Theory Meets Practice
...it's about TIME!

Roger Wattenhofer
„People who are really serious about software should make their own hardware.”

Alan Kay
„People who are really serious about algorithms should make their own software.“

... or wait a long time for algorithms to be discovered.
Theory Meets Practice?
Today, we look much cuter!

And we’re usually carefully deployed

Power

Radio

Processor

Memory

Sensors
A Sensor Network After Deployment

multi-hop communication
A Typical Sensor Node: TinyNode 584

- TI MSP430F1611 microcontroller @ 8 MHz
- 10k SRAM, 48k flash (code), 512k serial storage
- 868 MHz Xemics XE1205 multi channel radio
- Up to 115 kbps data rate, 200m outdoor range

<table>
<thead>
<tr>
<th>State</th>
<th>Current Draw</th>
<th>Power Consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>uC sleep with timer on</td>
<td>6.5 uA</td>
<td>0.0195 mW</td>
</tr>
<tr>
<td>uC active, radio off</td>
<td>2.1 mA</td>
<td>6.3 mW</td>
</tr>
<tr>
<td>uC active, radio idle listening</td>
<td>16 mA</td>
<td>48 mW</td>
</tr>
<tr>
<td>uC active, radio TX/RX at +12dBm</td>
<td>62 mA</td>
<td>186 mW</td>
</tr>
<tr>
<td>Max. Power (uC active, radio TX/RX at +12dBm + flash write)</td>
<td>76.9 mA</td>
<td>230.7mW</td>
</tr>
</tbody>
</table>
The PermaSense Project
Matterhorn Field Site Installations

Sensor node installations targeting 3 years unattended lifetime

Base station mounted under a combined sun/rain hood

Base station and solar panels on the field site at Matterhorn

Base station power supply, system monitoring and a backup GSM modem are housed separately
Example: Dozer

- Up to 10 years of network life-time
- Mean energy consumption: 0.066 mW
- Operational network in use > 3 years
- High availability, reliability (99.999%)

[Burri et al., IPSN 2007]
Is Dozer a theory-meets-practice success story?

- **Good news**
  - Theory people can develop good systems!
  - Sensor network (systems) people write that Dozer is one of the “best sensor network systems papers”, or: “In some sense this is the first paper I'd give someone working on communication in sensor nets, since it nails down how to do it right.”

- **Bad news**: Dozer does not have an awful lot of theory inside
- **Ugly news**: Dozer v2 has even less theory than Dozer v1
- **Hope**: Still subliminal theory ideas in Dozer?
Energy-Efficient Protocol Design

• Communication subsystem is the main energy consumer
  – Power down radio as much as possible

<table>
<thead>
<tr>
<th>TinyNode</th>
<th>Power Consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>uC sleep, radio off</td>
<td>0.015 mW</td>
</tr>
<tr>
<td>Radio idle, RX, TX</td>
<td>30 – 40 mW</td>
</tr>
</tbody>
</table>

• Issue is tackled at various layers
  – MAC
  – Topology control / clustering
  – Routing

👉 Orchestration of the whole network stack to achieve duty cycles of ~ 0.1%
Dozer System

- Tree based routing towards data sink
  - No energy wastage due to multiple paths
  - Current strategy: SPT

- TDMA based link scheduling
  - Each node has two independent schedules
  - No global time synchronization

- The parent initiates each TDMA round with a beacon
  - Enables integration of disconnected nodes
  - Children tune in to their parent’s schedule
Dozer System

- Parent decides on its children data upload times
  - Each interval is divided into upload slots of equal length
  - Upon connecting each child gets its own slot
  - Data transmissions are always ack’ed

- No traditional MAC layer
  - Transmissions happen at exactly predetermined point in time
  - Collisions are explicitly accepted
  - Random jitter resolves schedule collisions

Clock drift, queuing, bootstrap, etc.

![Diagram showing data transfer and time slots](image-url)
Dozer in Action
Energy Consumption

- Leaf node
- Few neighbors
- Short disruptions

• Relay node
• No scanning

0.28% duty cycle

0.32% duty cycle
no theory 😞
Theory for sensor networks, what is it good for?!

How many lines of pseudo code //
Can you implement on a sensor node?

The best algorithm is often complex //
And will not do what one expects.

Theory models made lots of progress //
Reality, however, they still don’t address.

My advice: invest your research £££s //
in ... impossibility results and lower bounds!
Example: Clock Synchronization
...it's about TIME!
Clock Synchronization in Practice

• Many different approaches for clock synchronization

Global Positioning System (GPS)

Radio Clock Signal

AC-power line radiation

Synchronization messages
Oscillators in Sensor Nodes

- Structure
  - External oscillator with a nominal frequency (e.g. 32 kHz or 7.37 MHz)
  - Counter register which is incremented with oscillator pulses
  - Works also when CPU is in sleep state

<table>
<thead>
<tr>
<th>Platform</th>
<th>System clock</th>
<th>Crystal oscillator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mica2</td>
<td>7.37 MHz</td>
<td>32 kHz, 7.37 MHz</td>
</tr>
<tr>
<td>TinyNode 584</td>
<td>8 MHz</td>
<td>32 kHz</td>
</tr>
<tr>
<td>Tmote Sky</td>
<td>8 MHz</td>
<td>32 kHz</td>
</tr>
</tbody>
</table>
Clocks Experience Drift

- 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

This is a drift of up to 50 μs per second or 0.18s per hour
Messages Experience Jitter in the Delay

• Problem: Jitter in the message delay
  – Various sources of errors (deterministic and non-deterministic)

• Solution: Timestamping packets at the MAC layer [Maróti et al.]
  – → Jitter in the message delay is reduced to a few clock ticks
Clock Synchronization in Networks?

- *Time, Clocks, and the Ordering of Events in a Distributed System*
- *Internet Time Synchronization: The Network Time Protocol (NTP)*
- *Reference Broadcast Synchronization (RBS)*
  J. Elson, L. Girod and D. Estrin, OSDI 2002
- *Timing-sync Protocol for Sensor Networks (TPSN)*
  S. Ganeriwal, R. Kumar and M. Srivastava, SenSys 2003
- *Flooding Time Synchronization Protocol (FTSP)*
  M. Maróti, B. Kusy, G. Simon and Á. Lédeczi, SenSys 2004
- and many more ...

FTSP: State of the art clock sync protocol for networks.
Variants of Clock Synchronization Algorithms

Tree-like Algorithms
e.g. FTSP

Distributed Algorithms
e.g. GTSP
[Sommer et al., IPSN 2009]

Bad local skew

All nodes consistently average errors to all neighbors
FTSP vs. GTSP: Global Skew

- Network synchronization error (global skew)
  - Pair-wise synchronization error between any two nodes in the network

FTSP (avg: 7.7 μs)

GTSP (avg: 14.0 μs)
FTSP vs. GTSP: Local Skew

- 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

Clock Synchronization Protocol

Hardware Clock
Clock Synchronization in Theory?

• Given a communication network
  1. Each node equipped with hardware clock with drift
  2. Message delays with jitter

• Goal: Synchronize Clocks (“Logical Clocks”)
  • Both global and local synchronization!

worst-case (but constant)
Time Must Behave!

• Time (logical clocks) should not be allowed to stand still or jump

Let’s be more careful (and ambitious):
• Logical clocks should always move forward
  • Sometimes faster, sometimes slower is OK.
  • But there should be a minimum and a maximum speed.
  • As close to correct time as possible!
Formal Model

- Hardware clock $H_v(t) = \int_{[0,t]} h_v(\tau) \, d\tau$ with clock rate $h_v(t) \in [1-\varepsilon, 1+\varepsilon]$

- Logical clock $L_v(\cdot)$ which increases at rate at least 1 and at most $\beta$

- Message delays $\in [0,1]$

- Employ a synchronization algorithm to update the logical clock according to hardware clock and messages from neighbors

Clock drift $\varepsilon$ is typically small, e.g. $\varepsilon \approx 10^{-4}$ for a cheap quartz oscillator

Logical clocks with rate much less than 1 behave differently...

Neglect fixed share of delay, normalize jitter
Local Skew

Tree-like Algorithms
e.g. FTSP

Distributed Algorithms
e.g. GTSP

Bad local skew
Synchronization Algorithms: An Example ("A_{max}"")

- Question: How to update the logical clock based on the messages from the neighbors?

- Idea: Minimizing the skew to the fastest neighbor
  - Set clock to maximum clock value you know, forward new values immediately

- First all messages are slow (1), then suddenly all messages are fast (0)!

![Diagram showing clock values and message updates over time with skew D]
Everybody’s expectation, 10 years ago (”solved“)

Lower bound of log\(D / \log\log D\) [Fan & Lynch, PODC 2004]

Blocking algorithm

All natural algorithms [Locher et al., DISC 2006]

Kappa algorithm

Tight lower bound [Lenzen et al., PODC 2009]

Dynamic Networks! [Kuhn et al., SPAA 2009]

Tight lower bound [Lenzen et al., PODC 2009]

together [JACM 2010]
Clock Synchronization vs. Car Coordination

• In the future cars may travel at high speed despite a tiny safety distance, thanks to advanced sensors and communication.
Clock Synchronization vs. Car Coordination

• In the future cars may travel at high speed despite a tiny safety distance, thanks to advanced sensors and communication

   ________

   ________

   ________

• How fast & close can you drive?

• Answer possibly related to clock synchronization
  – clock drift ↔ cars cannot control speed perfectly
  – message jitter ↔ sensors or communication between cars not perfect
Example: Clock Synchronization?!!
...it's about TIME!

Roger Wattenhofer
One Big Difference Between Theory and Practice, Usually!

Physical Reality...

Worst Case Analysis!

Practice

Theory
As we have seen, FTSP does have a local skew problem. But it’s not all that bad...

However, tests revealed another (severe!) problem: FTSP does not scale. Global skew grows exponentially with network size...
The PulseSync Protocol

1) Remove self-amplifying of synchronization error
2) Send fast synchronization pulses through the network
   - Speed-up the initialization phase
   - Faster adaptation to changes in temperature or network topology

FTSP
Expected time
= $D \cdot \frac{B}{2}$

PulseSync
Expected time
= $D \cdot t_{\text{pulse}}$

[Mathematical expressions and diagrams as shown in the slide]
Evaluation

- Testbed setup
  - 20 Crossbow Mica2 sensor nodes
  - PulseSync implemented in TinyOS 2.1
  - FTSP from TinyOS 2.1

- Network topology
  - Single-hop setup, basestation
  - Virtual network topology (white-list)
  - Acknowledgments for time sync beacons

![Diagram of network topology with 20 sensor nodes and a probe beacon]
Experimental Results

- Global Clock Skew
  - Maximum synchronization error between any two nodes

<table>
<thead>
<tr>
<th>Synchronization Error</th>
<th>FTSP</th>
<th>PulseSync</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average (t&gt;2000s)</td>
<td>23.96 µs</td>
<td>4.44 µs</td>
</tr>
<tr>
<td>Maximum (t&gt;2000s)</td>
<td>249 µs</td>
<td>38 µs</td>
</tr>
</tbody>
</table>
Experimental Results

- Synchronization error vs. hop distance
Everybody's expectation, five years ago ("solved")

Lower bound \( \log D / \log \log D \) [Fan & Lynch, PODC 2004]

All natural algorithms [Locher et al., DISC 2006]

Blocking algorithm [Lenzen et al., FOCS 2008]

Tight lower bound [Lenzen et al., PODC 2008]

Dynamic Networks [Kuhn et al., SPAA 2009]
Merci!

Questions & Comments?

Thanks to my co-authors
Nicolas Burri
Christoph Lenzen
Thomas Locher
Philipp Sommer
Pascal von Rickenbach
Open Problems

• global vs. local skew
• worst-case vs. reality (Gaussian?)
• accuracy vs. convergence
• accuracy vs. energy efficiency
• dynamic networks
• fault-tolerance (Byzantine clocks)
• applications, e.g. coordinating physical objects (example with cars)