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

FC Practice  SC Theory

It’s a BOY

$(a + b)^2 = a^2 + 2ab + b^2$
Why is there so little interaction?

Theory is useless...

Practice is trivial...

Practice

Theory
Systems people don’t read theory papers

- Sometimes for good reasons...
  - unreadable
  - don’t matter that much (only getting out the last %)
  - wrong models
  - theory is lagging behind
  - bad theory merchandising/branding
    - systems papers provide easy to remember acronyms
    - “On the Locality of Bounded Growth” vs. “Smart Dust”
  - good theory comes from surprising places
    - difficult to keep up with
    - having hundreds of workshops does not help

- If systems people don’t read theory papers, maybe theory people should build systems themselves?
Systems Perspective: Dozer
Today, we look much cuter!

And we’re usually carefully deployed
A Sensor Network After Deployment

multi-hop communication
A Typical Sensor Node: TinyNode 584

[Shockfish SA, The Sensor Network Museum]

- 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 > 2 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!
  – Dozer is to the best of my knowledge more energy-efficient and reliable than all other published systems protocols… for many years already!
  – 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
Energy Consumption

- Leaf node
- Few neighbors
- Short disruptions

- Relay node
- No scanning

Roger Wattenhofer @ SOFSEM 2010 – 18
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**
Clock Devices 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>
Clock 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
Sender/Receiver Synchronization

- Round-Trip Time (RTT) based synchronization

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

\[ \theta = \frac{(t_2 - (t_1 + \delta)) - (t_4 - (t_3 + \delta))}{2} = \frac{(t_2 - t_1) + (t_3 - t_4)}{2} \]

- Receiver synchronizes to sender's clock
- Propagation delay \( \delta \) and clock offset \( \theta \) can be calculated
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.]
  $\rightarrow$ 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.
Flooding Time Synchronization Protocol (FTSP)

- Each node maintains both a local and a global time
- Global time is synchronized to the local time of a reference node
  - Node with the smallest id is elected as the reference node
- Reference time is flooded through the network periodically

- Timestamping at the MAC Layer is used to compensate for deterministic message delays
- Compensation for clock drift between synchronization messages using a linear regression table
Best tree for tree-based clock synchronization?

- Finding a good tree for clock synchronization is a tough problem
  - Spanning tree with small (maximum or average) stretch.

- Example: Grid network, with $n = m^2$ nodes.

- No matter what tree you use, the maximum stretch of the spanning tree will always be at least $m$ (just try on the grid figure right…)

- In general, finding the minimum max stretch spanning tree is a hard problem, however approximation algorithms exist [Emek, Peleg, 2004].
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) vs. 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:

![Graph showing comparison between FTSP and GTSP synchronization error](image-url)
Synchronized clocks are essential for many applications:
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!
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 $h_v(t) \in [1-\epsilon, 1+\epsilon]$.
  
  Clock drift $\epsilon$ is typically small, e.g. $\epsilon \approx 10^{-4}$ for a cheap quartz oscillator.

- Logical clock $L_v(\cdot)$ which increases at rate at least 1 and at most $\beta$.
  
  Logical clocks with rate much less than 1 behave differently...

- Message delays $\in [0,1]$.
  
  Neglect fixed share of delay, normalize jitter.

- Employ a synchronization algorithm to update the logical clock according to hardware clock and messages from neighbors.
Variants of Clock Synchronization Algorithms

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 the clock to the maximum clock value received from any neighbor (if larger than local clock value)
    – forward new values immediately

• Optimum global skew of about $D$

• Poor local property
  – First all messages take 1 time unit...
  – ...then we have a fast message!

Allow $\beta = \infty$, i.e. logical clock may jump forward

Roger Wattenhofer @ SOFSEM 2010 – 36
Local Skew: Overview of Results

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]

$1 \log D \quad \sqrt{D} \quad D \quad \ldots$

Kappa algorithm
[Lenzen et al., FOCS 2008]

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

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

together
[JACM 2010]
Enforcing Clock Skew

- Messages between two neighboring nodes may be fast in one direction and slow in the other, or vice versa.

- A constant skew between neighbors may be „hidden“.

- In a path, the global skew may be in the order of $D/2$. 
Local Skew: Lower Bound

\( l_0 = D \)

- Add \( l_0/2 \) skew in \( l_0/(2\epsilon) \) time, messing with clock rates and messages

- Afterwards: Continue execution for \( l_0/(4(\beta-1)) \) time (all \( h_x = 1 \))
  
  \( \rightarrow \) Skew reduces by at most \( l_0/4 \) \( \rightarrow \) at least \( l_0/4 \) skew remains

  \( \rightarrow \) Consider a subpath of length \( l_1 = l_0\cdot\epsilon/(2(\beta-1)) \) with at least \( l_1/4 \) skew

  \( \rightarrow \) Add \( l_1/2 \) skew in \( l_1/(2\epsilon) = l_0/(4(\beta-1)) \) time \( \rightarrow \) at least \( 3/4 \cdot l_1 \) skew in subpath

- Repeat this trick \( (+1/2,-1/4,+1/2,-1/4,...) \log_{2(\beta-1)/\epsilon} D \) times

Theorem: \( \Omega(\log_{(\beta-1)/\epsilon} D) \) skew between neighbors
Local Skew: Upper Bound

- Surprisingly, up to small constants, the $\Omega(\log_{(\beta-1)/\epsilon} D)$ lower bound can be matched with clock rates $\in [1, \beta]$ (tough part, not in this talk).
- We get the following picture [Lenzen et al., PODC 2009]:

<table>
<thead>
<tr>
<th>max rate $\beta$</th>
<th>$1+\epsilon$</th>
<th>...</th>
<th>...</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>local skew</td>
<td>$\infty$</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

We can have both smooth and accurate clocks!

... because too large clock rates will amplify the clock drift $\epsilon$. 

Roger Wattenhofer @ SOFSEM 2010 – 40
Local Skew: Upper Bound

- Surprisingly, up to small constants, the $\Omega(\log_{(\beta-1)/\epsilon} D)$ lower bound can be matched with clock rates $\in [1, \beta]$ (tough part, not in this talk).
- We get the following picture [Lenzen et al., PODC 2009]:

<table>
<thead>
<tr>
<th>max rate $\beta$</th>
<th>$1+\epsilon$</th>
<th>$1+\Theta(\epsilon)$</th>
<th>$1+\sqrt{\epsilon}$</th>
<th>2</th>
<th>large</th>
</tr>
</thead>
<tbody>
<tr>
<td>local skew</td>
<td>$\infty$</td>
<td>$\Theta(\log D)$</td>
<td>$\Theta(\log_{1/\epsilon} D)$</td>
<td>$\Theta(\log_{1/\epsilon} D)$</td>
<td>$\Theta(\log_{1/\epsilon} D)$</td>
</tr>
</tbody>
</table>

We can have both smooth and accurate clocks!

... because too large clock rates will amplify the clock drift $\epsilon$.

- In practice, we usually have $1/\epsilon \approx 10^4 > D$. In other words, our initial intuition of a constant local skew was not entirely wrong! 😊
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
Is Our Theory Practical?!?
...it's about TIME!
One Big Difference Between Theory and Practice, Usually!

Physical Reality...

Practice

Worst Case Analysis!

Theory

\[ \Theta \left( \frac{W}{\sqrt{n \log n}} \right) \]

\( P \neq NP \)
„Industry Standard“ FTSP in Practice

- 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...
Why?

- How does the network diameter affect synchronization errors?

- Examples for sensor networks with large diameter
  Bridge, road or pipeline monitoring

Deployment at Golden Gate Bridge with 46 hops
[Kim et al., IPSN 07]
Multi-hop Clock Synchronization

• Nodes forward their current estimate of the reference clock
  – Each synchronization beacon is affected by a \textit{random jitter} $J$

\begin{align*}
0 & \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow \ldots \rightarrow d
\end{align*}

• Sum of the jitter grows with the square-root of the distance
  – \[ \text{stddev}(J_1 + J_2 + J_3 + J_4 + J_5 + \ldots + J_d) = \sqrt{d} \times \text{stddev}(J) \]

• This is bad but does not explain exponential behavior of FTSP...

• In addition FTSP uses linear regression to compensate for clock drift
  – Jitter is amplified before it is sent to the next hop!
  – Amplification leads to exponential behavior...
Linear Regression (FTSP)

- Simulation of FTSP with regression tables of different sizes (k = 2, 8, 32)

![Graph showing the relationship between network diameter and average global skew for different table sizes. The x-axis represents network diameter ranging from 0 to 50, and the y-axis represents average global skew (us) with a log scale. The graph includes lines for Table Size 2, Table Size 8, and Table Size 32, each with error bars.](image)
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·B/2

PulseSync
Expected time
= D·t_{pulse}
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]

---

0 → 1 → 2 → 3 → 4 → ... → 20

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

![Graph showing synchronization error vs. hop distance for FTSP and PulseSync.](image)
Beyond the list?

- Problem: So far PulseSync works for list topology only

- Instead schedule synchronization beacons without collisions
  - Time information has to propagate quickly through the network
  - Avoid loss of synchronization pulses due to collisions

This is known as **wireless broadcasting**, a well-studied problem (in theory...!)

- In other words, for the first time in my life as a researcher, theory and practice play ping pong.
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)

- more open problems in SOFSEM paper
Summary

Everybody’s expectation, five years ago (“solved”)

Lower bound of $\log D / \log \log D$

[Fan & Lynch, PODC 2004]

All natural algorithms

[Locher et al., DISC 2006]

Blocking algorithm

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

Dynamic network

[Kuhn et al., SPAA 2009]

Tight lower bound

[Lenzen et al., PODC 2008]

Blocking algorithm

[Lenzen et al., DISC 2006]

Evacuating’s expectation, five years ago (“solved”)

Dynamic Networks!

[Lenzen et al., PODC 2009]
Thank You!

Questions & Comments?

Thanks to my co-authors
Nicolas Burri
Michael Kuhn
Christoph Lenzen
Thomas Locher
Philipp Sommer
Pascal von Rickenbach
Clock Synchronization