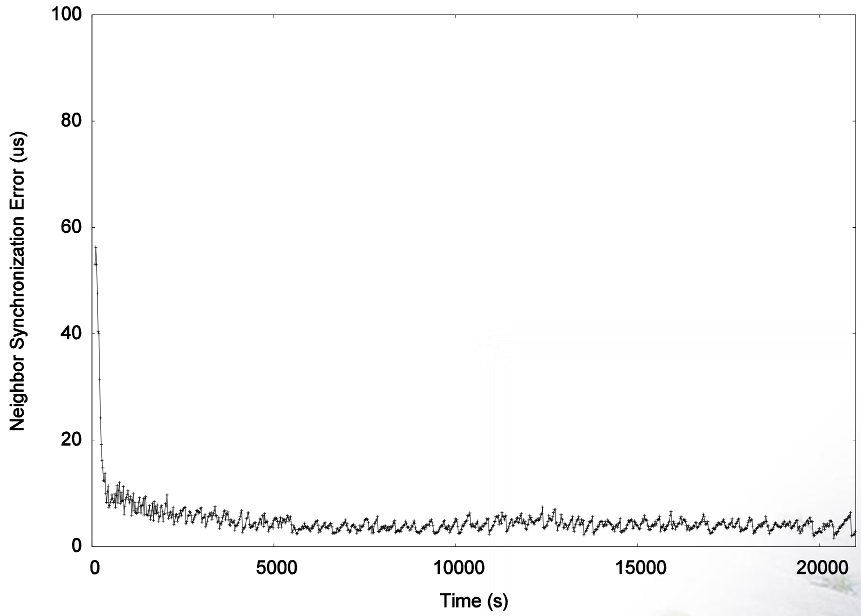
Experimental Results (2)

- Neighbor synchronization error *(local clock skew)*
  - 5.3 µs with FTSP, 4.0 µs with GTSP

FTSP

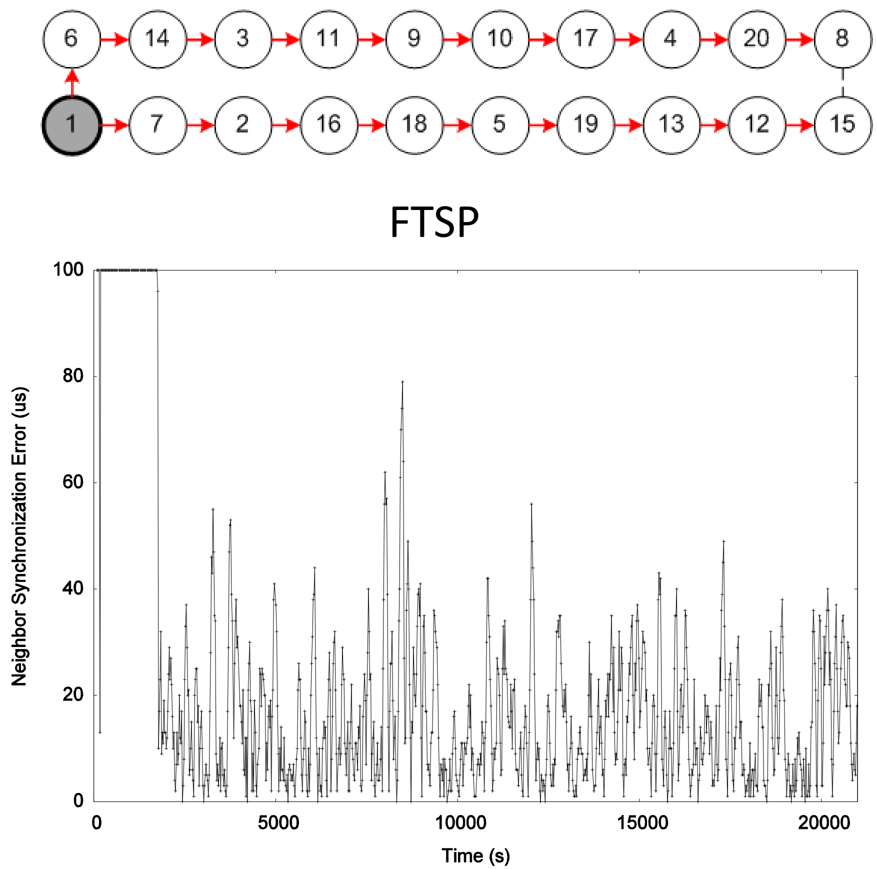


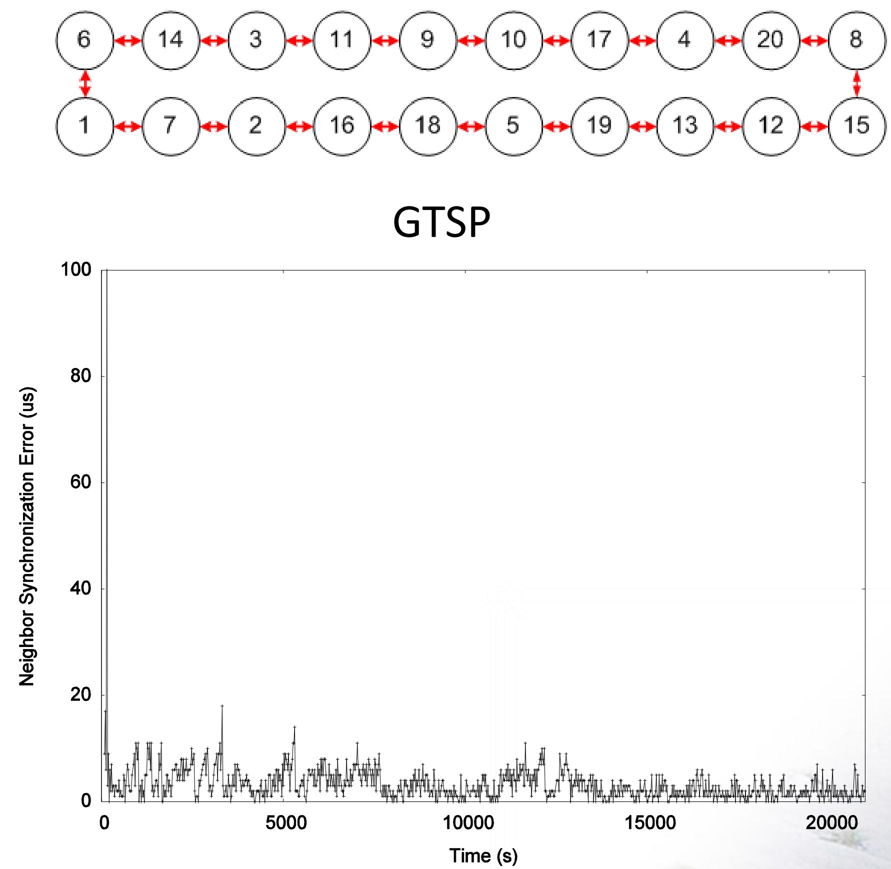
GTSP



Neighbor Synchronization Error: FTSP vs. GTSP

- FTSP has a large clock error for neighbors with large stretch in the tree (Node 8 and Node 15)





Multi-Hop Time Synchronization in Practice

- Is this really a problem in practice?

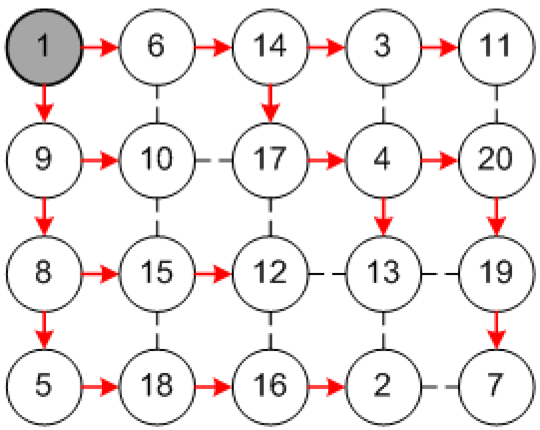
  Ring topology of 20 nodes seems to be „artificial“!?

- Finding a tree-embedding with low stretch is hard

  In a n = m*m grid you will have two neighbors with a stretch of at least $\sqrt{n}$

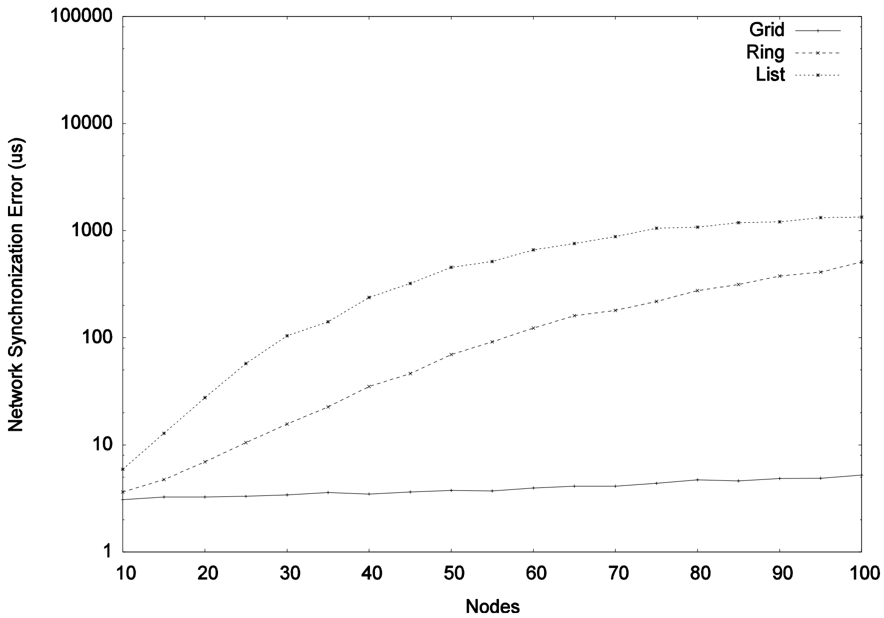
  Example: FTSP on a 5x4 grid topology

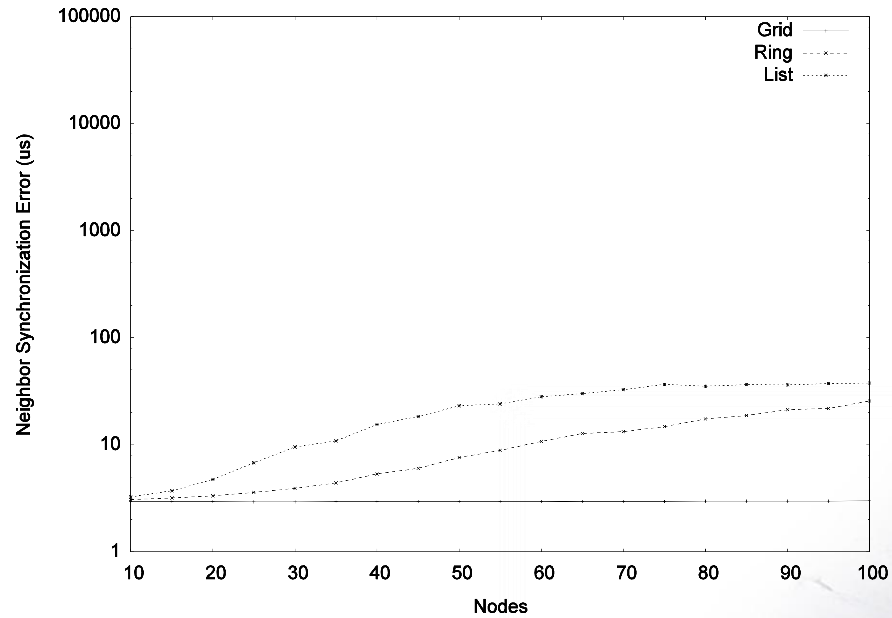
  Node 2 and 7 have a distance of 13 hops!



Simulation Results

- Simulation of GTSP for larger network topologies
  - Network error of ~1 ms for 100 nodes in a line topology
  - Neighbor error below 100 µs for the same topology





Conclusions and Future Work

- Gradient Time Synchronization Protocol (GTSP)
  - Distributed time synchronization algorithm (no leader)
  - Improves the synchronization error between neighboring nodes while still providing precise network-wide synchronization
  - Bridging the gap between theory and practice

- Is there a „perfect“ clock synchronization protocol?
  - Goal: Minimizing local and global skew at the same time