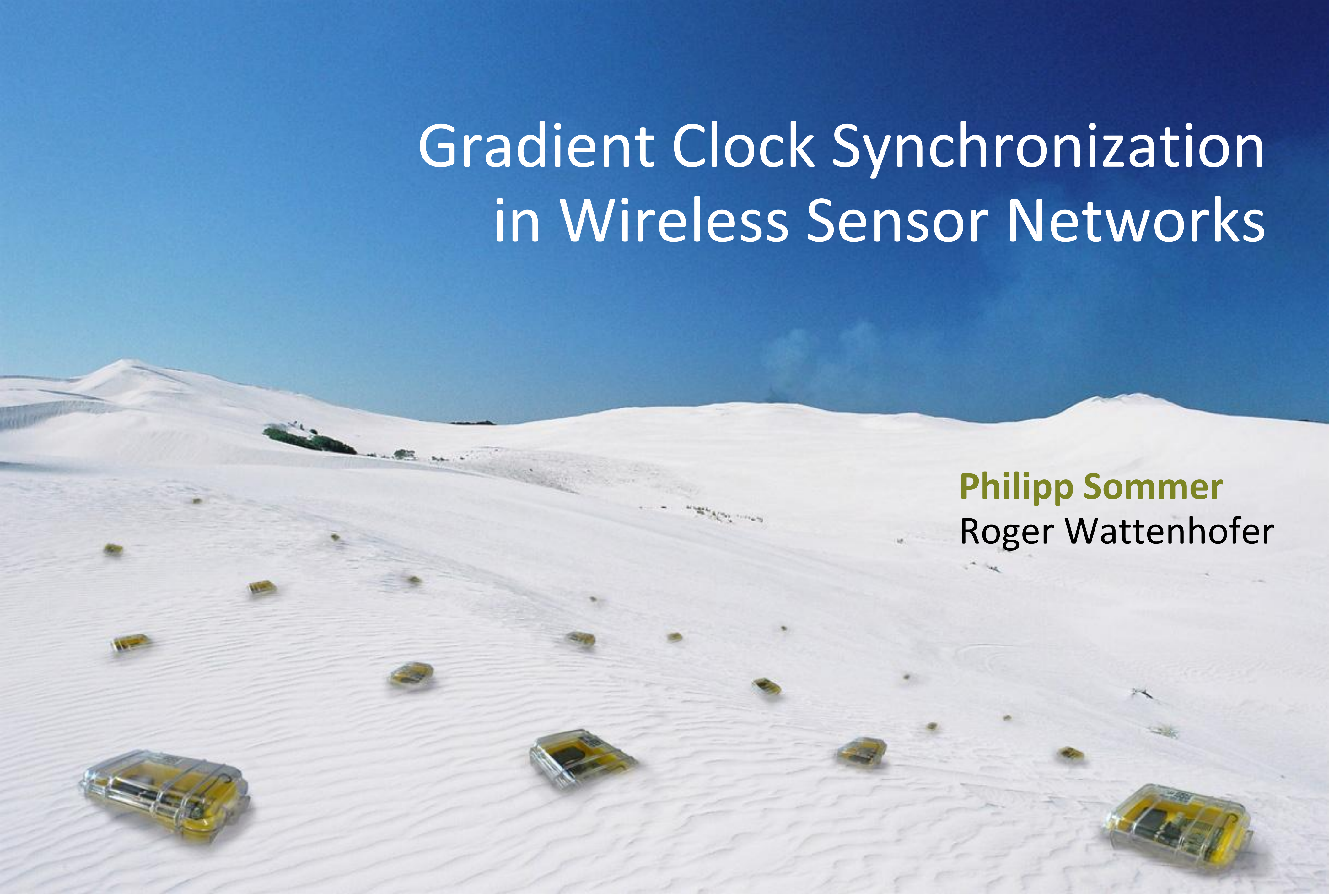


Gradient Clock Synchronization in Wireless Sensor Networks

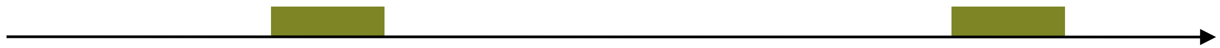
Philipp Sommer
Roger Wattenhofer

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

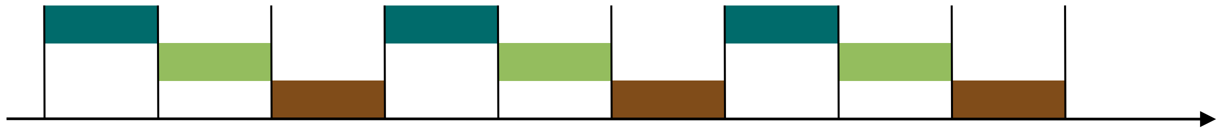
Distributed Computing Group

Time in Sensor Networks

- Synchronized clocks are essential for many applications:
  - Global — Time-stamping sensed data/events at different locations
  - 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



Time Synchronization in (Sensor) Networks

- *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

Clock Synchronization in Practice?

- 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

  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

Hardware Clocks Experience Drift

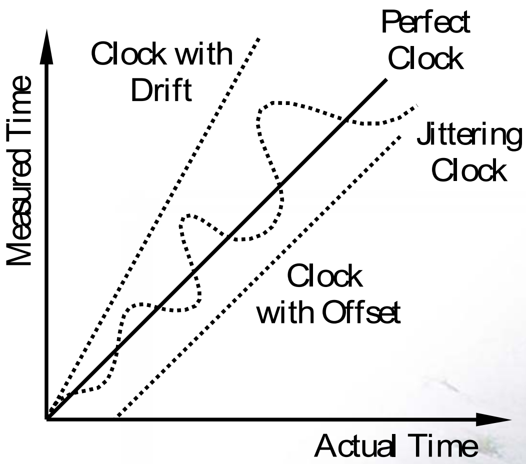
- Hardware clock H(t)

  Timer/Counter register of the microcontroller

  External crystal quartz (32kHz, 7.37 MHz)

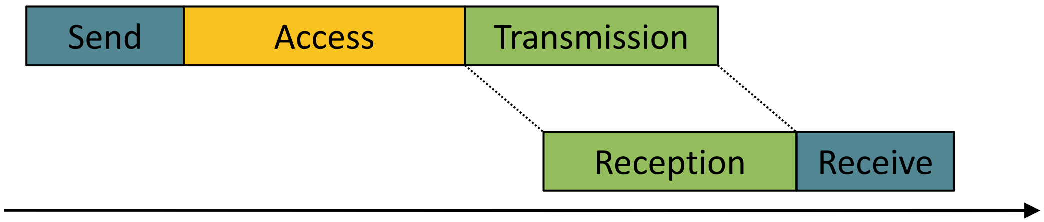
- Accuracy

  Clock drift: random deviation from the nominal rate dependent on power supply, temperature, etc. (30-100 ppm)



Messages Experience Jitter

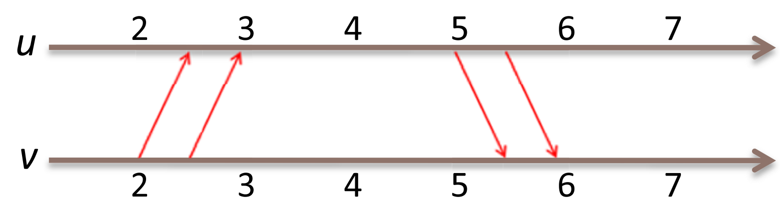
- Problem: Jitter in the message delay
  - Various sources of errors (deterministic and non-deterministic)
  - Asymmetric packet delays

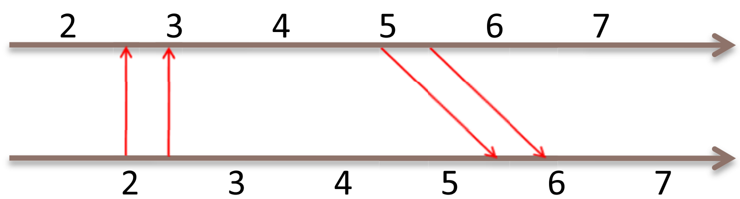


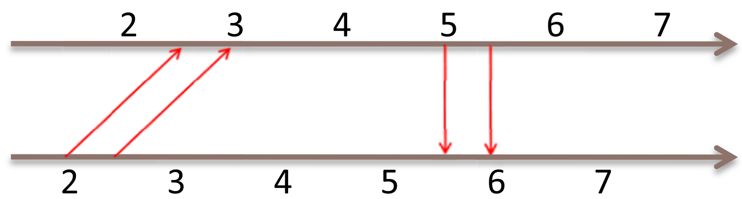
- Solution: Timestamping packets at the MAC layer
  - But still there is some jitter

Limits on the Synchronization Accuracy

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



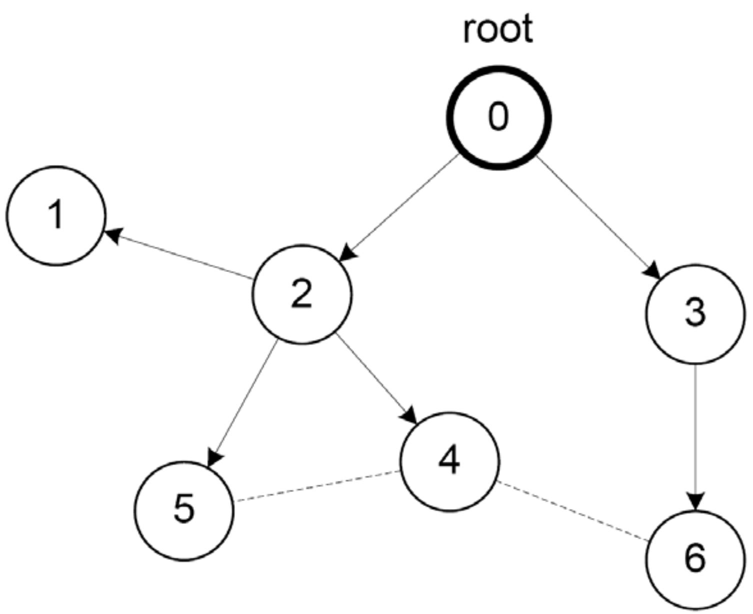




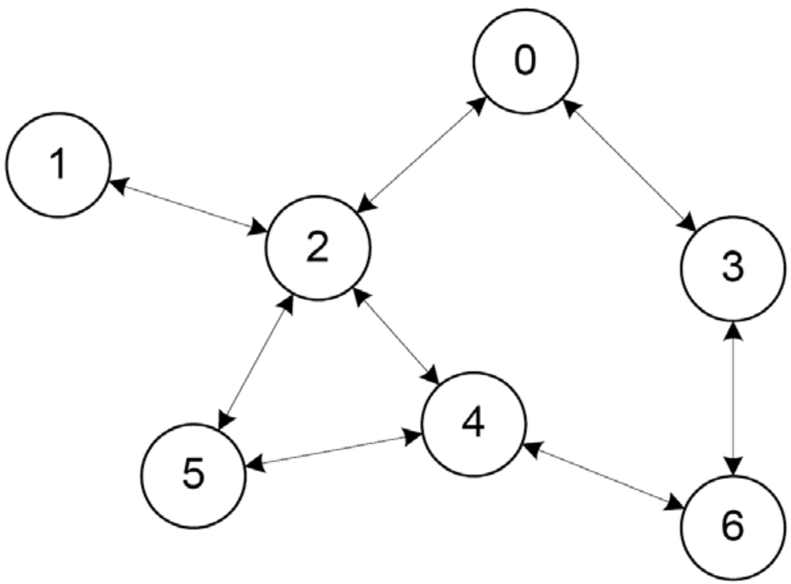
- Messages between two neighboring nodes may be fast in one direction and slow in the other, or vice versa.
- Error increases with distance from the reference node

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

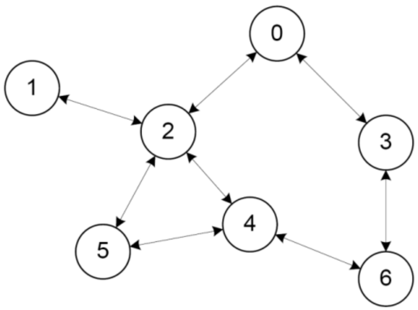


GTSP

Gradient Time Synchronization Protocol (GTSP)

[Sommer et al., IPSN 09]

- Synchronize clocks with **all** neighboring nodes
  - There is no reference node and no tree
  - Broadcast periodic synchronization beacons



- Node maintains a logical clock L(t)
  - Estimation of the global clock of the sensor network
  - Computed in software as a function of the hardware clock *H(t)*
  - Logical clock rate $l_i(t)$ can be adjusted to compensate for drift

GTSP Details

- Problem: How to synchronize clocks without having a leader?

  Follow the node with the fastest/slowest clock?

- Solution: Go to the average clock value/rate of all neighbors (including node itself)

Clock Rate	Clock Offset*
$l_i(t_{k+1}) = \frac{\left(\sum_{j \in \mathcal{N}_i} \frac{x_j(t_k)}{h_i(t_k)}\right) + l_i(t_k)}{\lvert\mathcal{N}_i\rvert + 1}$	$\theta_i(t_{k+1}) = \theta_i(t_k) + \frac{\sum_{j \in \mathcal{N}_i} L_j(t_k) - L_i(t_k)}{\lvert\mathcal{N}_i\rvert + 1}$

*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)

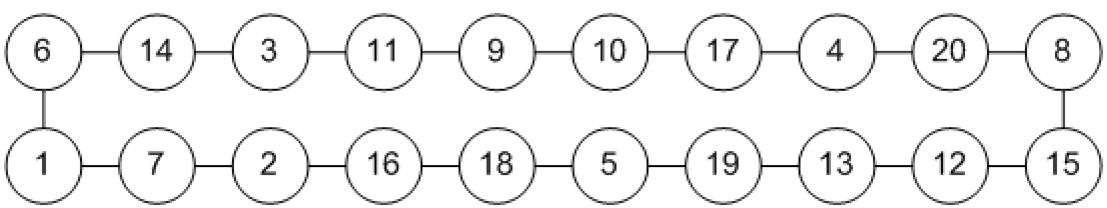
  Hardware clock: System clock divided by 8 = 921 kHz

  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



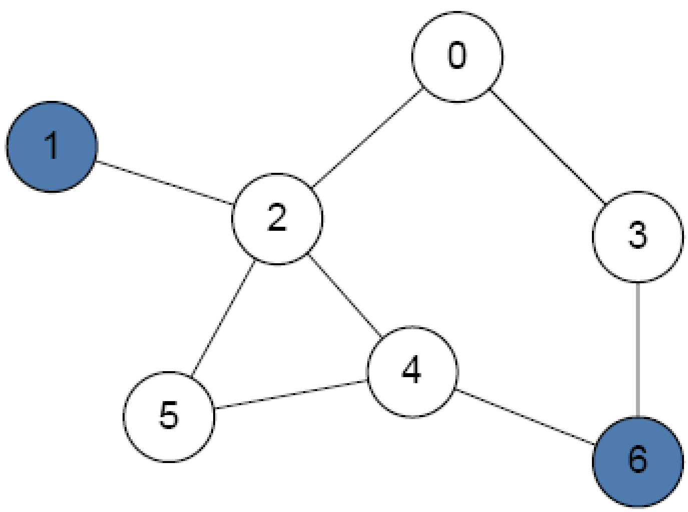
How to evaluate Synchronization Accuracy?

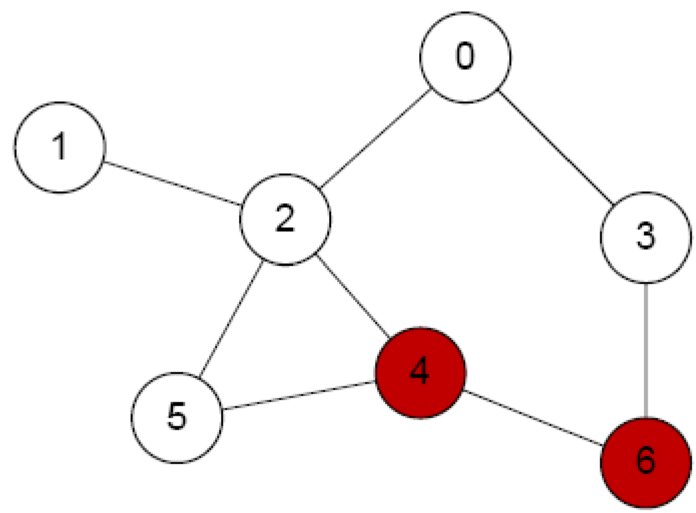
- Network synchronization error (**global clock skew**)

  Pair-wise synchronization error between **any** nodes in the network

- Neighbor Synchronization error (**local clock skew**)

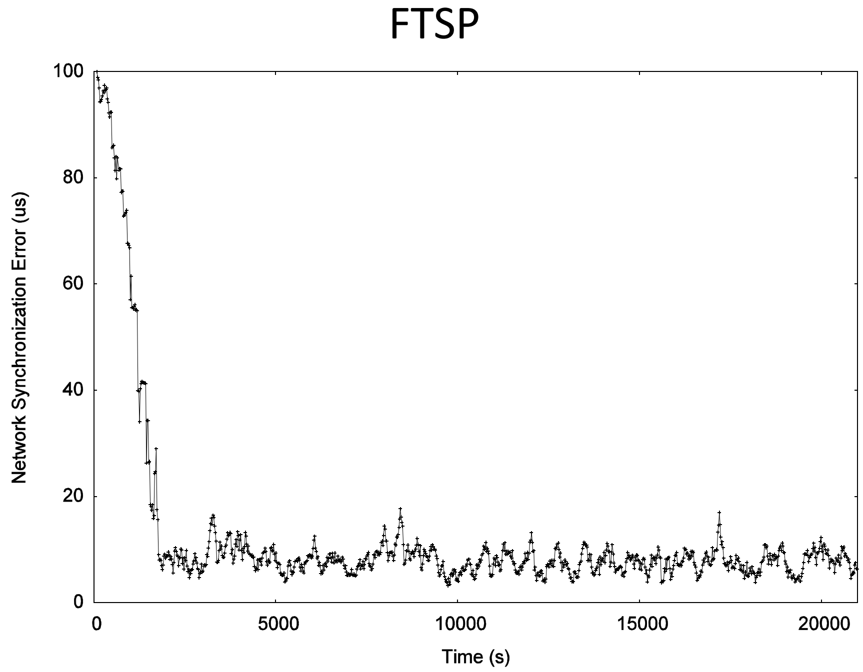
  Pair-wise synchronization error between neighboring nodes

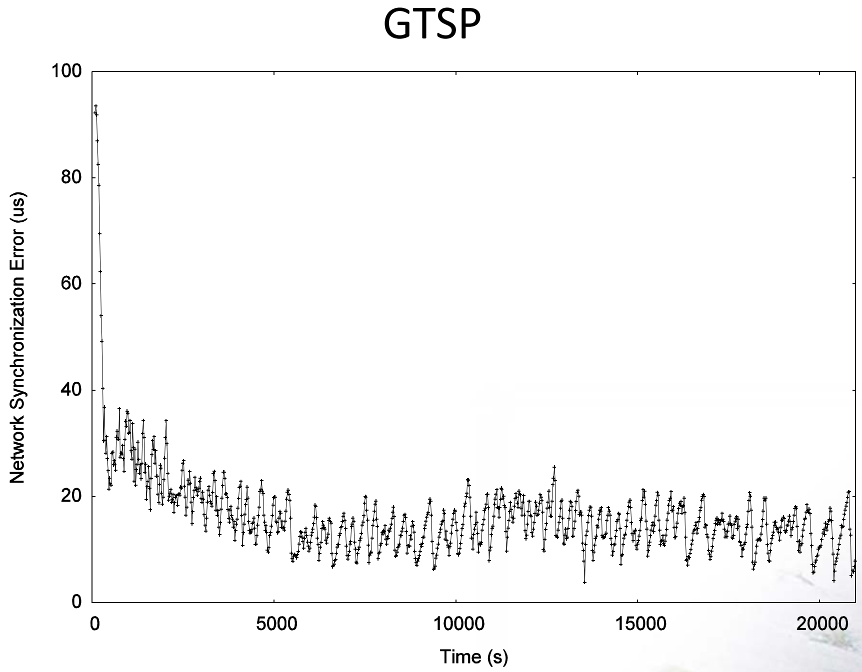


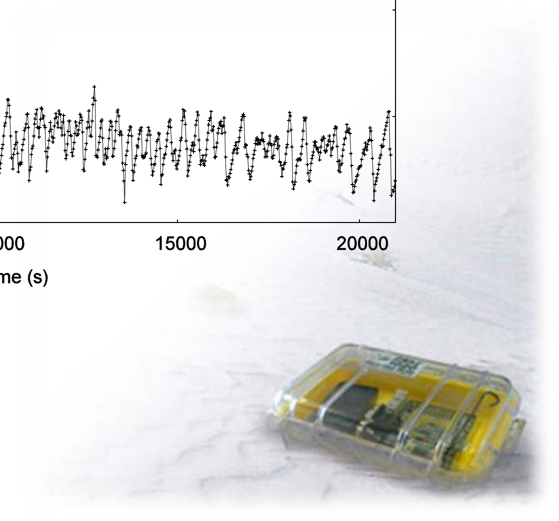


Experimental Results

- 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



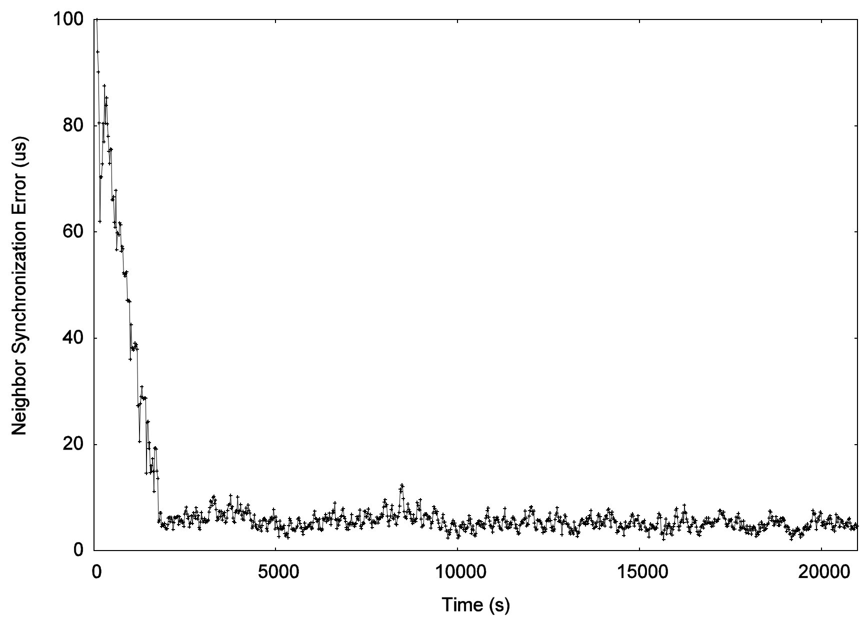




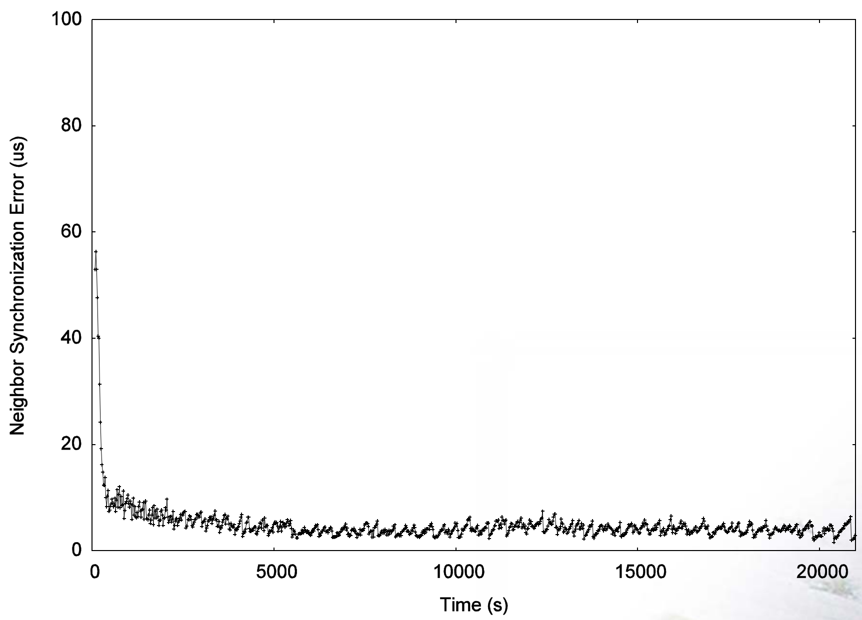
Experimental Results (2)

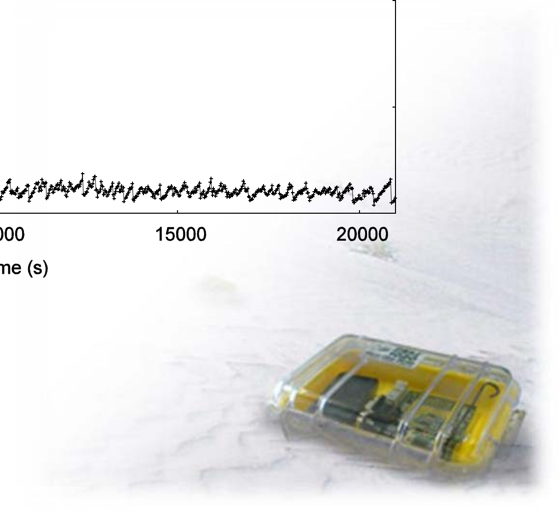
- Neighbor synchronization error (**local clock skew**)
  - 5.3 µs with FTSP, 4.0 µs with GTSP
  - GTSP takes slightly more time to stabilize

FTSP



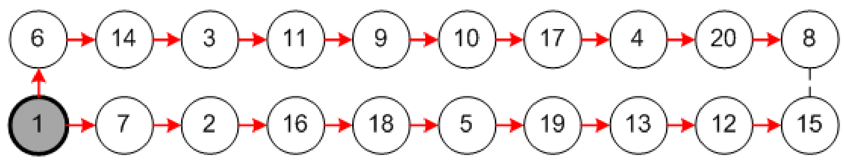
GTSP

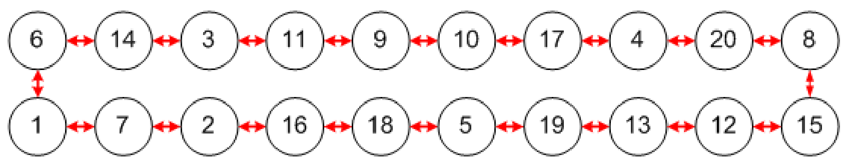




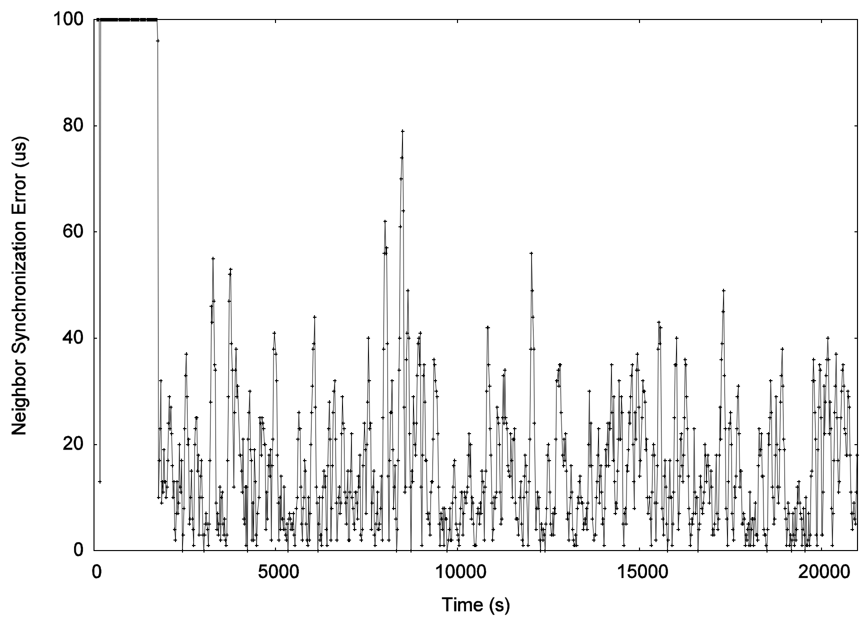
Neighbor Synchronization Error: FTSP vs. GTSP

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

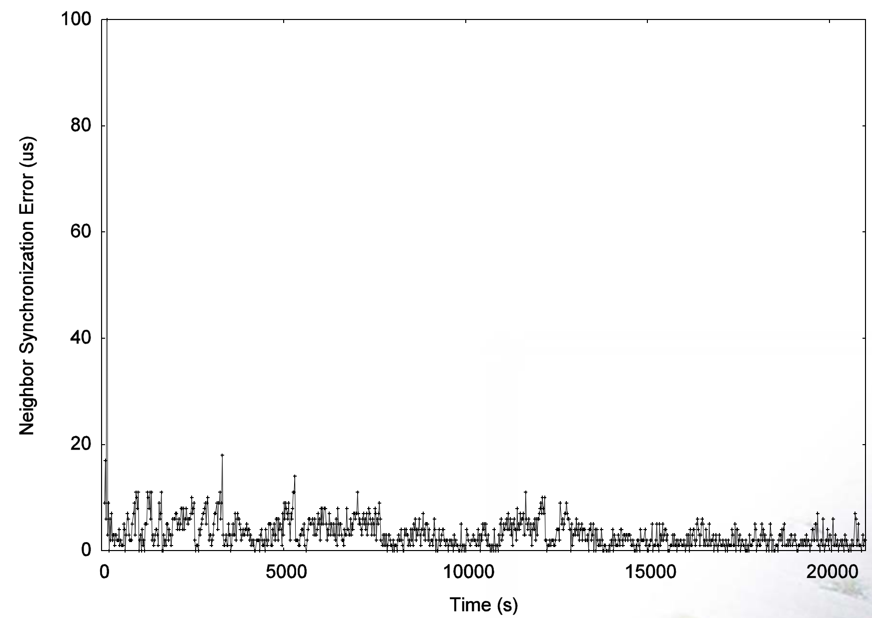




FTSP



GTSP



Multi-Hop Time Synchronization in Practice

- Is gradient clock synchronization relevant in practice?

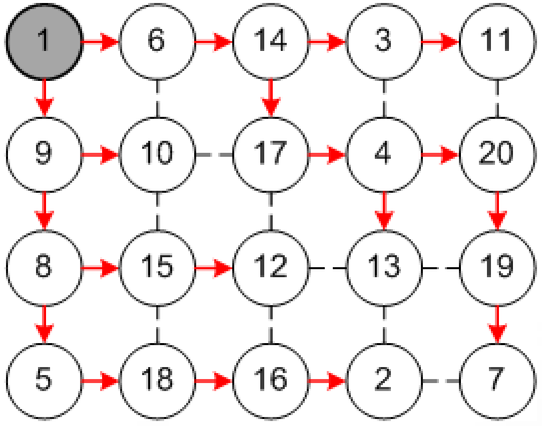
  Ring topology of 20 nodes seems to be „artificial“!?

- Finding a tree-embedding with low stretch is hard

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

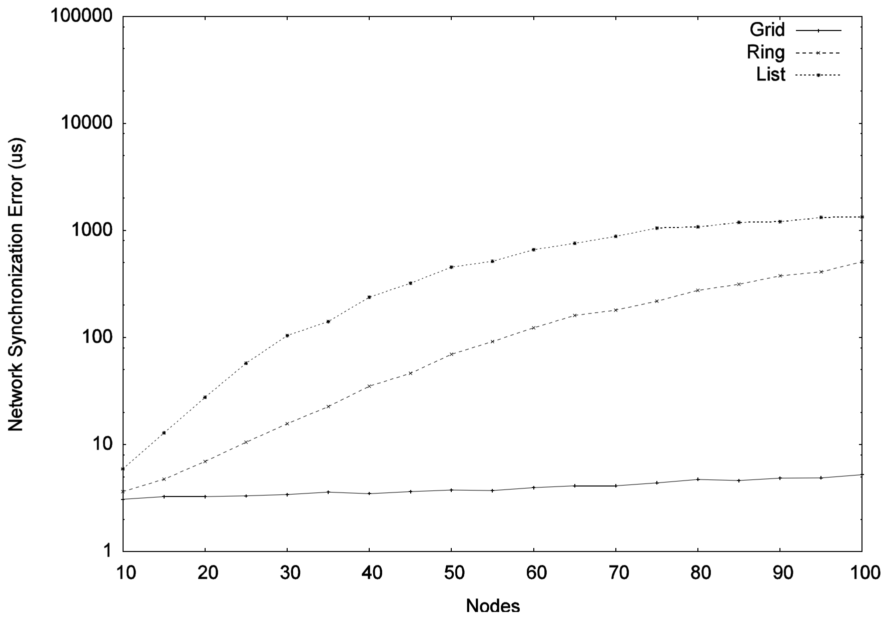
  Example: FTSP on a 5x4 grid topology

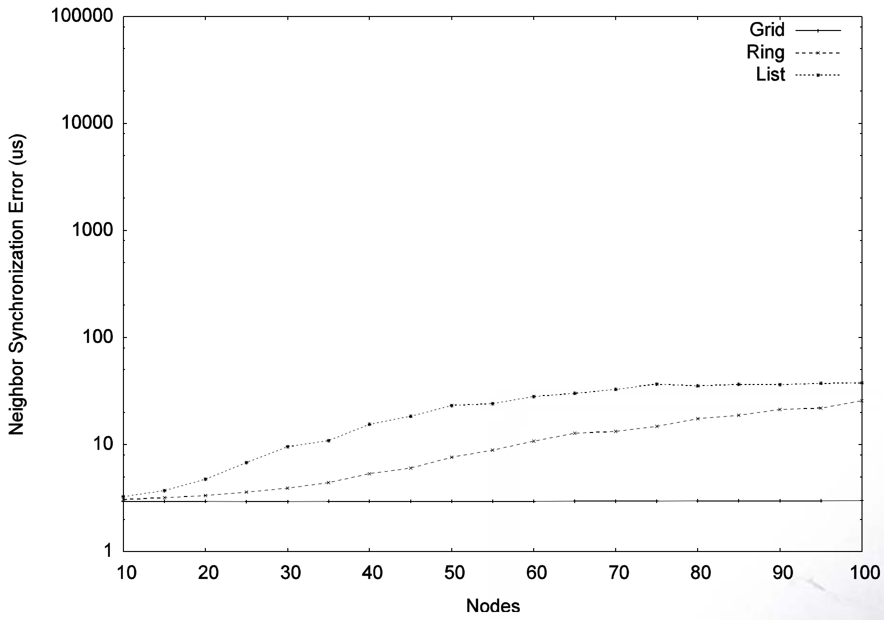
  Node 2 and 7 have a distance of 13 hops!



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 clock synchronization algorithm (no leader)
  - Improves the synchronization error between neighboring nodes while still providing precise network-wide synchronization

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