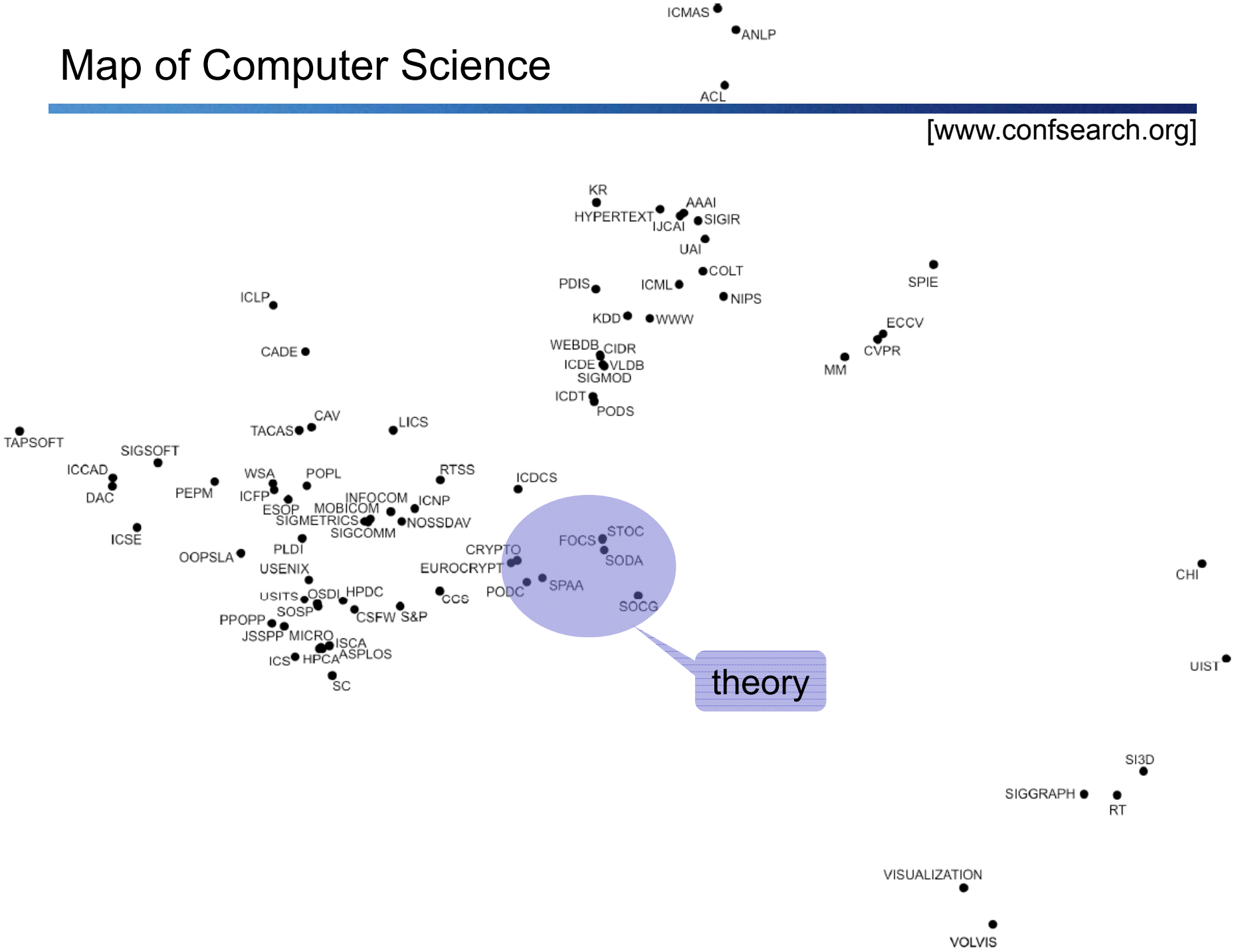
Theory Meets Practice

...it's about TIME!

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

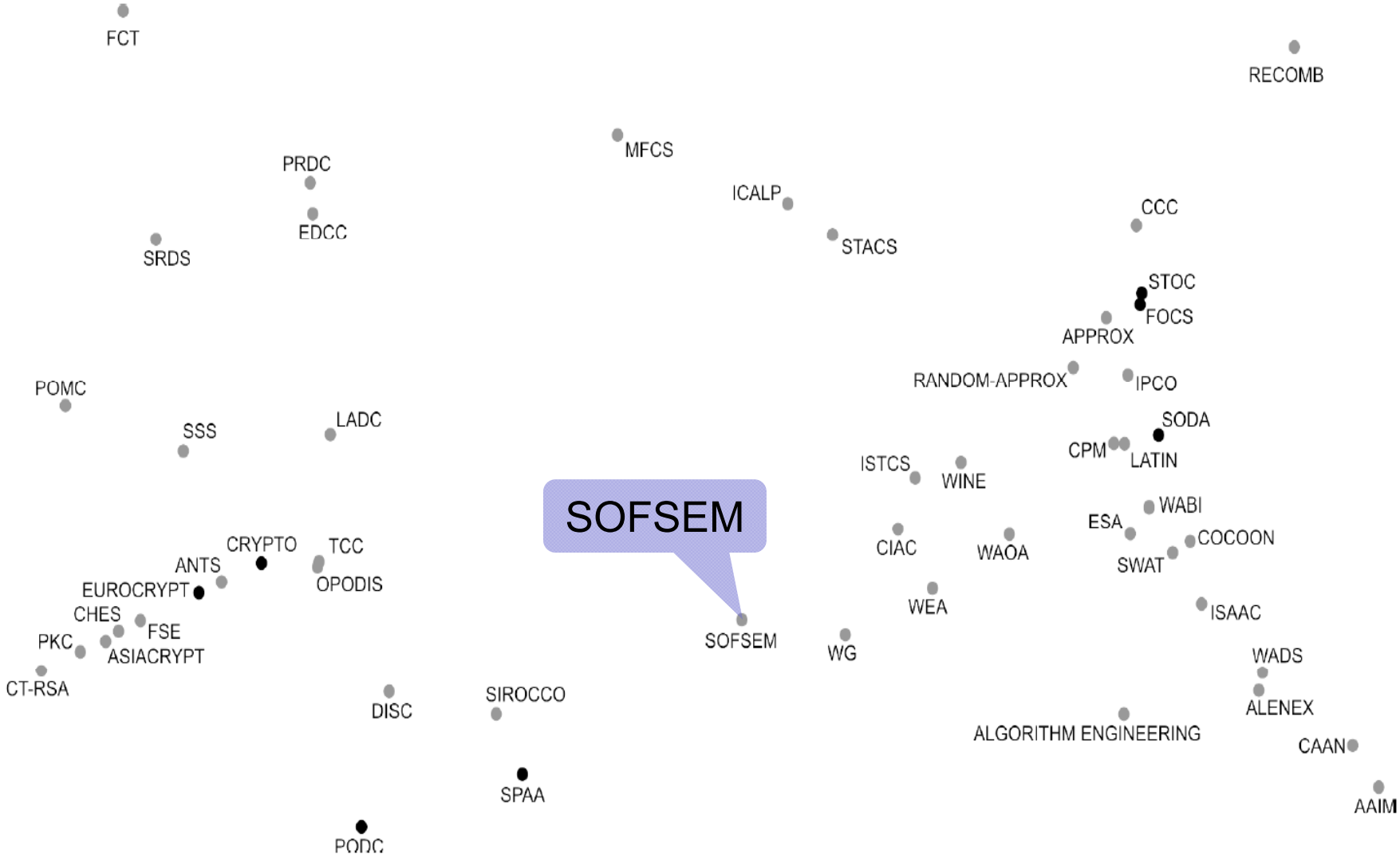
Map of Computer Science

[www.confsearch.org]



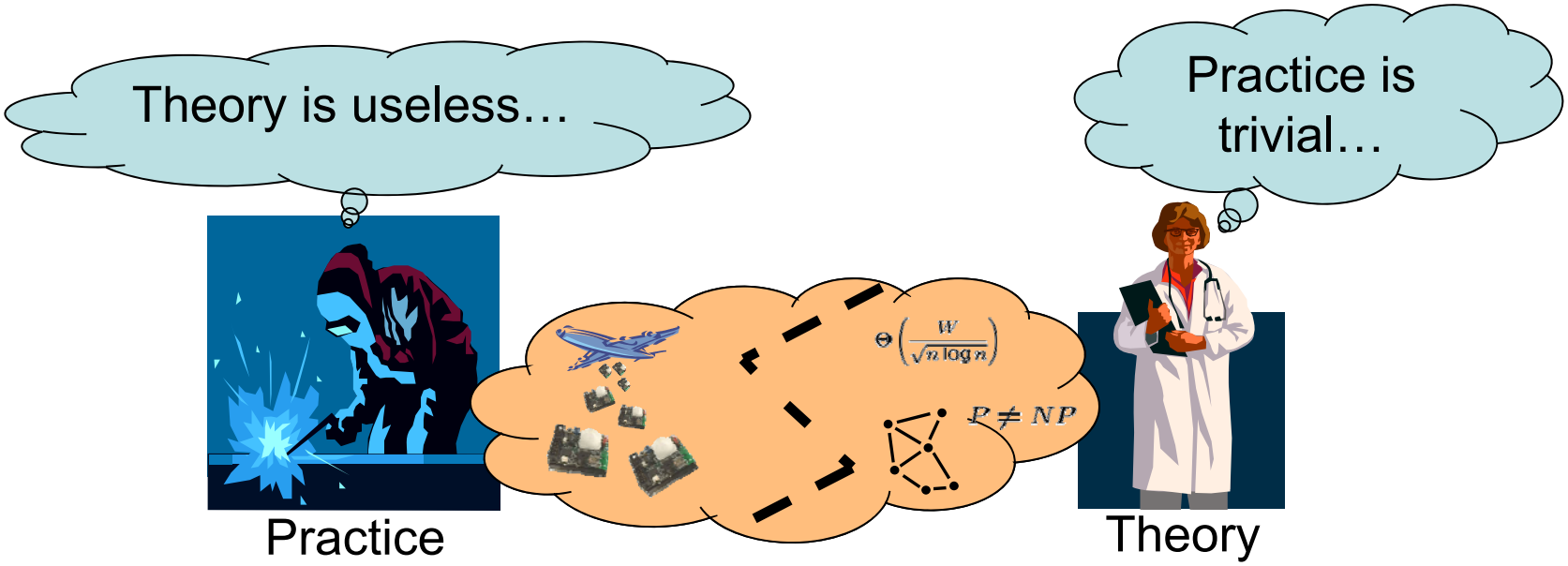
Zooming in on Theory

[www.confsearch.org]



Theory Meets Practice?

Why is there so little interaction?

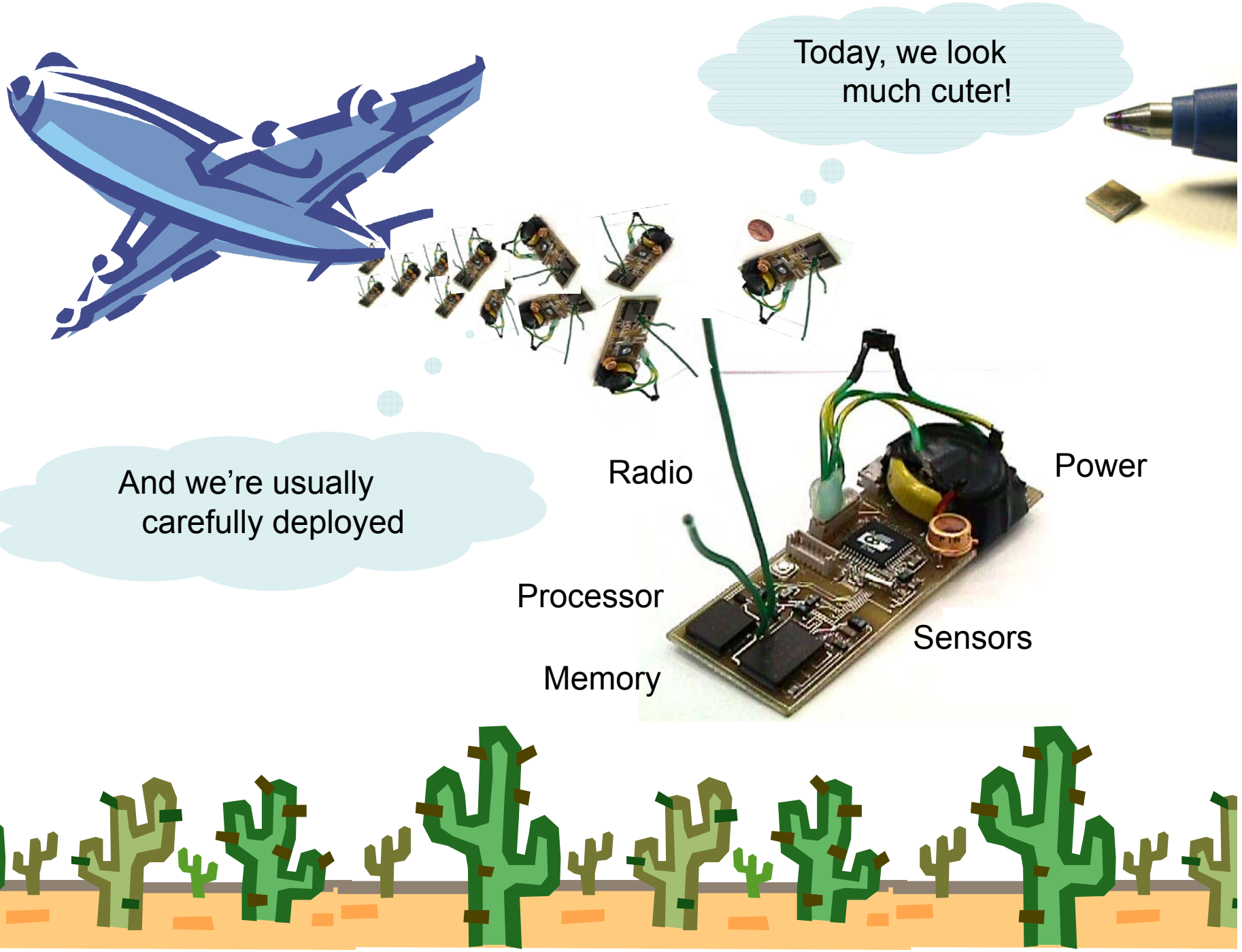


Systems people don't read theory papers

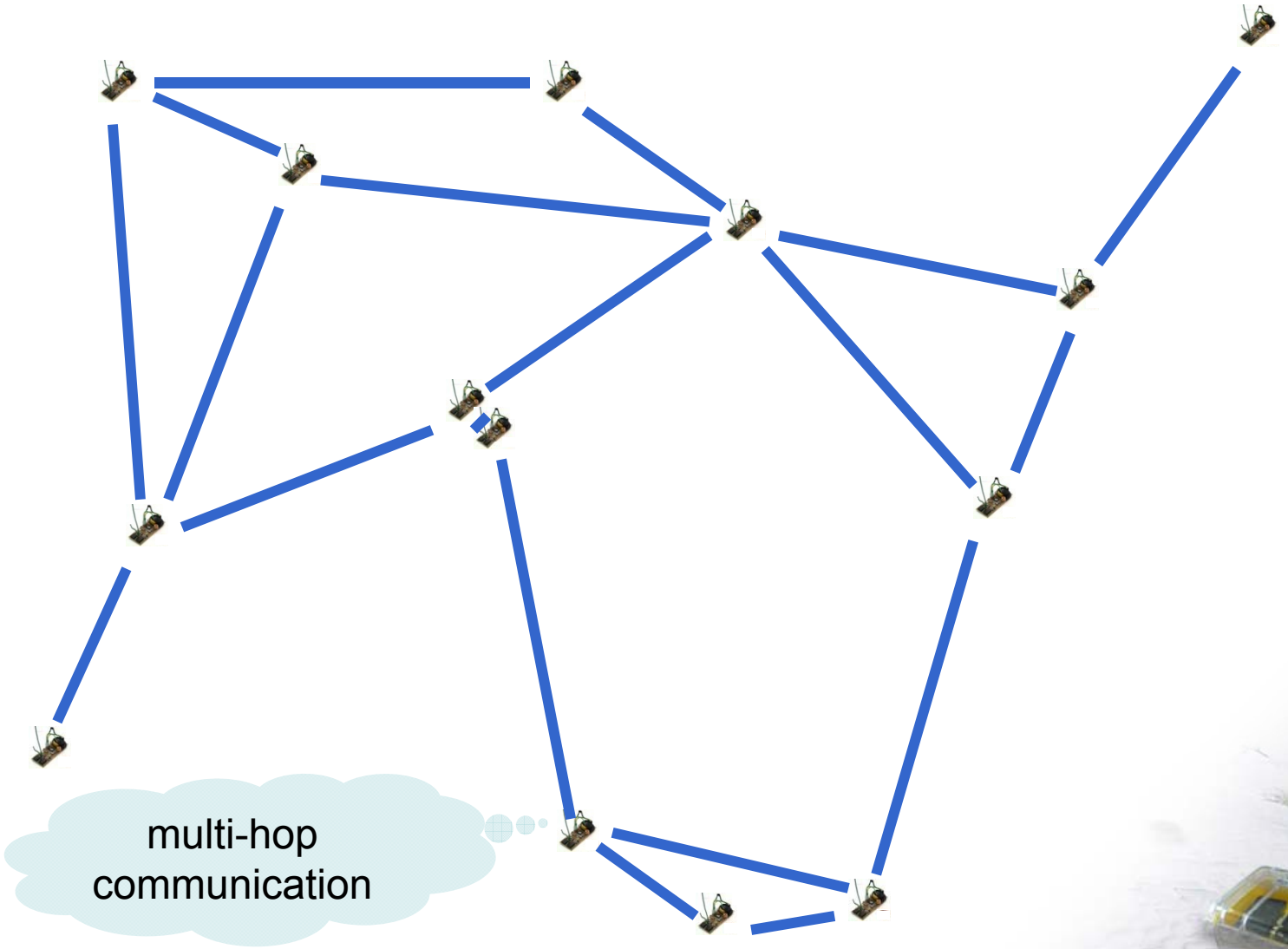
- Sometimes for good reasons…
  - unreadable
  - don't matter that much (only getting out the last %)
  - wrong models
  - theory is lagging behind
  - bad theory merchandising/branding
    - systems papers provide easy to remember acronyms
    - "On the Locality of Bounded Growth" vs. "Smart Dust"
  - 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
Radio
Power
Processor
Sensors
Memory

A Sensor Network After Deployment

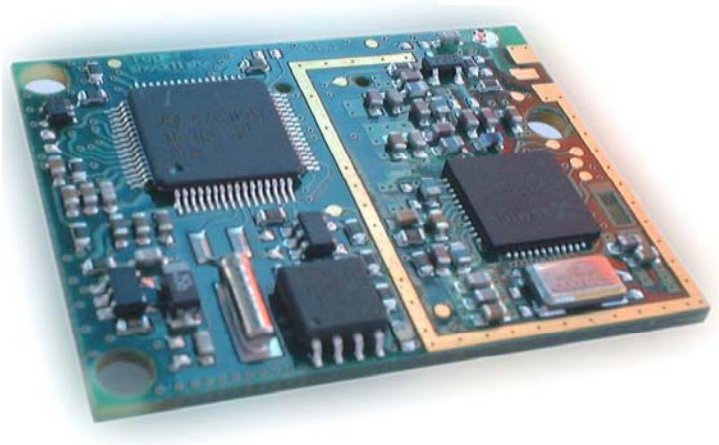


A Typical Sensor Node: TinyNode 584

[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

	Current Draw	Power Consumption
uC sleep with timer on	6.5 uA	0.0195 mW
uC active, radio off	2.1 mA	6.3 mW
uC active, radio idle listening	16 mA	48 mW
uC active, radio TX/RX at +12dBm	62 mA	186 mW
Max. Power (uC active, radio TX/RX at +12dBm + flash write)	76.9 mA	230.7mW



ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

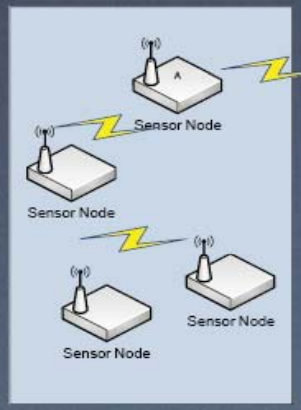
The PermaSense Project
Matterhorn Field Site Installations

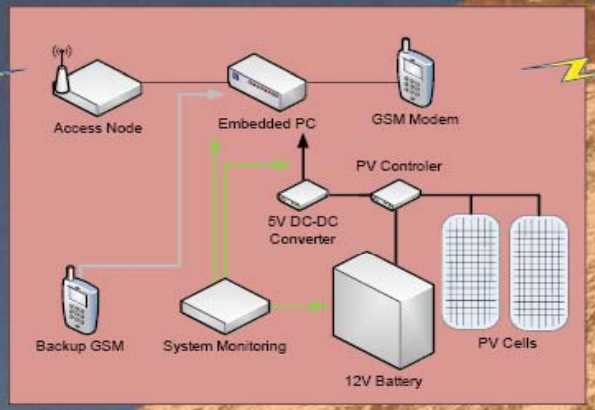
TIK
Computer Engineering and Networks Laboratory



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

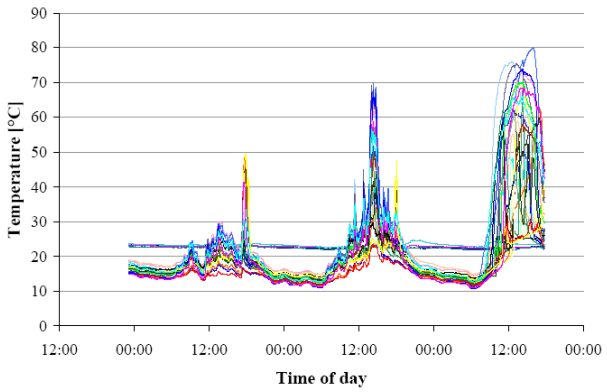
Go

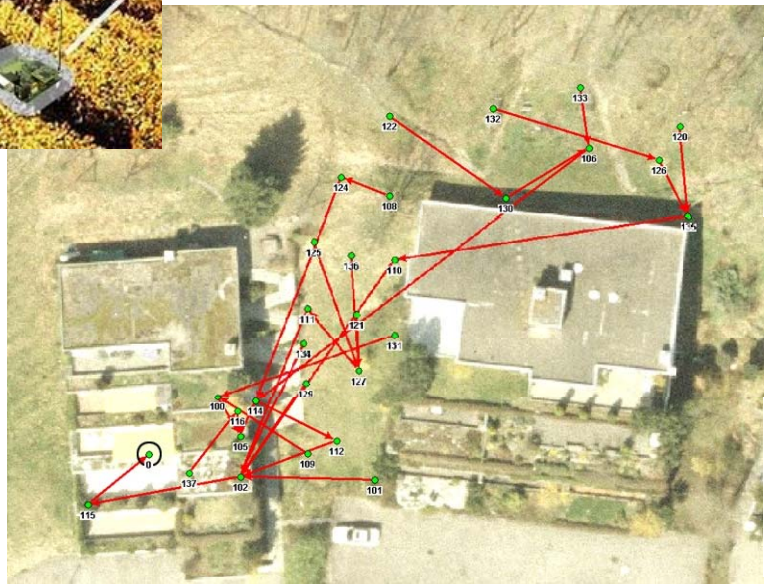

Base station power supply, system monitoring and a backup GSM modem are housed separately

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]

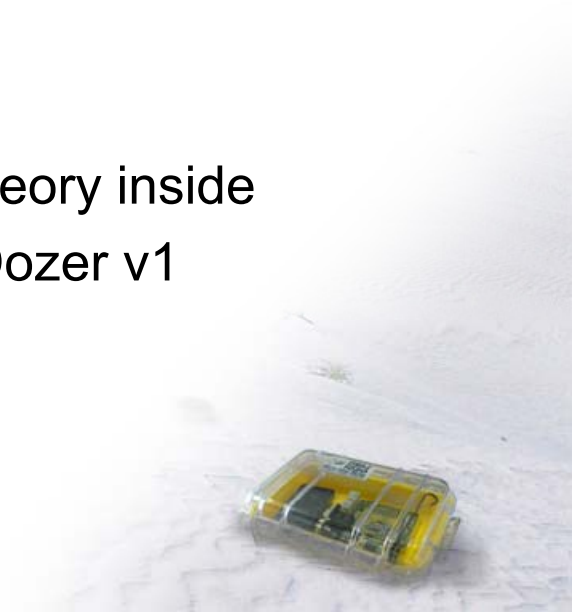




Is Dozer a theory-meets-practice success story?

- 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

TinyNode	Power Consumption
uC sleep, radio off	0.015 mW
Radio idle, RX, TX	30 – 40 mW

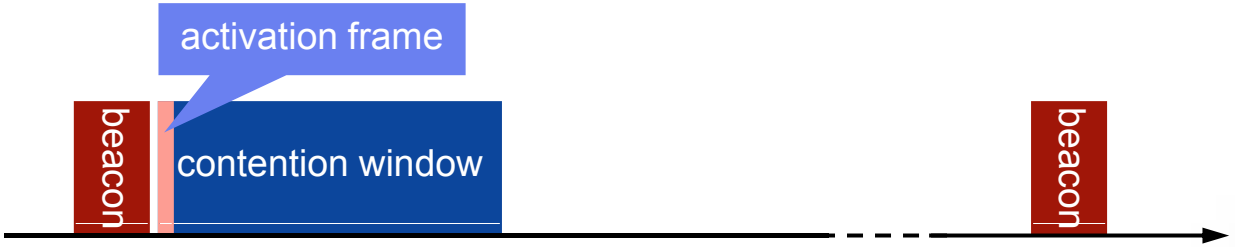
- Issue is tackled at various layers
  - MAC
  - Topology control / clustering
  - Routing

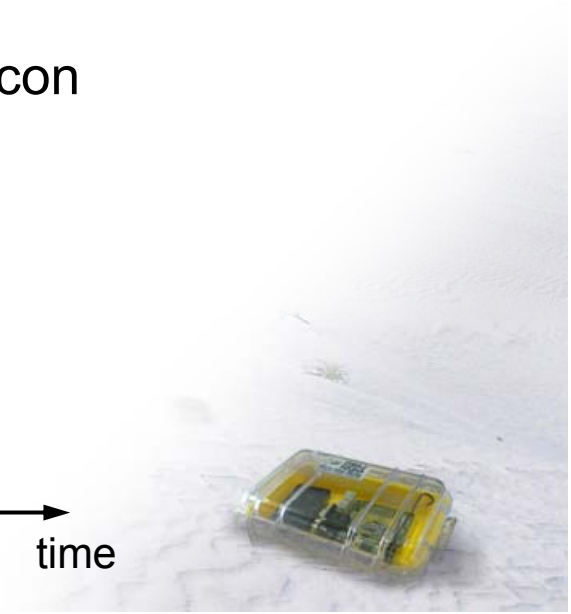
➡ Orchestration of the whole network stack to achieve duty cycles of ~ 0.1%

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

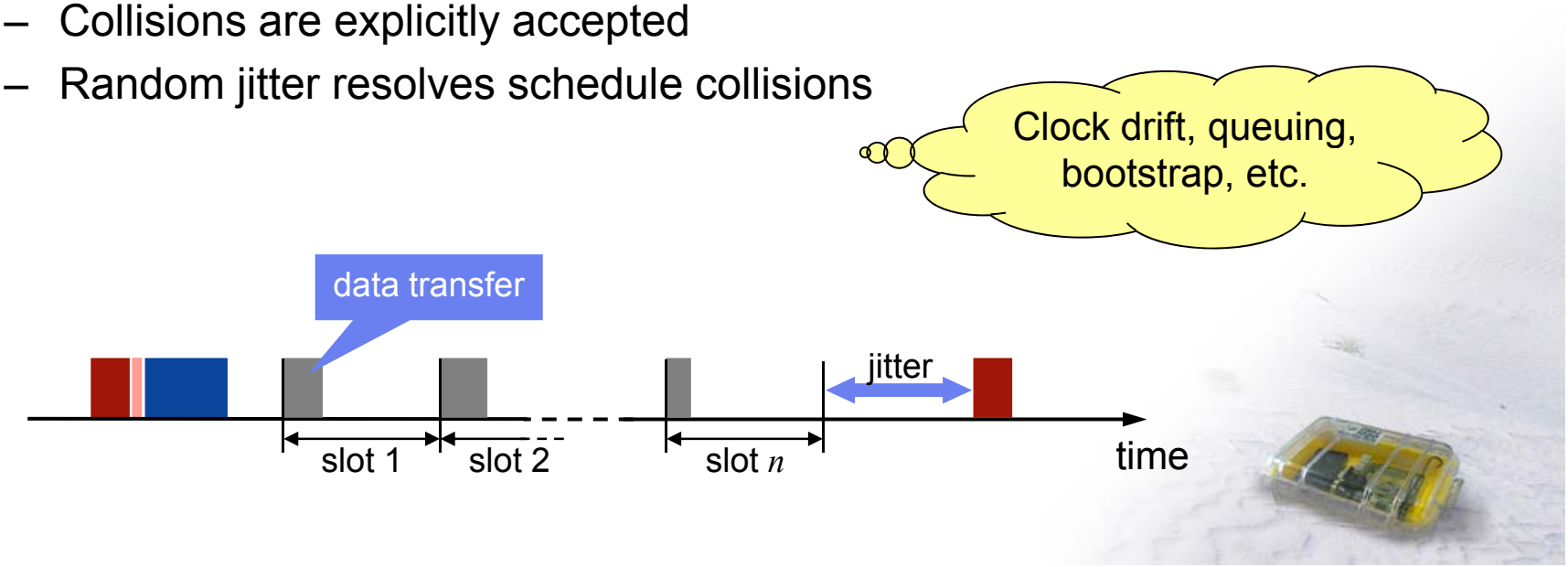




Dozer System

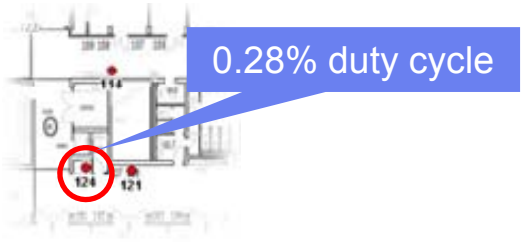
- Parent decides on its children data upload times
  - Each interval is divided into upload slots of equal length
  - Upon connecting each child gets its own slot
  - Data transmissions are always ack'ed

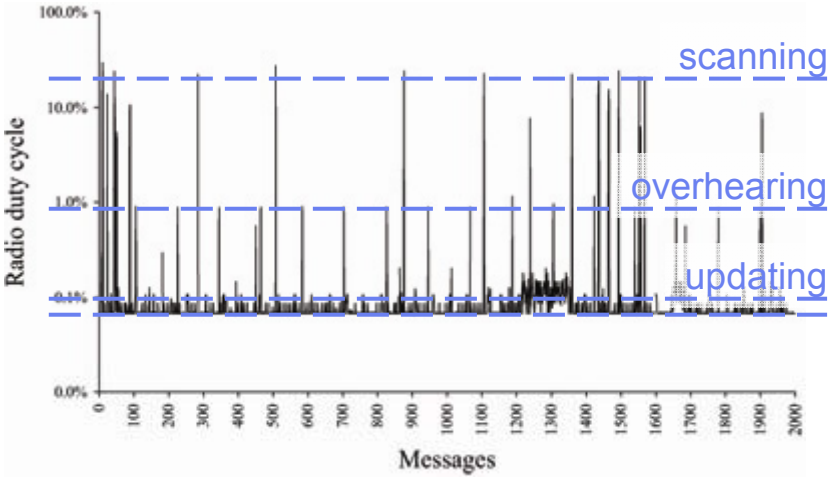
- No traditional MAC layer
  - Transmissions happen at exactly predetermined point in time
  - Collisions are explicitly accepted
  - Random jitter resolves schedule collisions



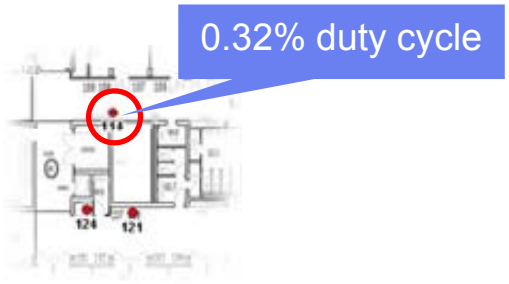
Dozer in Action

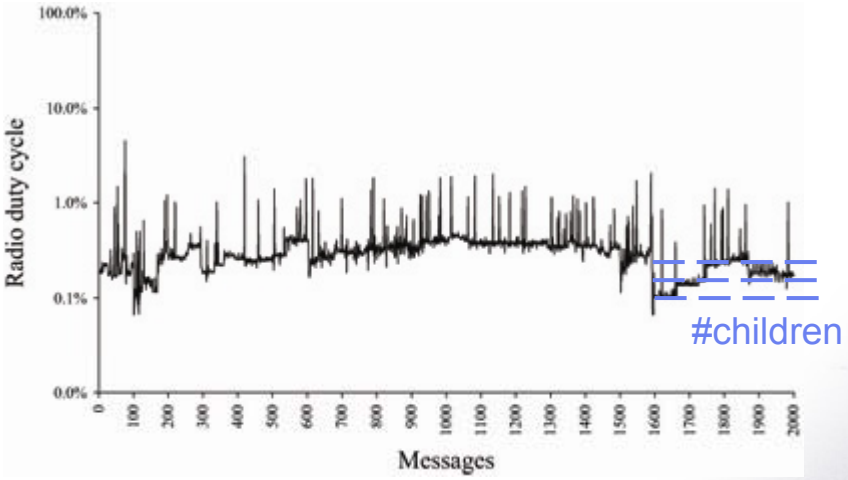
Energy Consumption



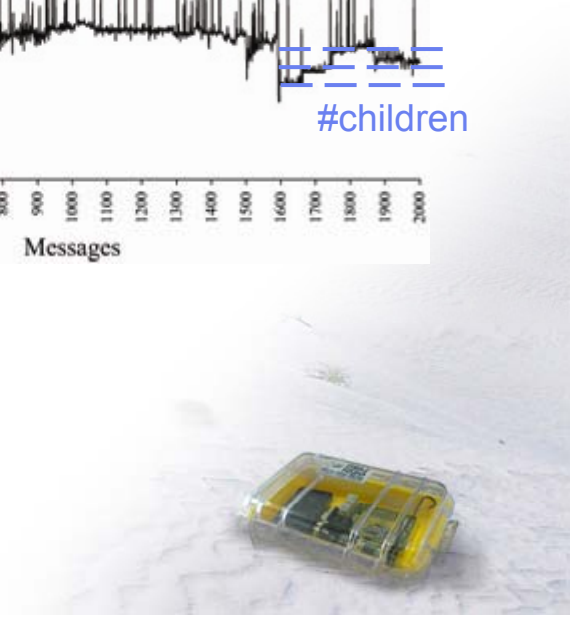


- Leaf node
- Few neighbors
- Short disruptions





- Relay node
- No scanning



Example: Clock Synchronization

...it's about TIME!

Clock Synchronization in Practice

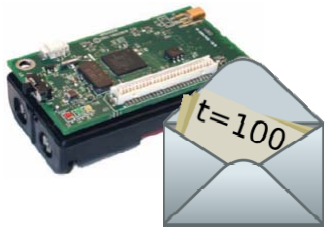
- Many different approaches for clock synchronization

Global Positioning System (GPS)

Radio Clock Signal

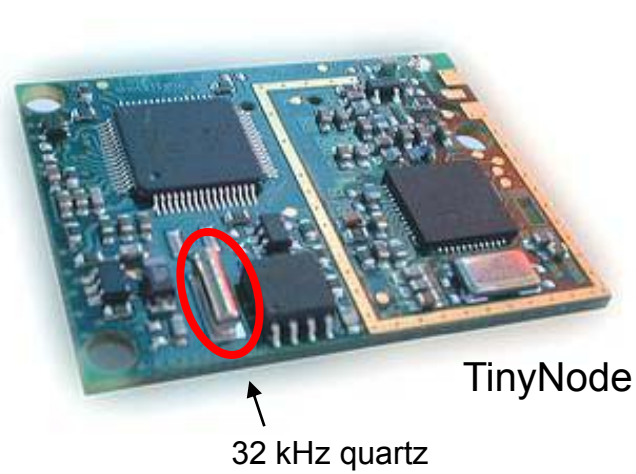
AC-power line radiation

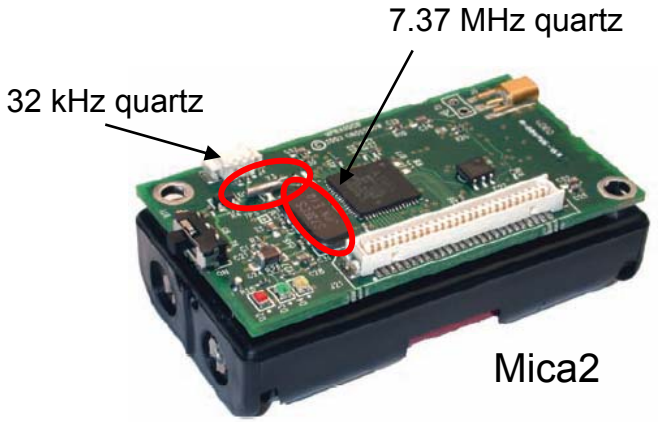
Synchronization messages



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

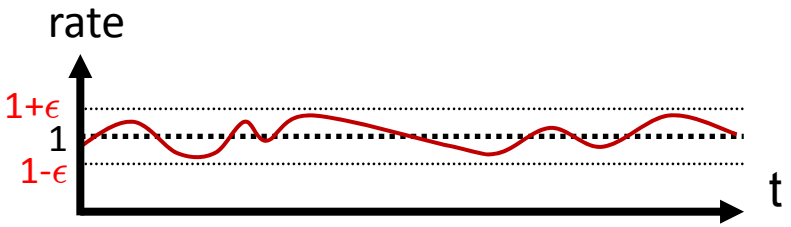




Platform	System clock	Crystal oscillator
Mica2	7.37 MHz	32 kHz, 7.37 MHz
TinyNode 584	8 MHz	32 kHz
Tmote Sky	8 MHz	32 kHz

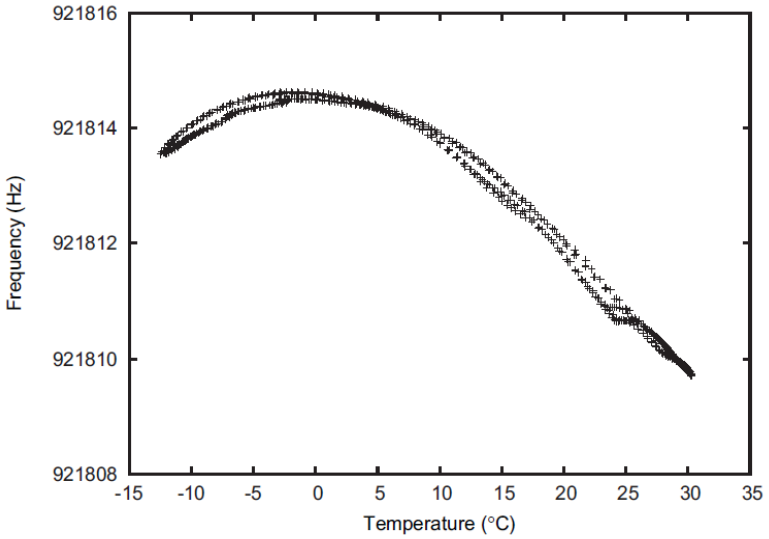
Clock Drift

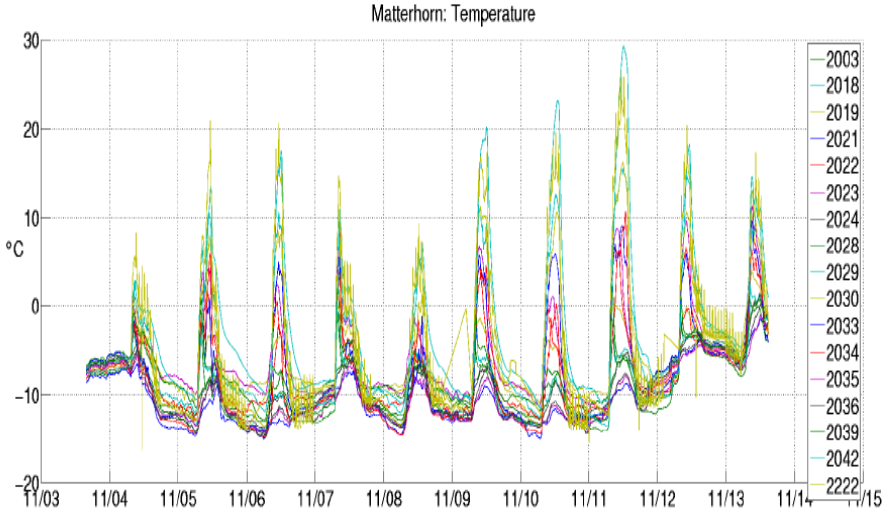
- Accuracy
  - Clock drift: random deviation from the nominal rate dependent on power supply, temperature, etc.



This is a drift of up to 50 µs per second or 0.18s per hour

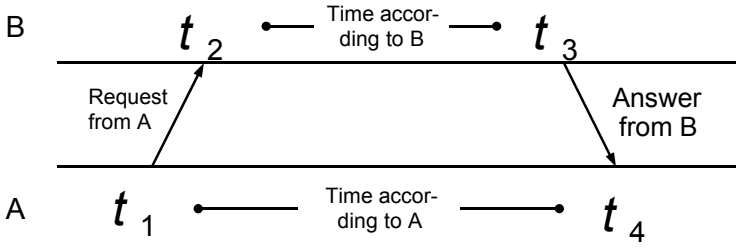
  - E.g. TinyNodes have a maximum drift of 30-50 ppm at room temperature





Sender/Receiver Synchronization

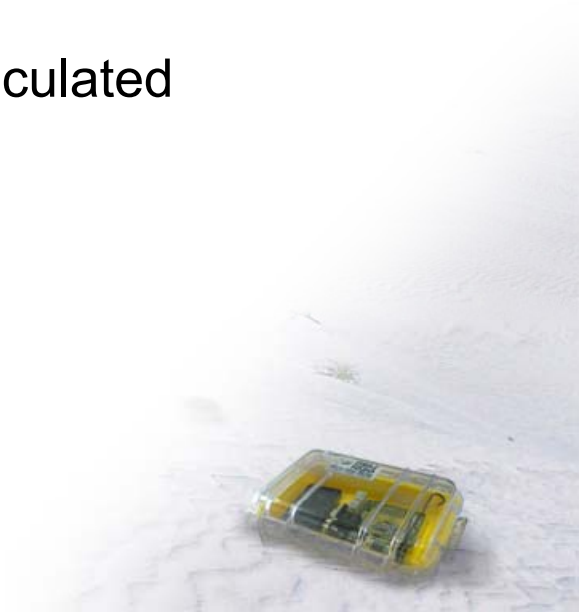
- Round-Trip Time (RTT) based synchronization



- Receiver synchronizes to sender‘s clock
- Propagation delay δ and clock offset θ can be calculated

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

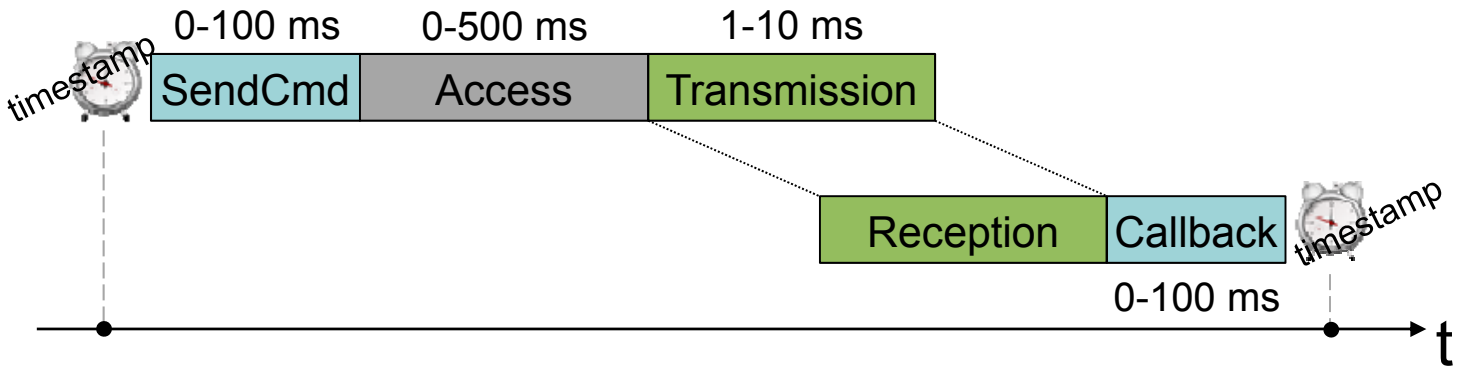
$$\theta = \frac{(t_2 - (t_1 + \delta)) - (t_4 - (t_3 + \delta))}{2} = \frac{(t_2 - t_1) + (t_3 - t_4)}{2}$$



Messages Experience Jitter in the Delay

- Problem: Jitter in the message delay

  Various sources of errors (deterministic and non-deterministic)



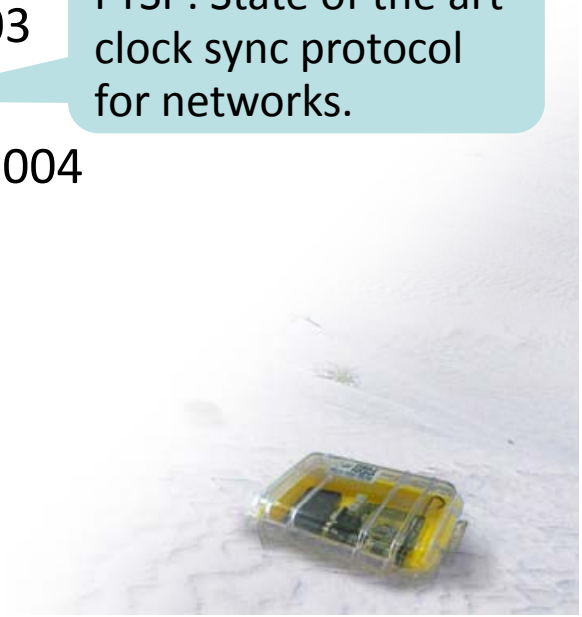
- Solution: Timestamping packets at the MAC layer [Maróti et al.]

  → Jitter in the message delay is reduced to a few clock ticks

Clock Synchronization in Networks?

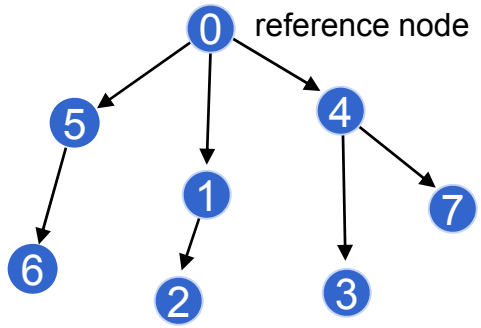
- *Time, Clocks, and the Ordering of Events in a Distributed System*
  L. Lamport, Communications of the ACM, 1978.
- *Internet Time Synchronization: The Network Time Protocol (NTP)*
  D. Mills, IEEE Transactions on Communications, 1991
- *Reference Broadcast Synchronization (RBS)*
  J. Elson, L. Girod 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

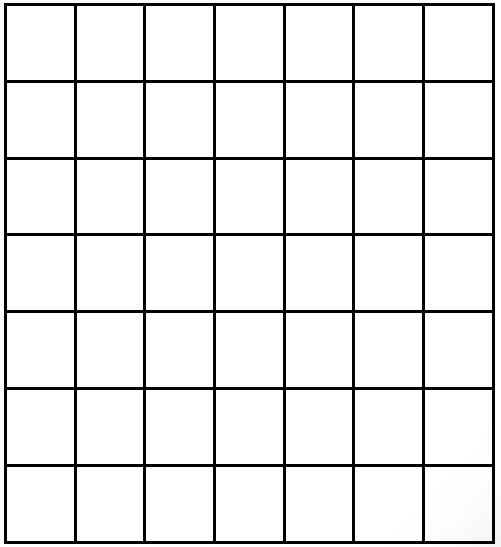


- Timestamping at the MAC Layer is used to compensate for deterministic message delays
- Compensation for clock drift between synchronization messages using a linear regression table

Best tree for tree-based clock synchronization?

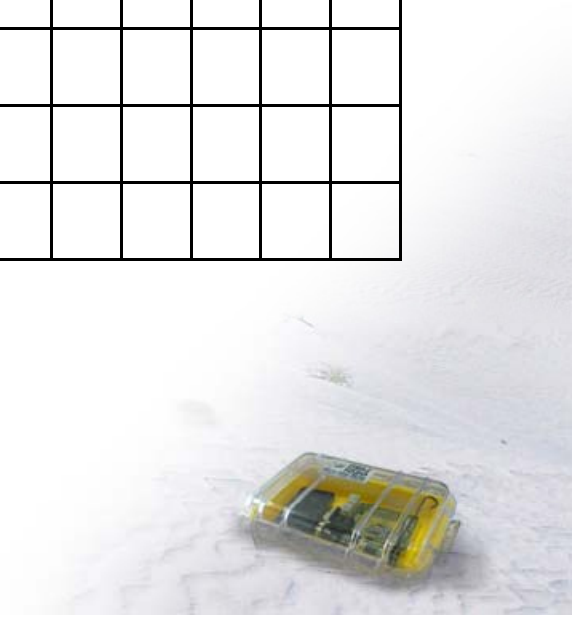
- Finding a good tree for clock synchronization is a tough problem
  - Spanning tree with small (maximum or average) stretch.

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



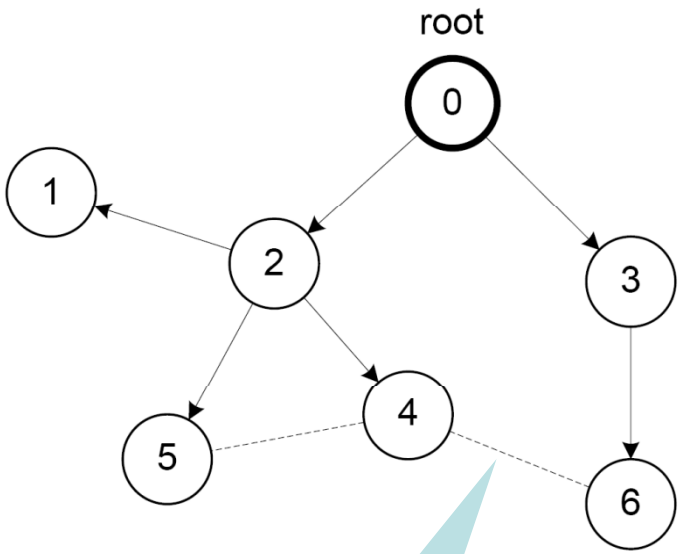
- No matter what tree you use, the maximum stretch of the spanning tree will always be at least m (just try on the grid figure right…)

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



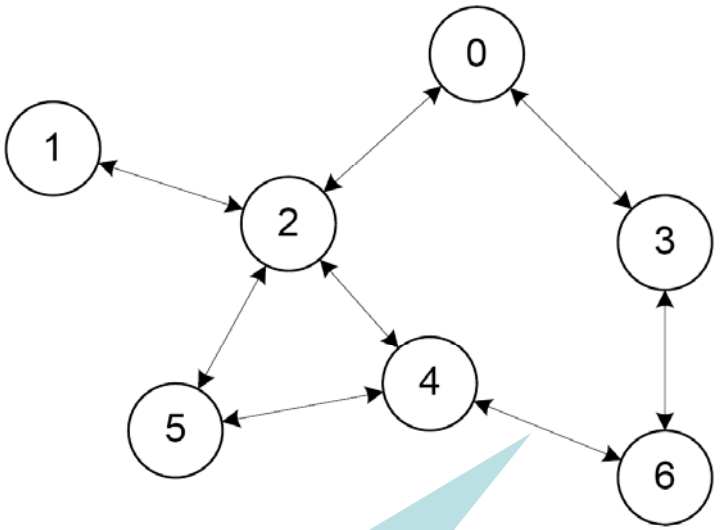
Variants of Clock Synchronization Algorithms

Tree-like Algorithms
e.g. FTSP

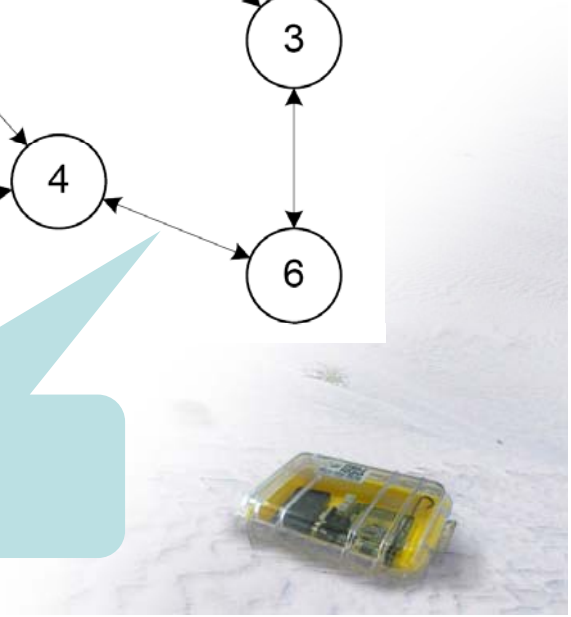


Bad local skew

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

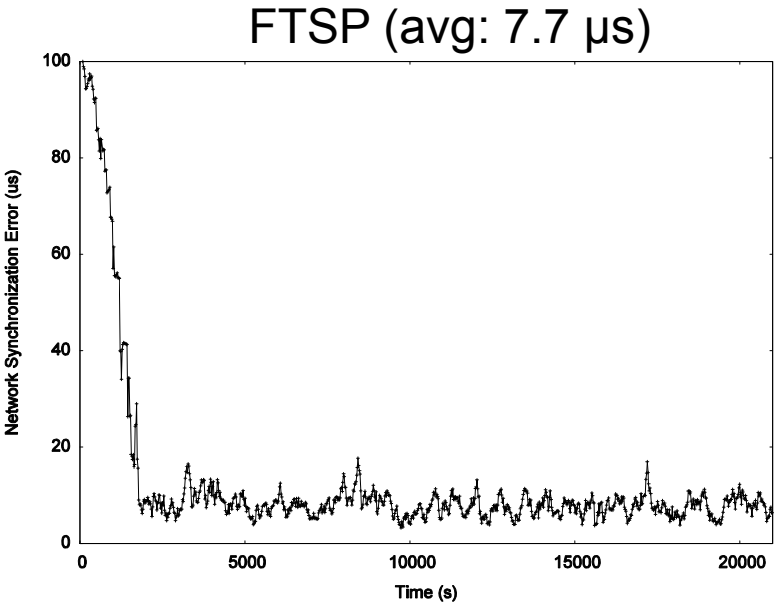


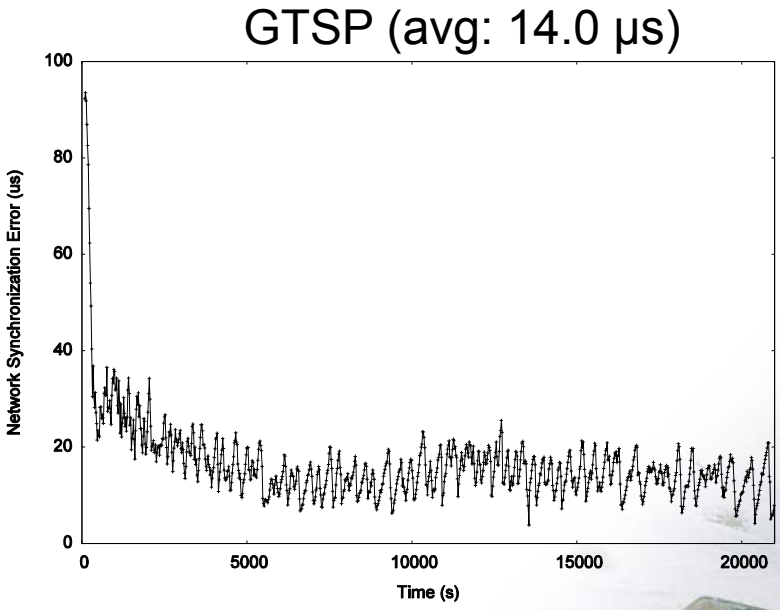
All nodes consistently average errors to *all* neighbors

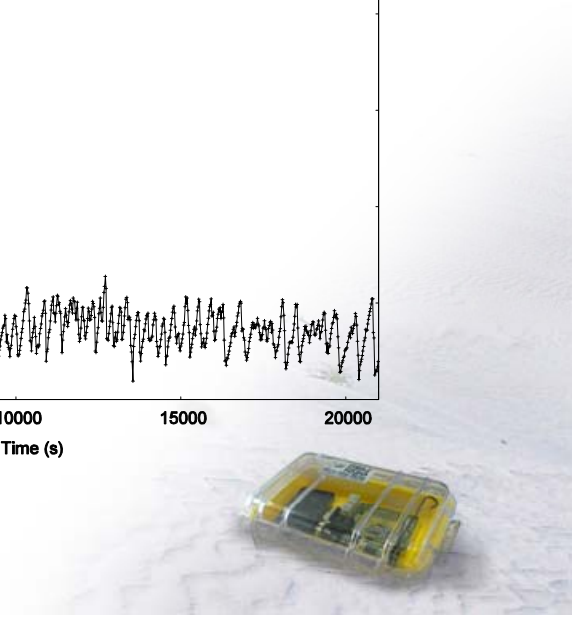


FTSP vs. GTSP: Global Skew

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





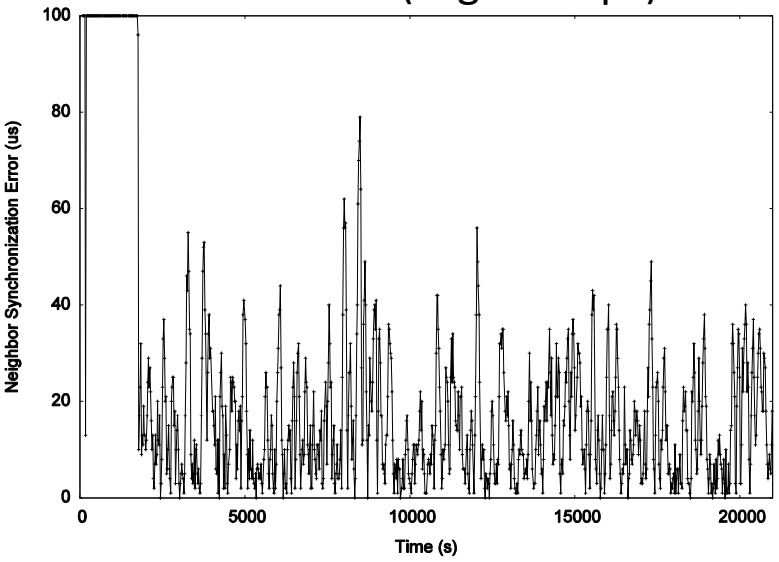


FTSP vs. GTSP: Local Skew

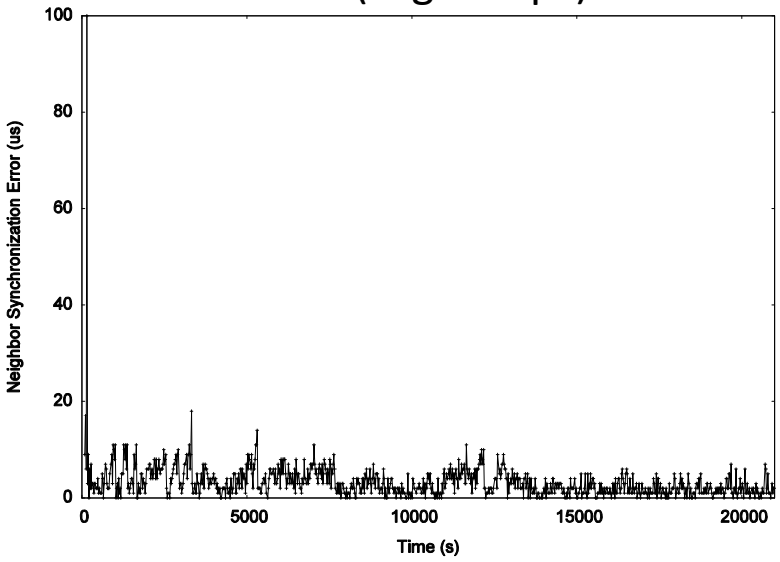
- Neighbor Synchronization error (local skew)
  - Pair-wise synchronization error between neighboring nodes

- Synchronization error between two direct neighbors:

FTSP (avg: 15.0 µs)

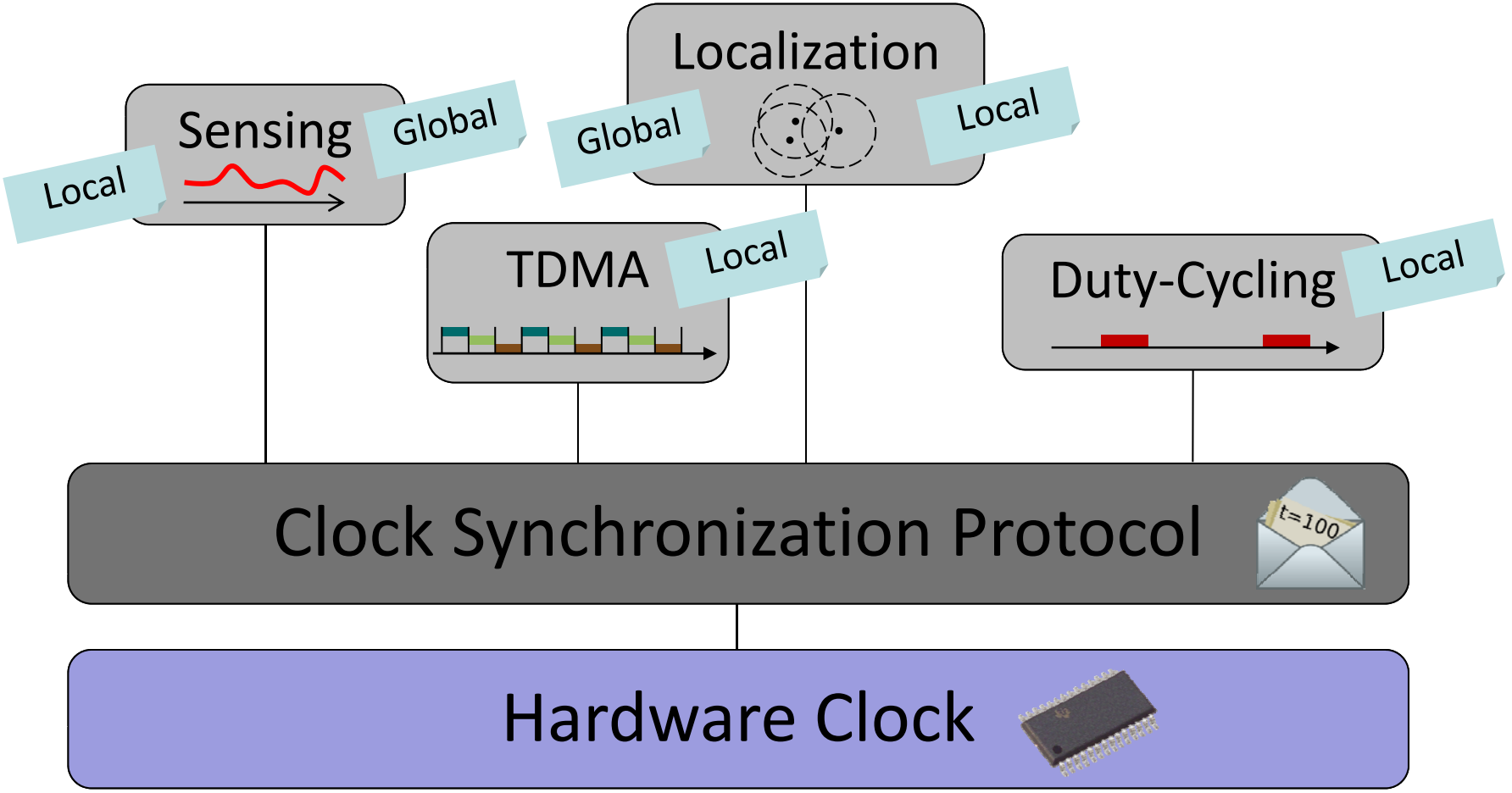


GTSP (avg: 2.8 µs)



Time in (Sensor) Networks

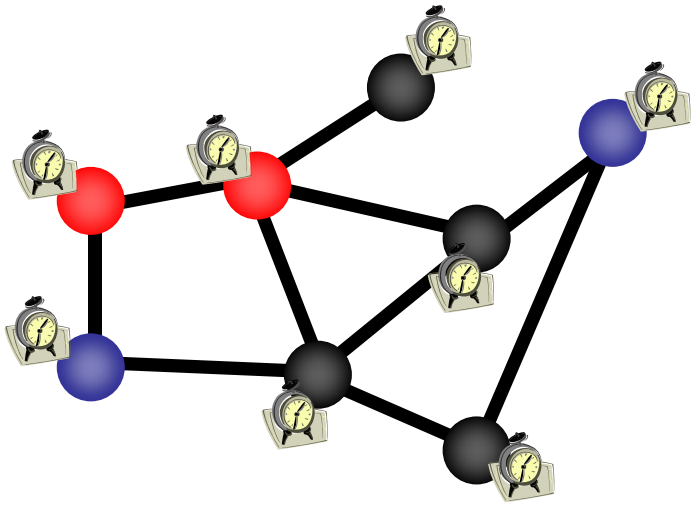
- Synchronized clocks are essential for many applications:



Clock Synchronization in Theory?

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

worst-case (but constant)



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

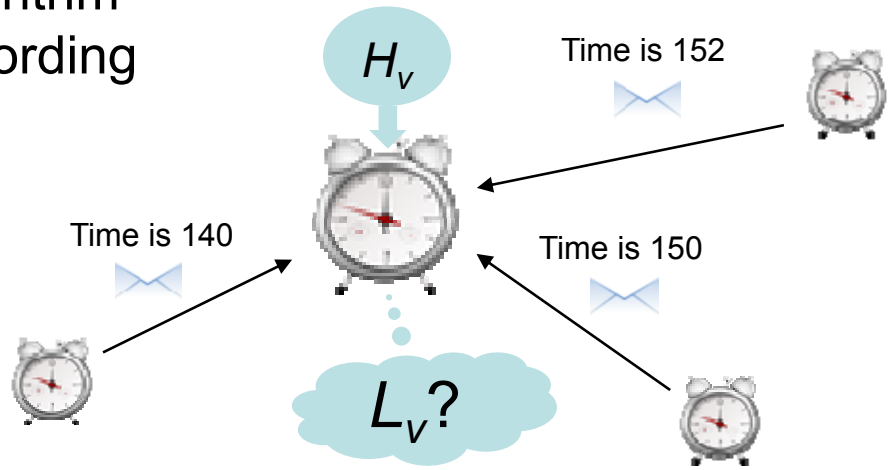
- Logical clock $L_v(\cdot)$ which increases at rate at least 1 and at most β

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

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

Neglect fixed share of delay, normalize jitter

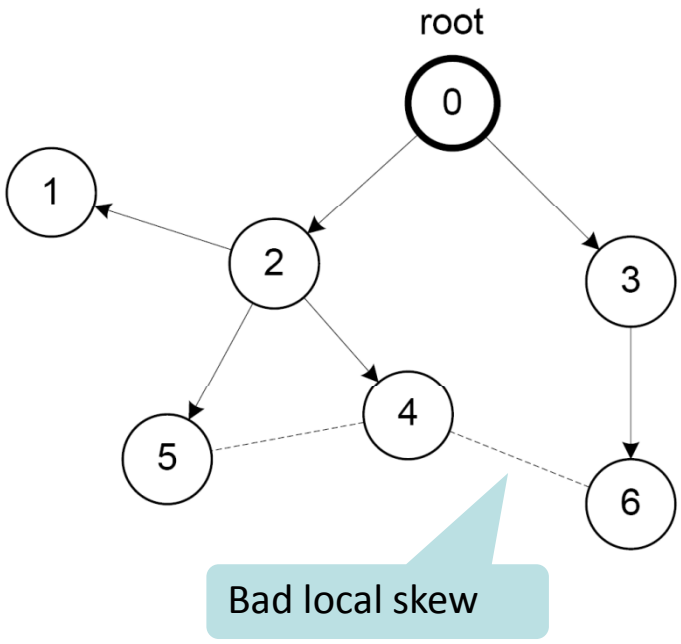
- Employ a synchronization algorithm to update the logical clock according to hardware clock and messages from neighbors



Variants of Clock Synchronization Algorithms

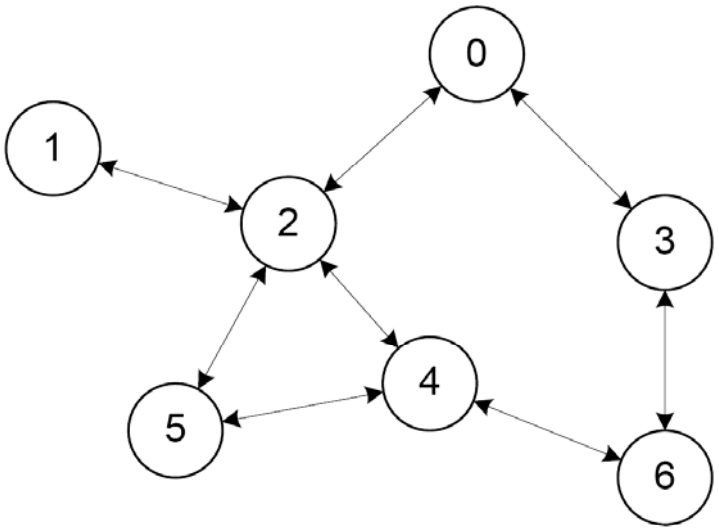
Tree-like Algorithms

e.g. FTSP



Distributed Algorithms

e.g. GTSP

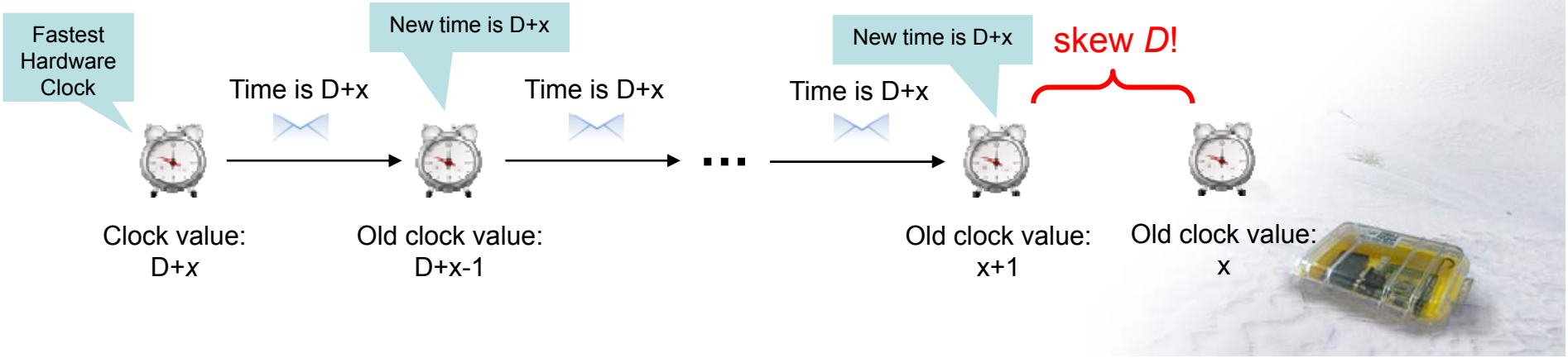


Synchronization Algorithms: An Example ("A^{max}")

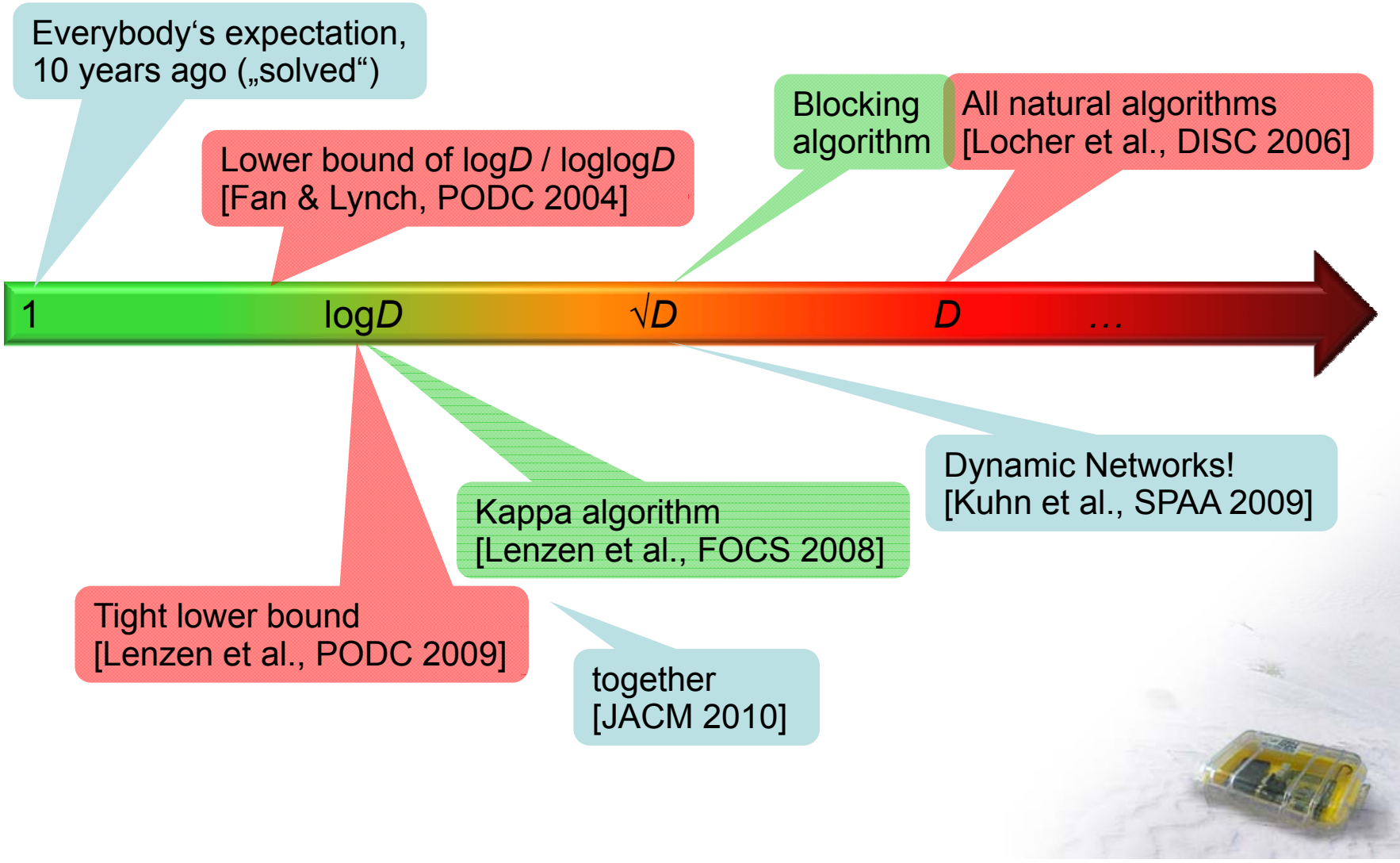
- Question: How to update the logical clock based on the messages from the neighbors?

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

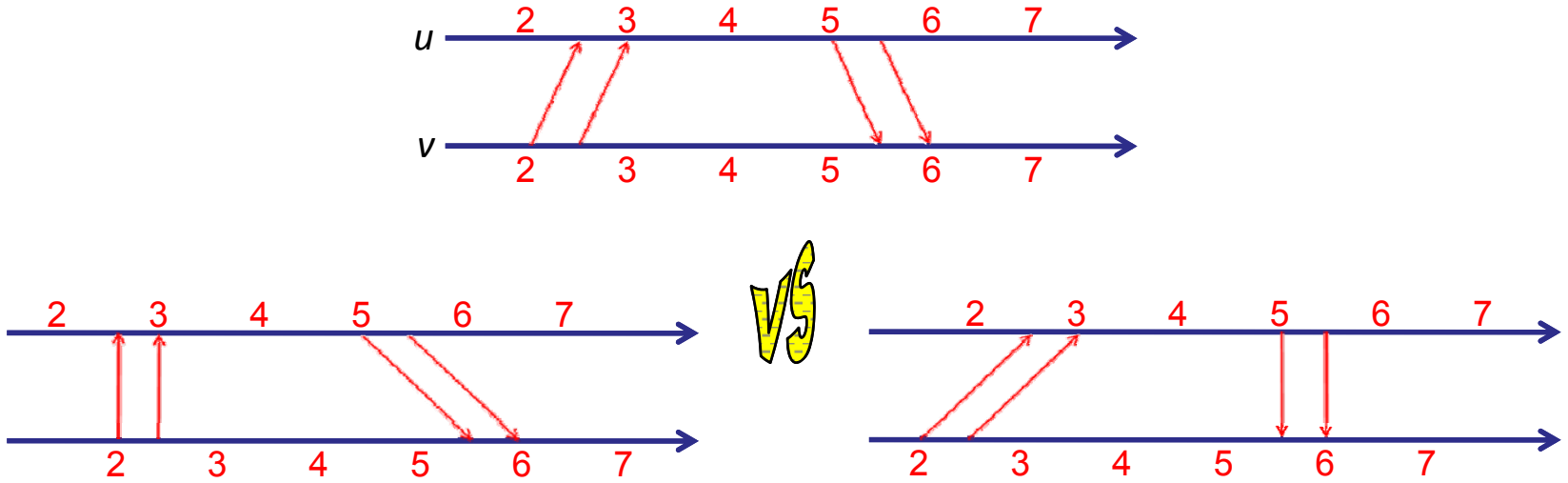
- Idea: Minimizing the skew to the fastest neighbor
  - Set the clock to the maximum clock value received from any neighbor (if larger than local clock value)
  - forward new values immediately
- Optimum global skew of about *D*
- Poor local property
  - First all messages take 1 time unit…
  - …then we have a fast message!



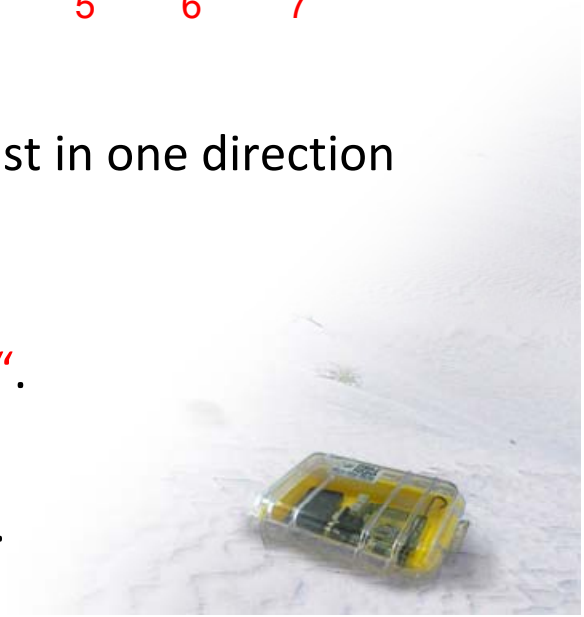
Local Skew: Overview of Results



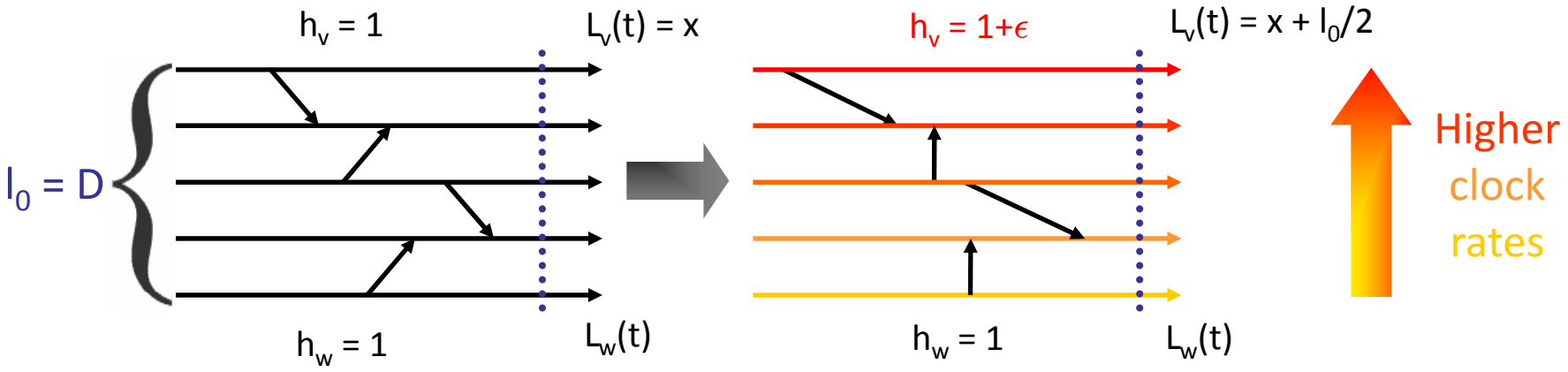
Enforcing Clock Skew



- 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 (Single-Slide Proof!)



- Add $l_0/2$ skew in $l_0/(2\epsilon)$ time, messing with clock rates and messages
- Afterwards: Continue execution for $l_0/(4(\beta\text{-}1))$ time (all $h_x = 1$)
  - → Skew reduces by at most $l_0/4$ → at least $l_0/4$ skew remains
  - → Consider a subpath of length $l_1 = l_0 \cdot \epsilon/(2(\beta\text{-}1))$ with at least $l_1/4$ skew
  - → Add $l_1/2$ skew in $l_1/(2\epsilon) = l_0/(4(\beta\text{-}1))$ time → at least $3/4 \cdot l_1$ skew in subpath
- Repeat this trick (+½,-¼,+½,-¼,…) $\log_{2(\beta\text{-}1)/\epsilon} D$ times

Theorem: $\Omega(\log_{(\beta\text{-}1)/\epsilon} D)$ skew between neighbors

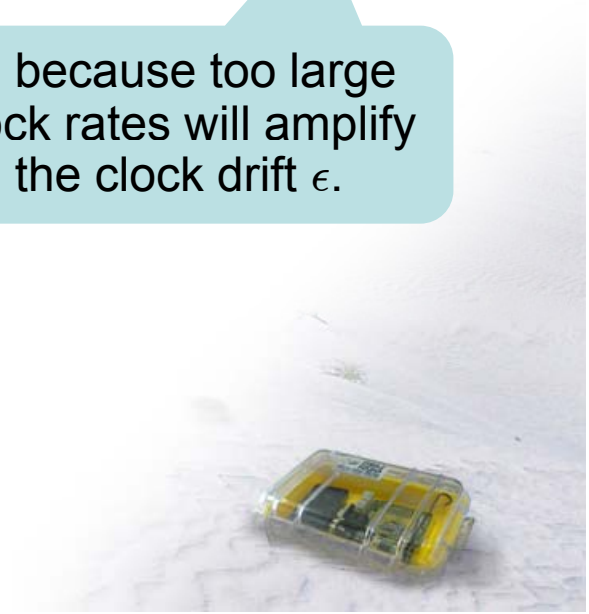
Local Skew: Upper Bound

- 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]:

max rate β	$1+\epsilon$
local skew	∞

We can have both smooth and accurate clocks!

... because too large clock rates will amplify the clock drift ϵ.



Local Skew: Upper Bound

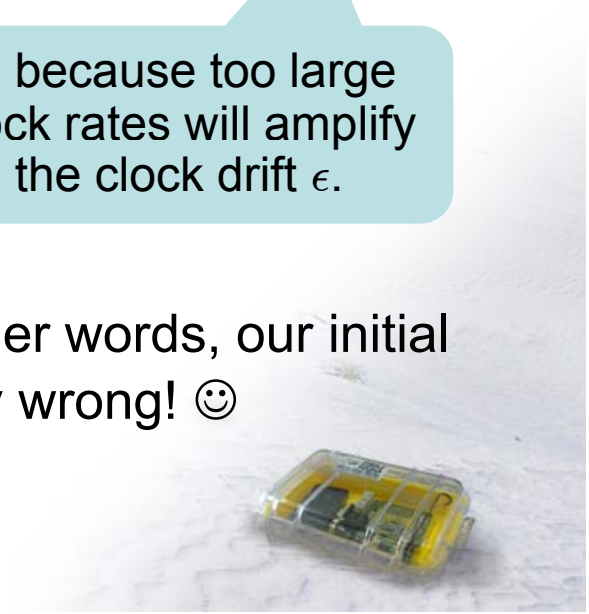
- Surprisingly, up to small constants, the $\Omega(\log_{(\beta-1)/\epsilon} D)$ lower bound can be matched with clock rates $\in [1,\beta]$ (tough part, not in this talk)
- We get the following picture [Lenzen et al., PODC 2009]:

max rate β	$1+\epsilon$	$1+\Theta(\epsilon)$	$1+\sqrt{\epsilon}$	2	large
local skew	∞	$\Theta(\log D)$	$\Theta(\log_{1/\epsilon} D)$	$\Theta(\log_{1/\epsilon} D)$	$\Theta(\log_{1/\epsilon} D)$

We can have both smooth and accurate clocks!

... because too large clock rates will amplify the clock drift ϵ.

- 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

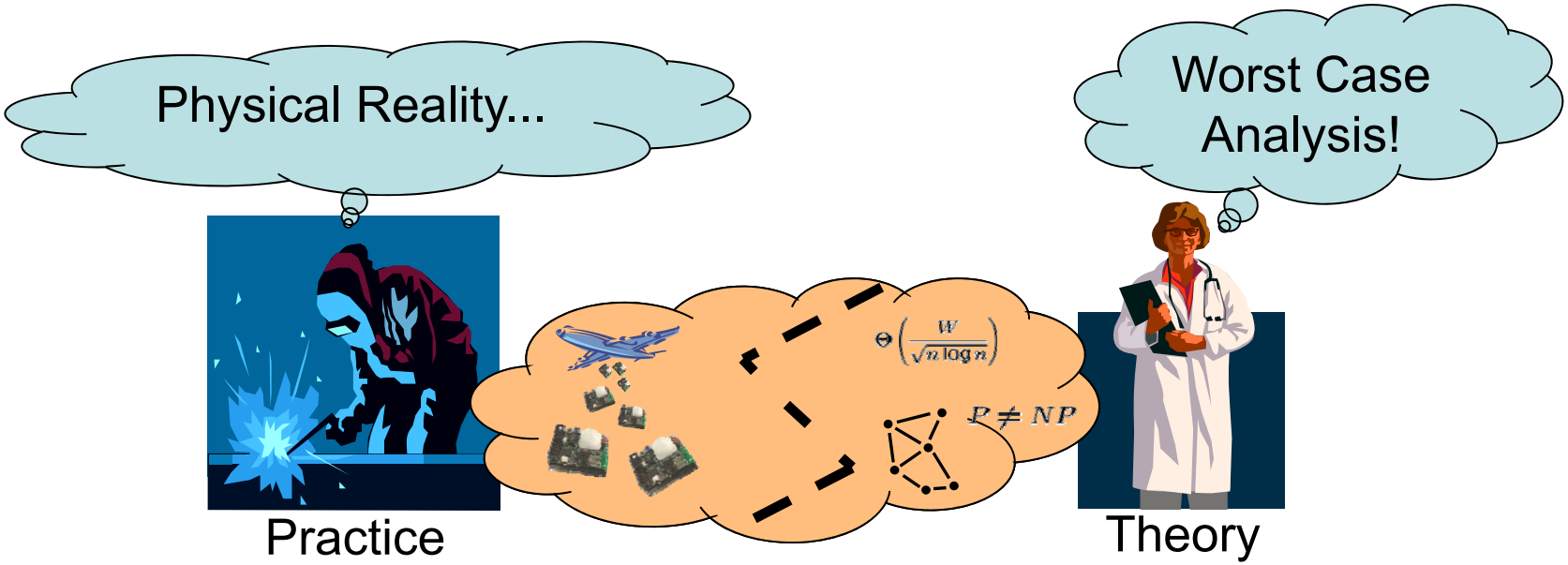
- In the future cars may travel at high speed despite a tiny safety distance, thanks to advanced sensors and communication

- 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

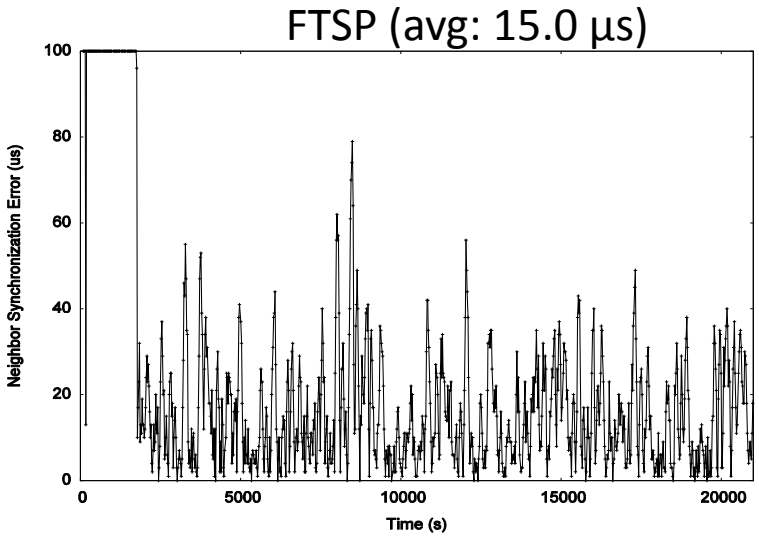
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

One Big Difference Between Theory and Practice, Usually!

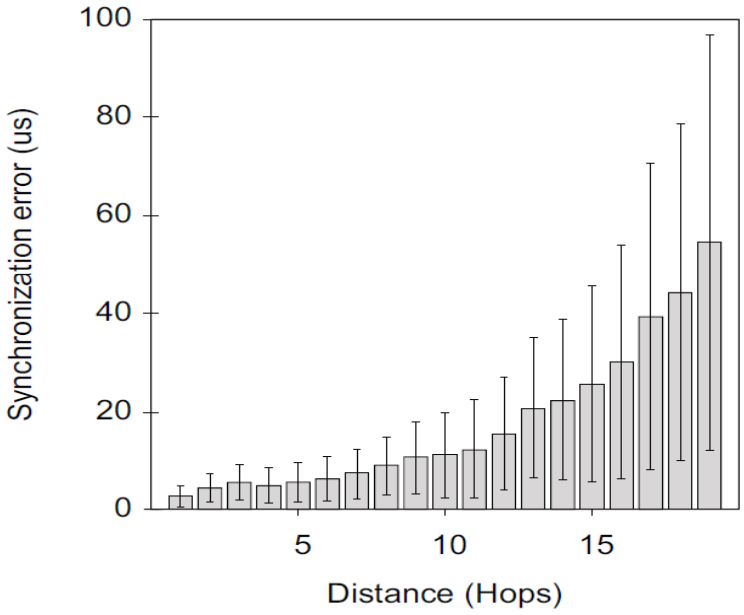


„Industry Standard“ FTSP in Practice

- As we have seen FTSP does have a local skew problem
- But it’s not all that bad…

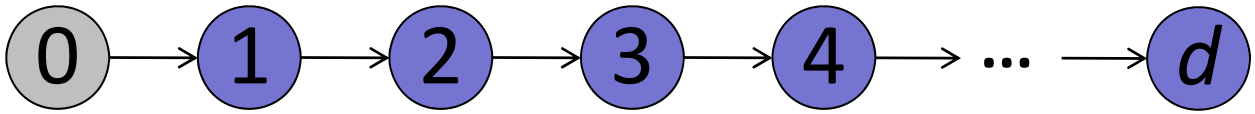


- However, tests revealed another (severe!) problem:
- FTSP does not scale: Global skew grows exponentially with network size…



Why?

- How does the network diameter affect synchronization errors?



- Examples for sensor networks with large diameter

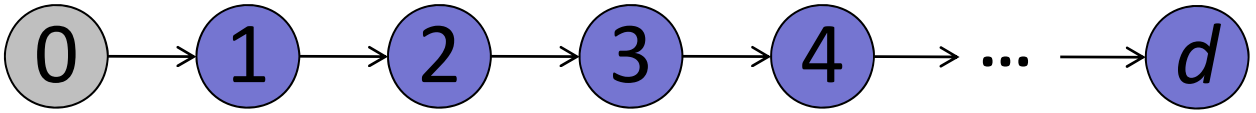
  Bridge, road or pipeline monitoring



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 **random jitter *J***



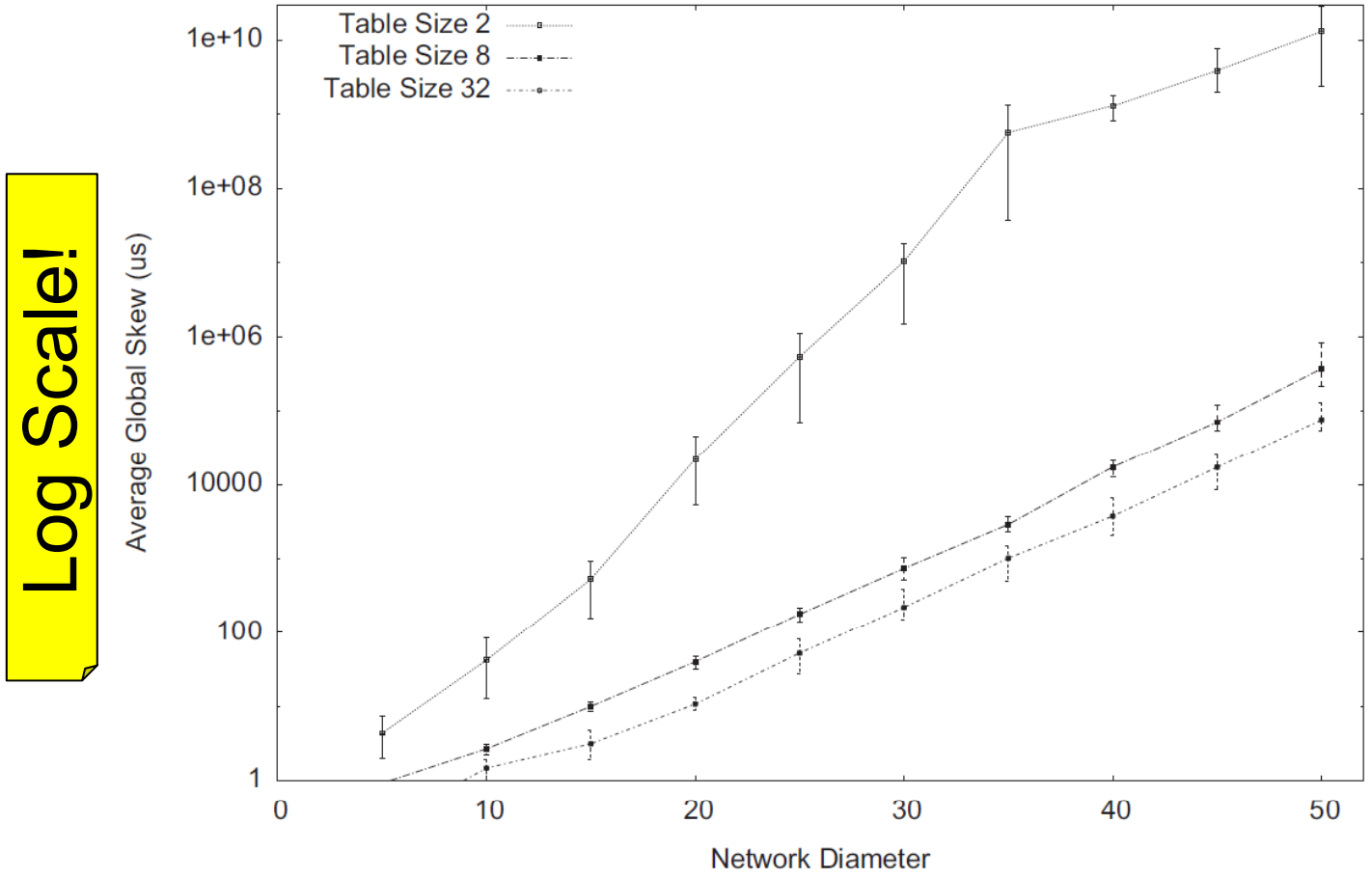
- Sum of the jitter grows with the square-root of the distance
  - stddev($J_1 + J_2 + J_3 + J_4 + J_5 + \ldots J_d$) = $\sqrt{d}$×stddev(J)

- 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)

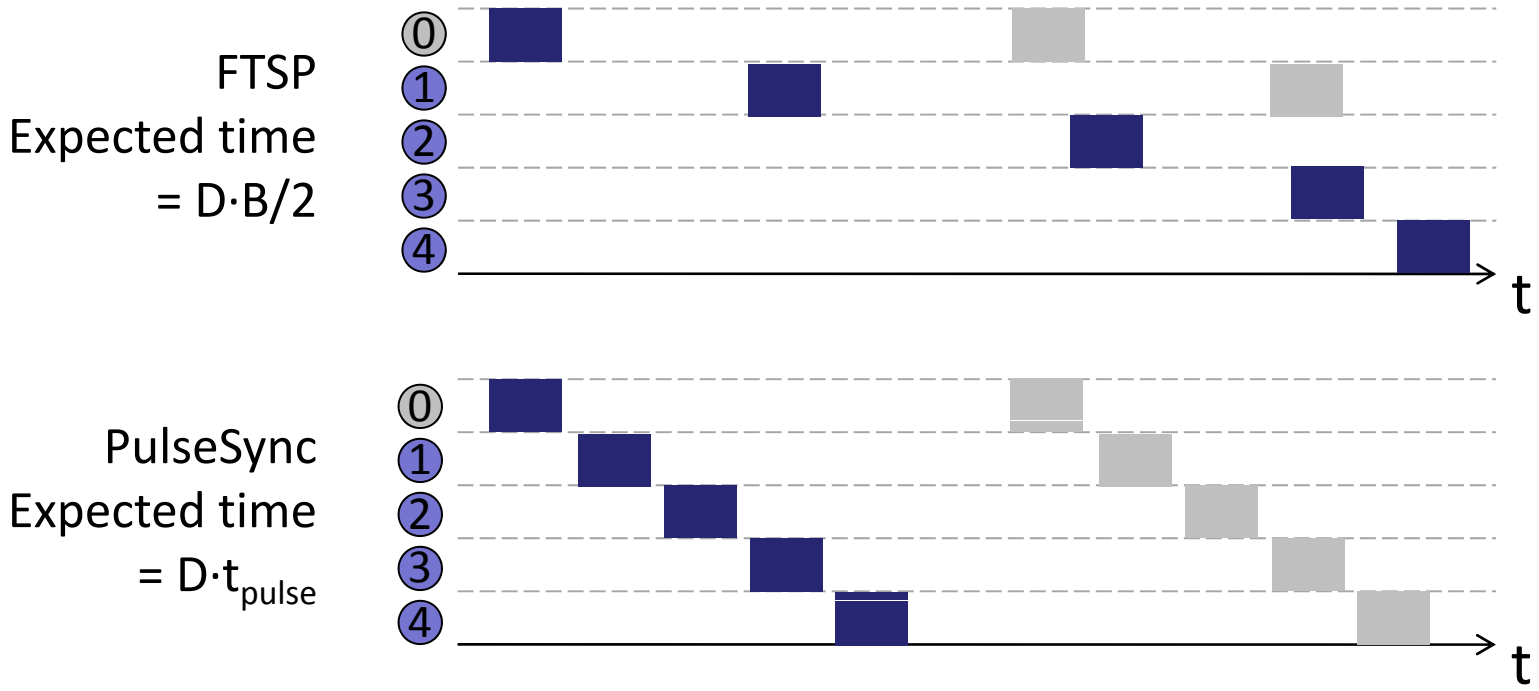
- Simulation of FTSP with regression tables of different sizes (k = 2, 8, 32)



The PulseSync Protocol

[Lenzen et al., SenSys 2009]

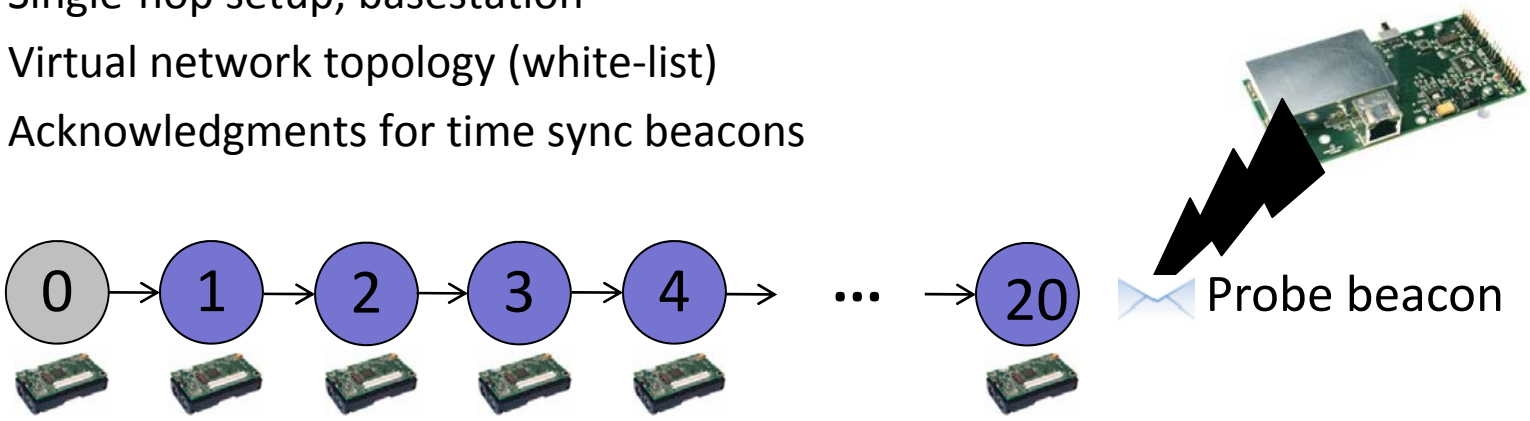
1) Remove self-amplifying of synchronization error
2) Send fast synchronization pulses through the network
   - Speed-up the initialization phase
   - Faster adaptation to changes in temperature or network topology



Evaluation

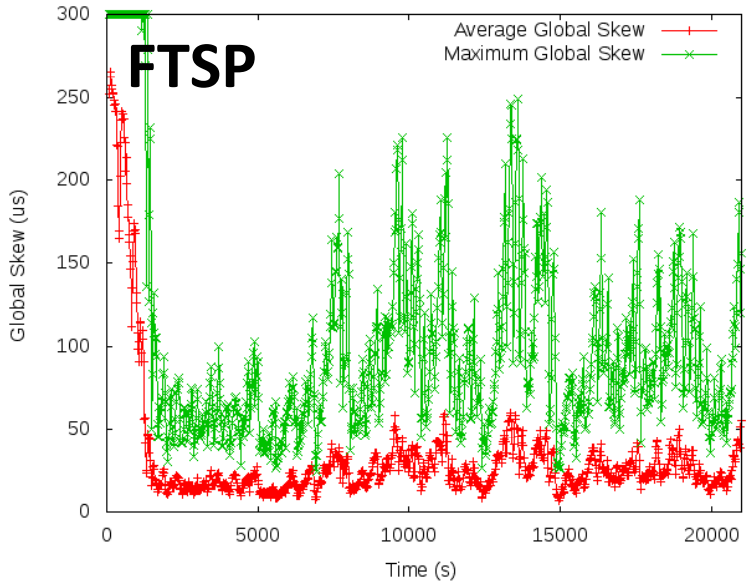
- Testbed setup
  - 20 Crossbow Mica2 sensor nodes
  - PulseSync implemented in TinyOS 2.1
  - FTSP from TinyOS 2.1

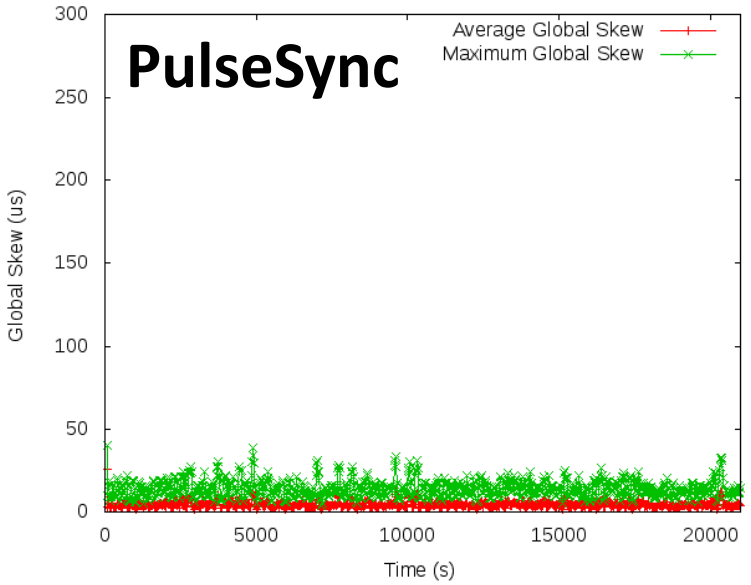
- Network topology
  - Single-hop setup, basestation
  - Virtual network topology (white-list)
  - Acknowledgments for time sync beacons



Experimental Results

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

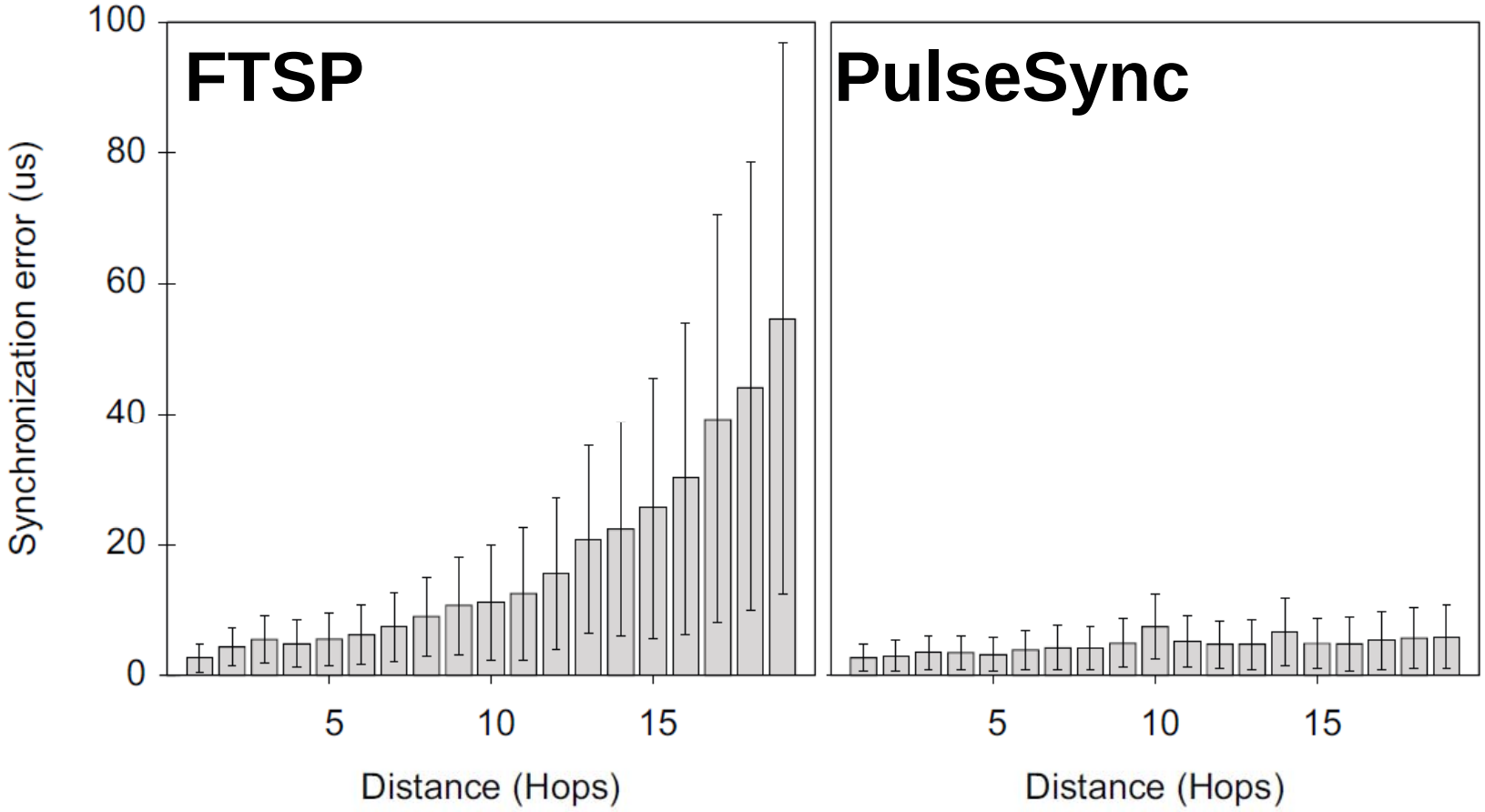




Synchronization Error	FTSP	PulseSync
Average (t>2000s)	23.96 µs	4.44 µs
Maximum (t>2000s)	249 µs	38 µs

Experimental Results

- Synchronization error vs. hop distance



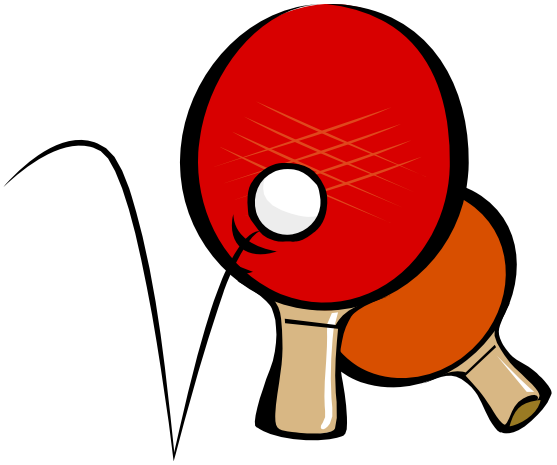
Beyond the list?

- 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.

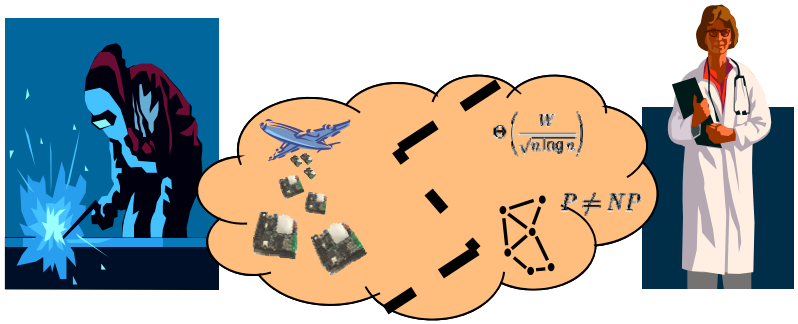


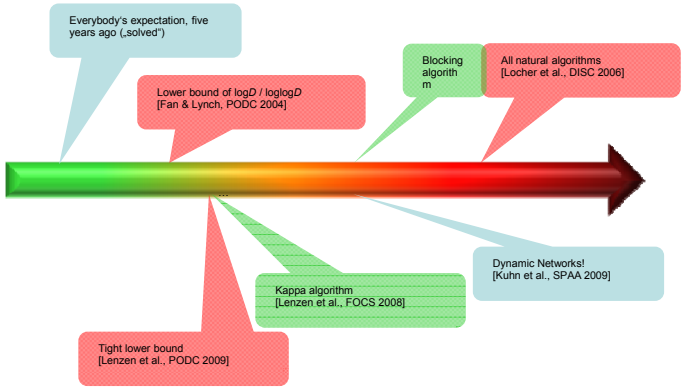
Open Problems

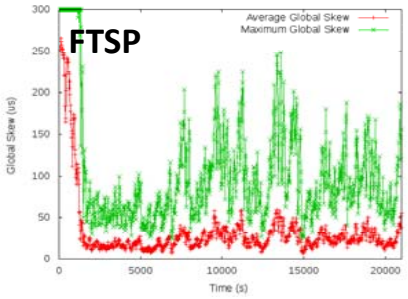
- global vs. local skew
- worst-case vs. reality (Gaussian?)
- accuracy vs. convergence
- accuracy vs. energy efficiency
- dynamic networks
- fault-tolerance (Byzantine clocks)
- applications, e.g. coordinating physical objects (example with cars)

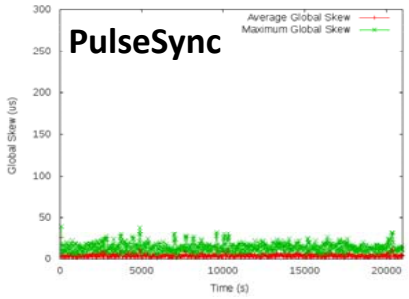
- more open problems in SOFSEM paper

Summary


$\Theta\left(\frac{W}{\sqrt{n}\log n}\right)$
$P \neq NP$


Everybody's expectation, five years ago („solved")
Lower bound of log*D* / loglog*D* [Fan & Lynch, PODC 2004]
Blocking algorithm
All natural algorithms [Locher et al., DISC 2006]
Dynamic Networks! [Kuhn et al., SPAA 2009]
Kappa algorithm [Lenzen et al., FOCS 2008]
Tight lower bound [Lenzen et al., PODC 2009]


FTSP
Average Global Skew
Maximum Global Skew
Global Skew (us)
Time (s)


PulseSync
Average Global Skew
Maximum Global Skew
Global Skew (us)
Time (s)

Thank You!

Questions & Comments?

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

YouTube Clock Synchronization