

Algorithms for Sensor Networks

GRAAL/AEOLUS School on Hot Topics in Network Algorithms



Algorithms for Sensor Networks ...what is it good for?!

Let's take a practical viewpoint...

- Algorithms for Sensor Networks, What is it Good For?
- Absolutely nothing!? The merit of theory and algorithms in the context of wireless sensor and ad hoc networks is often questioned. Admittedly, coming up with theory success stories that will be accepted by practitioners is not easy. In my talk I will discuss the current score of the Theory vs. Practice game after playing seven years for the Theory team. (Probably due to a "seven year itch" I recently also started playing for the Practice team...)



Absolutely nothing?!?

Hypothesis: Impact(Theory) $\rightarrow \epsilon$



Scoring for Theory

- “Theory is important, even if it sometimes does not have impact”
 - sometimes decades later, e.g., **number theory** for cryptography
- Packet switching (very important for sensor networks) was promoted by theory guys in the early 60s:
 - **Paul Baran, Donald Davies, Leonard Kleinrock**, et al.
 - Later followed by Lawrence Roberts, Robert Kahn, Vinton Cerf, et al.

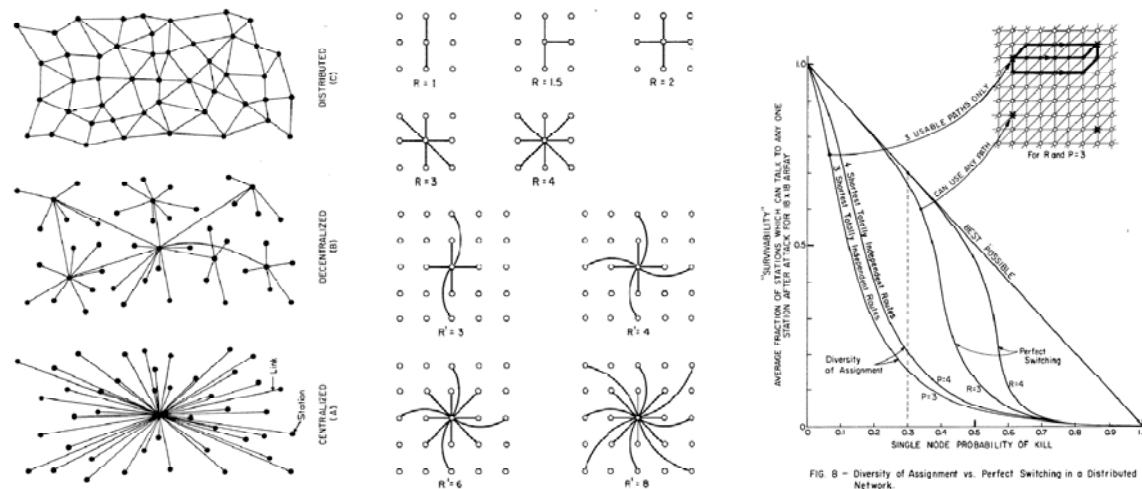


FIG 8 - Diversity of Assignment vs. Perfect Switching in a Distributed Network.



Scoring for Systems

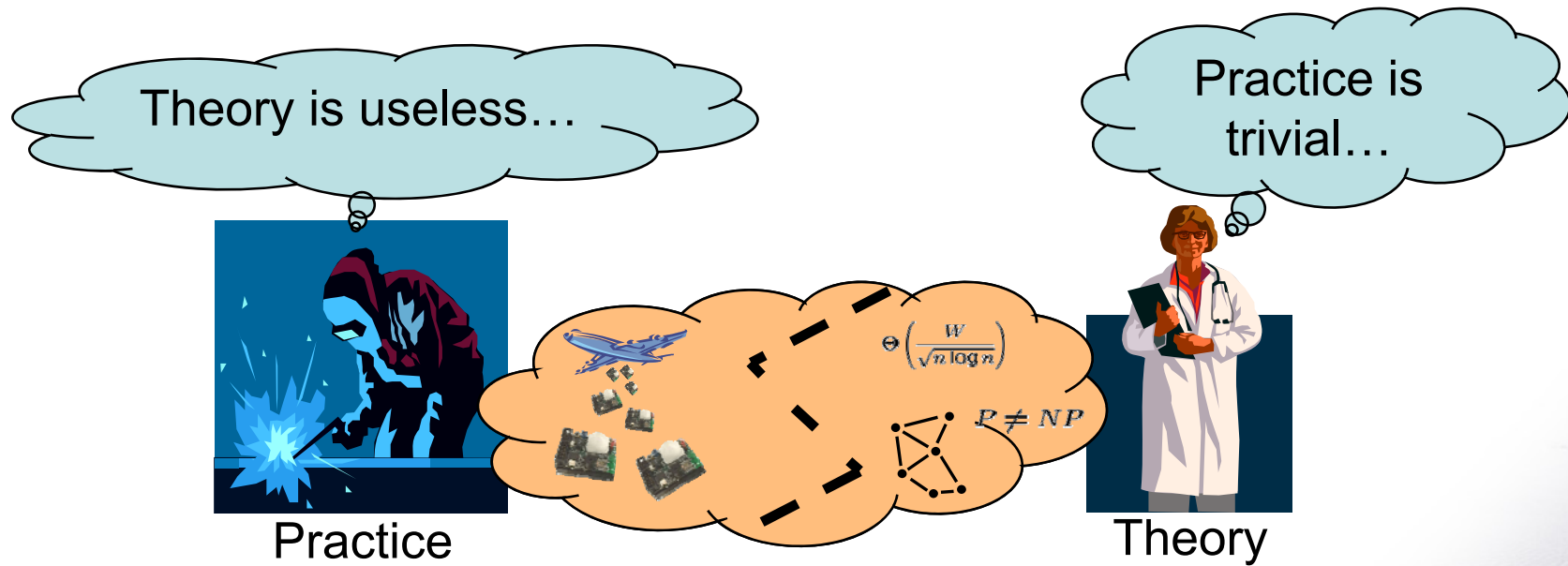
- Baran et al. was almost 50 years ago
- Systems people get it “right” quite often...
- Many important difficult problems are “not really theoretical”...



Impact(Recent Theory) → ε!

Why? (More theory whining)

- Why does theory not have impact on practical systems?



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 rarely comes from the top 5 US Universities
 - having hundreds of workshops does not help,
is just a good excuse for not following up research

- ... do I sound embittered?!? :-)



Why recent theory does not have impact on real systems...

1) Systems people don't read theory papers

2) Theory people don't build systems

Maybe theory people should build systems themselves?!?

3) Ergo, theory does not have practical impact



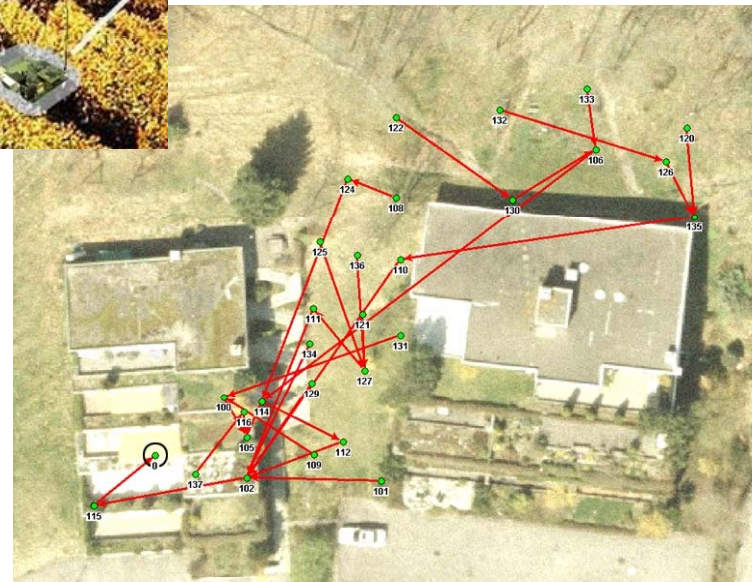
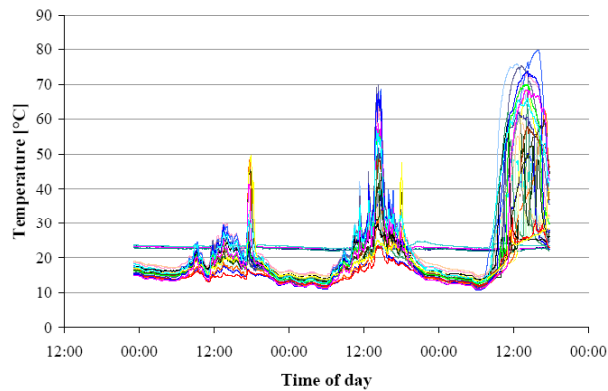
Systems Perspective: Dozer

Example: Dozer

[Burri, von Rickenbach, W, IPSN 2007]

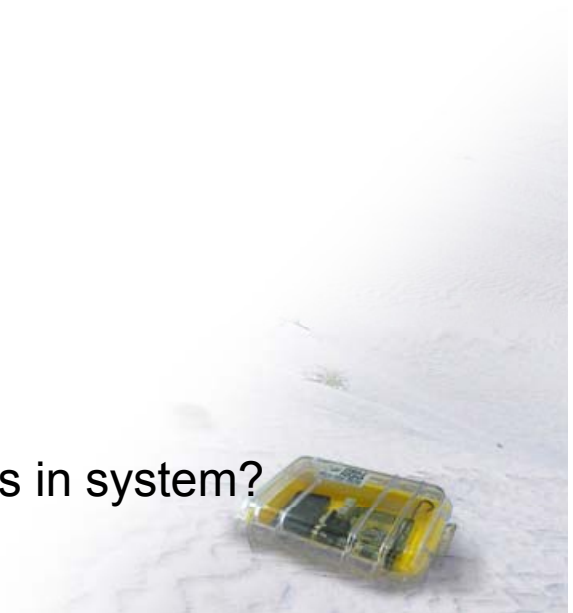


- Up to 10 years of network life-time
- Mean energy consumption: 0.066 mW
- Operational network in use > 1 year
- High availability, reliability (99.999%)



Is Dozer a theory-meets-systems 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 more than 1 year already!
- **Bad** news
 - Dozer does not have an awful lot of theory inside
- **Ugly** news
 - Dozer v2 has even less theory than Dozer v1
- **Hope**
 - Despite not being aware still subliminal theory ideas in system?



Environmental Monitoring



- Continuous data gathering
- Unattended operation
- Low data rates
- Battery powered
- ~~Network latency~~
- ~~Dynamic bandwidth demands~~

Energy conservation is crucial to prolong network lifetime



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 $\sim 1\%$



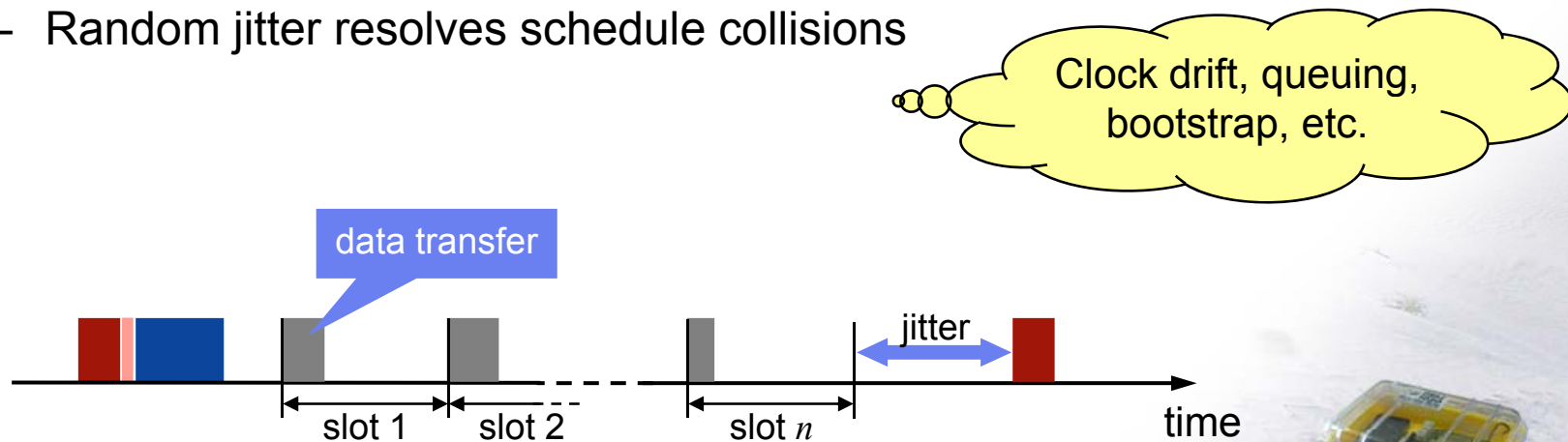
Dozer System

- 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 System

- Lightweight backchannel
 - Beacon messages comprise commands
- Bootstrap
 - Scan for a full interval
 - Suspend mode during network downtime
- Potential parents
 - Avoid costly bootstrap mode on link failure
 - Periodic refresh the list

periodic channel
activity check



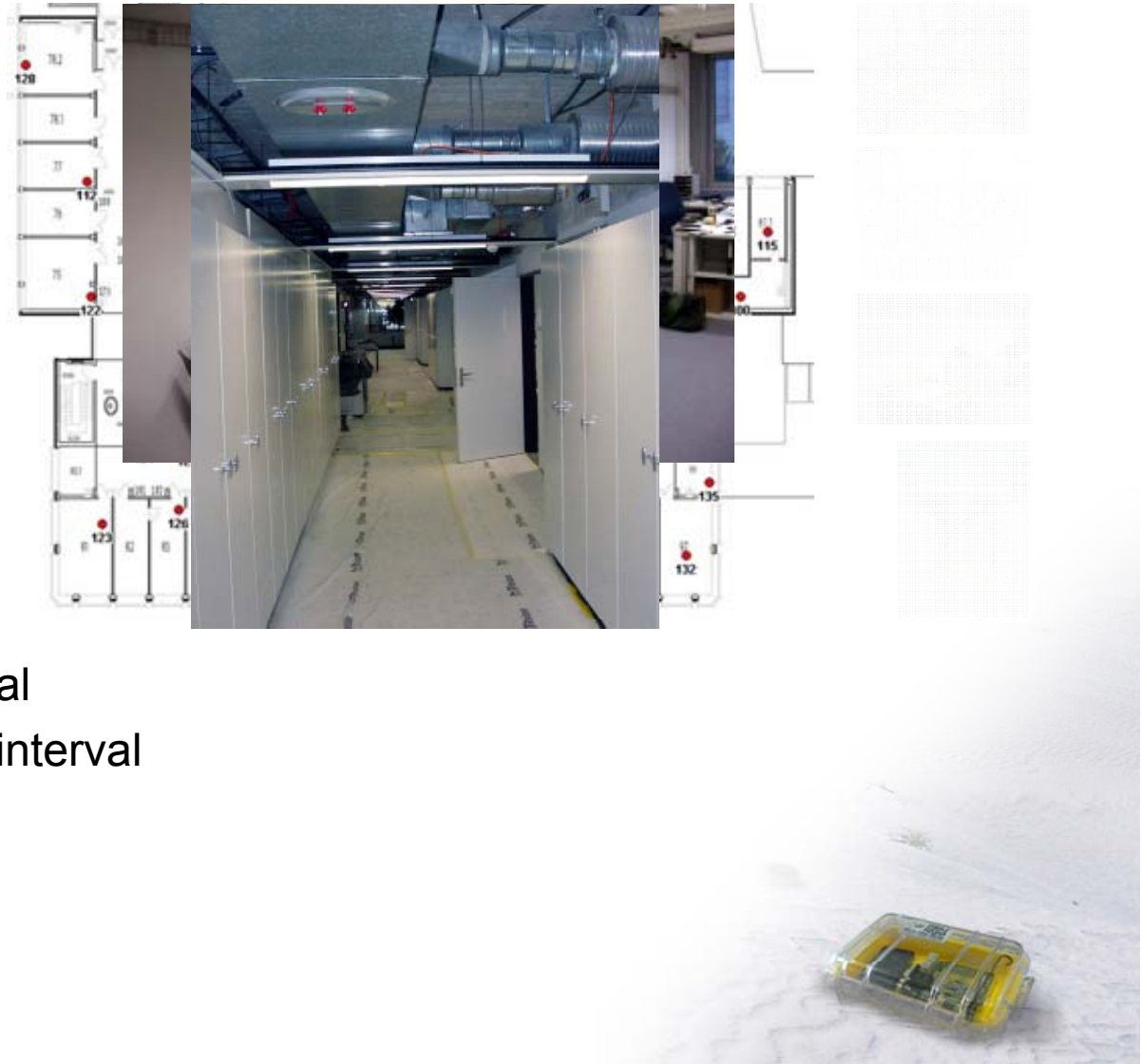
Dozer System

- Clock drift compensation
 - fixed guard times
- Application scheduling
 - TinyOS is single threaded and non-preemptive
 - TDMA is highly time critical
- Queuing strategy
 - Fixed size buffers

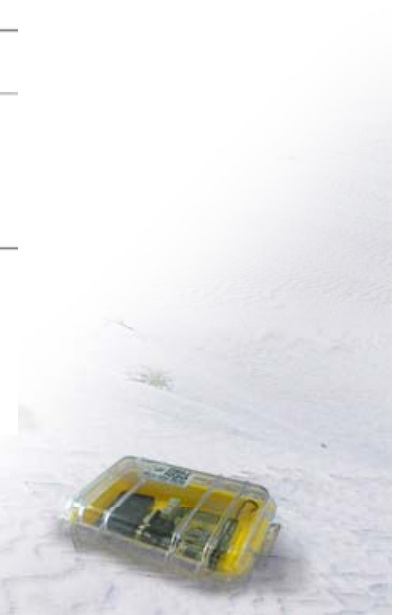
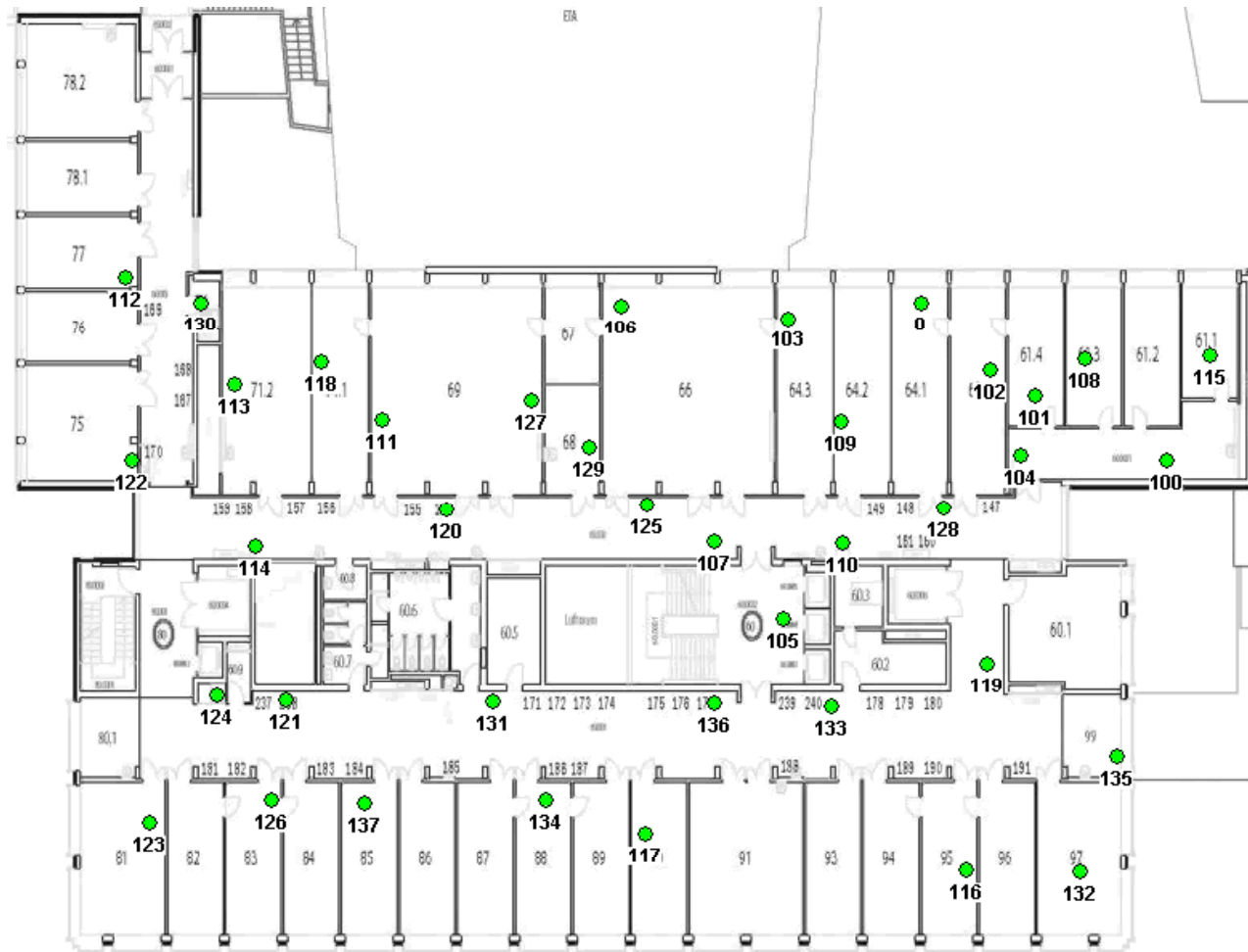


Evaluation

- Platform
 - TinyNode
 - MSP 430
 - Semtech XE1205
 - TinyOS 1.x
- Testbed
 - 40 Nodes
 - Indoor deployment
 - > 1 month uptime
 - 30 sec beacon interval
 - 2 min data sampling interval

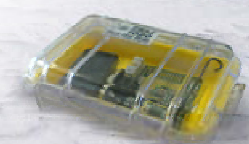
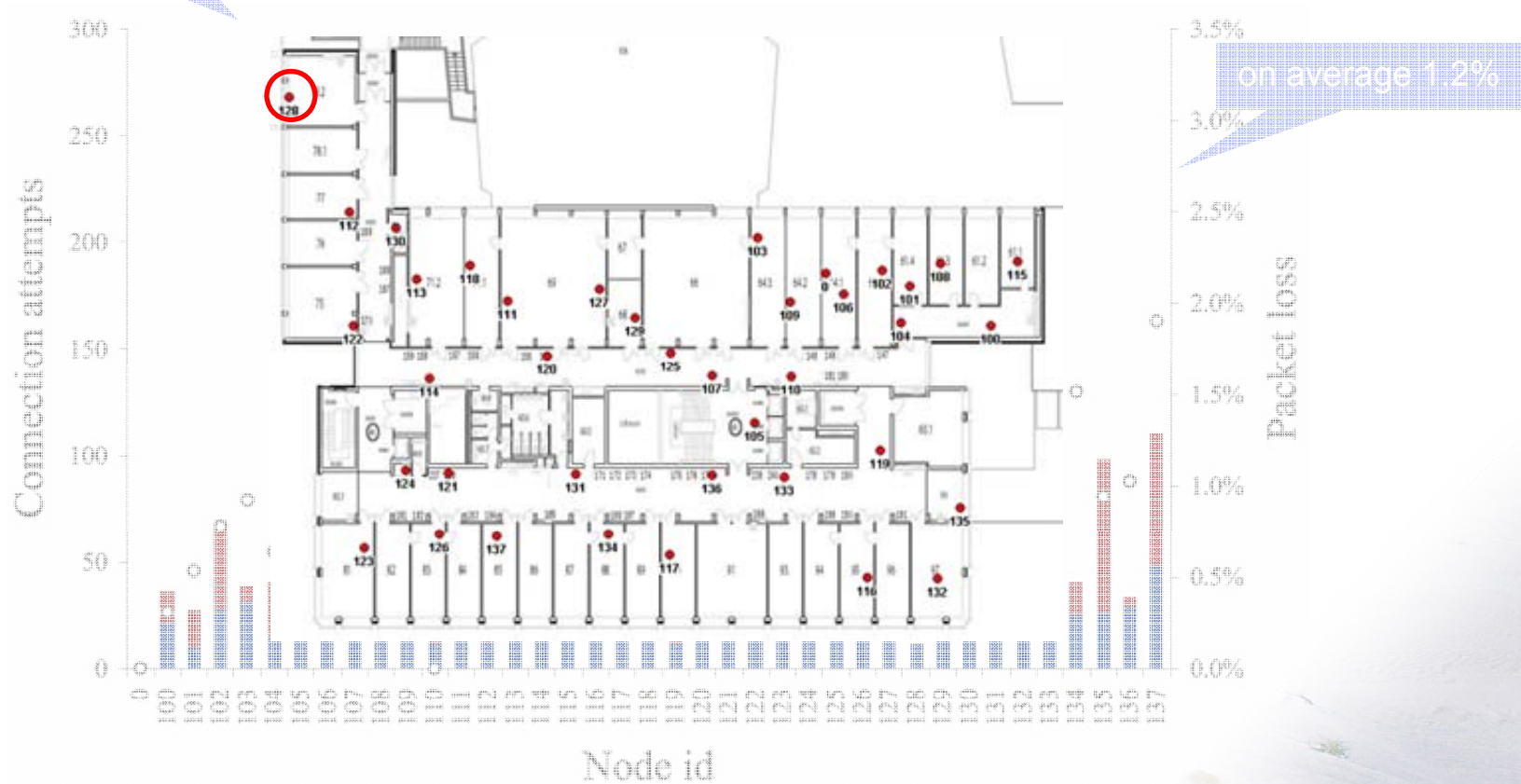


Dozer in Action



Tree Maintenance

1 week of operation



Energy Consumption

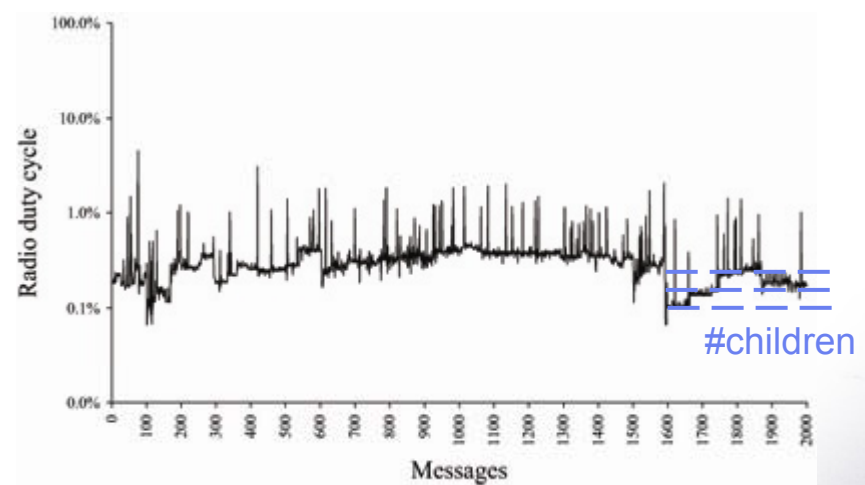
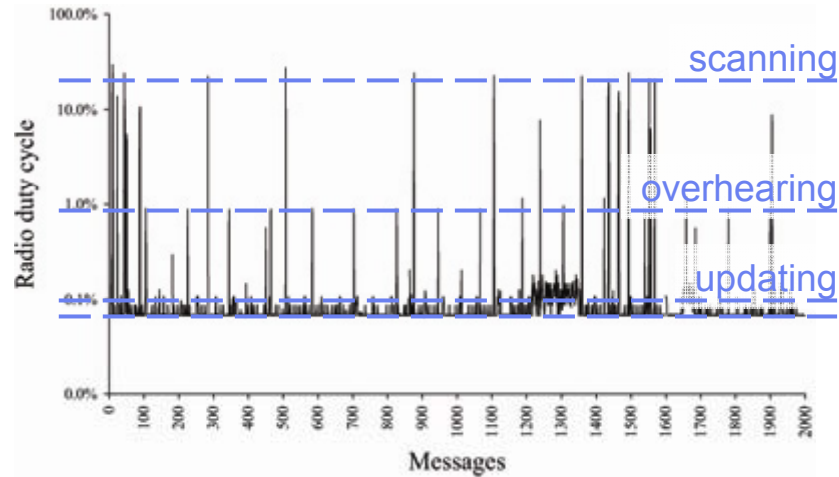
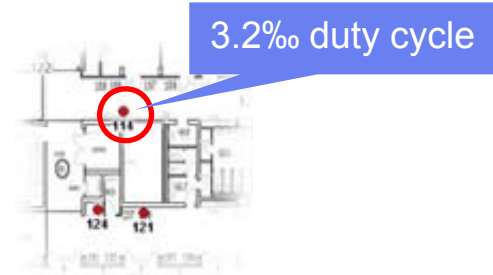
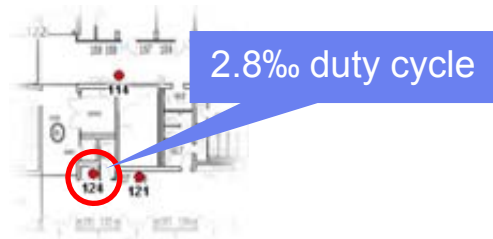
on average 1.67‰



➡ Mean energy consumption of 0.082 mW



Energy Consumption



- Leaf node
- Few neighbors
- Short disruptions

- Relay node
- No scanning



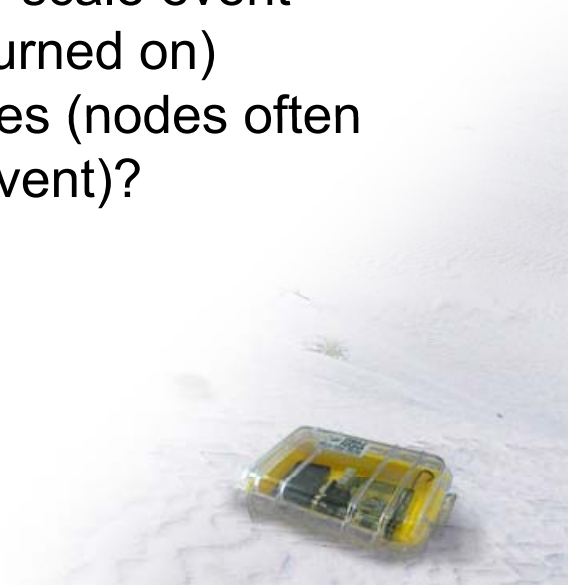
Dozer Conclusions & Possible Future Work

- Conclusions
 - Dozer achieves duty cycles in the magnitude of 1‰.
 - Abandoning collision avoidance was the right thing to do.
- Possible Future work
 - Incorporate clock drift compensation.
 - Optimize delivery latency of sampled sensor data.
 - Make use of multiple frequencies to further reduce collisions.



Open problem

- Continuous data gathering is somewhat well understood, both practically and theoretically, in contrast to the two other paradigms, event detection and query processing.
- One possible open question is about **event detection**. Assume that you have a battery-operated sensor network, both sensing and having your radio turned on costs energy. How can you build a network that raises an alarm quickly if some large-scale event (many nodes will notice the event if sensors are turned on) happens? What if nodes often sense false positives (nodes often sense something even if there is no large-scale event)?



again, what is theory good for?!?

My own private answer

- Understanding the basic principles and limitations!
- In other words, **lower bounds and impossibility results**
- On the following slides, I showcase a few examples



Time Synchronization

Rating

- Area maturity



- Practical importance



- Theoretical importance



Overview

- Motivation
- Clock Sources
- Reference-Broadcast Synchronization (RBS)
- Time-sync Protocol for Sensor Networks (TPSN)
- Gradient Clock Synchronization



Motivation

- Time synchronization is essential for many applications
 - Coordination of wake-up and sleeping times (energy efficiency)
 - TDMA schedules
 - Ordering of collected sensor data/events
 - Co-operation of multiple sensor nodes
 - Estimation of position information (e.g. shooter detection)
- Goals of clock synchronization
 - Compensate *offset* between clocks
 - Compensate *drift* between clocks



Properties of Synchronization Algorithms

- External versus internal synchronization
 - External sync: Nodes synchronize with an external clock source (UTC)
 - Internal sync: Nodes synchronize to a common time
 - to a leader, to an averaged time, or to anything else
- Instant versus periodic synchronization
 - Periodic synchronization required to compensate clock drift
- A-priori versus a-posteriori
 - A-posteriori clock synchronization triggered by an event
- Local versus global synchronization



Clock Sources

- Radio Clock Signal:
 - Clock signal from a reference source (atomic clock) is transmitted over a longwave radio signal
 - DCF77 station near Frankfurt, Germany transmits at 77.5 kHz with a transmission range of up to 2000 km
 - Accuracy limited by the distance to the sender, Frankfurt-Zurich is about 1ms.
 - Special antenna/receiver hardware required

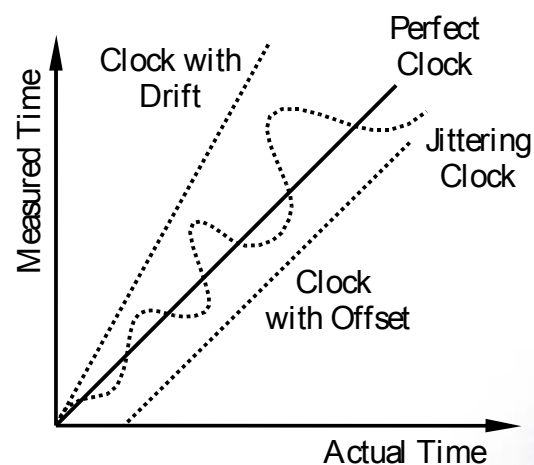
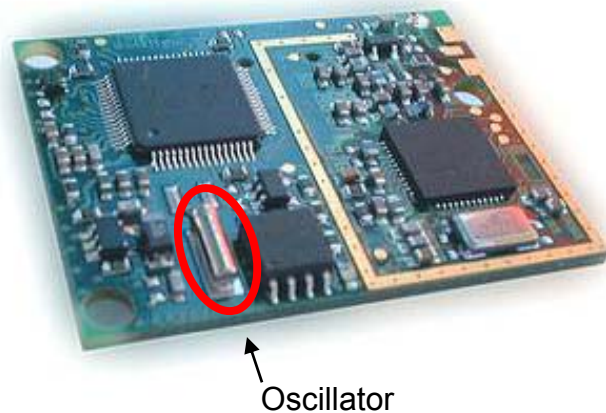


- Global Positioning System (GPS):
 - Satellites continuously transmit own position and time code
 - Line of sight between satellite and receiver required
 - Special antenna/receiver hardware required



Clock Devices in Sensor Nodes

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

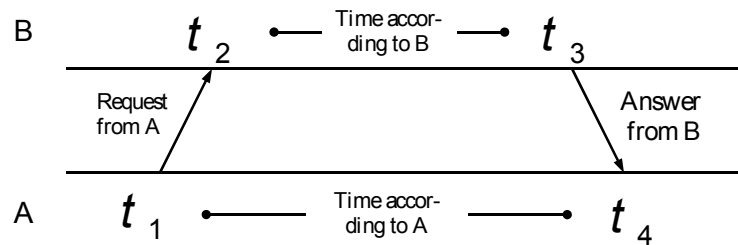


This is a drift of up to 50 μ s per second or 0.18s per hour



Sender/Receiver Synchronization

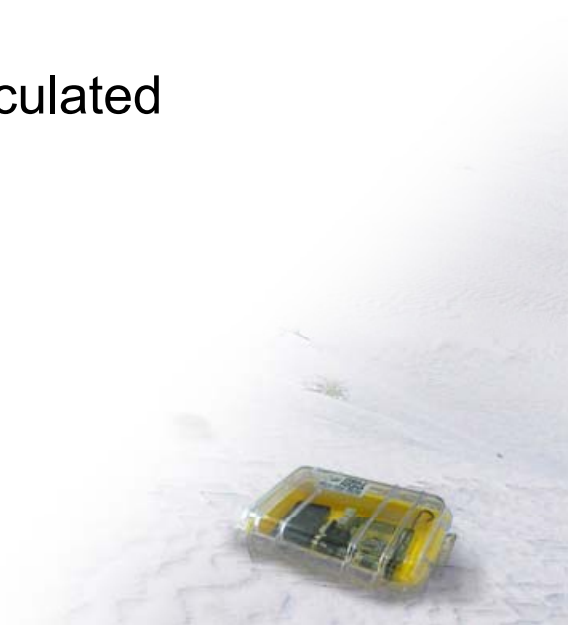
- Round-Trip Time (RTT) based synchronization



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

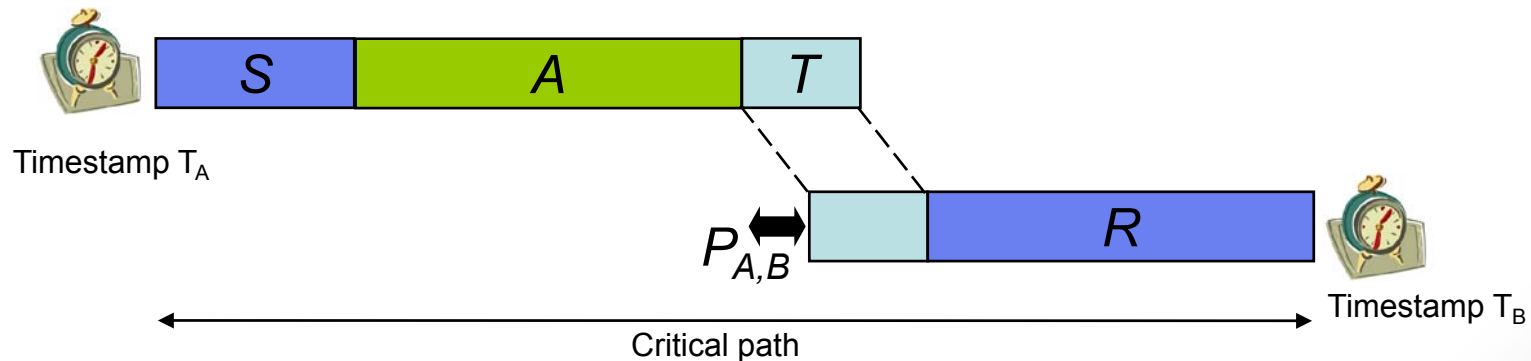
$$\delta = \frac{(t_4 - t_1) - (t_3 - t_2)}{2}$$

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



Disturbing Influences on Packet Latency

- Influences
 - Sending Time S (up to 100ms)
 - Medium Access Time A (up to 500ms)
 - Transmission Time T (tens of milliseconds, depending on size)
 - Propagation Time $P_{A,B}$ (microseconds, depending on distance)
 - Reception Time R (up to 100ms)



- Asymmetric packet delays due to *non-determinism*
- Solution: timestamp packets at MAC Layer



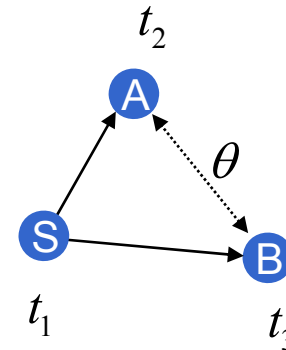
Reference-Broadcast Synchronization (RBS)

- A sender synchronizes a set of receivers with one another
- Point of reference: beacon's arrival time

$$t_2 = t_1 + S_S + A_S + P_{S,A} + R_A$$

$$t_3 = t_1 + S_S + A_S + P_{S,B} + R_B$$

$$\theta = t_2 - t_3 = (P_{S,A} - P_{S,B}) + (R_A - R_B)$$

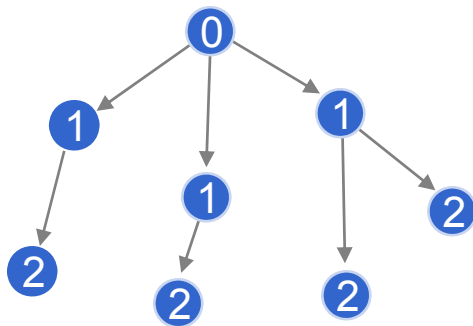


- Only sensitive to the **difference** in propagation and reception time
- Time stamping at the interrupt time when a beacon is received
- After a beacon is sent, all receivers exchange their reception times to calculate their clock offset
- **Post-synchronization** possible
- Least-square linear regression to tackle clock drifts



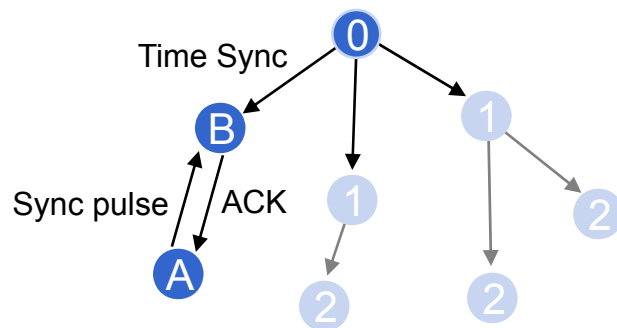
Time-sync Protocol for Sensor Networks (TPSN)

- Traditional sender-receiver synchronization (RTT-based)
- *Initialization phase: Breadth-first-search flooding*
 - Root node at level 0 sends out a *level discovery* packet
 - Receiving nodes which have not yet an assigned level set their **level** to +1 and start a random timer
 - After the timer is expired, a new level discovery packet will be sent
 - When a new node is deployed, it sends out a *level request* packet after a random timeout



Time-sync Protocol for Sensor Networks (TPSN)

- *Synchronization phase*
 - Root node issues a *time sync* packet which triggers a random timer at all level 1 nodes
 - After the timer is expired, the node asks its parent for synchronization using a *synchronization pulse*
 - The parent node answers with an *acknowledgement*
 - Thus, the requesting node knows the round trip time and can calculate its clock offset
 - Child nodes receiving a synchronization pulse also start a random timer themselves to trigger their own synchronization

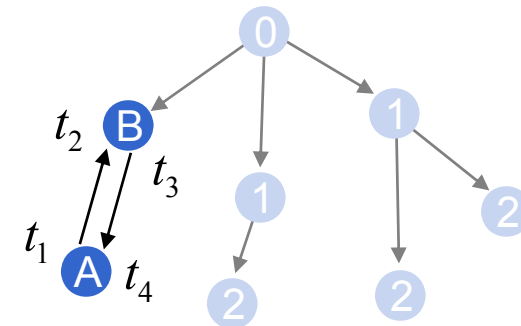


Time-sync Protocol for Sensor Networks (TPSN)

$$t_2 = t_1 + S_A + A_A + P_{A,B} + R_B$$

$$t_4 = t_3 + S_B + A_B + P_{B,A} + R_A$$

$$\theta = \frac{(S_A - S_B) + (A_A - A_B) + (P_{A,B} - P_{B,A}) + (R_B - R_A)}{2}$$

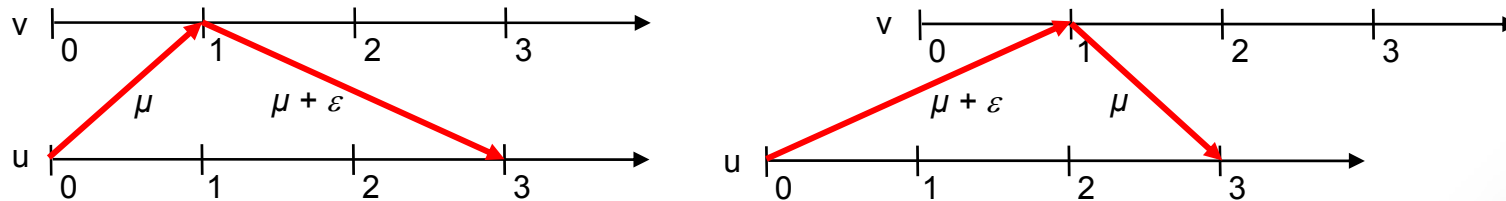


- Time stamping packets at the MAC layer
- In contrast to RBS, the signal propagation time might be negligible
- Authors claim that it is “about two times” better than RBS
- Again, clock drifts are taken into account using periodical synchronization messages
- Problem: What happens in a non-tree topology (e.g. **ring**)?!?
 - Two neighbors may have exceptionally bad synchronization



Theoretical Bounds for Clock Synchronization

- Network Model:
 - Each node i has a local clock $L_i(t)$
 - Network with n nodes, diameter D .
 - Reliable point-to-point communication with minimal delay μ
 - Jitter ε is the uncertainty in message delay
- Two neighboring nodes u, v cannot distinguish whether message is faster from u to v and slower from v to u , or vice versa. Hence clocks of neighboring nodes can be up to ε off.



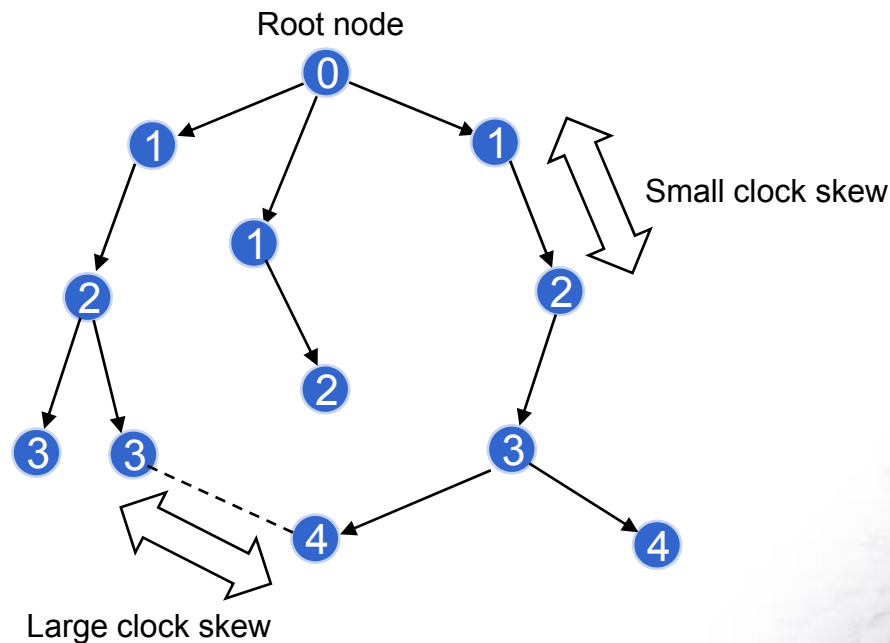
- Hence, two nodes at distance D may have clocks which are εD off.
- This can be achieved by a simple **flooding** algorithm: Whenever a node receives a new minimum value, it sets its clock to the new value and forwards its new clock value to all its neighbors.






Clock Synchronization

1. **Global** property: Minimize clock skew between any two nodes
2. **Local** (“gradient”) property: Small clock skew between two nodes if the distance between the nodes is small.
3. Clock should **not** be allowed to **jump backwards**
 - You don’t want new events to be registered earlier than older events.

• Example:



Trivial Solution: Let $t = 0$ at all nodes and times

1. Global property: Minimize clock skew between any two nodes 
2. Local (gradient) property: Small clock skew between two nodes if the distance between the nodes is small. 
3. Clock should not be allowed to jump backwards 

- To prevent trivial solution, we need a fourth constraint:

4. Clock should always to move forward.

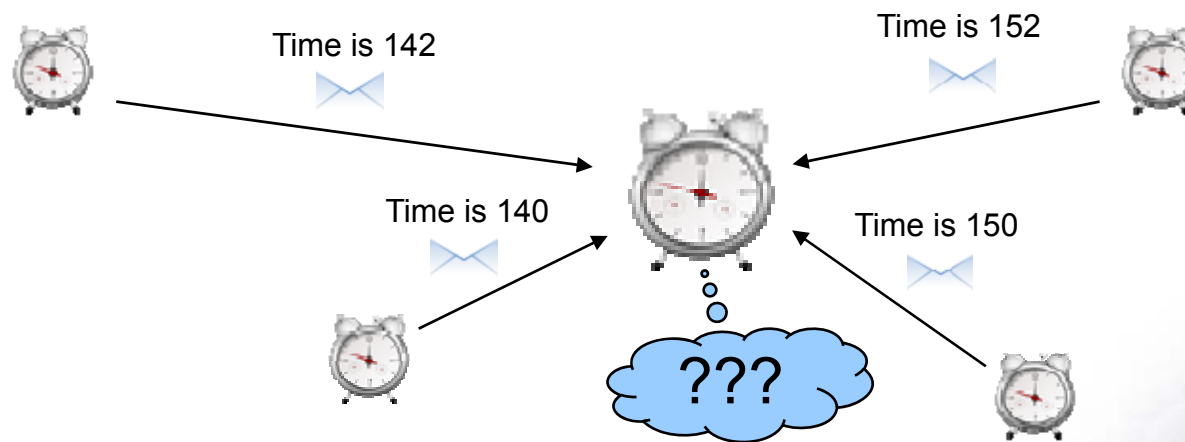
- Sometimes faster, sometimes slower is OK.
- But there should be a minimum and a maximum speed.



Gradient Clock Synchronization

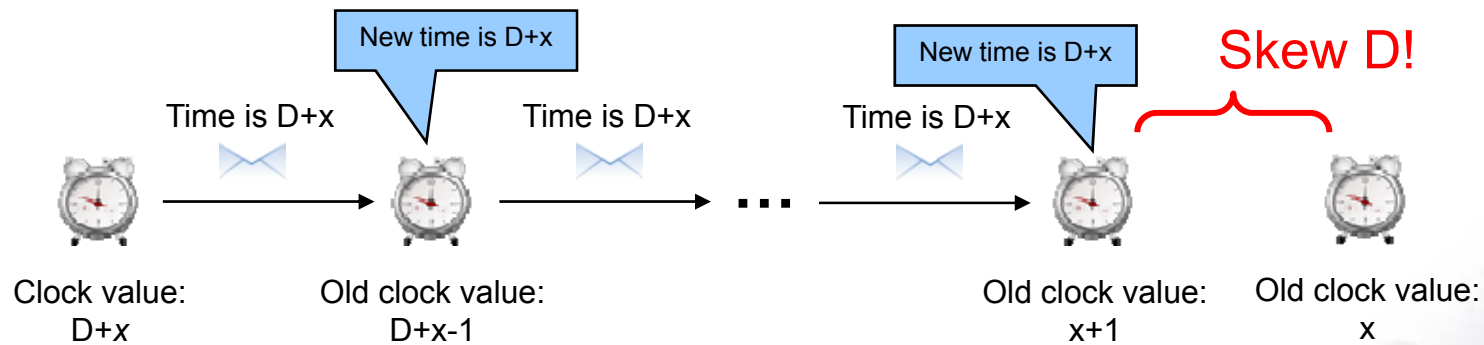
- Model

- Each node has a hardware clock $H_i(\cdot)$ with a clock rate $h_i(t) \in [L, U]$ where $0 < L < U$ and $U \geq 1$
- The time of node i at time t is $H_i(t) = \int_0^t h_i(t) dt$
- Each node has a logical clock $L_i(\cdot)$ which increases at the rate of $H_i(\cdot)$
- Employ a synchronization algorithm A to update the local clock with fresh clock values from neighboring nodes (clock cannot run backwards)
- Nodes inform their neighboring nodes when local clock is updated



Synchronization Algorithms: A^{\max}

- Question: How to update the local 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 greater than local clock value)
- Poor gradient algorithm: Fast propagation of the largest clock value could lead to a large skew between two neighboring nodes



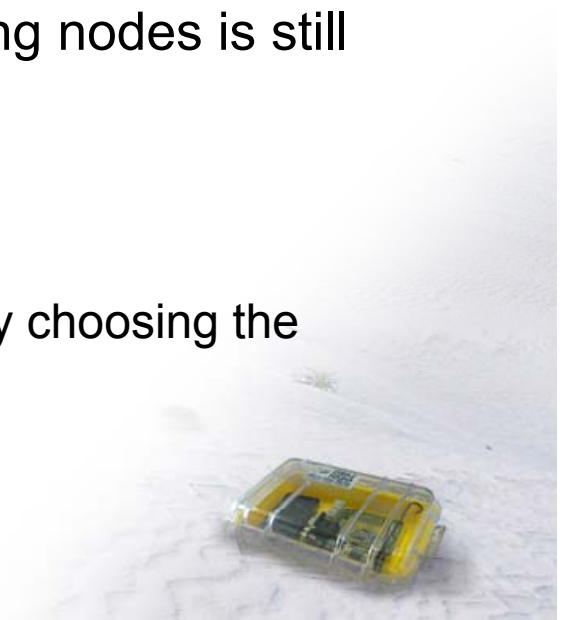
Synchronization Algorithms: A^{\max} '

- The problem of A^{\max} is that the clock is always increased to the maximum value
- Idea: Allow a constant slack γ between the maximum neighbor clock value and the own clock value
- The algorithm A^{\max} ' sets the local clock value $L_i(t)$ to

$$L_i(t) := \max(L_i(t), \max_{j \in N_i} L_j(t) - \gamma)$$

→ Worst-case clock skew between two neighboring nodes is still $\Theta(D)$ independent of the choice of γ !

- How can we do better?
 - Idea: Take the clock of all neighbors into account by choosing the **average** value

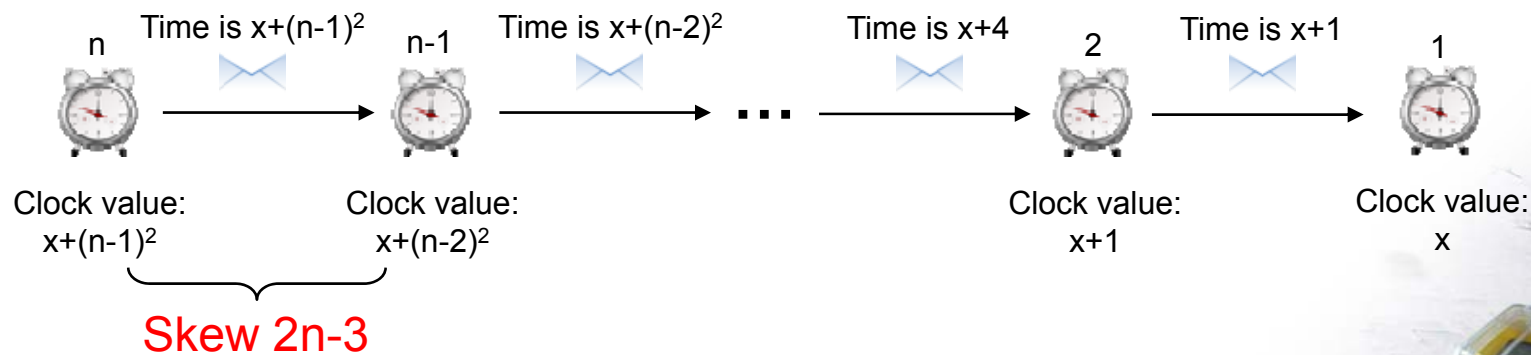


Synchronization Algorithms: A^{avg}

- A^{avg} sets the local clock to the average value of all neighbors:

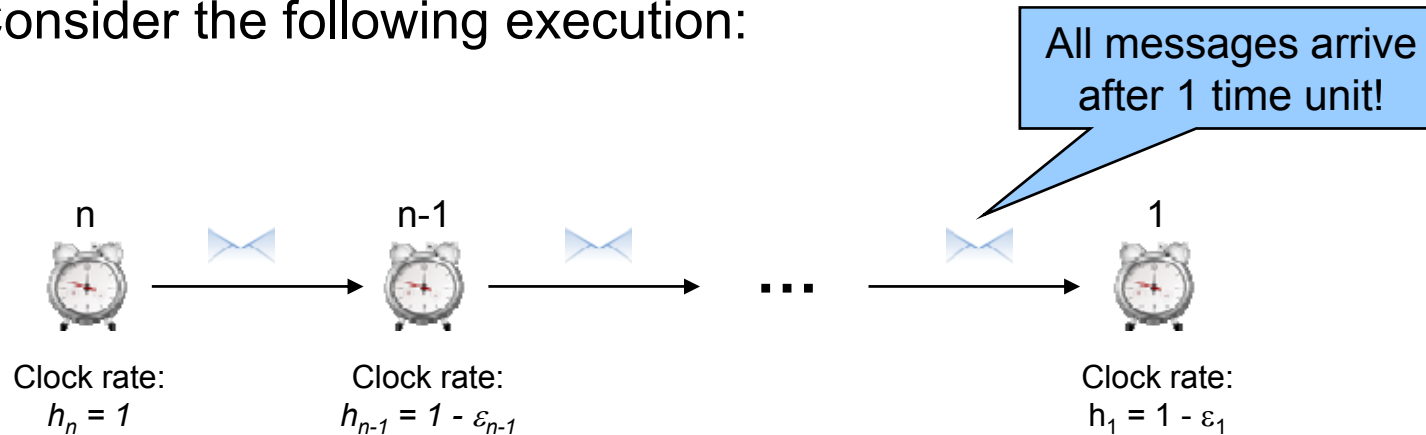
$$L_i(t) := \max(L_i(t), \frac{1}{|N_i|} \sum_{j \in N_i} L_j(t))$$

- Surprisingly, this algorithm is even worse!
- We will now prove that in a very natural execution of this algorithm, the clock skew becomes large!



Synchronization Algorithms: A^{avg}

Consider the following execution:



All ϵ_i for $i \in \{1, \dots, n-1\}$ are arbitrary values in the range $(0, 1)$

→ The clock rates can be viewed as *relative* rates compared to the fastest node n !

Theorem: In the given execution, the largest skew between neighbors is $2n-3 \in O(D)$.



Synchronization Algorithms: A^{avg}

We first prove two lemmas:

Lemma 1: In this execution it holds that $\forall t \forall i \in \{2, \dots, n\}$:
 $L_i(t) - L_{i-1}(t) \leq 2i - 3$, independent of the choices of $\varepsilon_i > 0$.

Proof:

Define $\Delta L_i(t) := L_i(t) - L_i(t-1)$. It holds that $\forall t \forall i: \Delta L_i(t) \leq 1$.

$L_1(t) = L_2(t-1)$ as node 1 has only one neighbor (node 2).

Since $\Delta L_2(t) \leq 1$ for all t , we know that $L_2(t) - L_1(t) \leq 1$ for all t .

Assume now that it holds for $\forall t \forall j \leq i: L_j(t) - L_{j-1}(t) \leq 2j - 3$.

We prove a bound on the skew between node i and $i+1$:

For $t = 0$ it is trivially true that $L_{i+1}(t) - L_i(t) \leq 2(i+1) - 3$.



Synchronization Algorithms: A^{avg}

Assume that it holds for all $t' \leq t$. For $t+1$ we have that

$$\begin{aligned} L_i(t + 1) &\geq \frac{L_{i+1}(t) + L_{i-1}(t)}{2} \\ &\geq \frac{L_{i+1}(t) + L_i(t) - (2i - 3)}{2} \\ &\geq \frac{L_{i+1}(t) + L_i(t + 1) - 1 - (2i - 3)}{2} \\ &\geq L_{i+1}(t + 1) - (2(i + 1) - 3). \end{aligned}$$

- The first inequality holds because the logical clock value is always at least the average value of its neighbors.
- The second inequality follows by induction.
- The third and fourth inequalities hold because $\Delta L_i(t) \leq 1$. □



Synchronization Algorithms: A^{avg}

Lemma 2: $\forall i \in \{1, \dots, n\}: \lim_{t \rightarrow \infty} \Delta L_i(t) = 1.$

Proof:

Assume $\Delta L_{n-1}(t)$ does not converge to 1.

Case (1):

$\exists \varepsilon > 0$ such that $\forall t: \Delta L_{n-1}(t) \leq 1 - \varepsilon.$

As $\Delta L_n(t)$ is always 1, if there is such an ε , then

$\lim_{t \rightarrow \infty} L_n(t) - L_{n-1}(t) = \infty$, a contradiction to Lemma 1.

Case (2):

$\Delta L_{n-1}(t) = 1$ only for some t , then there is an unbounded number of times t' where $\Delta L_{n-1}(t) < 1$, which also implies that

$\lim_{t \rightarrow \infty} L_n(t) - L_{n-1}(t) = \infty$, again contradicting Lemma 1.

Hence, $\lim_{t \rightarrow \infty} \Delta L_{n-1}(t) = 1$. Applying the same argument to the other nodes, it follows inductively that $\forall i \in \{1, \dots, n\}: \lim_{t \rightarrow \infty} \Delta L_i(t) = 1. \quad \square$



Synchronization Algorithms: A^{avg}

Theorem: In the given execution, the largest skew between neighbors is $2n-3$.

Proof:

We show that $\forall i \in \{2, \dots, n\}: \lim_{t \rightarrow \infty} L_i(t) - L_{i-1}(t) = 2i - 3$.

Since $L_1(t) = L_2(t-1)$, it holds that $\lim_{t \rightarrow \infty} L_2(t) - L_1(t) = \Delta L_1(t) = 1$, according to Lemma 2.

Assume that $\forall j \leq i: \lim_{t \rightarrow \infty} L_j(t) - L_{j-1}(t) = 2j - 3$.

According to Lemma 1 & 2, $\lim_{t \rightarrow \infty} L_{i+1}(t) - L_i(t) = Q$ for a value $Q \leq 2(i+1) - 3$. If (for the sake of contradiction) $Q < 2(i+1) - 3$, then

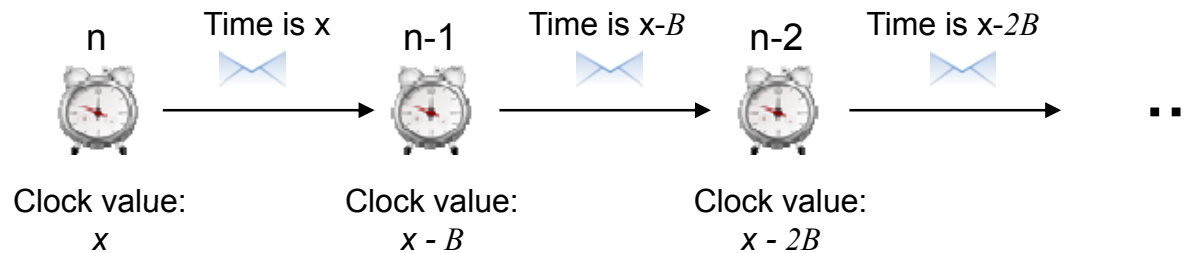
$$\begin{aligned} \lim_{t \rightarrow \infty} L_i(t) &= \lim_{t \rightarrow \infty} \frac{L_{i-1}(t-1) + L_{i+1}(t-1)}{2} \\ &= \lim_{t \rightarrow \infty} \frac{2L_i(t-1) - (2i-3) + Q}{2} \end{aligned}$$

and thus $\lim_{t \rightarrow \infty} \Delta L_i(t) < 1$, a contradiction to Lemma 2. \square



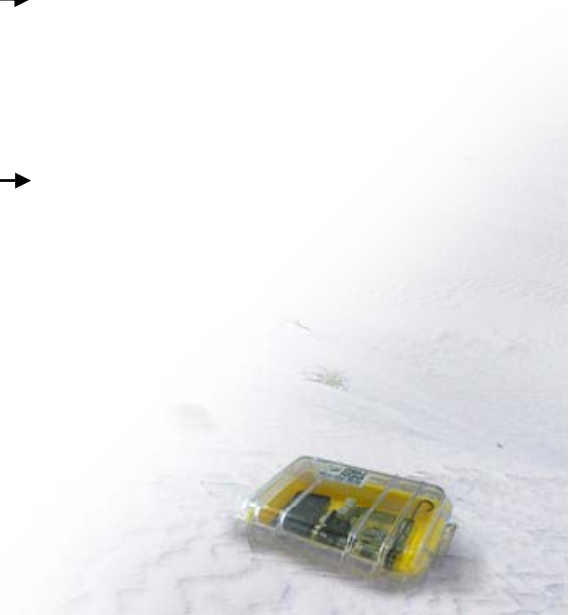
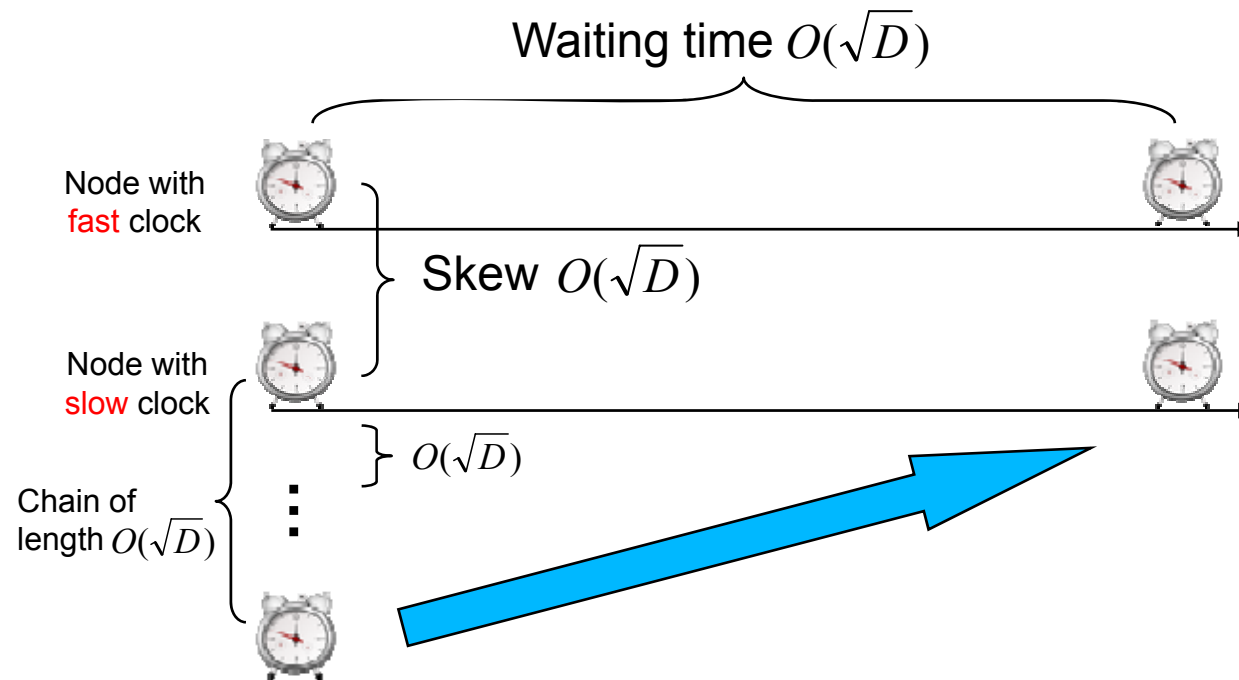
Synchronization Algorithms: A^{bound}

- Idea: Minimize the skew to the **slowest** neighbor
 - Update the local clock to the maximum value of all neighbors as long as no neighboring node's clock is more than B behind.
- Gives the slowest node time to catch up
- Problem: Chain of dependency
 - Node $n-1$ waits for node $n-2$, node $n-2$ waits for node $n-3$, ...
 - Chain of length $\Theta(n) = \Theta(D)$ results in $\Theta(D)$ waiting time
 - **$\Theta(D)$ skew!**



Synchronization Algorithms: A^{root}

- How long should we wait for a slower node to catch up?
 - Do it smarter: Set $B = O(\sqrt{D}) \rightarrow$ skew is allowed to be $O(\sqrt{D})$
 \rightarrow waiting time is at most $O(D/B) = O(\sqrt{D})$ as well



Synchronization Algorithms: A^{root}

- When a message is received, execute the following steps:

```
max := Maximum clock value of all neighboring nodes
min := Minimum clock value of all neighboring nodes

if (max > own clock and min +  $U\sqrt{D+1}$  > own clock
    own clock := min(max, min +  $U\sqrt{D+1}$ )
    inform all neighboring nodes about new clock value
end if
```

- This algorithm guarantees that the worst-case clock skew between neighbors is bounded by $O(\sqrt{D})$.
- In [Fan and Lynch, PODC 2004] it is shown that when logical clocks need to obey **minimum/maximum speed rules**, the skew of two **neighboring** clocks can be up to $\Omega(\log D / \log \log D)$.



Open Problem

- The obvious open problem is about **gradient clock synchronization**.
- Nodes in an arbitrary graph are equipped with an unmodifiable hardware clock and a modifiable logical clock. The logical clock must make progress roughly at the rate of the hardware clock, i.e., the clock rates may differ by a small constant. Messages sent over the edges of the graph have delivery times in the range $[0, 1]$. Given a bounded, variable drift on the hardware clocks, design a message-passing algorithm that ensures that the logical clock skew of adjacent nodes is as small as possible at all times.
- Indeed, there is a huge gap between upper bound of \sqrt{D} and lower bound of $\log D / \log \log D$.



Data Gathering

Rating

- Area maturity



- Practical importance



- Theoretical importance



Distributed Aggregation

Growing interest in **distributed aggregation!**

→ Sensor networks, distributed databases...

Aggregation functions?

→ *Distributive* (max, min, sum, count)

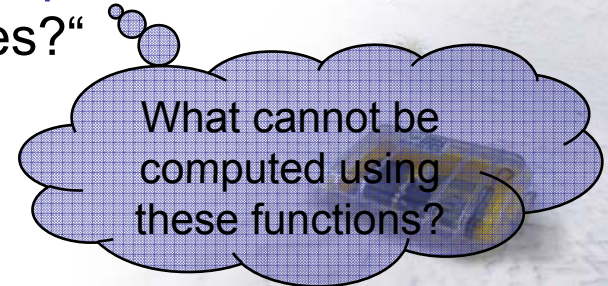
→ *Algebraic* (plus, minus, average)

→ *Holistic* (median, k^{th} smallest/largest value)



Combinations of these functions enable *complex queries!*

→ „What is the **average** of the **10% largest** values?“



Aggregation Model

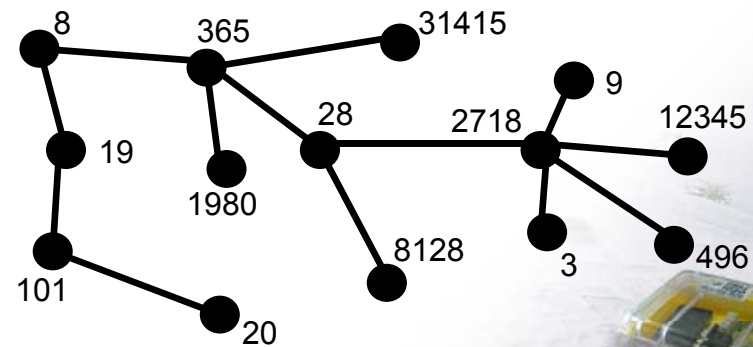
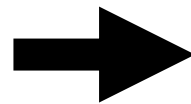
How **difficult** is it to compute these aggregation primitives?

Model:

- ❖ **Connected graph** $G = (V, E)$ of diameter D_G , $|V| = n$.
- ❖ Nodes v_i and v_j can communicate directly if $(v_i, v_j) \in E$.
- ❖ A **spanning tree** is available (diameter $D \leq 2D_G$)^o
- ❖ **Asynchronous model** of communication.
- ❖ All nodes hold a **single element**.^o
- ❖ Messages can contain only a **constant number** of elements.

Simple breadth-first construction!

Can easily be generalized to an arbitrary number of elements!



Distributive & Algebraic Functions

How **difficult** is it to compute these aggregation primitives?

→ We are interested in the **time complexity!**

Worst-case for every legal input and every execution scenario!

→ *Distributive* (sum, count...) and *algebraic* (plus, minus...) functions are **easy** to compute:

Slowest message arrives after 1 time unit!

Use a simple *flooding-echo* procedure → **convergecast!**

Time complexity: $\Theta(D)$

What about **holistic functions** (such as **k-selection**)???

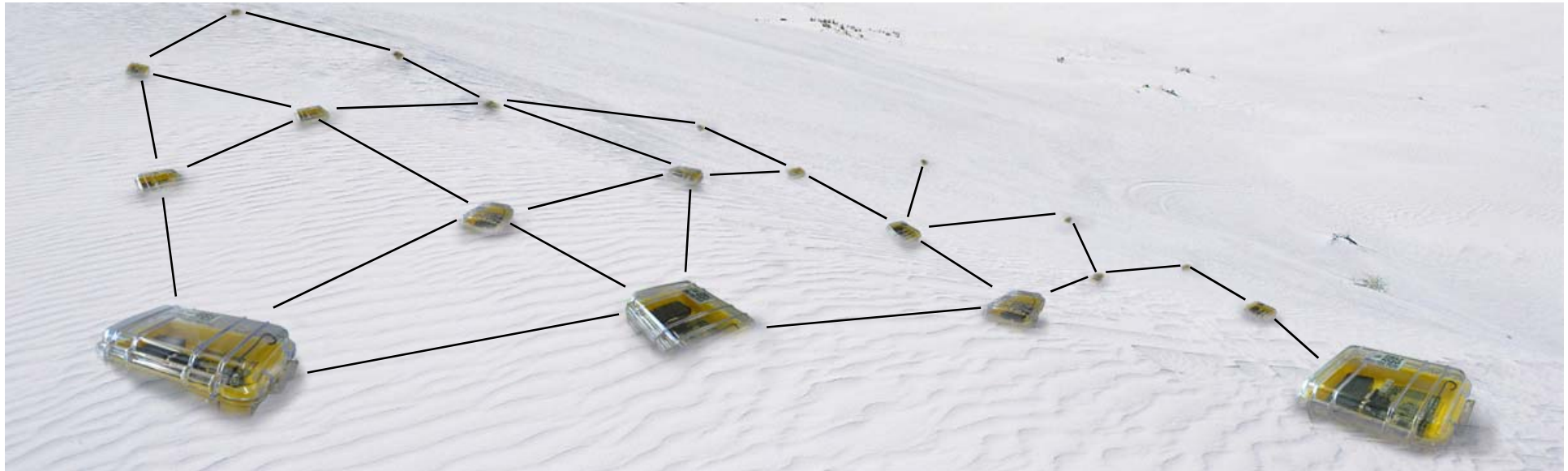
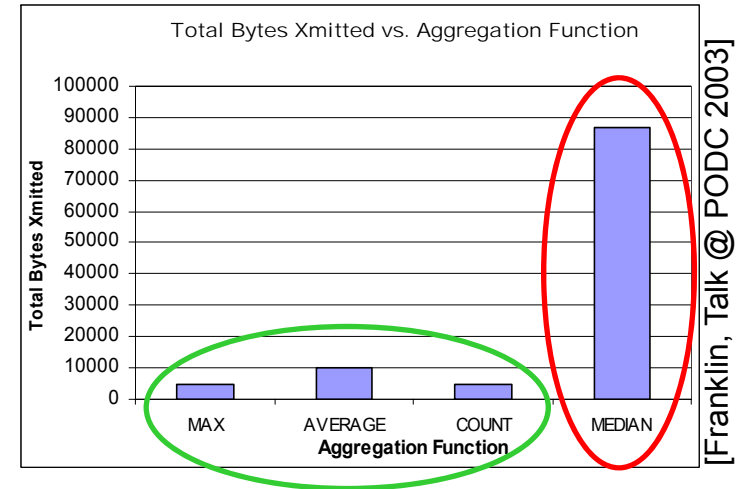
Is it (really) harder...?

Impossible to perform **in-network aggregation**?



Distributed Selection

- Database requests („SELECT ...“) consist of combinations of functions such as MAX, AVG, COUNT, k^{th} largest, etc.
- In a (sensor) network, **most functions are trivially computable** in diameter time.
- **Only selection** (median, k^{th} largest, 90% smallest values, etc.) is considered to be **impossible** (or at least difficult).

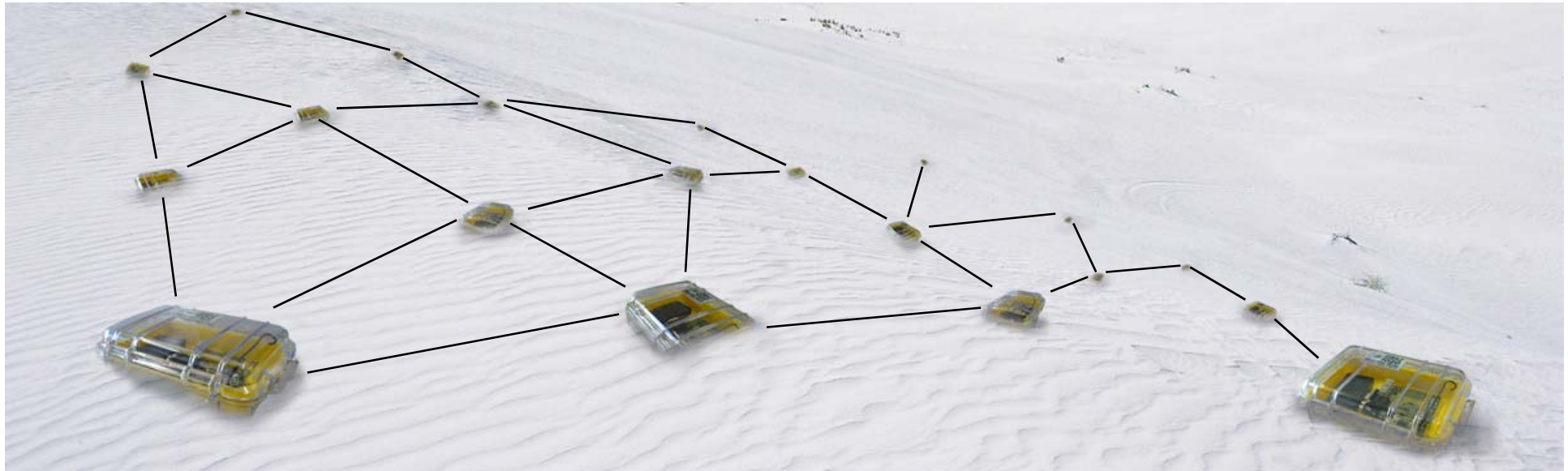


Results

- [Locher, Kuhn, W, SPAA 2007] showed that

Selection can be done in time $O(D \cdot \log_D n)$.
This is asymptotically optimal as there is a matching $\Omega(D \cdot \log_D n)$ lower bound. For deterministic algorithms: $O(D \cdot \log^2_D n)$.

$D = \text{diameter}$
 $n = \# \text{ of nodes}$

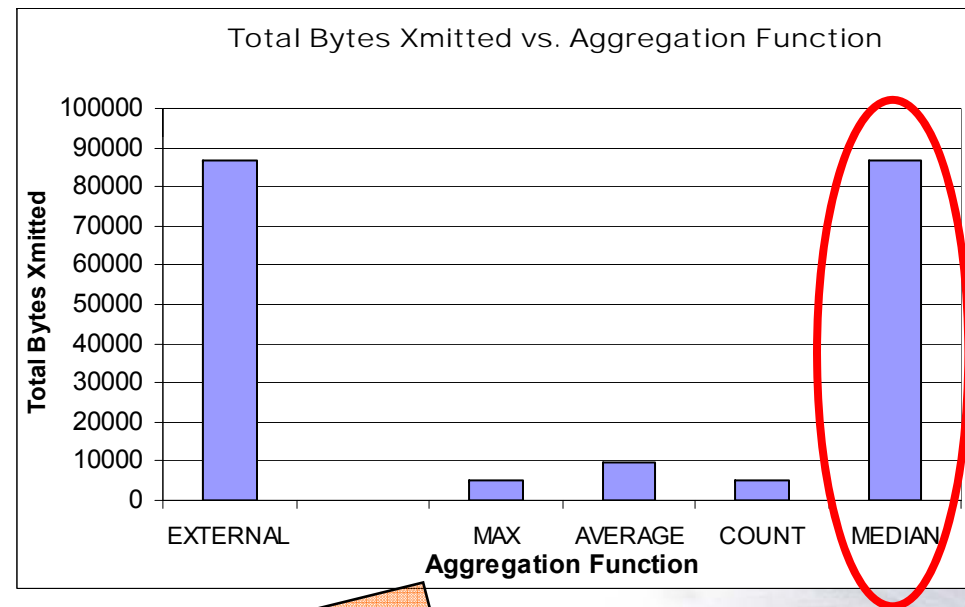


Holistic Functions

It is widely believed that *holistic* functions are **hard** to compute using in-network aggregation.

Example: TAG is an aggregation service for **ad-hoc sensor networks**
→ It is fast for other aggregates, but not for the **MEDIAN** aggregate:

„Thus, we have shown that (...) in network aggregation can reduce communication costs by an order of magnitude over centralized approaches, and that, even in the worst case (such as with **MEDIAN), it provides performance equal to the centralized approach.“**



TAG simulation: 2500 nodes in a 50x50 grid

Is it difficult?

However, there is quite a lot of literature on **distributed k-selection**:

A straightforward idea: Use the **sequential algorithm** by Blum et al. also in a distributed setting → Time Complexity: $O(D \cdot n^{0.9114})$.



A simple idea: Use **binary search** to find the k^{th} smallest value → Time Complexity: $O(D \cdot \log x_{\max})$, where x_{\max} is the maximum value.

→ Assuming that $x_{\max} \in O(n^{O(1)})$, we get $O(D \cdot \log n)$...



A better idea: Select values **randomly**, check how many values are smaller and repeat these two steps!

→ Time Complexity: $O(D \cdot \log n)$ in expectation!



Randomized Algorithm

Choosing elements **uniformly at random** is a good idea...

How is this done?

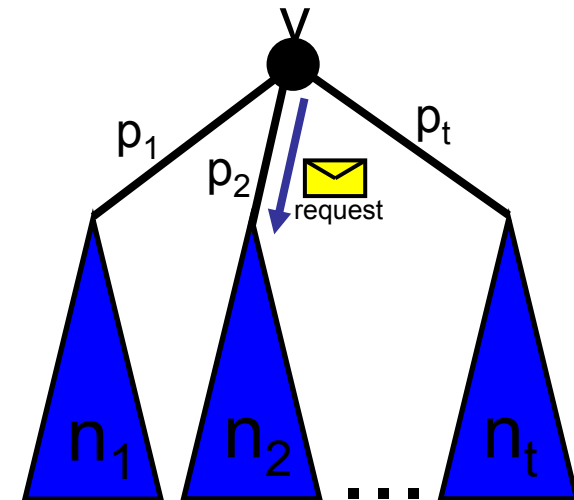
→ Assuming that all nodes know the **sizes** n_1, \dots, n_t of the **subtrees** rooted at their children v_1, \dots, v_t , the request is forwarded to node v_i with probability:

$$p_i := n_i / (1 + \sum_k n_k).$$

With probability $1 / (1 + \sum_k n_k)$ node v chooses itself.

Key observation: Choosing an element randomly **requires** $O(D)$ time!

Use **pipe-lining** to select *several random elements!*



D elements in $O(D)$ time!

Randomized Algorithm

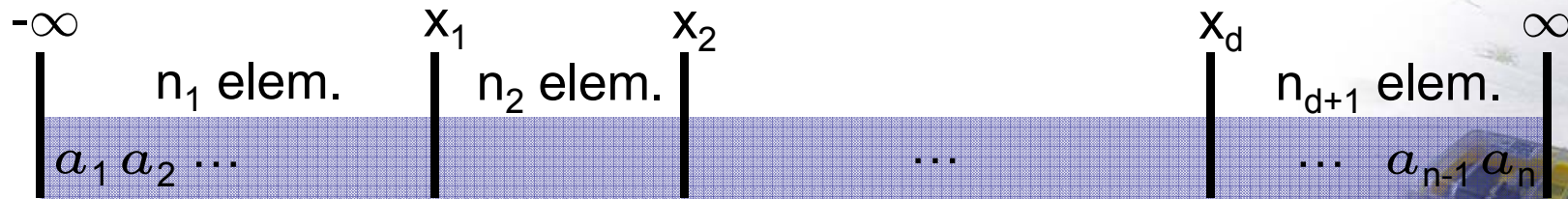
Our algorithm also operates in **phases** → The set of *candidates* **decreases** in each phase!

A *candidate* is a node whose element is possibly the solution.

A phase of the randomized algorithm:

1. Count the **number of candidates** in all subtrees
2. Pick **$O(D)$ elements** x_1, \dots, x_d uniformly at random
3. For all those elements, count the **number of smaller elements!**

Each step can be performed in **$O(D)$ time!**



Randomized Algorithm

Using these counts, the number of candidates can be reduced by a factor of D in a constant number of phases with high probability.

With probability at least $1-1/n^c$ for a constant $c \geq 1$.

We get the following result:

Theorem: The time complexity of the randomized algorithm is $O(D \cdot \log_D n)$ w.h.p.

We further proved a time lower bound of $\Omega(D \cdot \log_D n)$.

→ This simple randomized algorithm is asymptotically optimal!

More on that later...

The only remaining question: What can we do deterministically???



Deterministic Algorithm

Why is it difficult to find a good deterministic algorithm???

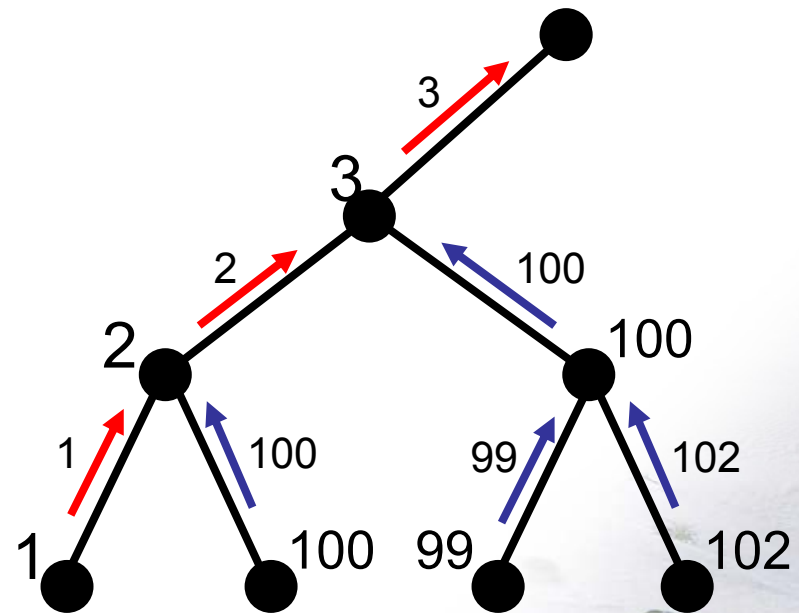
→ **Hard** to find a good **selection of elements** that **provably reduces** the set of **candidates**!

Simple idea: Always propagate the median of all received values!

Problem: In one phase, only the **h^{th} smallest element** is found if **h** is the **height of the tree**...

→ Time complexity: **$O(n / h)$**

One could do a lot better!!!
(Not shown in this course.)



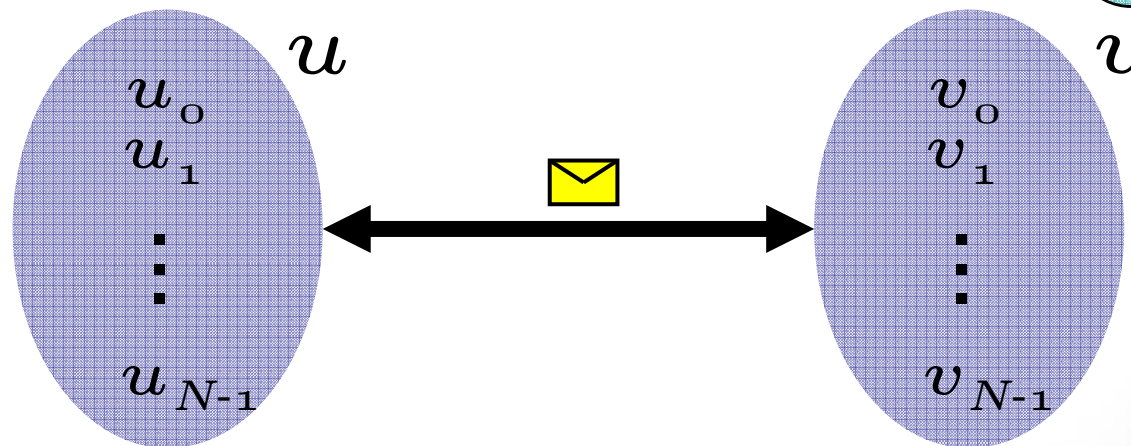
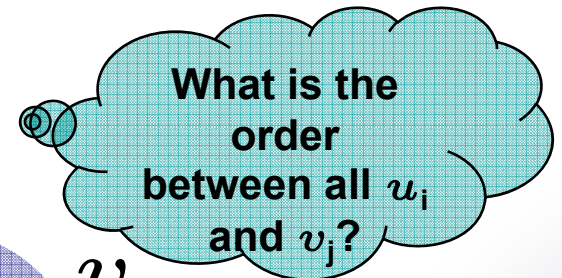
Lower Bound

The proof of the lower bound of $\Omega(D \cdot \log_D n)$ consists of **two parts**:

Part I. Find a lower bound for the case of **two nodes u and v** with N elements each.

Let $u_0 < u_1 < \dots < u_{N-1}$ and $v_0 < v_1 < \dots < v_{N-1}$.

How are the $2N$ elements **distributed** on u and v ?



Lower Bound

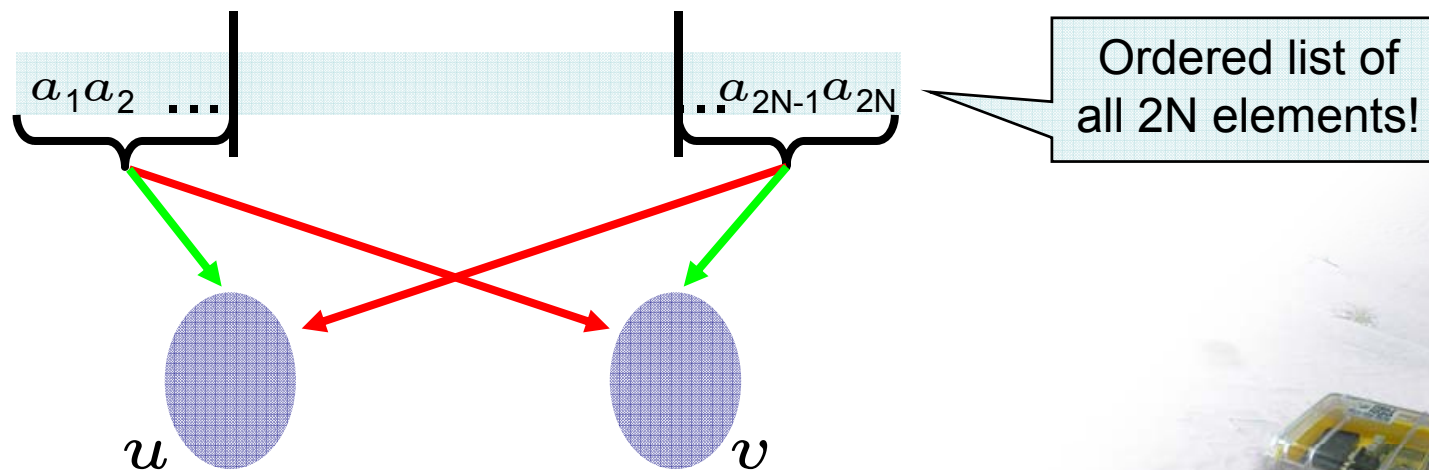
Assume $N = 2^b$. We use b independent Bernoulli variables

X_0, \dots, X_{b-1} to distribute the elements!

If $X_{b-1} = 0 \rightarrow N/2$ smallest elements go to u and the $N/2$ largest elements go to v .

If $X_{b-1} = 1$ it's the other way round.

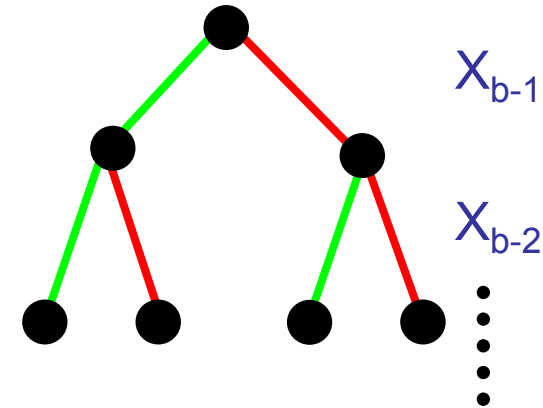
The remaining N elements are recursively distributed using the other variables X_0, \dots, X_{b-2} !



Lower Bound

Crucial observation: For all 2^b possibilities for X_0, \dots, X_{b-1} , the median is a **different element**.

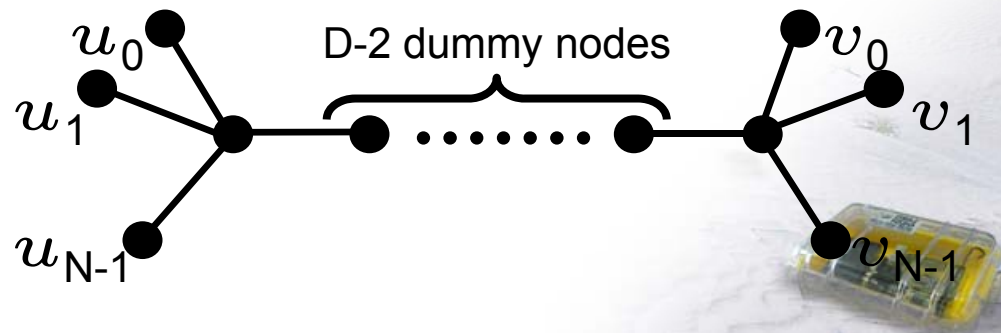
→ Determining all X_i is **equivalent** to finding the **median**!



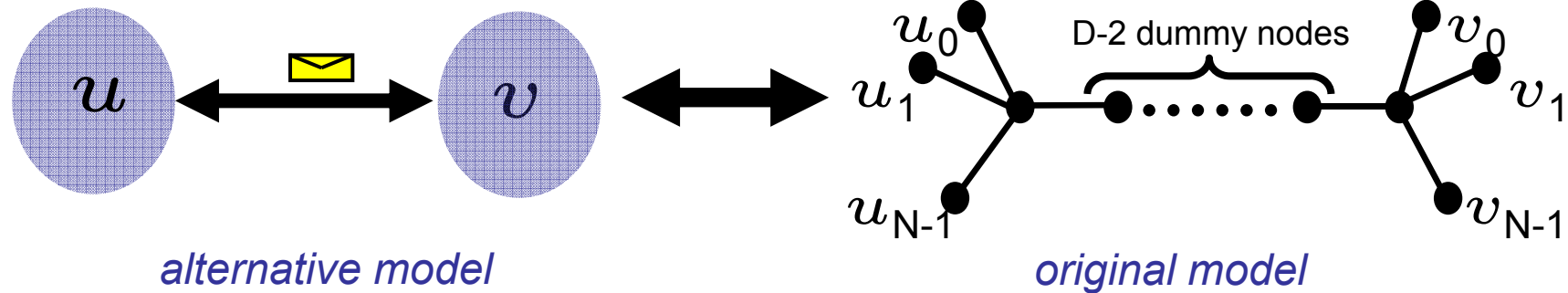
We showed that at least $\Omega(\log_{2B} N)$ rounds are required if B elements can be sent in a single round **in this model**!

Part II. Find a lower bound for the **original model**.

Look at the following graph G of diameter D :



Lower Bound



One can show that a time lower bound for the **alternative model** implies a **time lower bound** for the **original model**!

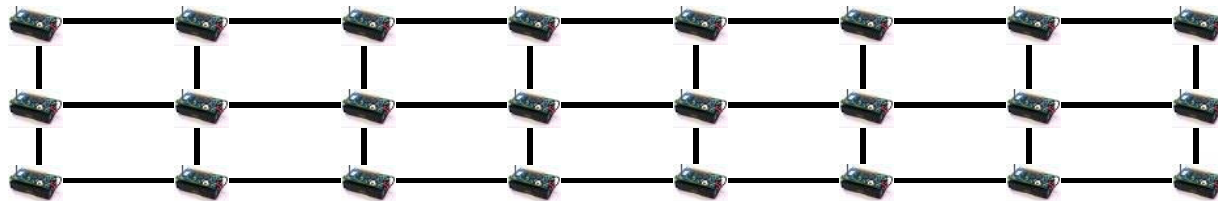
Theorem: $\Omega(D \cdot \log_D \min\{k, N-k\})$ rounds are needed to find the k^{th} smallest element.

$\Omega(D \cdot \log_D n)$ lower bound to find the **median**!

Median Summary

- Simple randomized algorithm with time complexity $O(D \cdot \log_D n)$ w.h.p.
- Easy to understand, easy to implement...
- Even asymptotically optimal! Our lower bound shows that no algorithm can be significantly faster!
- Deterministic algorithm with time complexity $O(D \cdot \log_D^2 n)$.
- If $\exists c \leq 1: D = n^c \rightarrow k$ -selection can be solved efficiently in $\Theta(D)$ time even deterministically!

Recall the 50x50 grid used to test out TAG!



Data Gathering 2

TinyDB and TinySQL

- Use paradigms familiar from relational databases to simplify the “programming” interface for the application developer.

```
SELECT roomno, AVERAGE(light), AVERAGE(volume)
FROM sensors
GROUP BY roomno
HAVING AVERAGE(light) > l AND AVERAGE(volume) > v
EPOCH DURATION 5min
```

- TinyDB then supports in-network aggregation to speed up communication.

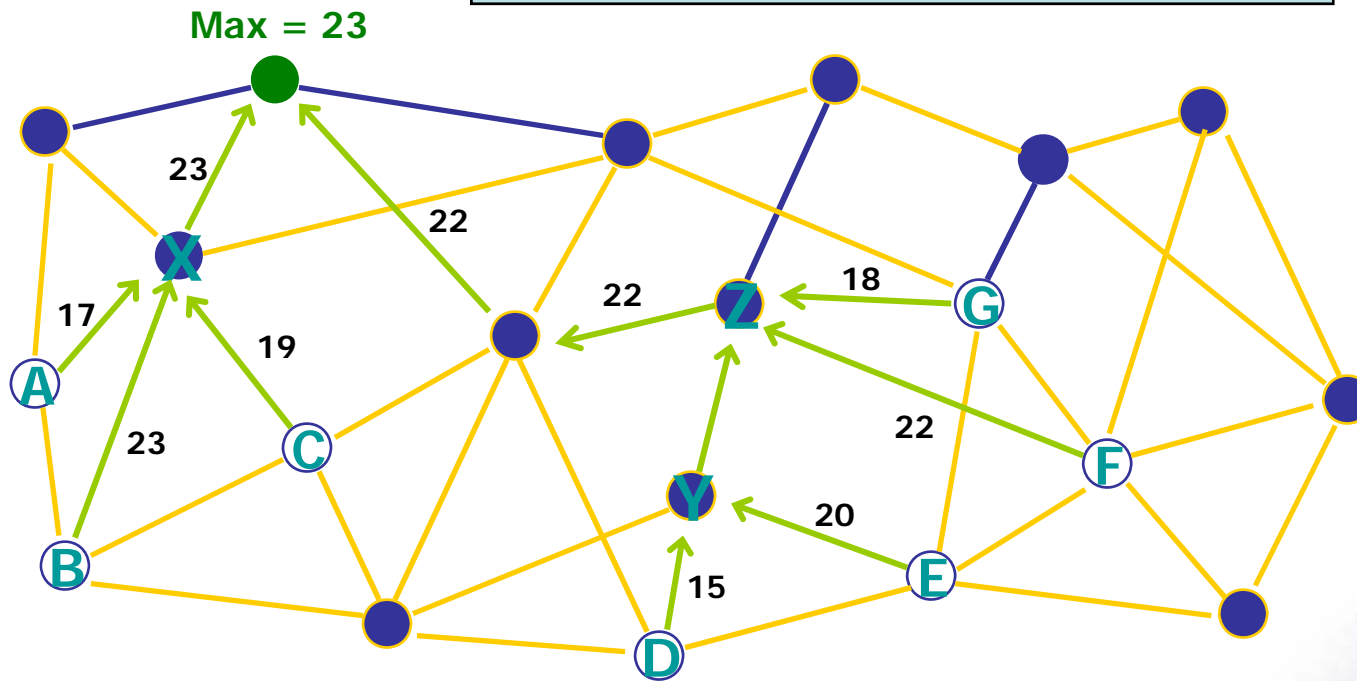
```
SELECT <aggregates>, <attributes>
[FROM {sensors | <buffer>}]
[WHERE <predicates>]
[GROUP BY <exprs>]
[SAMPLE PERIOD <const> | ONCE]
[INTO <buffer>]
[TRIGGER ACTION <command>]
```



Data Aggregation: Universal Spanning Tree

- `SELECT MAX(temp) FROM sensors WHERE node_id < "H".`

Average, Median, Count Distinct, ...?!



Selective data aggregation

- In sensor network applications
 - Queries can be frequent
 - **Sensor groups are time-varying**
 - Events happen in a dynamic fashion
- Option 1: Construct aggregation trees for each group
 - Setting up a good tree incurs communication overhead
- Option 2: Construct a single spanning tree
 - When given a sensor group, simply use the **induced tree**



Group-Independent (a.k.a. Universal) Spanning Tree

- Given
 - A set of nodes V in the Euclidean plane (or forming a metric space)
 - A root node $r \in V$
 - Define stretch of a **universal spanning tree** T to be

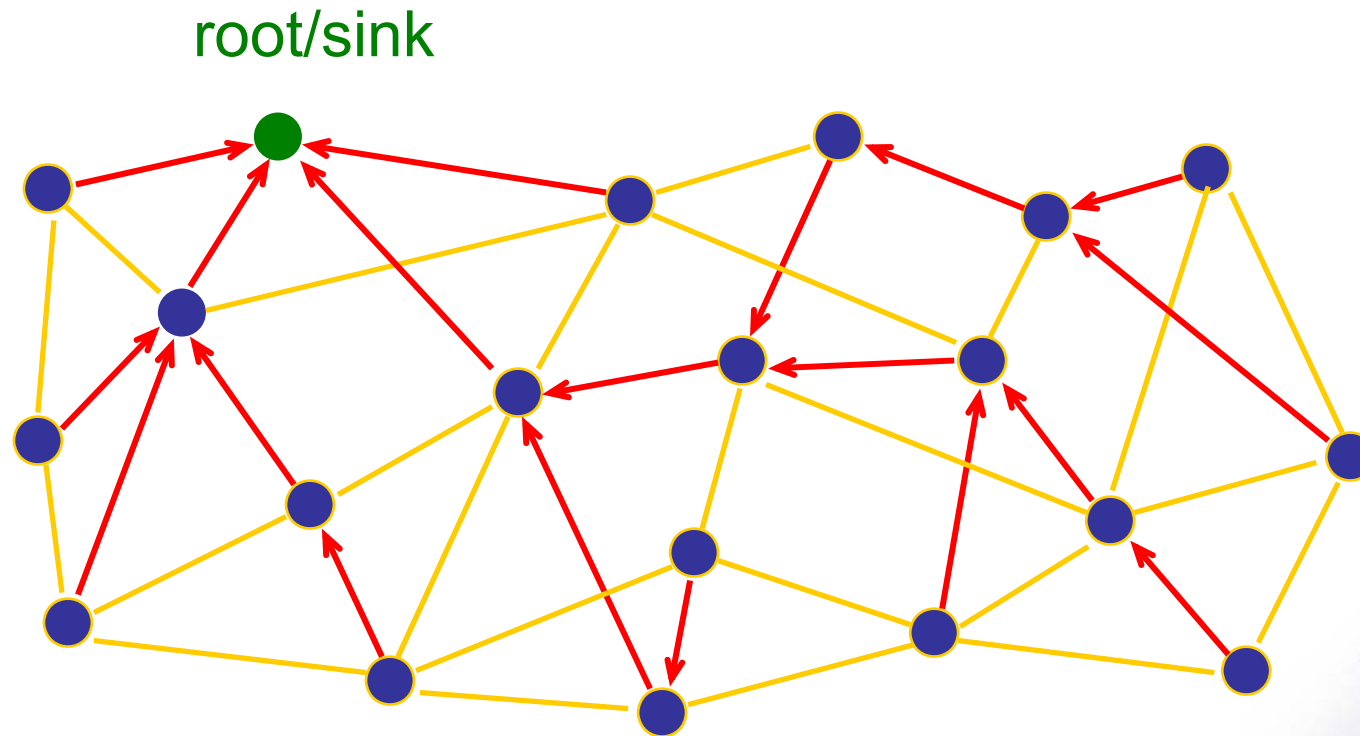
$$\max_{S \subseteq V} \frac{\text{cost}(\text{Induced tree of } S+r \text{ on } T)}{\text{cost}(\text{minimum Steiner tree of } S+r)}$$

- We're looking for a spanning tree T on V with minimum stretch.



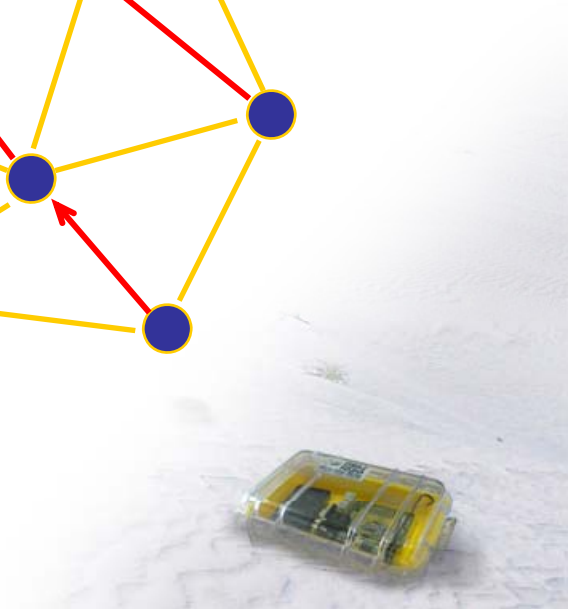
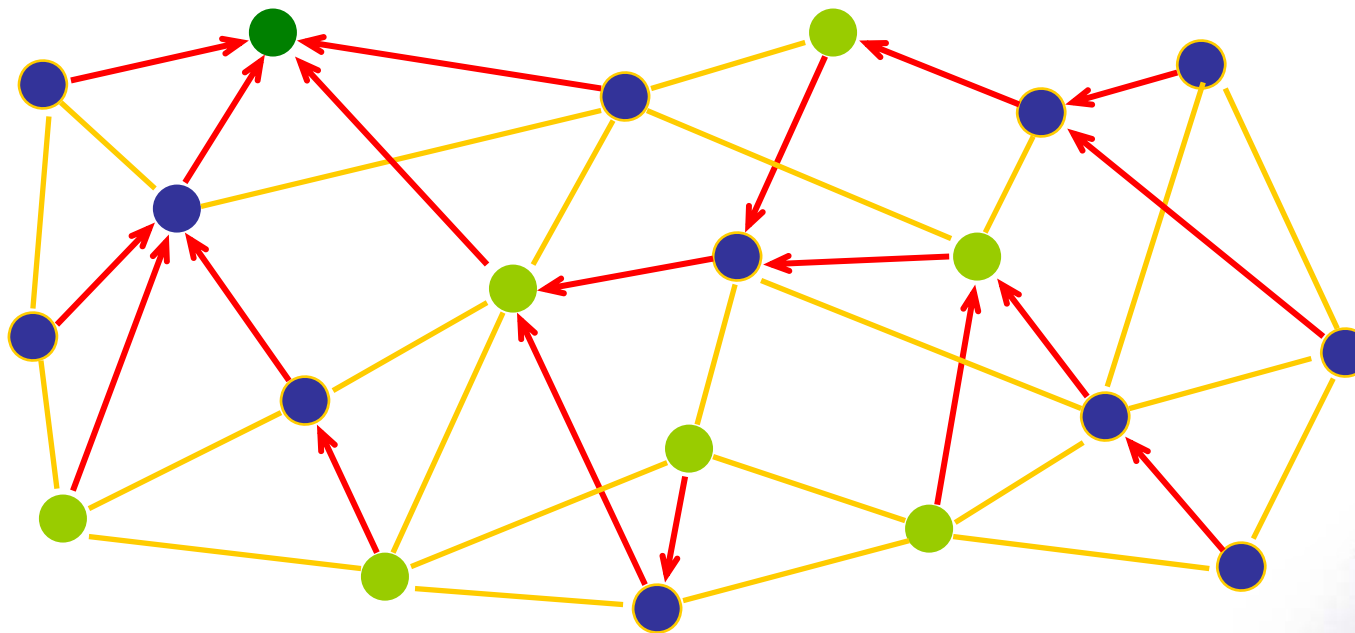
Example

- The **red** tree is the universal spanning tree. All links cost 1.



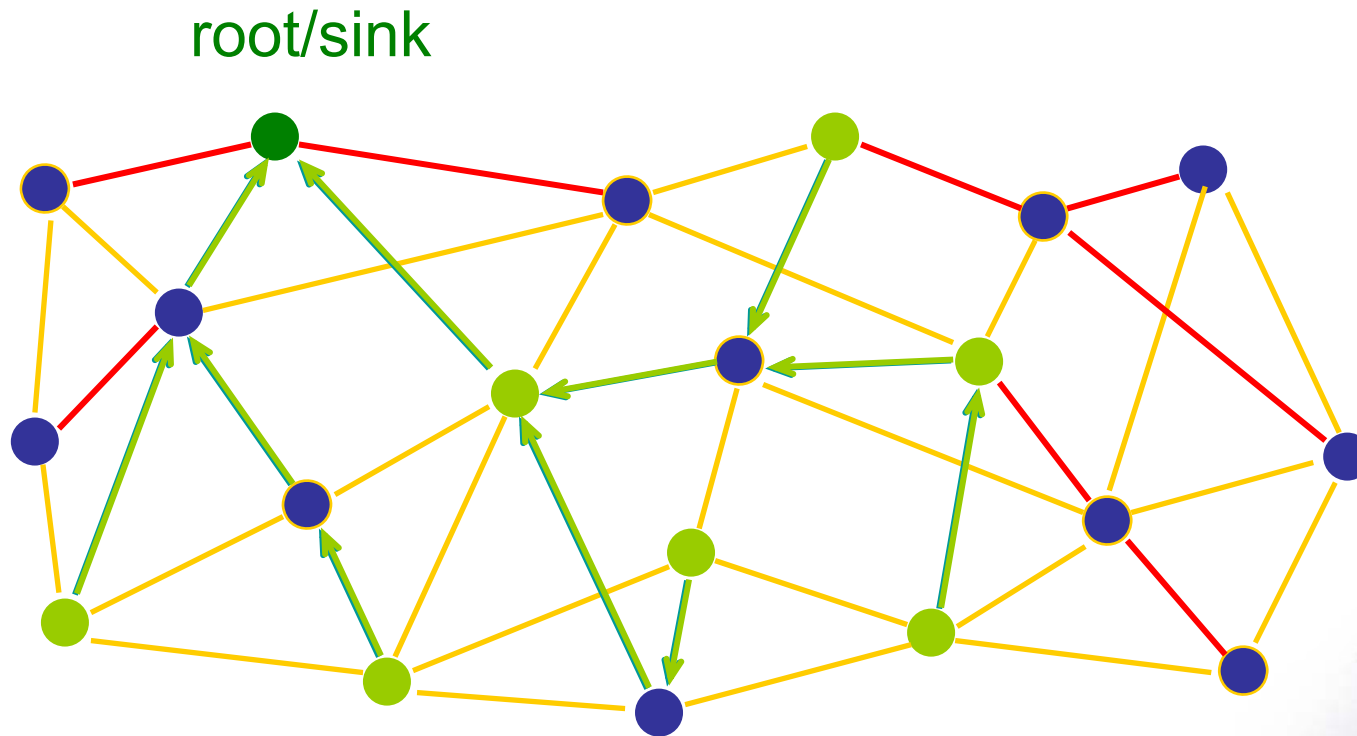
Given the lime subset...

root/sink



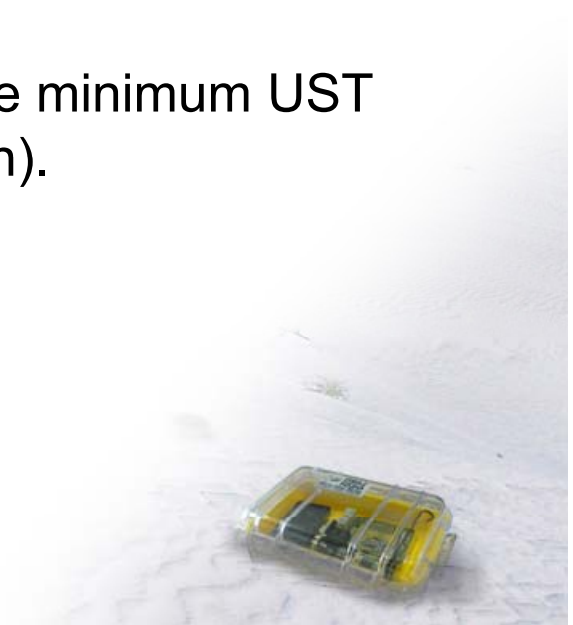
Induced Subtree

- The cost of the induced subtree for this set S is 11. The optimal was 8.



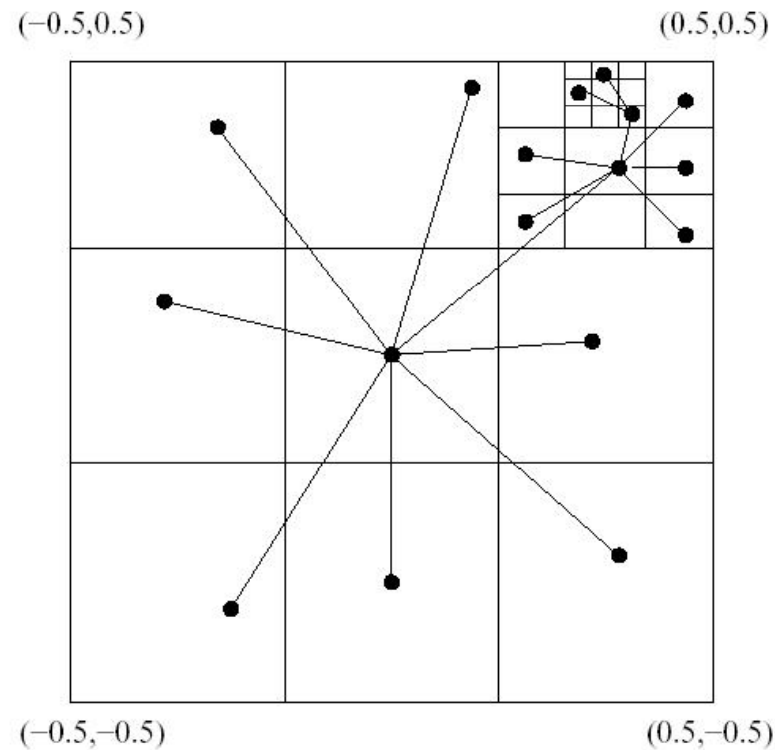
Main results

- [Jia, Lin, Noubir, Rajaraman and Sundaram, STOC 2005]
- Theorem 1: (Upper bound)
For the minimum UST problem on Euclidean plane, an approximation of $O(\log n)$ can be achieved within polynomial time.
- Theorem 2: (Lower bound)
No polynomial time algorithm can approximate the minimum UST problem with stretch better than $\Omega(\log n / \log \log n)$.
- Proofs: Not in this lecture.

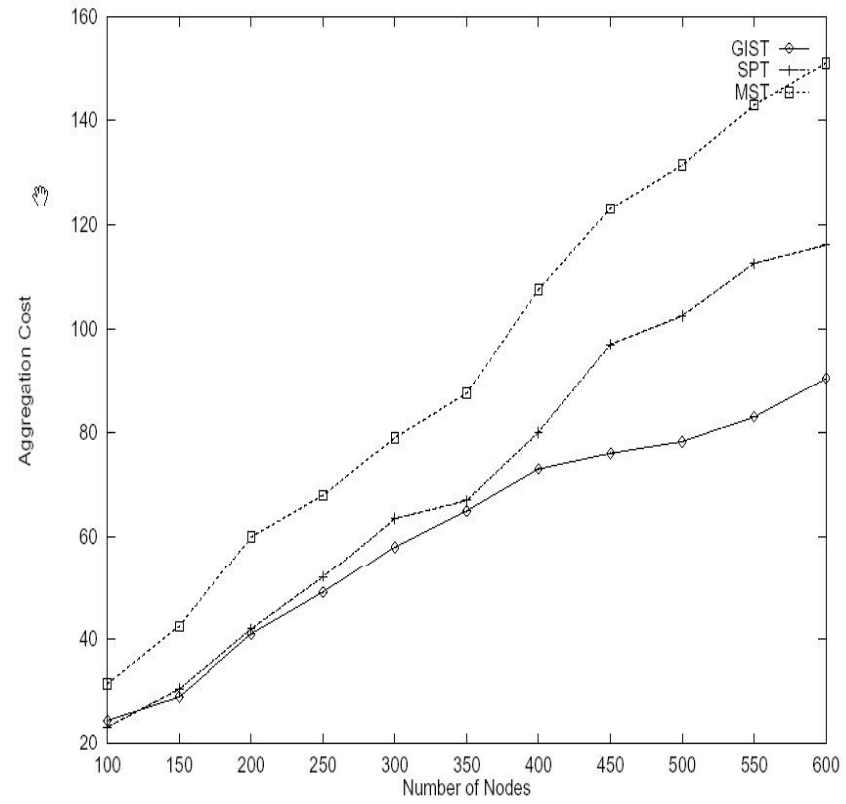
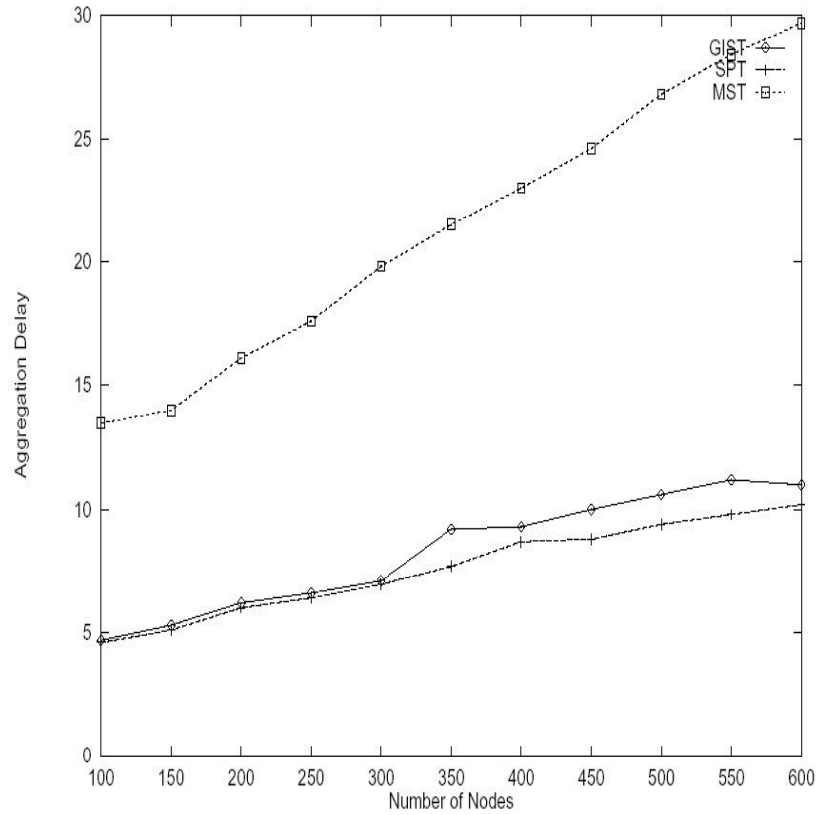


Algorithm sketch

- For the simplest Euclidean case:
- Recursively divide the plane and select random node.
- Results: The induced tree has logarithmic overhead. The aggregation delay is also constant.

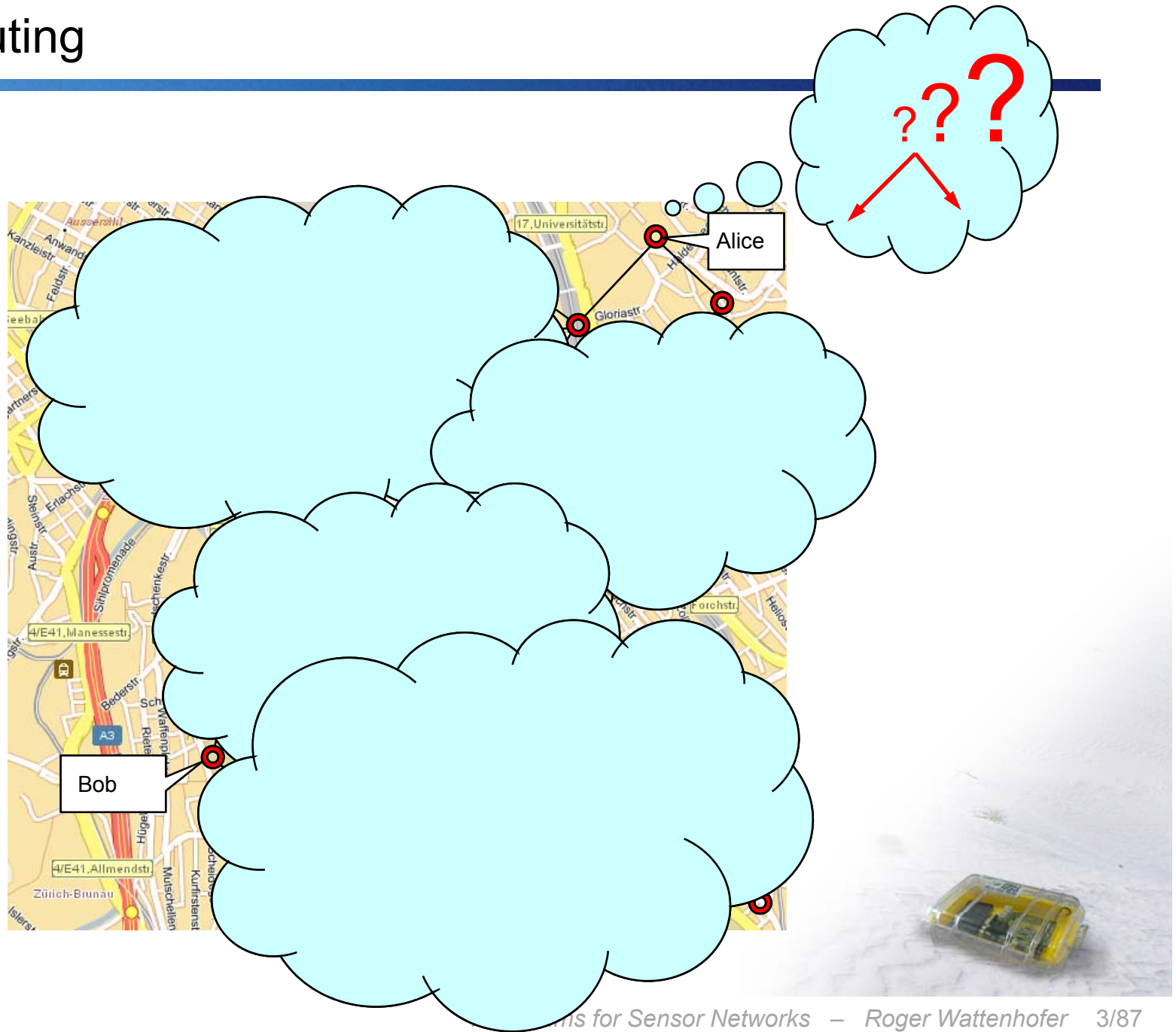


Simulation with random node distribution & random events

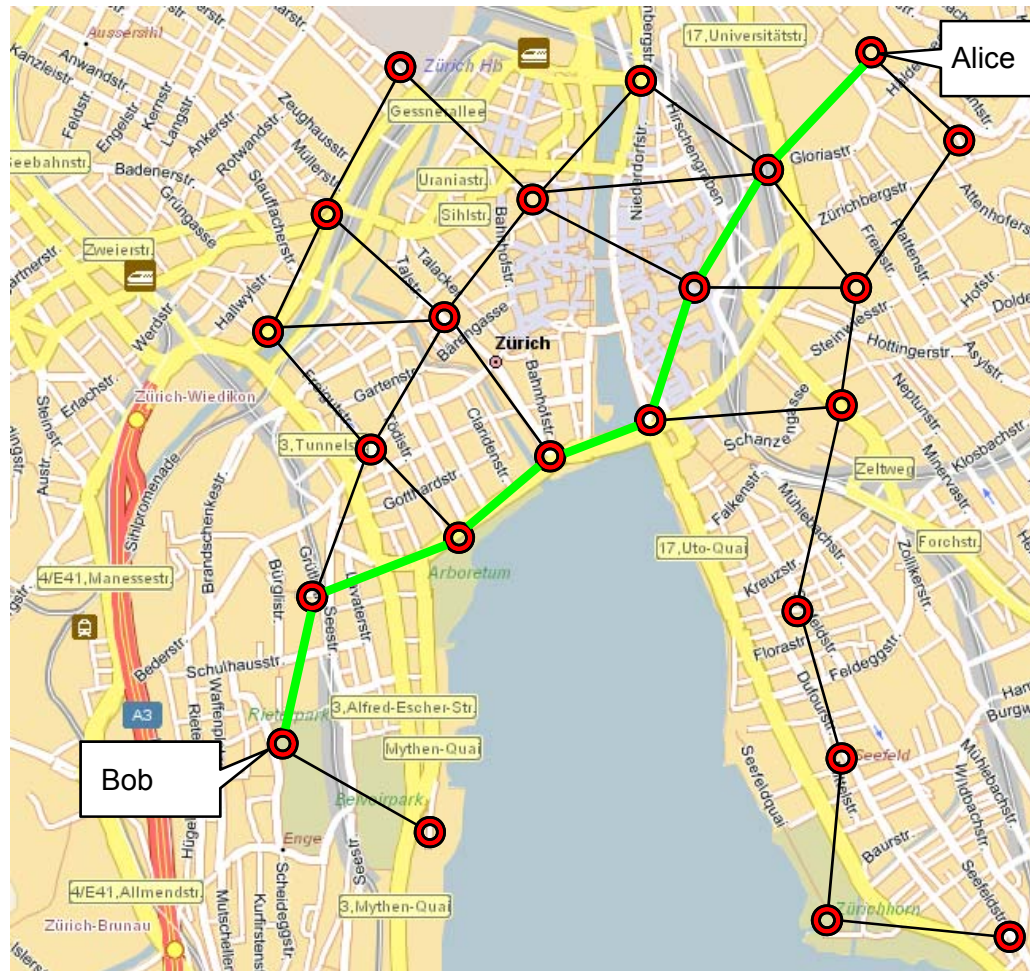


Geo-Routing

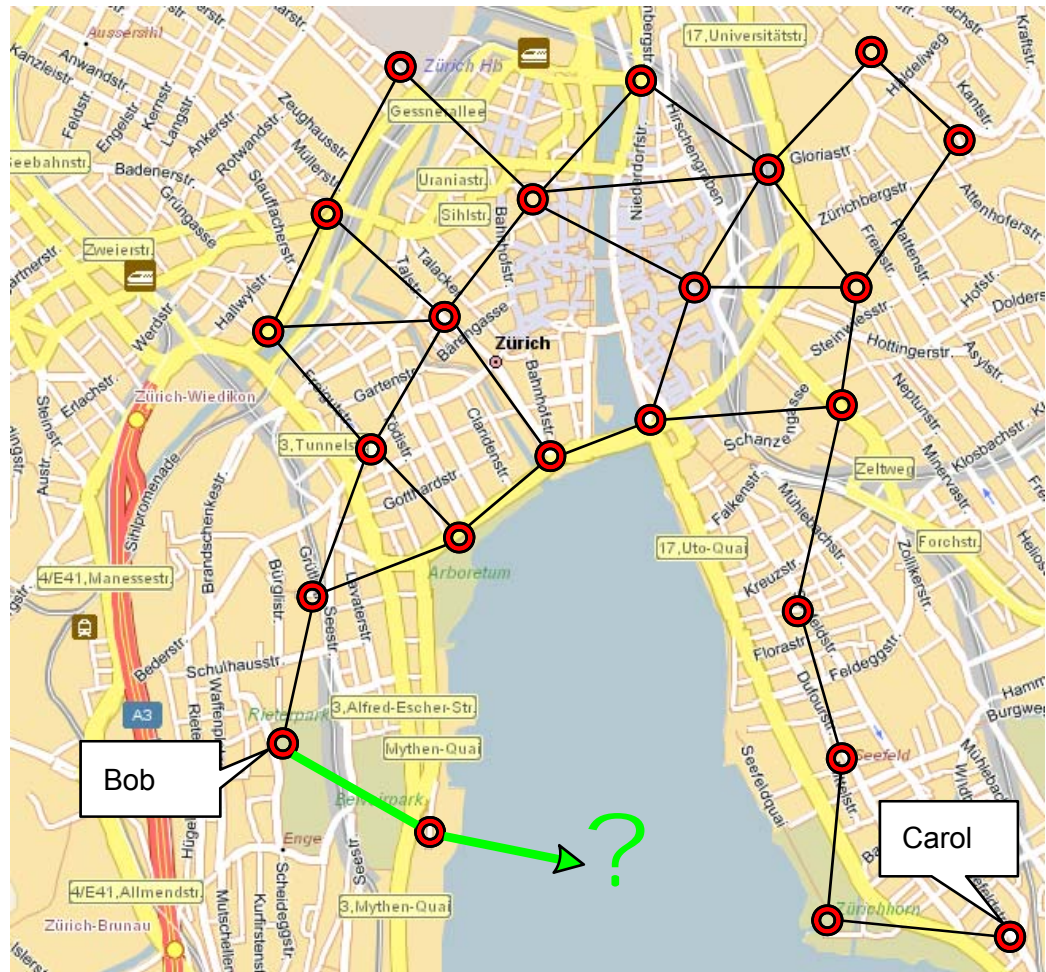
Geo-Routing



Greedy Geo-Routing?



Greedy Geo-Routing?



What is Geographic Routing?

- A.k.a. geometric, location-based, position-based, etc.

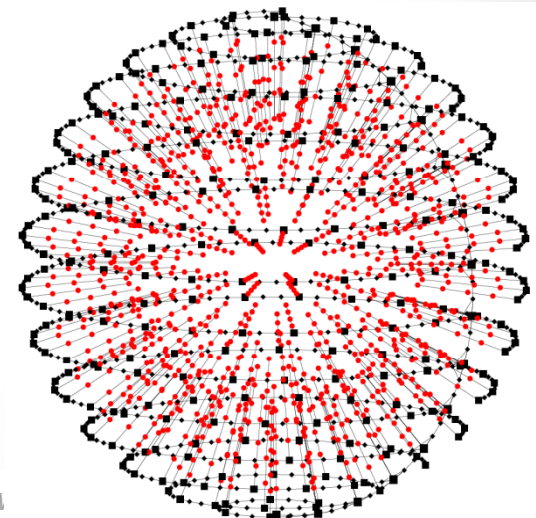
- Each node knows its own position and position of neighbors
- Source knows the position of the destination
- **No routing tables stored in nodes!**

- Geographic routing makes sense
 - Own position: GPS/Galileo, local positioning algorithms
 - Destination: Geocasting, location services, source routing++
 - **Learn about ad-hoc routing in general**



Geo-Routing Results

- Can be done (“face routing”)
 - [Kranakis, Singh, Urrutia, CCCG 1999]
 - [Bose, Morin, Stojmenovic, Urrutia, DIALM 1999]
 - later: others... “GPSR”
- At what cost?
 - Geo-routing cost (hops) is **quadratic** to optimal route [Kuhn, W, Zollinger, DIALM 2002]
- Can it be done in 3D?!?
 - Does a technique like face routing exist for 3D?
 - **No! There is no deterministic 3D geo-routing algo [Durocher, Kirkpatrick, Narayanan, ICDCN 2008]**
 - ... unless you use randomization [Flury, W, Infocom 2008]



Positioning

Rating

- Area maturity



- Practical importance



- Theoretical importance



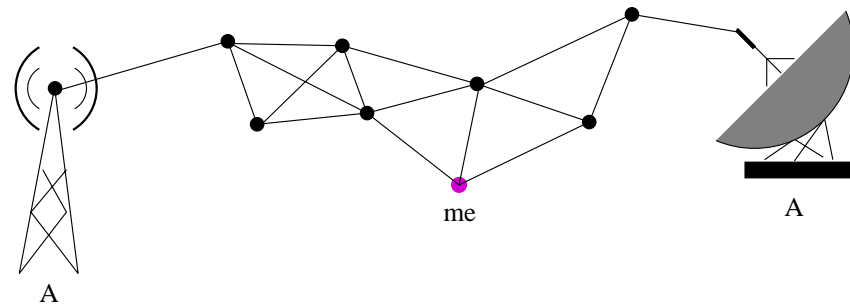
Overview

- Motivation
- GPS et al.
- Measurements
- Anchors
- Virtual Coordinates
- Heuristics
- Boundary Recognition
- Practice



Positioning

- Why positioning?
 - Sensor nodes without position information is often meaningless
 - Geo-routing



- Why **not GPS (or Galileo)**?
 - ~~Heavy, large, and expensive~~
 - ~~Battery drain~~
 - Not indoors
 - Accuracy?



- Idea: equip small fraction with GPS (**anchors**)



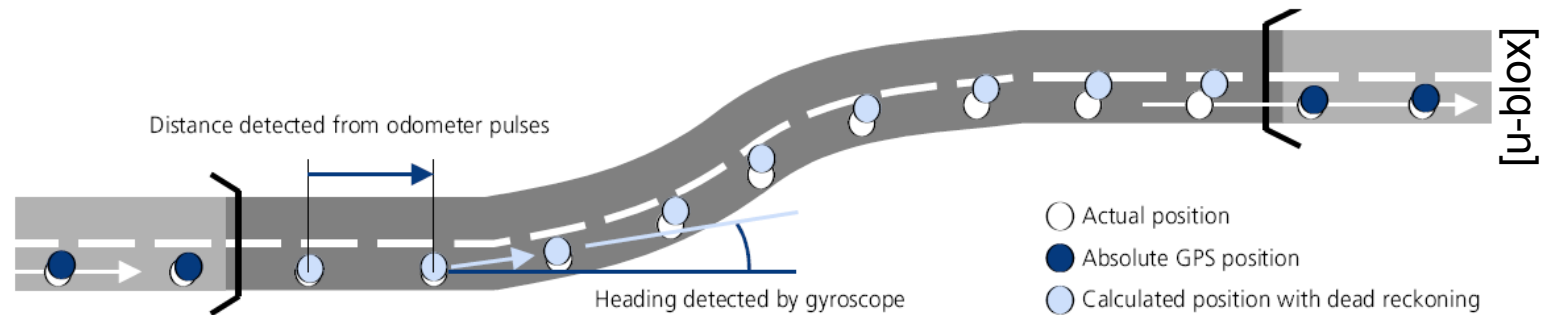
GPS

- A lot of recent progress, so attaching a GPS receiver to each sensor node becomes an alternative.
- Example, u-blox
 - footprint size: down to 4x4mm
 - Power supply: 1.8 - 4.8V
 - power consumption: 50 mW
 - power on: < 1 second
 - update rate: 4Hz
 - support for Galileo!
- So GPS is definitely becoming more attractive; however, some of the problems of GPS (indoors, accuracy, etc.) remain.



GPS Extensions

- GPS chips can be extended with other sensors such as gyroscope, direction indications, or tachometer pulses (in cars). With these additions, mobile devices get continued coverage indoors.



- Affordable technology has a distance error of less than 5% per distance travelled, and a direction error of less than 3 degrees per minute.



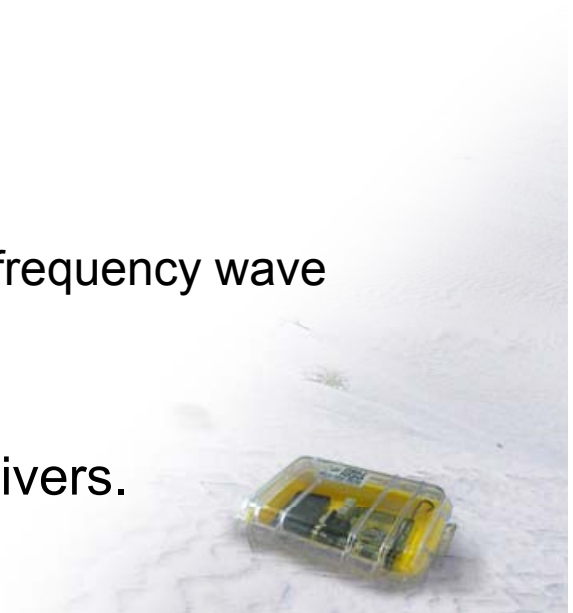
Measurements

Distance estimation

- Received Signal Strength Indicator (RSSI)
 - The further away, the weaker the received signal.
 - Mainly used for RF signals.
- Time of Arrival (ToA) or Time Difference of Arrival (TDoA)
 - Signal propagation time translates to distance.
 - RF, acoustic, infrared and ultrasound.

Angle estimation

- Angle of Arrival (AoA)
 - Determining the direction of propagation of a radio-frequency wave incident on an antenna array.
- Directional Antenna
- Special hardware, e.g., laser transmitter and receivers.



Example: Measuring distance with two radios

- Particularly interesting if the signal speed differs substantially, e.g. sound propagation is at about 331 m/s (depending on temperature, humidity, etc.), which is of course much less than the speed of light.

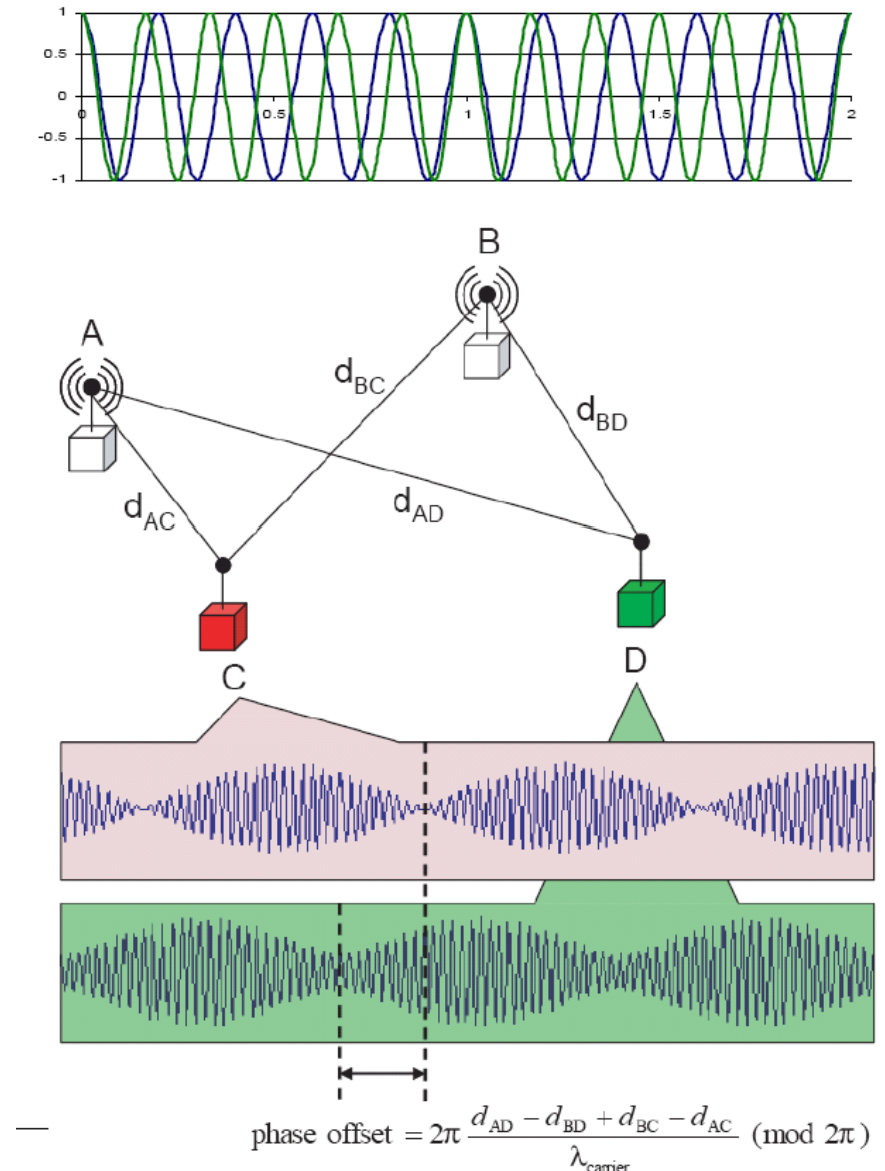


- If you have free sight you may achieve about a 1cm accuracy. But there are problems
 - (Ultra)sound does not travel far
 - For good results you need line of sight
 - You have to deal with reflections



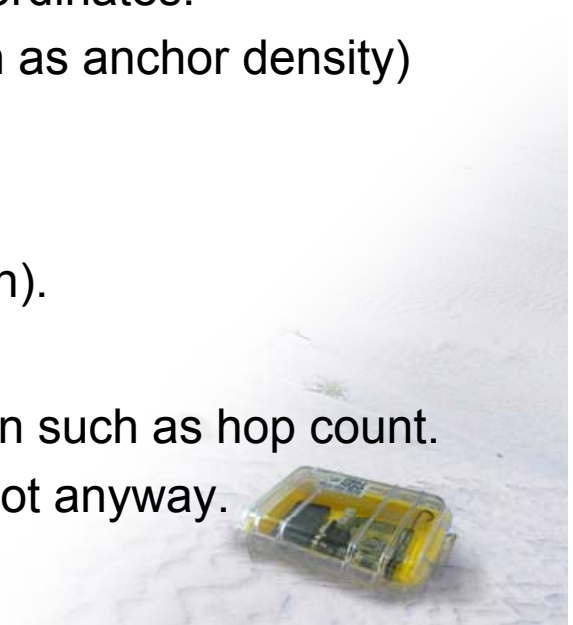
Inferometry

- Interferometry is the technique of superimposing (interfering) two or more waves, to detect differences between them
- Signals transmitted with a few hundred Hz difference at senders A and B will give different phase offsets at C and D. With that one can compute the total distance of the four points A, B, C, D.
- However, one needs to solve a linear equation system, and one needs very accurate time synchronization (μs order)



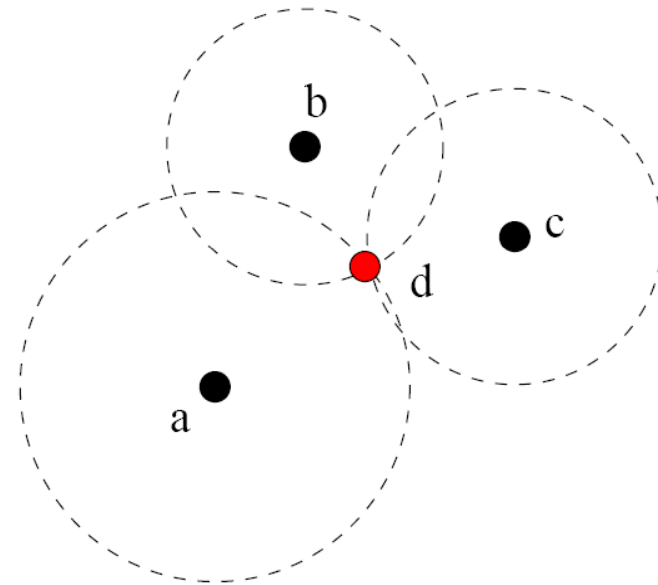
Positioning (a.k.a. Localization)

- Task: Given distance or angle measurements or mere connectivity information, find the locations of the sensors.
- **Anchor-based**
 - Some nodes know their locations, either by a GPS or as pre-specified.
- **Anchor-free**
 - Relative location only. Sometimes called virtual coordinates.
 - Theoretically cleaner model (less parameters, such as anchor density)
- **Range-based**
 - Use range information (distance or angle estimation).
- **Range-free**
 - No distance estimation, use connectivity information such as hop count.
 - It was shown that bad measurements don't help a lot anyway.



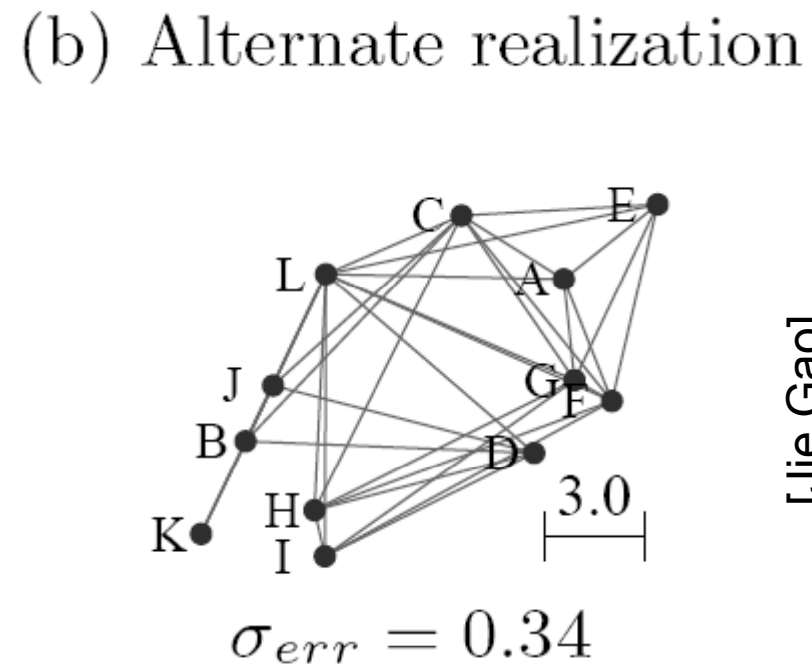
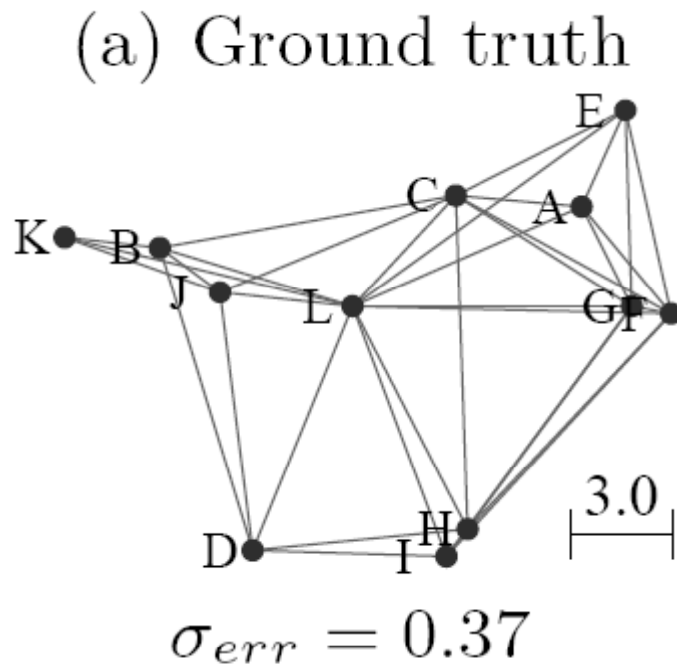
Trilateration and Triangulation

- Use geometry, measure the distances/angles to three anchors.
- **Trilateration**: use distances
 - Global Positioning System (GPS)
- **Triangulation**: use angles
 - Some cell phone systems
- How to deal with inaccurate measurements?
 - Least squares type of approach
 - What about strictly more than 3 (inaccurate) measurements?



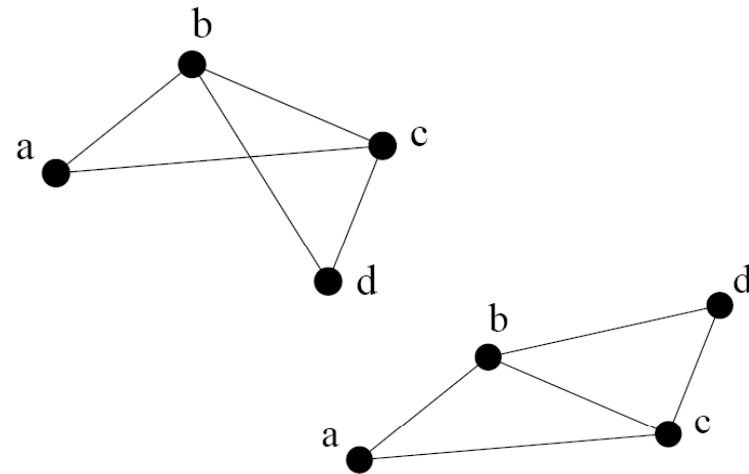
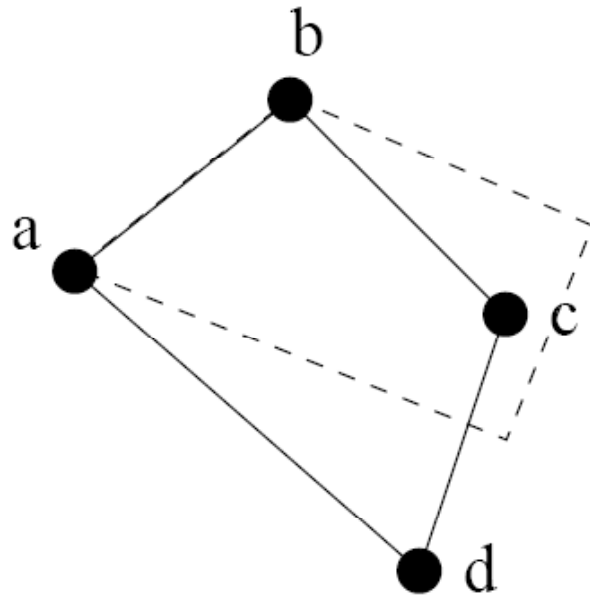
Ambiguity Problems

- Same distances, different realization.



[Jie Gao]

Continuous deformation, flips, etc.



[Jie Gao]

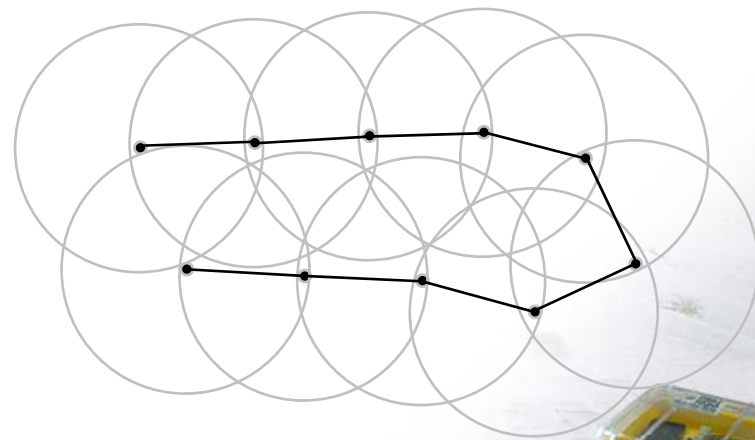
- Rigidity theory: Given a set of rigid bars connected by hinges, rigidity theory studies whether you can move them continuously.



Simple hop-based algorithms

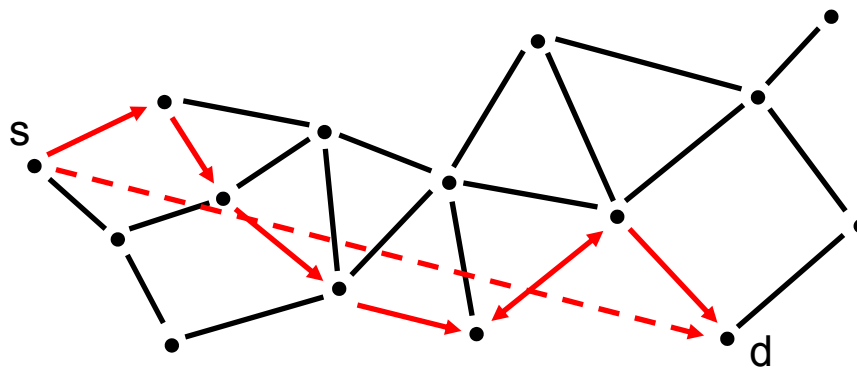
- Algorithm
 - Get graph **distance h** to anchor(s)
 - Intersect circles around anchors
 - radius = distance to anchor
 - Choose point such that **maximum error is minimal**
 - Find **enclosing circle** (ball) of minimal radius
 - Center is calculated location

- In higher dimensions: $1 < d \leq h$
 - Rule of thumb: **Sparse graph**
 - bad performance



How about no anchors at all...?

- In absence of anchors...
 - ...nodes are clueless about **real coordinates**.
- For many applications, real coordinates are not necessary
 - **Virtual coordinates** are sufficient
 - Geometric Routing requires only virtual coordinates
 - Require no routing tables
 - Resource-frugal and scalable

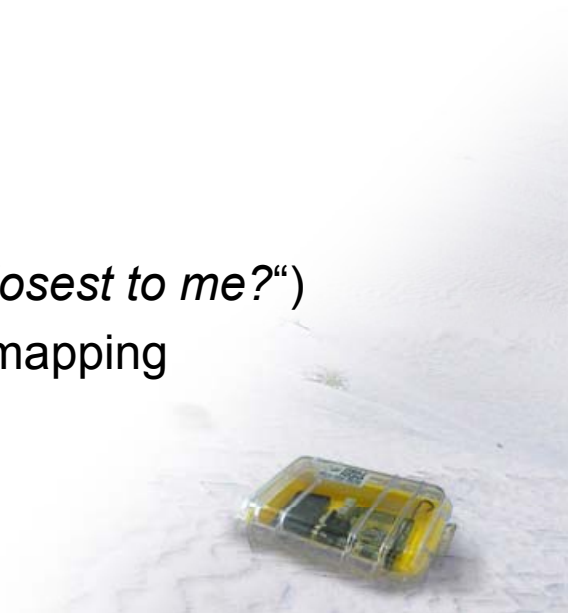


Virtual Coordinates

- Idea:
Close-by nodes have similar coordinates
Distant nodes have very different coordinates

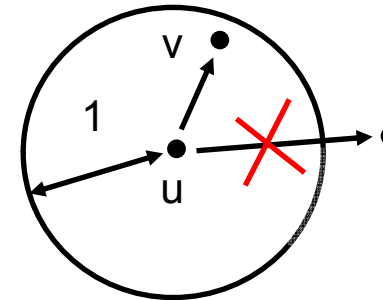
→ Similar coordinates imply physical proximity!

- Applications
 - Geometric Routing
 - Locality-sensitive queries
 - Obtaining meta information on the network
 - Anycast services („Which of the service nodes is closest to me?“)
 - Outside the sensor network domain: e.g., Internet mapping



Model

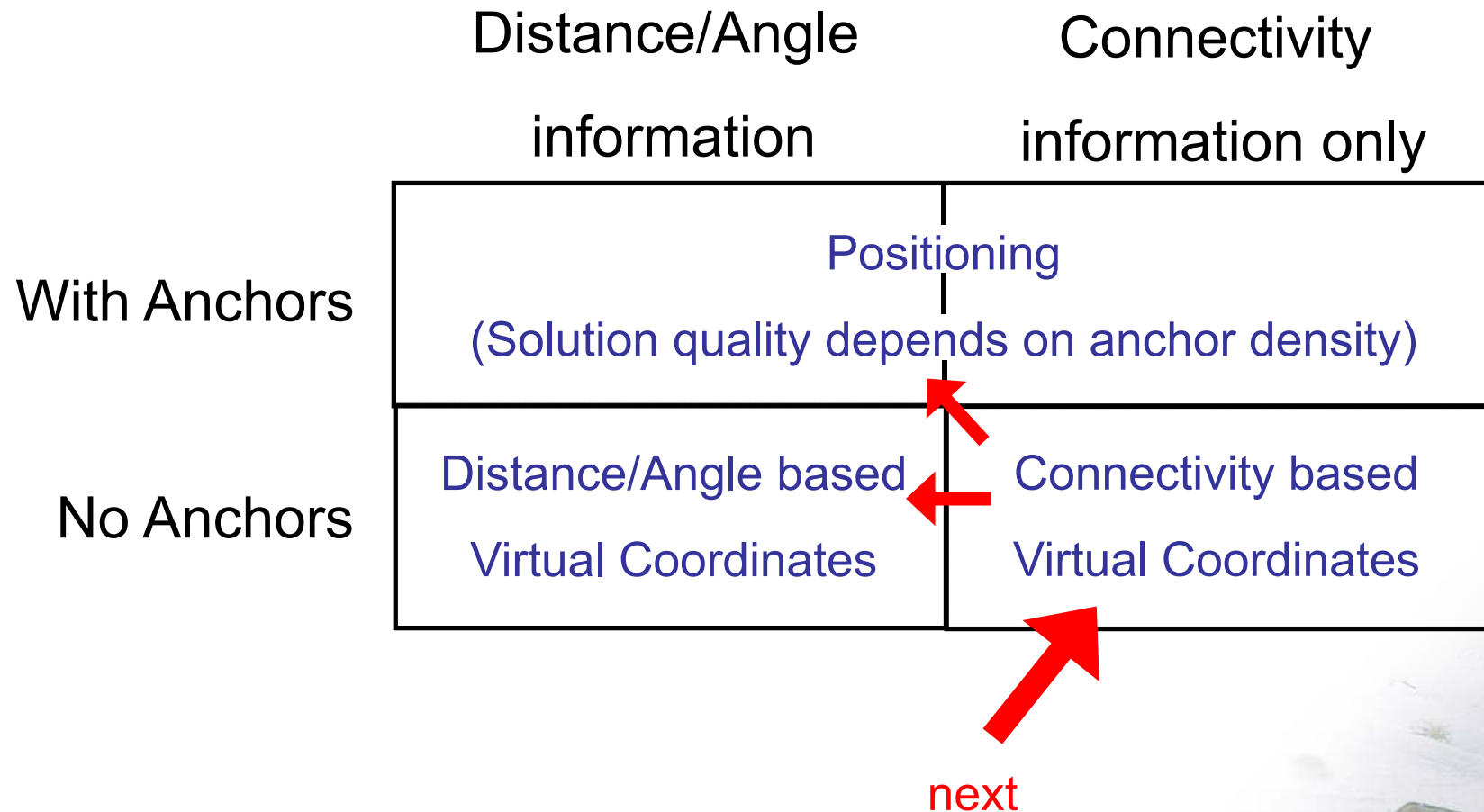
- **Unit Disk Graph (UDG)** to model wireless multi-hop network
 - Two nodes can communicate iff Euclidean distance is at most 1



- Sensor nodes may not be capable of
 - Sensing directions to neighbors
 - Measuring distances to neighbors
- Goal: Derive topologically correct coordinate information from **connectivity information** only.
 - Even the simplest nodes can derive connectivity information

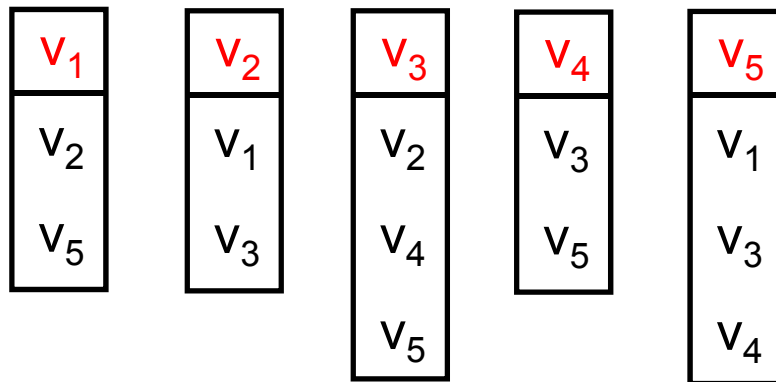


Context



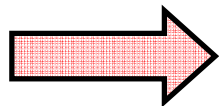
Virtual Coordinates \longleftrightarrow UDG Embedding

- Given the **connectivity information** for each node...

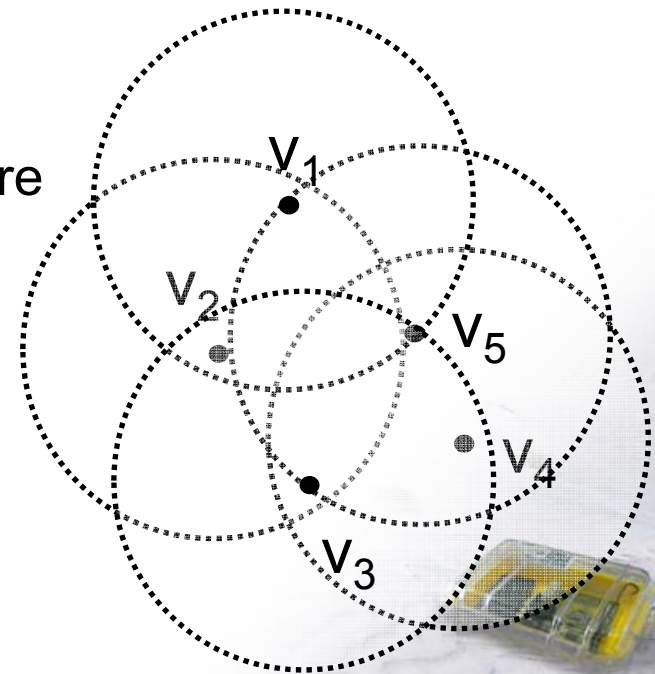


...and knowing the underlying graph is a UDG...

...find a UDG embedding in the plane such that all connectivity requirements are fulfilled! (\rightarrow Find a realization of a UDG)



This problem is NP-hard!
(Simple reduction to *UDG-recognition* problem, which is NP-hard)
[Breu, Kirkpatrick, Comp.Geom.Theory 1998]



UDG Approximation – Quality of Embedding

- Finding an exact realization of a UDG is NP-hard.
→ Find an embedding $r(G)$ which **approximates a realization**.
- Particularly,
→ Map adjacent vertices (**edges**) to points which are close together.
→ Map non-adjacent vertices („**non-edges**“) to far apart points.
- Define **quality of embedding** $q(r(G))$ as:

Ratio between longest edge to shortest non-edge in the embedding.

Let $\rho(u,v)$ be the distance between points u and v in the embedding.

$$q(r(G)) := \frac{\max_{\{u,v\} \in E} \rho(u,v)}{\min_{\{u',v'\} \notin E} \rho(u',v')}$$

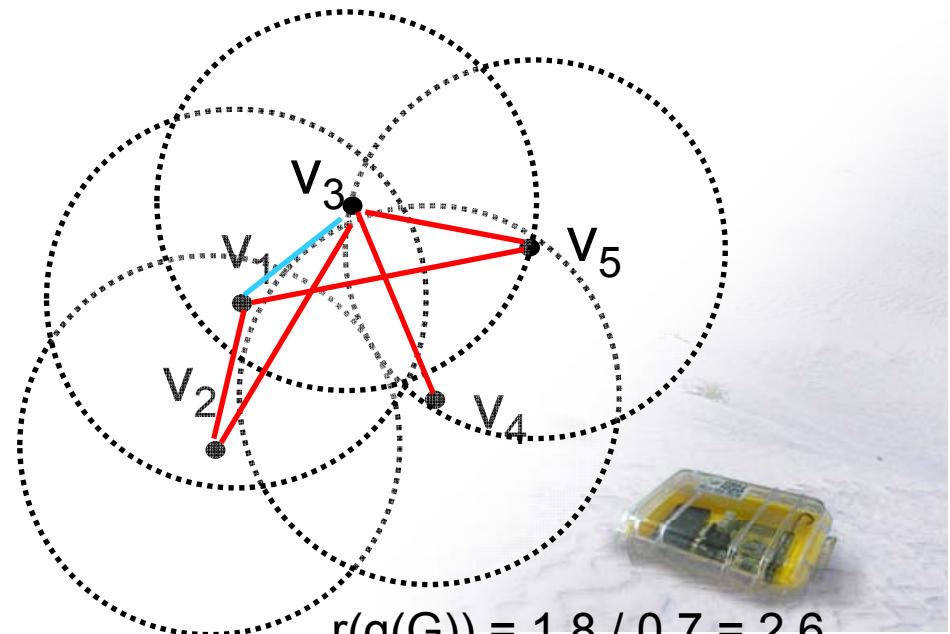
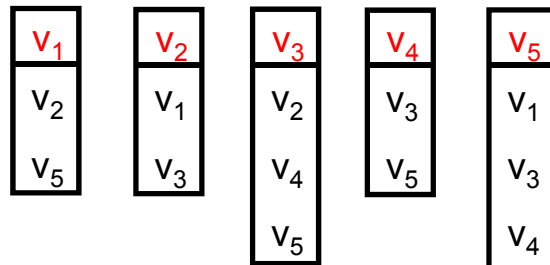
UDG Approximation

- For each UDG G , there exists an embedding $r(G)$, such that, $q(r(G)) \leq 1$.
(a realization of G)

$$q(r(G)) := \frac{\max_{\{u,v\} \in E} \rho(u,v)}{\min_{\{u',v'\} \notin E} \rho(u',v')}$$

- Finding such an embedding is NP-hard
- An algorithm ALG achieves **approximation ratio α** if for all unit disk graphs G , $q(r_{\text{ALG}}(G)) \leq \alpha$.

- Example:



$$r(q(G)) = 1.8 / 0.7 = 2.6$$

Some Results

- There are a few virtual coordinates algorithms
Almost all of them evaluated only by **simulation on random graphs**
- In fact there are very few **provable approximation algorithms**

There is an algorithm which achieves an approximation ratio of $O(\log^{2.5} n)$, n being the number of nodes in G .
[Pemmaraju and Pirwani, 2007]

Plus there are lower bounds on the approximability.

There is no algorithm with approximation ratio better than $\sqrt{3}/2 - \epsilon$ unless $P = NP$.



Approximation Algorithm: Overview

- Four major steps
 1. Compute **metric** on MIS of input graph → **Spreading constraints**
(Key conceptual difference to previous approaches!)
 2. **Volume-respecting**, high dimensional **embedding**
 3. **Random projection** to 2D
 4. Final embedding

UDG Graph G with MIS M .



Approximate pairwise distances between nodes such that, MIS nodes are neatly spread out.



Volume respecting embedding of nodes in R^n with small distortion.



Nodes spread out fairly well in R^2 .



Final embedding of G in R^2 .



Lower Bound: Quasi Unit Disk Graph

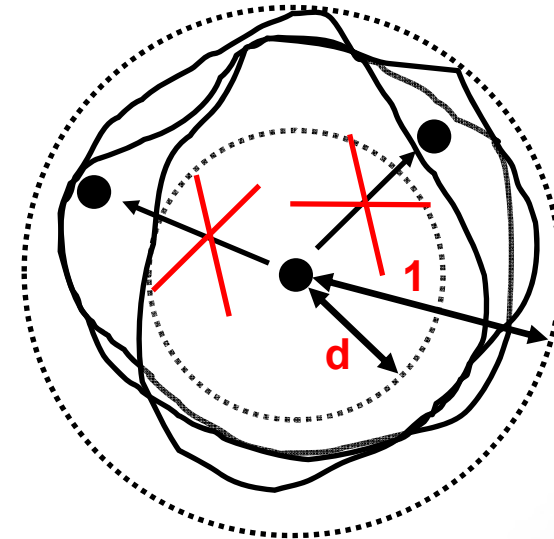
- Definition **Quasi Unit Disk Graph**:

Let $V \in \mathbb{R}^2$, and $d \in [0,1]$. The symmetric Euclidean graph $G=(V,E)$, such that for any pair $u,v \in V$

$\text{dist}(u,v) \leq d \Rightarrow \{u,v\} \in E$

$\text{dist}(u,v) > 1 \Rightarrow \{u,v\} \notin E$

is called *d-quasi unit disk graph*.



Note that between d and 1 , the existence of an edge is unspecified.



Reduction

- We want to show that finding an embedding with $q(r(G)) \leq \sqrt{3/2} - \epsilon$, where ϵ goes to 0 for $n \rightarrow \infty$ is NP-hard.
- We prove an equivalent statement:

Given a unit disk graph $G=(V,E)$, it is NP-hard to find a realization of G as a d -quasi unit disk graph with $d \geq \sqrt{2/3} + \epsilon$, where ϵ tends to 0 for $n \rightarrow \infty$.

- Even when allowing non-edges to be smaller than 1, embedding a unit disk graph remains NP-hard!
- It follows that finding an approximation ratio better than $\sqrt{3/2} - \epsilon$ is also NP-hard.



Reduction

- Reduction from 3-SAT (each variable appears in at most 3 clauses)
- Given a instance C of this 3-SAT, we give a polynomial time construction of $G_C=(V_C, E_C)$ such that the following holds:

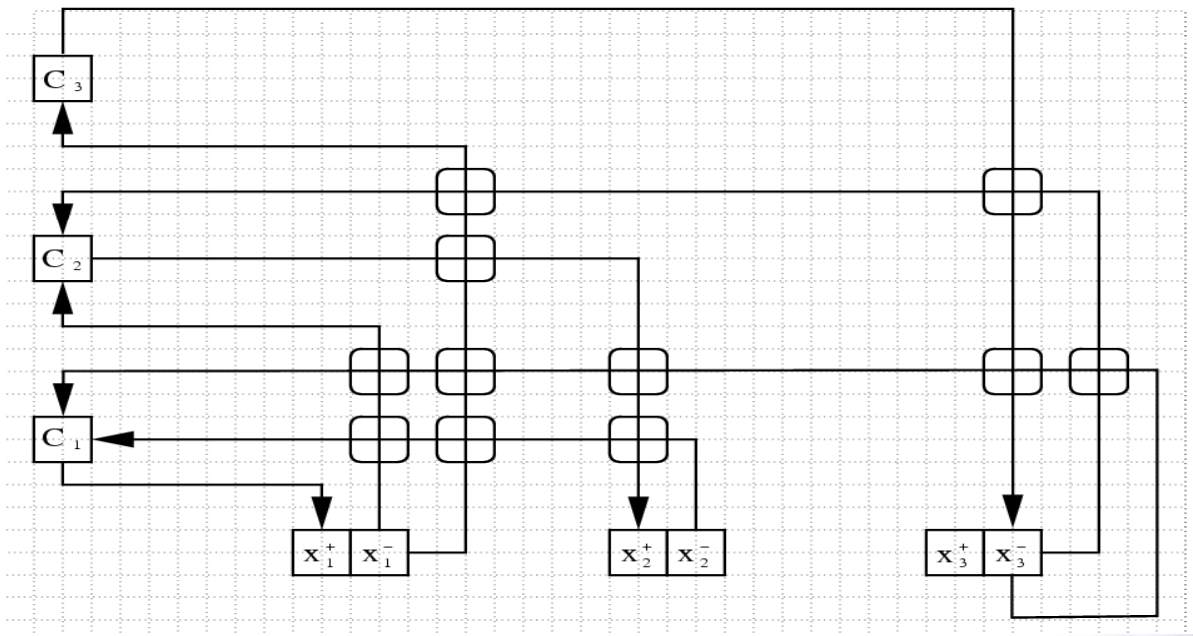
- C is satisfiable $\Rightarrow G_C$ is realizable as a unit disk graph
- C is not satisfiable $\Rightarrow G_C$ is not realizable as a d -quasi unit disk graph with $d \geq \sqrt{2/3} + \epsilon$

- Unless $P=NP$, there is no approximation algorithm with approximation ratio better than $\sqrt{3/2} - \epsilon$.



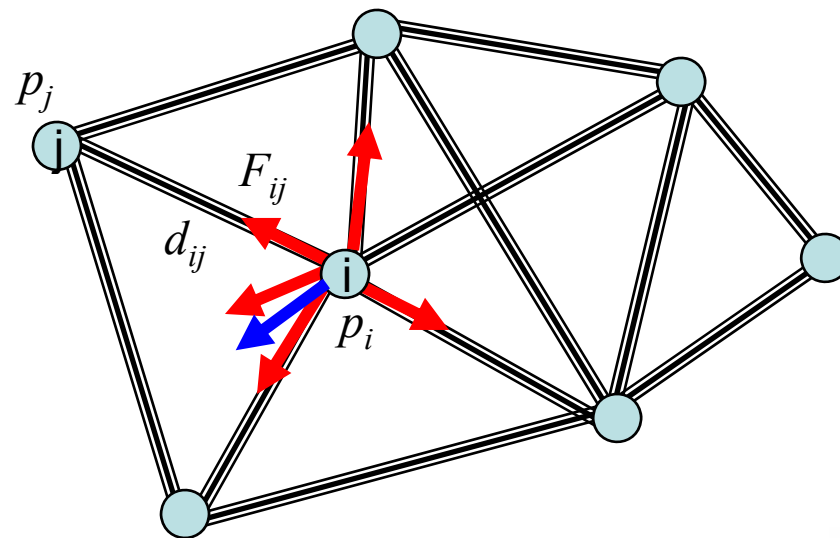
Proof idea

- Construct a grid drawing of the SAT instance.
- Grid drawing is *orientable* iff SAT instance is satisfiable.
- Grid components (clauses, literals, wires, crossings,...) are composed of nodes \rightarrow Graph G_C .
- G_C is *realizable as a d-quasi unit disk graph* with $d \geq \sqrt{2/3} + \epsilon$ iff grid drawing is orientable.



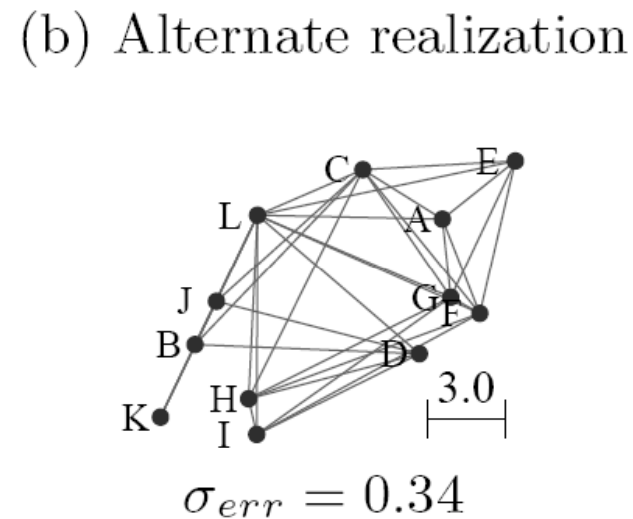
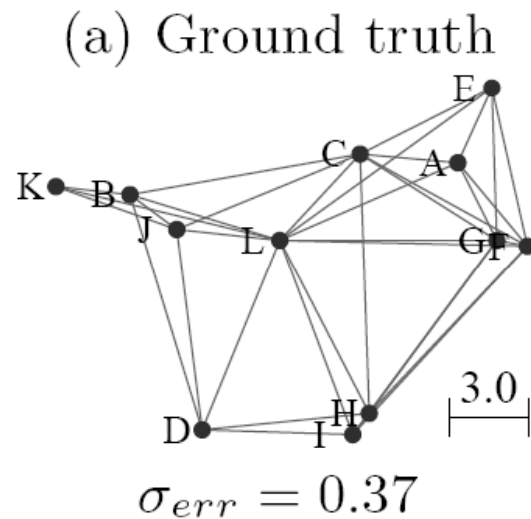
Heuristics: Spring embedder

- Nodes are “masses”, edges are “springs”.
- Length of the spring equals the distance measurement.
- Springs put forces to the nodes, nodes move, until stabilization.
- Force: $F_{ij} = d_{ij} - r_{ij}$, along the direction $p_i p_j$.
- Total force on n_i : $F_i = \sum F_{ij}$.
- Move the node n_i by a small distance (proportional to F_i).



Spring Embedder Discussion

- Problems:
 - may deadlock in local minimum
 - may never converge/stabilize (e.g. just two nodes)
- Solution: Need to start from a reasonably good initial estimation.



[Jie Gao]



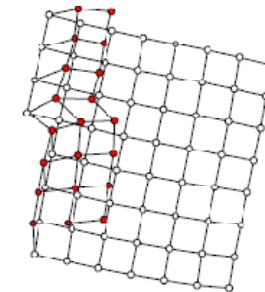
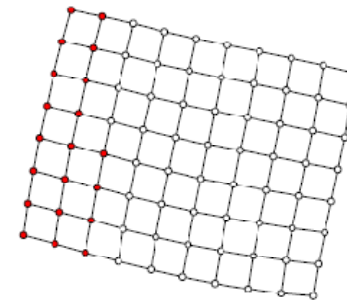
Heuristics: Priyantha et al.

N.B. Priyantha, H. Balakrishnan, E. Demaine, S. Teller:
Anchor-Free Distributed Localization
in **Sensor Networks**, *SenSys*, 2003.

iterative process minimizes the layout energy

$$E(p) = \sum_{\{i,j\} \in E} \left(\|p_i - p_j\| - \ell_{ij} \right)^2$$

- ▶ fact: layouts can have *foldovers* without violating the distance constraints
- ▶ problem: optimization can converge to such a local optimum
- ▶ solution: find a good initial layout *fold-free* → already close to the global optimum (=“real layout”)

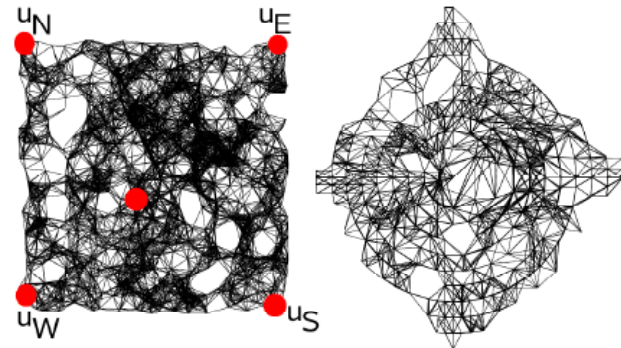


[Fleischer & Pich]

Continued

Phase 1: compute initial layout

- ▶ determine periphery nodes u_N, u_S, u_W, u_E
- ▶ determine central node u_C
- ▶ use polar coordinates



$$\rho_v = d(v, u_C) \quad \theta_v = \arctan \left(\frac{d(v, u_N) - d(v, u_S)}{d(v, u_W) - d(v, u_E)} \right)$$

as positions of node v

[Fleischer & Pich]

Phase 2: Spring Embedder



Heuristics: Gotsman et al.

C. Gotsman, Y. Koren [5]. **Distributed Graph Layout for Sensor Networks**, *GD*, 2004.

- ▶ initial placement: spread sensors

$$\frac{\sum_{\{i,j\} \in E} \exp(-l_{ij}) \|p_i - p_j\|^2}{\sum_{i < j} \|p_i - p_j\|^2} \rightarrow \min$$

- ▶ linear algebra:
minimized by second highest eigenvector v_2 of A where

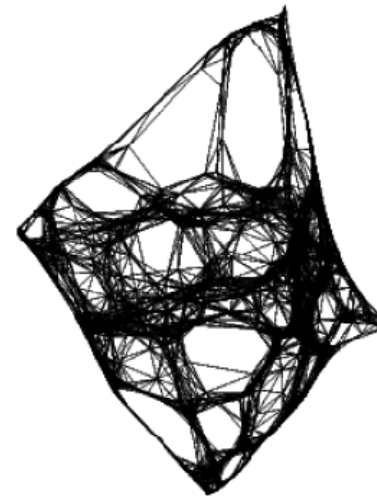
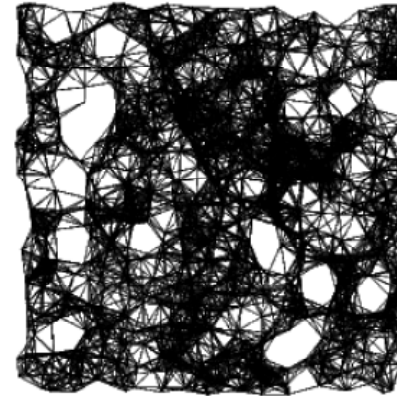
$$a_{ij} = -\frac{\exp(-l_{ij})}{\sum_{j: \{i,j\} \in E} \exp(-l_{ij})}$$

$$a_{ii} = 1$$

- ▶ x, Ax, A^2x, A^3x, \dots converges to v_2

- ▶ $x_i \leftarrow \frac{1}{2} \left(x_i + \frac{\sum_{j: \{i,j\} \in E} \exp(-l_{ij} x_j)}{\sum_{j: \{i,j\} \in E} \exp(-l_{ij})} \right)$

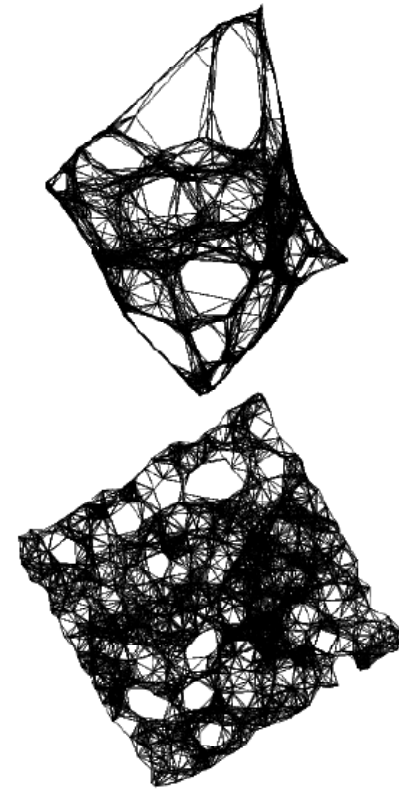
- ▶ compute third eigenvector v_3 ,
use v_2, v_3 as coordinates



[Fleischer & Pich]

Continued

- ▶ distributed optimization (spring model)
- ▶ alternative: *majorization*
- ▶ compute sequence of layouts $p^{(0)}, p^{(1)}, p^{(2)}, \dots$ with $E(p^{(0)}) \geq E(p^{(1)}) \geq E(p^{(2)}) \geq \dots$
 - ▶ solve linear equation $L^{(t+1)}p^{(t+1)} = L^{(t)}p^{(t)}$ in distributed manner



[Fleischer & Pich]

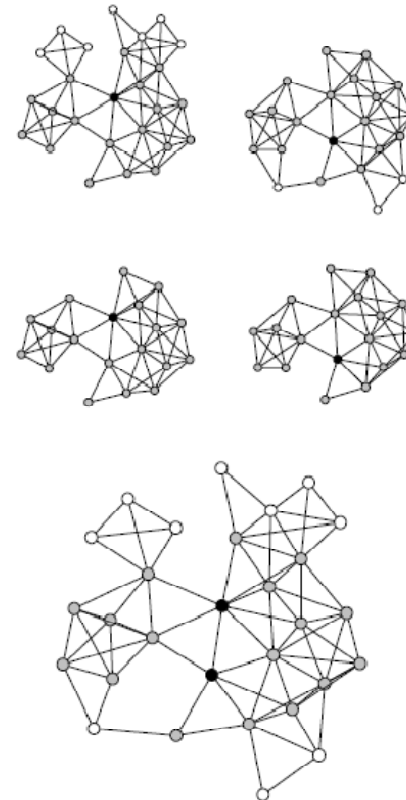


Heuristics: Shang et al.

Y. Shang, W. Ruml [7].

Improved MDS-based Localization, *IEEE Infocom*, 2004.

- ▶ compute a local map for each node (local MDS of the 2-hop neighborhood)
- ▶ merge local map patches into a global map (use incremental or binary-tree strategy)
- ▶ apply distributed optimization to the result

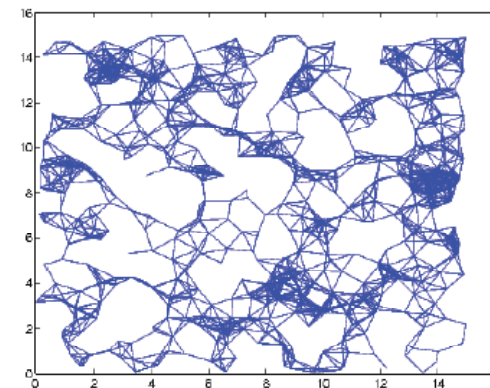
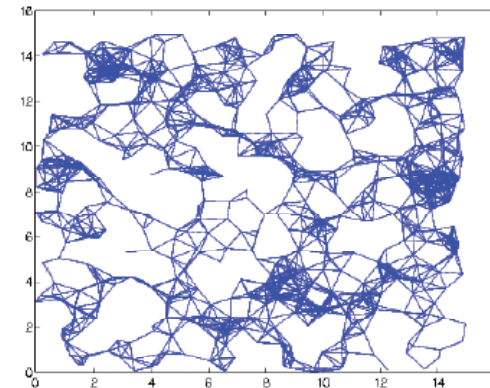


[Fleischer & Pich]

Heuristics: Bruck et al.

J. Bruck, J. Gao, A. Jiang [8]. **Localization and Routing in Sensor Networks by Local Angle Information**, *Mobile Ad Hoc Networking & Computing*, 2005.

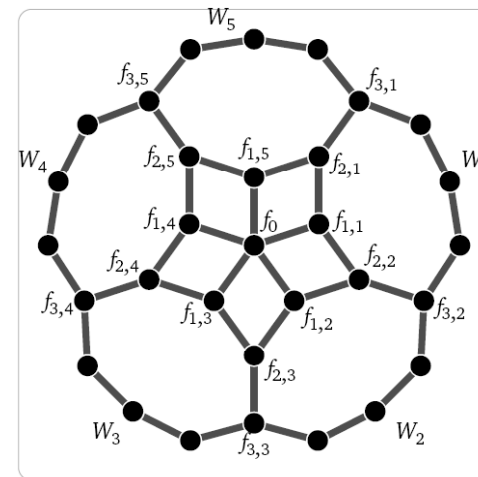
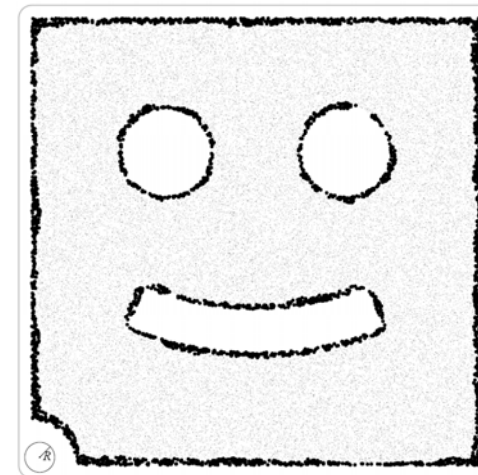
- ▶ Choose an edge e as x -axis to obtain absolute angles.
- ▶ Form an LP whose variables are the edge lengths $\ell(e)$.
- ▶ For all edges $0 \leq \ell(e) \leq 1$.
- ▶ For any cycle e_1, \dots, e_p :
$$\sum_{i=1}^p \ell(e_i) \cos \theta_i = 0$$
 and
$$\sum_{i=1}^p \ell(e_i) \sin \theta_i = 0$$
.
- ▶ Non-adjacent node pair constraints.
- ▶ Crossing-edge constraints.



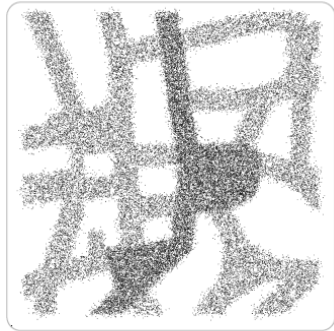
[Fleischer & Pich]

Boundary recognition

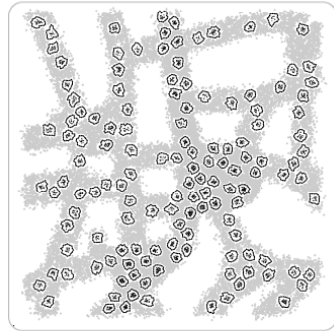
- Related problem, given a connectivity graph, what is the boundary?
- So far heuristics only, one heuristic uses the idea of a independent nodes; specifically, an interior (non-boundary) node sees enough (e.g. 5) independent neighbors which in turn have a ring around them; these are called “flowers”. Flowers can be grown and connected to compute the boundary.
- However, this is only a heuristic, and does not always work...



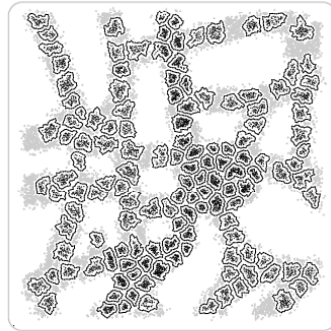
Boundary recognition



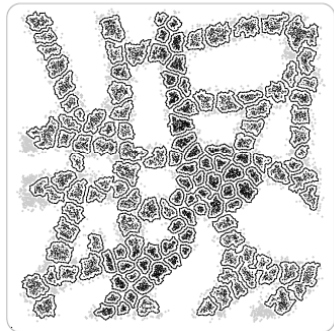
(a) Network



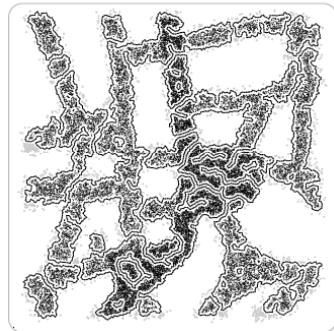
(b) Identified flowers



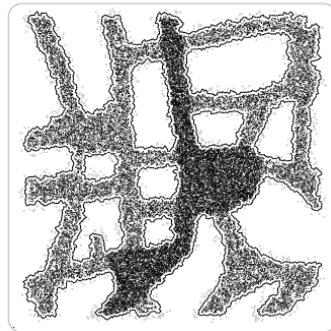
(c) Growing FGDs



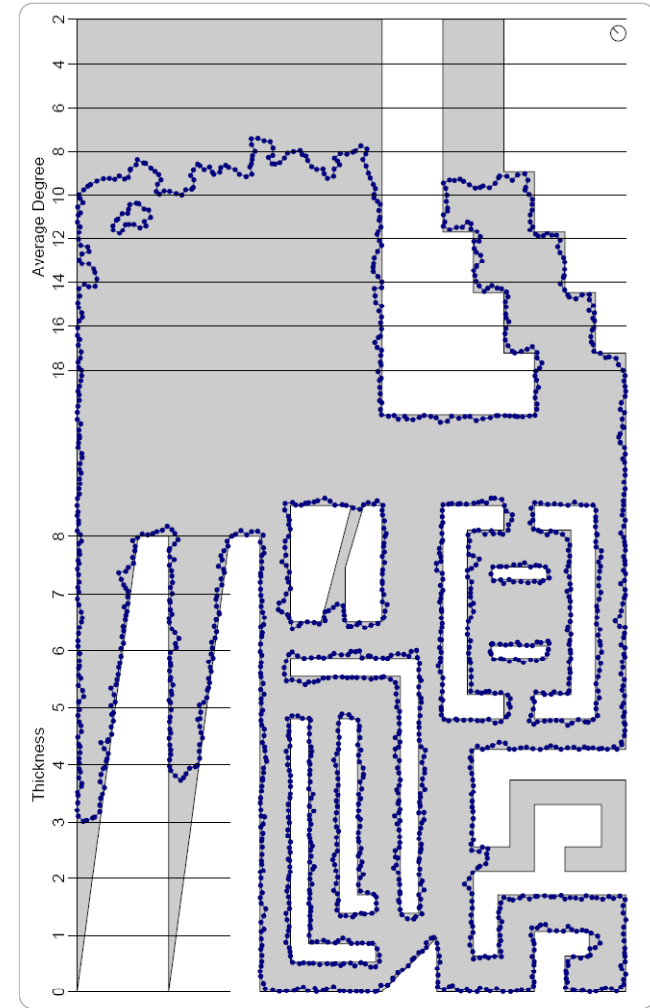
(d) FGDs beginning to merge



(e) Mergings FGDs

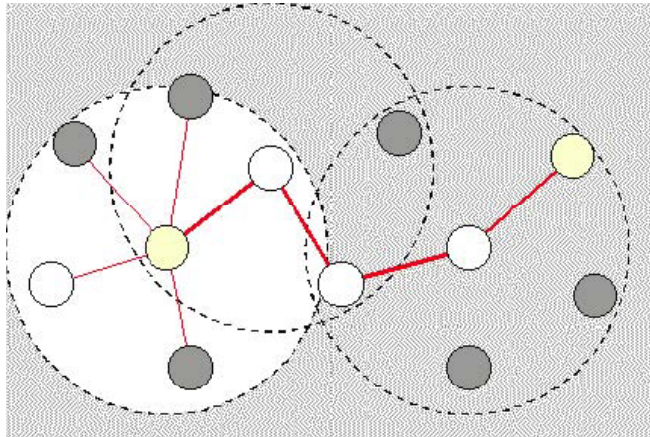


(f) Final state



Practical lessons

Theory



Practice



RSSI in sensor networks: good, but not for “reasonable” localization

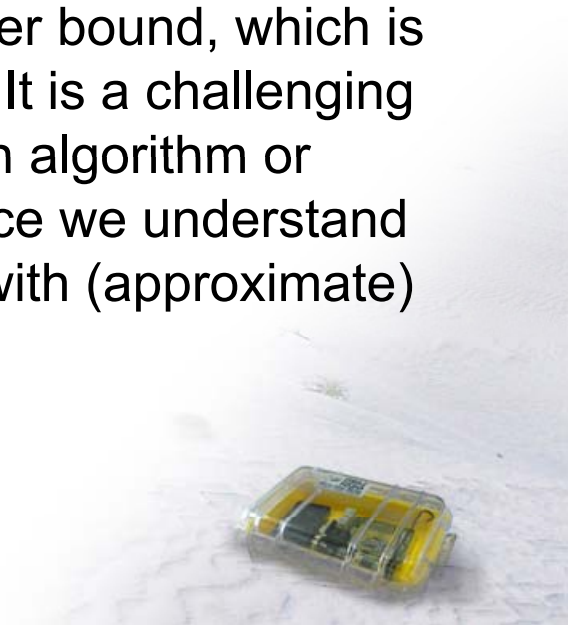
For exact indoor localization

- Buy special hardware (e.g., UWB)
- Place huge amount of short range anchors for single-hop localization



Open problem

- One tough open problem of this chapter obviously is the **UDG embedding** problem: Given the adjacency matrix of a unit disk graph, find positions for all nodes in the Euclidean plane such that the ratio between the maximum distance between any two adjacent nodes and the minimum distance between any two non-adjacent nodes is as small as possible.
- There is a large gap between the best known lower bound, which is a constant, and the polylogarithmic upper bound. It is a challenging task to either come up with a better approximation algorithm or prove a stronger (non-constant) lower bound. Once we understand this better, we can try networks with anchors, or with (approximate) distance/angle information.
- Generally, beyond GPS this area is in its infancy.



more...



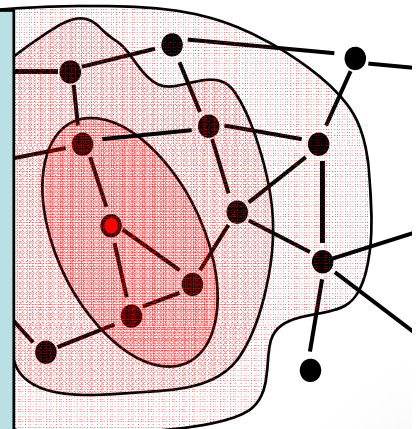
Global Optimization with Local Information?

- Towards a theory for understanding large-scale networks/systems

- Nodes in network
 - nodes must have access to local information

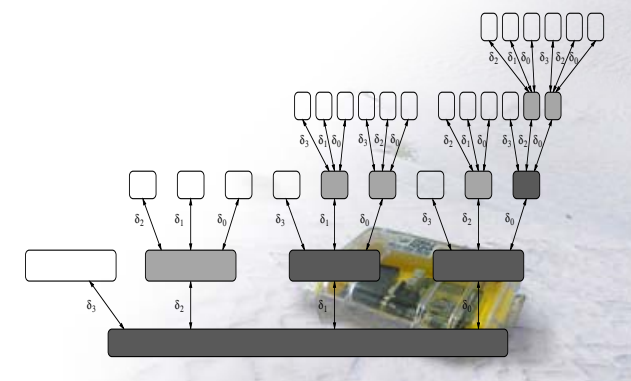
- We proved that traditional network models are insufficient
 - now we have a new paradigm
 - basis for understanding large-scale networks

We've seen this already!



- [Linial, SIAM J. Comput. 1995]
- [Kuhn, Meierwald, W, PODC 2004]
- [Gfeller, Moscari, PODC 2007]
-

Organization & dynamic systems

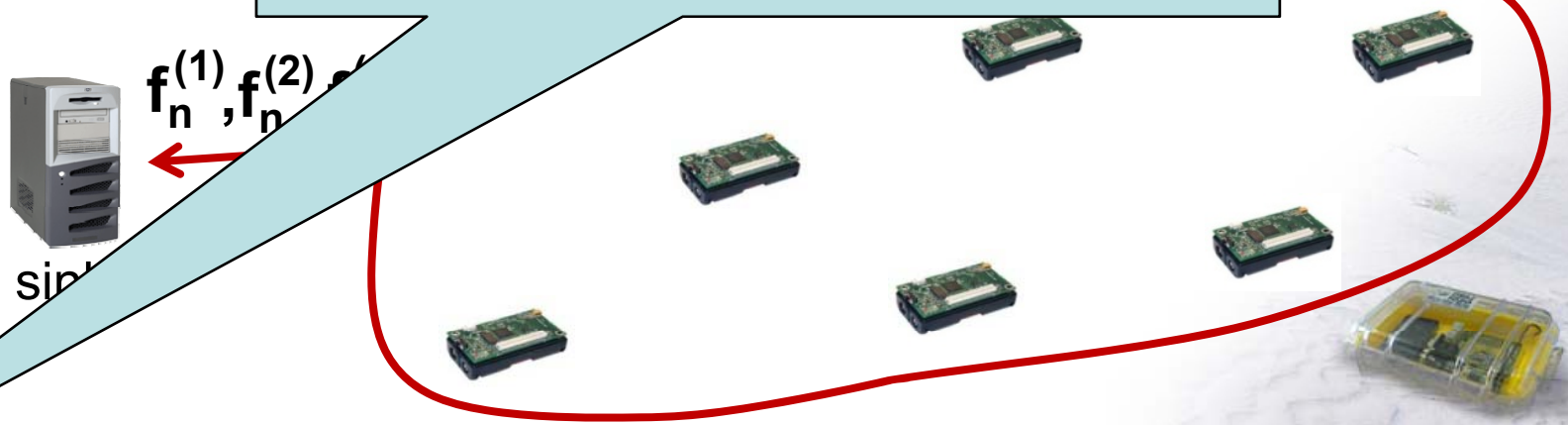


Data Gathering in Wireless Sensor Networks

- Data gathering & aggregation
 - Classic application of sensor networks
 - Sensor nodes periodically sense environment
 - Relevant

- Functional O
 - Sink period
 - At what ra

We've seen this already!



Summary

- Lower Bounds and Impossibility Results
 - Clock Synchronization
 - Distributed Selection / Median
 - Geo-Routing
 - Positioning
 - Local Algorithms
 - Data Gathering / Capacity



1 : 1

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!

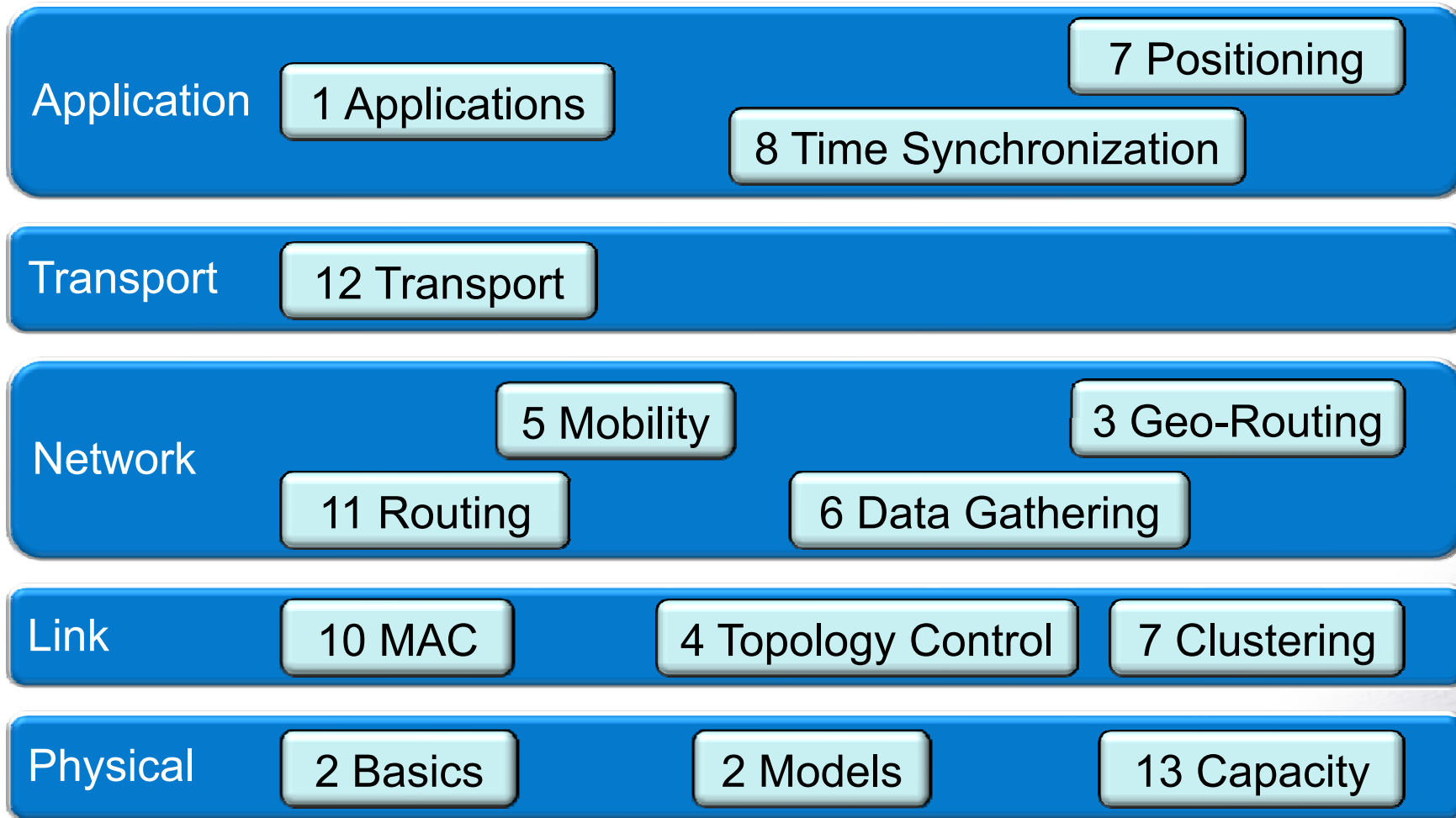


[Ali G]



Conclusions & Discussion

Some topics

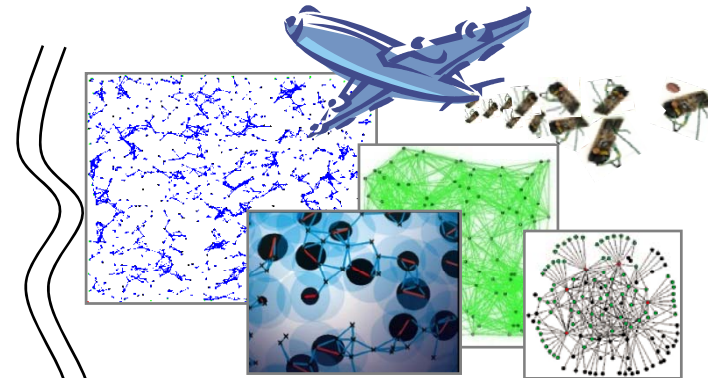
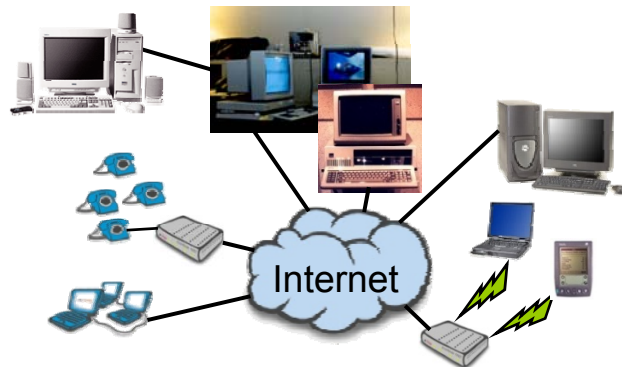


General Trend in Information Technology

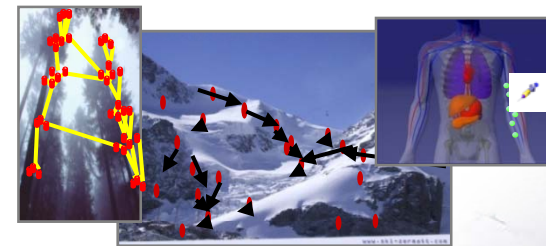


Centralized Systems

Networked Systems



Large-scale Distributed Systems



New Applications and System Paradigms

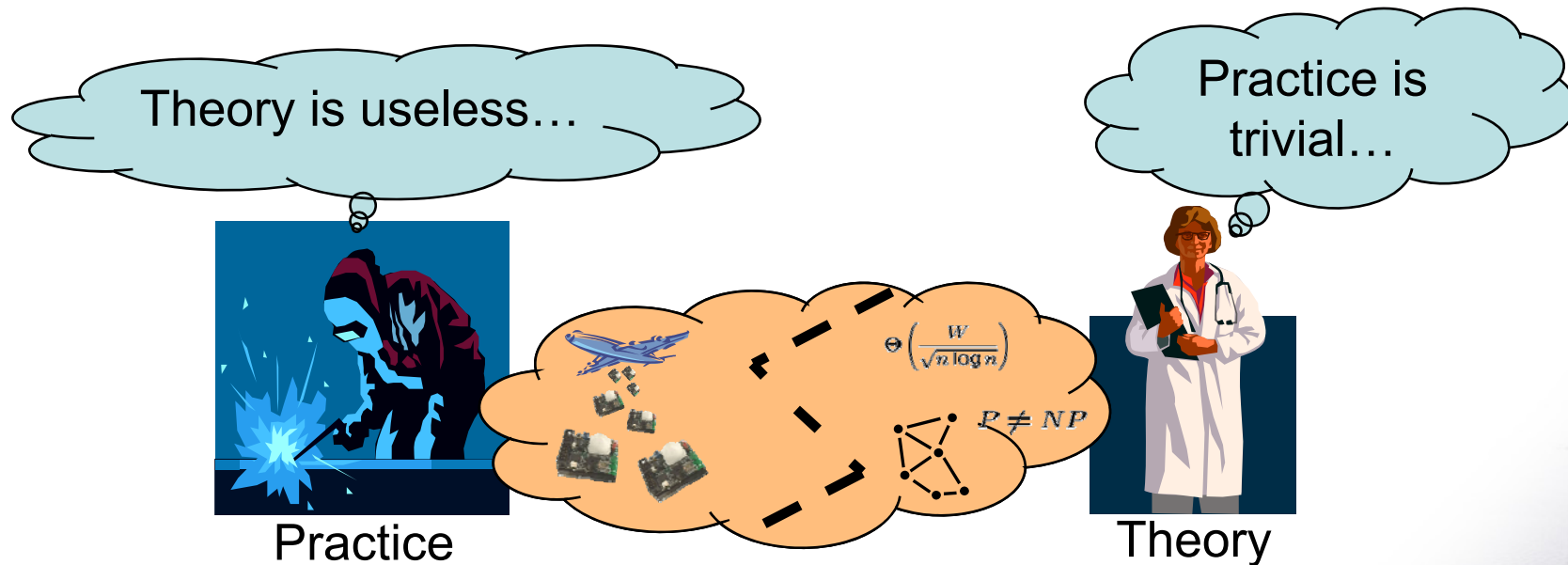
My Own Private View on Networking Research

| Class | Analysis | Communi- cation model | Node distribution | Other drawbacks | Popu- larity |
|---------------------|-------------------|-----------------------------|----------------------|-------------------------------|-----------------|
| Imple- mentation | Testbed | Reality | Reality(?) | “Too specific” | 5% |
| Heuristic | Simulation | UDG to SINR | Random, and more | Many...! (no benchmarks) | 80% |
| Scaling law | Theorem/ proof | SINR, and more | Random | Existential (no protocols) | 10% |
| Algorithm | Theorem/ proof | UDG, and more | Any (worst- case) | Worst-case unusual | 5% |



Conclusions

- We have seen **many stories about sensor networks**. These stories show that there still is quite a bit of research ahead of us.



- The stories also show that **theory and practice are not really connecting** well in this area. If even a group doing both cannot combine the theory and practice, one shall not be surprised that the two camps largely ignore each other.



Thank You!

Questions & Comments?