Data Gathering

Chapter 6

Rating

- Area maturity

- Practical importance

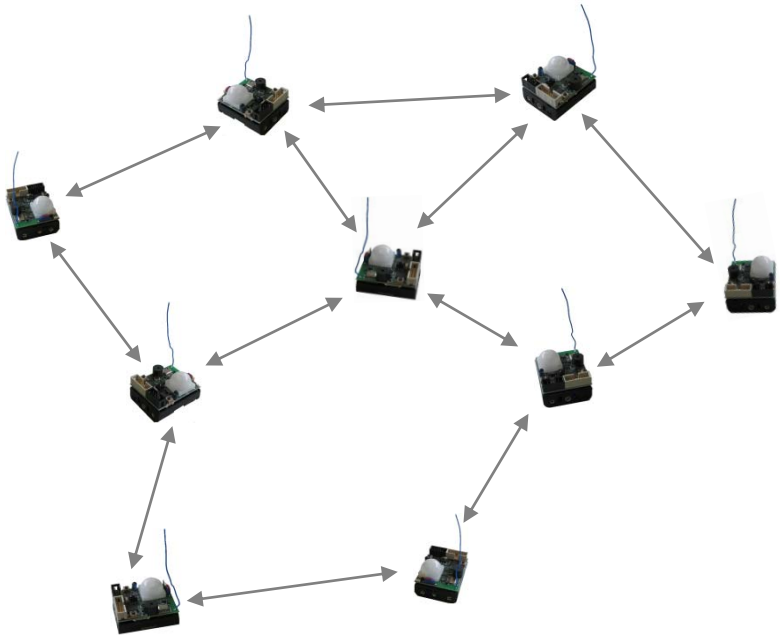
- Theoretical importance

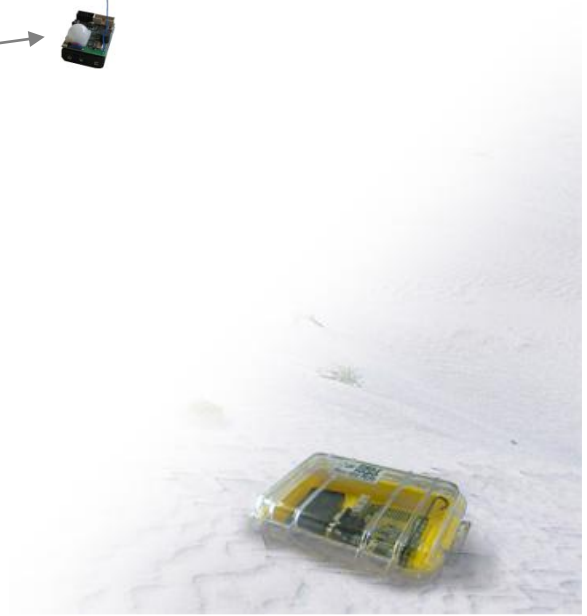
Overview

- Motivation
- Data gathering with coding
  - Self-coding
    - Excursion: Shallow Light Tree
  - Foreign coding
  - Multicoding
- Data gathering with aggregation
  - Max, Min, Average, Median, …
  - Universal data gathering tree
- Energy-efficient data gathering: Dozer

Sensor networks

- Sensor nodes
  - Processor & memory
  - Short-range radio
  - Battery powered
- Requirements
  - Monitoring geographic region
  - Unattended operation
  - Long lifetime
- Variants
  - Data gathering (continuous)
  - DB requests
  - Event detection



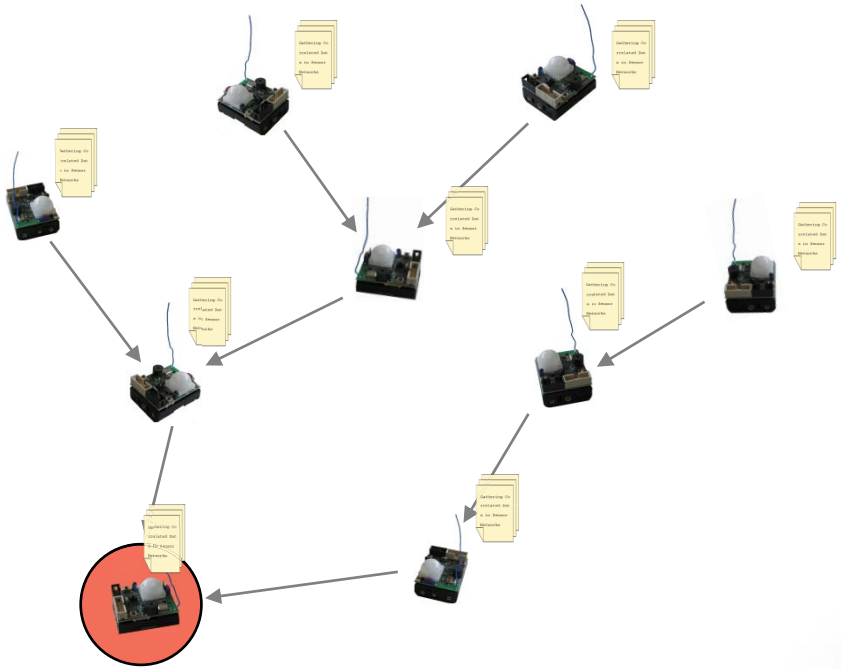


Data gathering

- All nodes produce relevant information about their vicinity periodically.
- Data is conveyed to an information sink for further processing.

➡ Routing scheme

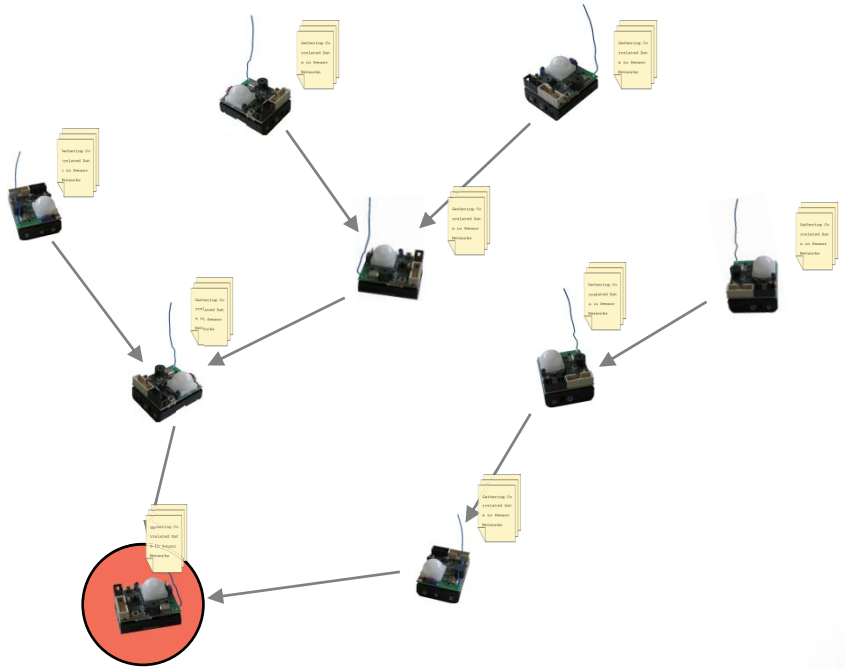
On which path is node u's data forwarded to the sink?



Time coding

- The simplest trick in the book: If the sensed data of a node changes not too often (e.g. temperature), the node only needs to send a new message when its data (significantly) changes.

- Improvement: Only send change of data, not actual data (similar to video codecs)



More than one sink?

- Use the anycast approach, and send to the closest sink.
- In the simplest case, a source wants to minimize the number of hops. To make anycast work, we only need to implement the regular distance-vector routing algorithm.
- However, one can imagine more complicated schemes where e.g. sink load is balanced, or even intermediate load is balanced.

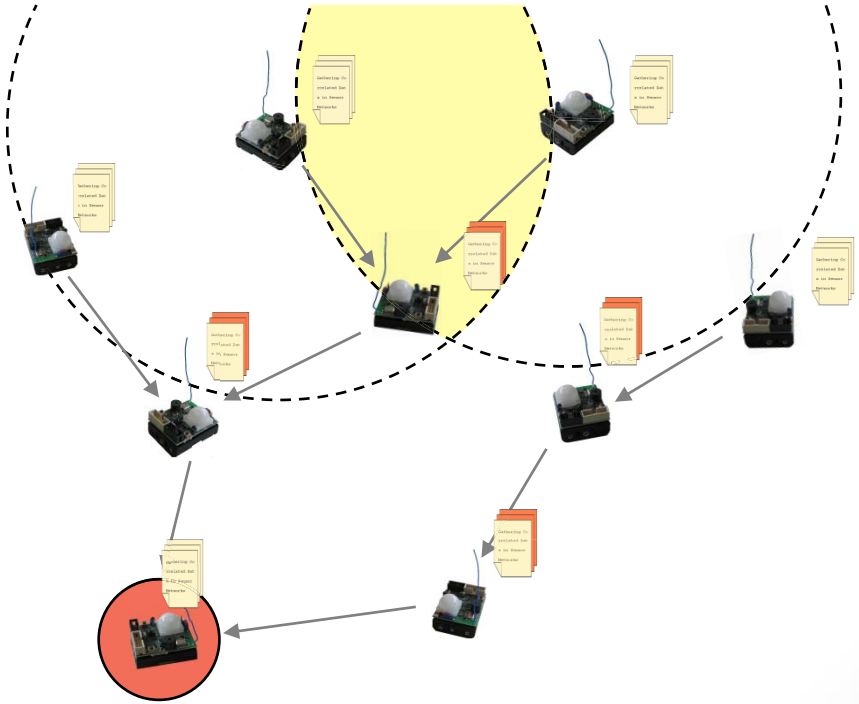
Correlated Data

- Different sensor nodes partially monitor the same spatial region.

  ➡ Data correlation

- Data might be processed as it is routed to the information sink.

  ➡ In-network coding

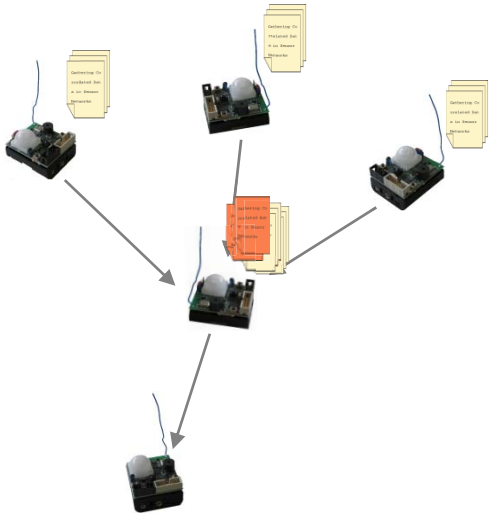


At which node is node u's data encoded?

Find a routing scheme and a coding scheme to deliver data packets from all nodes to the sink such that the overall energy consumption is minimal.

Coding strategies

- Multi-input coding
  - Exploit correlation among several nodes.
  - Combined aggregation of all incoming data.



➡ Recoding at intermediate nodes

➡ Synchronous communication model

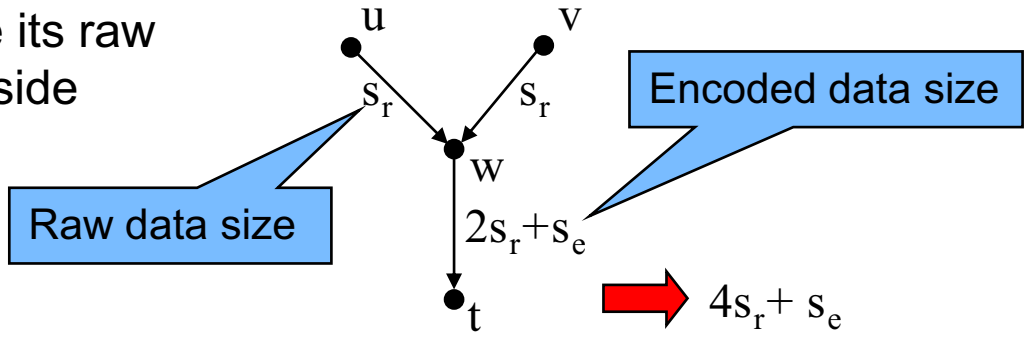
- Single-input coding
  - Encoding of a nodes data only depends on the side information of one other node.

➡ No recoding at intermediate nodes

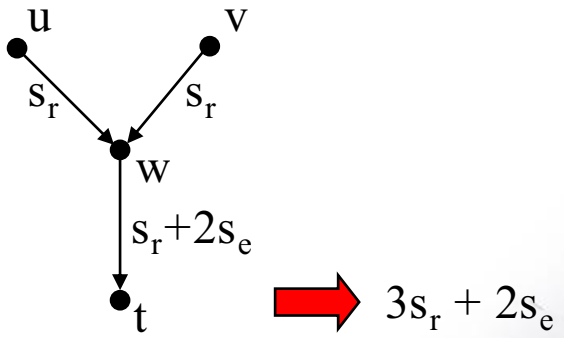
➡ No waiting for belated information at intermediate nodes

Single-input coding

- Self-coding
  - A node can only encode its raw data in the presence of side information.



- Foreign coding
  - A node can use its raw data to encode data it is relaying.



Self-coding

- The cost of an optimal topology

$$c_{opt} = \sum_{u \in B} \left(s_r \cdot \mathsf{ST}(S_u, u, t) + \sum_{v \in S_u} s_e \cdot \mathsf{SP}(v, t) \right).$$

Set of nodes that encode with data from u

Set of nodes with no side information

Steiner tree

Shortest path

- Two ways to lower-bound this equation:
  - $c_{opt} \geq \sum_{u \in V} s_e \cdot \mathsf{SP}(u, t)$
  - $c_{opt} \geq s_r \cdot c(\mathsf{MST})$

Algorithm

- LEGA (Low Energy Gathering Algorithm)
- Based on the shallow light tree (SLT)
- Compute SLT rooted at the sink t.
- The sink t transmits its packet p_t
- Upon reception of a data packet p_j at node v_i
  - Encode p_i with $p_j \rightarrow p_i^j$
  - Transmit p_i^j to the sink t
  - Transmit p_i to all children

Size = s_r

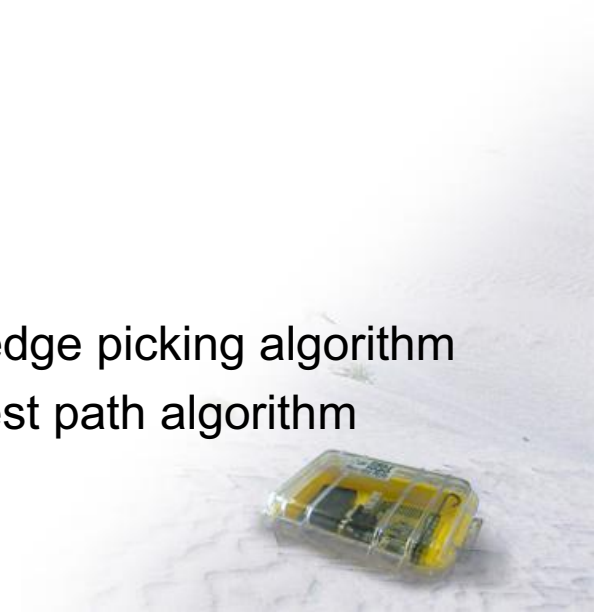
Size = s_e

Excursion: Shallow-Light Tree (SLT)

- Introduced by [Awerbuch, Baratz, Peleg, PODC 1990]
- Improved by [Khuller, Raghavachari, Young, SODA 1993]
  - new name: Light-Approximate-Shortest-Path-Tree (LAST)

- Idea: Construct a spanning tree for a given root r that is both a MST-approximation as well as a SPT-approximation for the root r. In particular, for any $\gamma > 0$
  - $c(\mathrm{SLT}) \leq (1 + \sqrt{2}/\gamma) \cdot c(\mathrm{MST})$
  - $d_{SLT}(v_i, r) \leq (1 + \sqrt{2}\gamma) \cdot \mathrm{SP}(v_i, r)$

- Remember:
  - MST: Easily computable with e.g. Prim's greedy edge picking algorithm
  - SPT: Easily computable with e.g. Dijkstra's shortest path algorithm



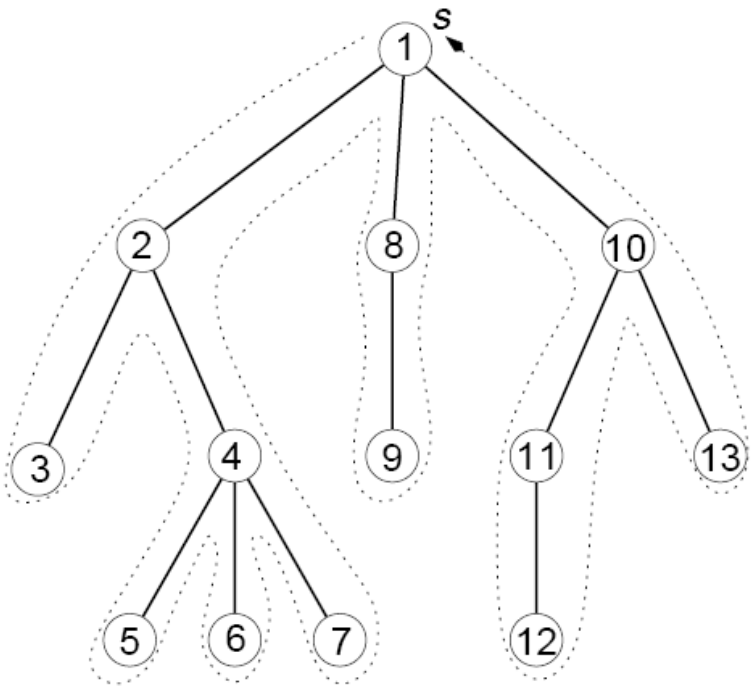
MST vs. SPT

- Is a good SPT not automatically a good MST (or vice versa)?

Result & Preordering

- Main Theorem: Given an $\alpha > 1$, the algorithm returns a tree T rooted at r such that all shortest paths from r to u in T have cost at most α the shortest path from r to u in the original graph (for all nodes u). Moreover the total cost of T is at most $\beta = 1+2/(\alpha-1)$ the cost of the MST.

- We need an ingredient: A preordering of a rooted tree is generated when ordering the nodes of the tree as visited by a depth-first search algorithm.

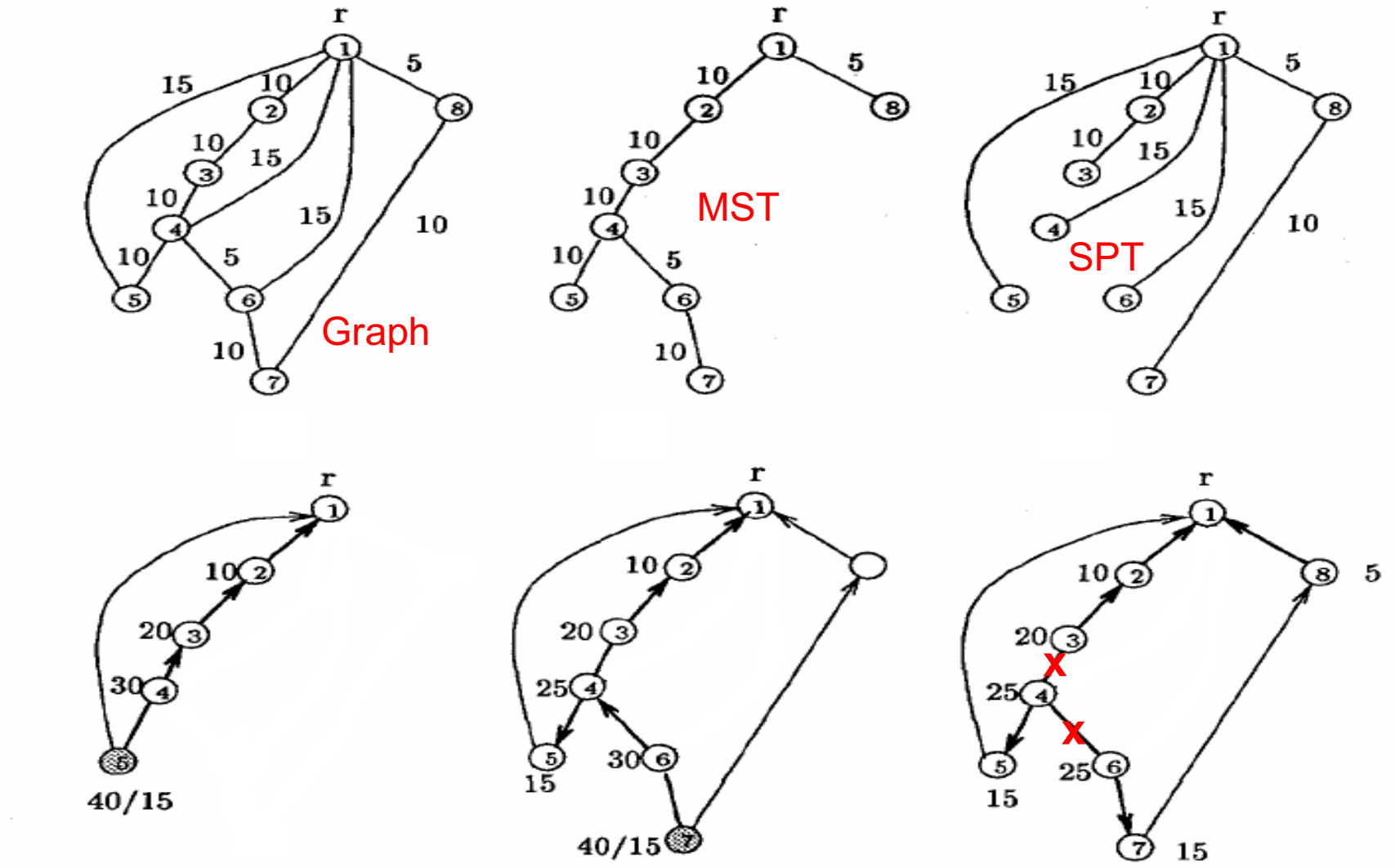


The SLT Algorithm

1. Compute MST H of Graph G;
2. Compute all shortest paths (SPT) from the root r.
3. Compute preordering of MST with root r.
4. For all nodes v in order of their preordering do
   - Compute shortest path from r to u in H. If the cost of this shortest path in H is more than a factor α more than the cost of the shortest path in G, then just add the shortest path in G to H.
5. Now simply compute the SPT with root r in H.

- Sounds crazy… but it works!

An example, $\alpha = 2$



Proof of Main Theorem

- The SPT α-approximation is clearly given since we included all necessary paths during the construction and in step 5 only removed edges which were not in the SPT.

- We need to show that our final tree is a β-approximation of the MST. In fact we show that the graph H before step 5 is already a β-approximation!

- For this we need a little helper lemma first…

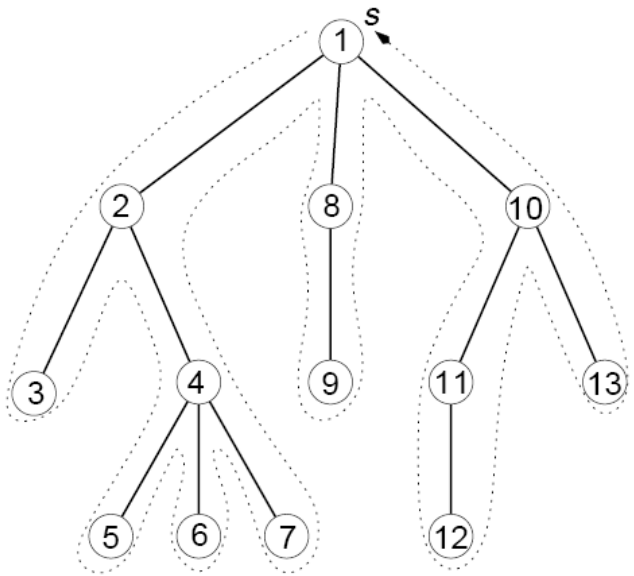
A preordering lemma

- Lemma: Let T be a rooted spanning tree, with root r, and let z_0, z_1, …, z_k be arbitrary nodes of T in preorder. Then,

$$\sum_{i=1}^{k} d_T(z_{i-1}, z_i) \leq 2 \cdot cost(T).$$

- "Proof by picture": Every edge is traversed at most twice.

- Remark: Exactly like the 2-approximation algorithm for metric TSP.



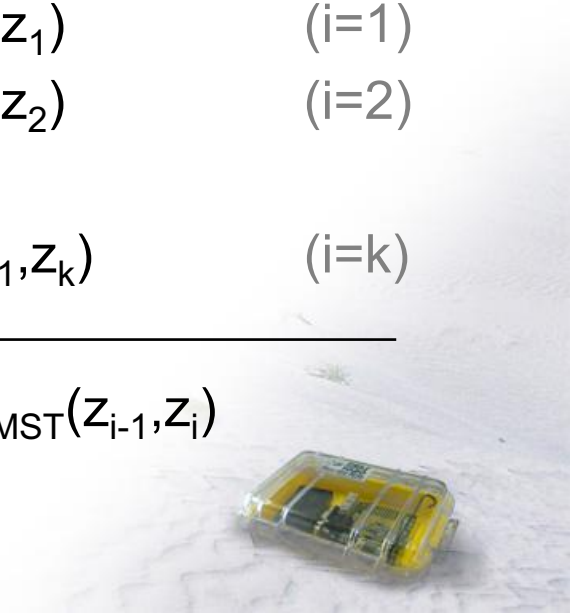
Proof of Main Theorem (2)

- Let $z_1, z_2, \ldots, z_k$ be the set of k nodes for which we added their shortest paths to the root r in the graph in step 4. In addition, let z_0 be the root r. The node z_i can only be in the set if (for example) $d_G(r,z_{i-1}) + d_{MST}(z_{i-1},z_i) > \alpha d_G(r,z_i)$, since the shortest path (r,z_{i-1}) and the path on the MST (z_{i-1},z_i) are already in H when we study z_i.

- We can rewrite this as $\alpha d_G(r,z_i) - d_G(r,z_{i-1}) < d_{MST}(z_{i-1},z_i)$. Summing up:

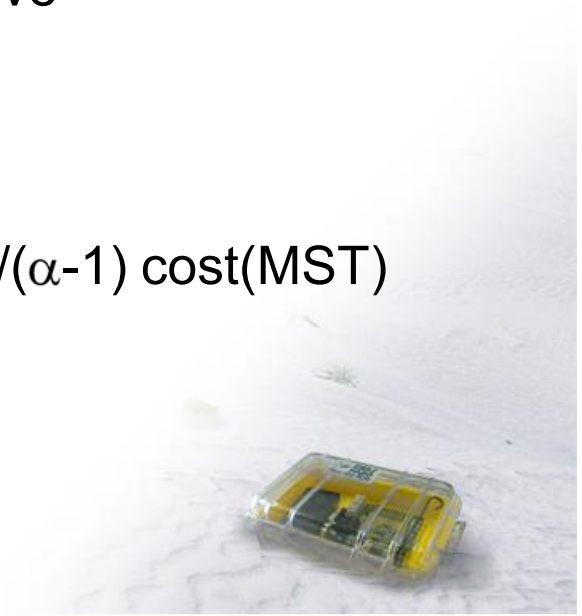
$$\begin{aligned}
&\alpha d_G(r,z_1) - d_G(r,z_0) && < && d_{MST}(z_0,z_1) && (i=1)\\
&\alpha d_G(r,z_2) - d_G(r,z_1) && < && d_{MST}(z_1,z_2) && (i=2)\\
&\quad \ldots && \ldots && \ldots &&\\
&\alpha d_G(r,z_k) - d_G(r,z_{k-1}) && < && d_{MST}(z_{k-1},z_k) && (i=k)
\end{aligned}$$

$$\Sigma_{i=1\ldots k}(\alpha\text{-}1)\, d_G(r,z_i) \quad + \cancel{d_G(r,z_k)} \quad < \quad \Sigma_{i=1\ldots k}\, d_{MST}(z_{i-1},z_i)$$



Proof of Main Theorem (3)

- In other words, $(\alpha\text{-}1)\ \Sigma_{i=1\ldots k}\, d_G(r,z_i) < \Sigma_{i=1\ldots k}\, d_{MST}(z_{i-1},z_i)$
- All we did in our construction of H was to add exactly at most the cost $\Sigma_{i=1\ldots k}\, d_G(r,z_i)$ to the cost of the MST. In other words, $\text{cost}(H) \leq \text{cost}(MST) + \Sigma_{i=1\ldots k}\, d_G(r,z_i)$.
- Using the inequality on the top of this slide we have $\text{cost}(H) < \text{cost}(MST) + 1/(\alpha\text{-}1)\ \Sigma_{i=1\ldots k}\, d_{MST}(z_{i-1},z_i)$.
- Using our preordering lemma we have $\text{cost}(H) \leq \text{cost}(MST) + 1/(\alpha\text{-}1)\ 2\text{cost}(MST) = 1+2/(\alpha\text{-}1)\ \text{cost}(MST)$
- That's exactly what we needed: $\beta = 1+2/(\alpha\text{-}1)$.

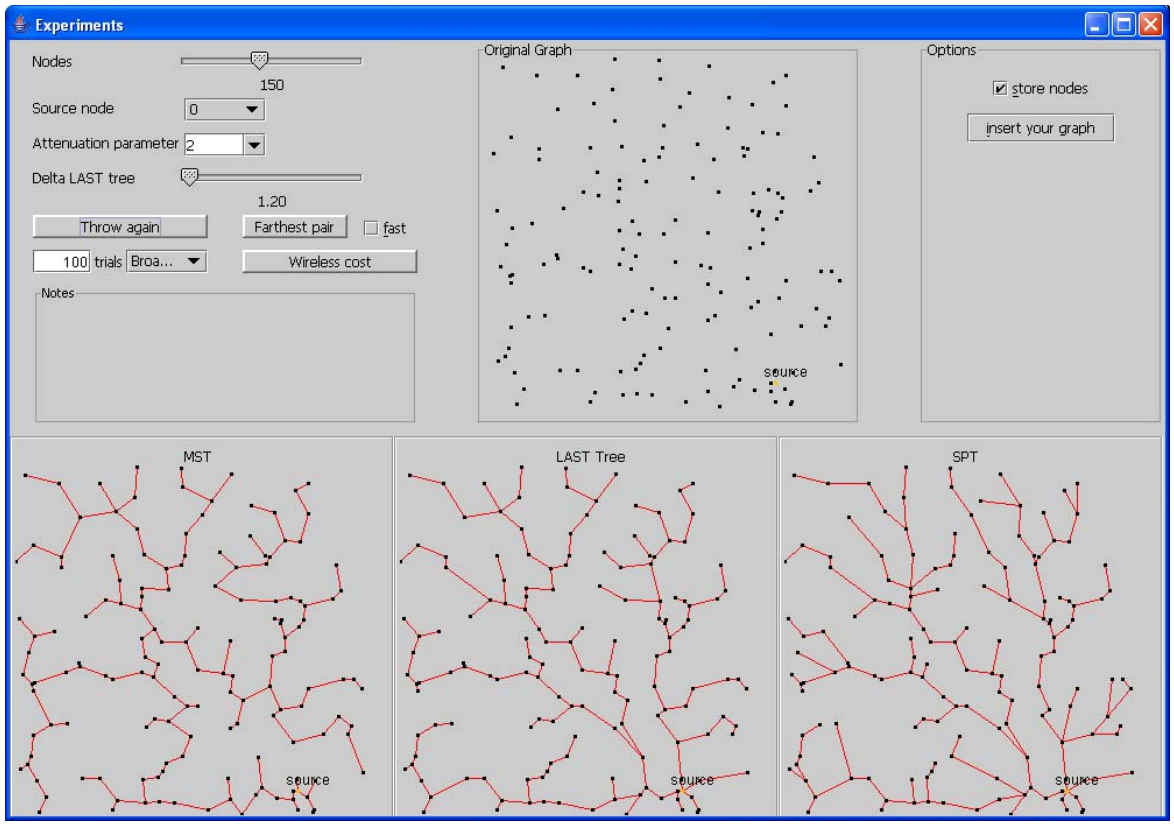


How the SLT can be used

- The SLT has many applications in communication networks.

- Essentially, it bounds the cost of unicasting (using the SPT) and broadcasting (using the MST).

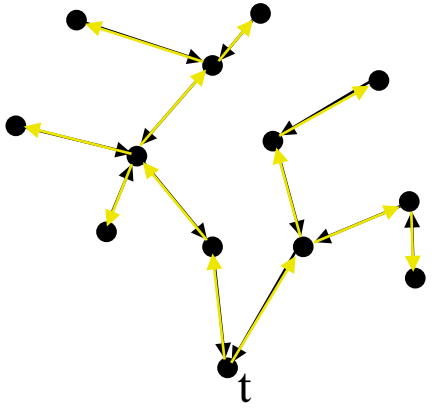
- Remark: If you use $\alpha = 1 + \sqrt{2}$, then $\beta = 1+2/(\alpha-1) = \alpha$.



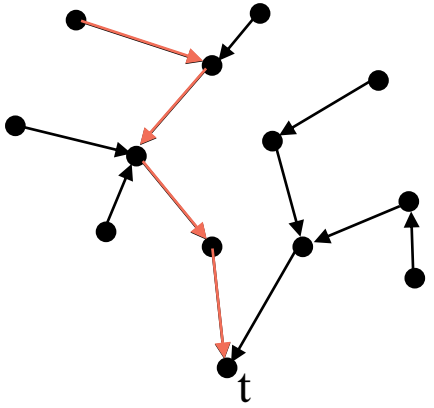
[www.dia.unisa.it/~ventre]

Analysis of LEGA

Theorem: LEGA achieves a $2(1+\sqrt{2})$-approximation of the optimal topology. (We use $\alpha = 1+\sqrt{2}$.)



$\Rightarrow s_r \cdot c(\mathsf{SLT})$

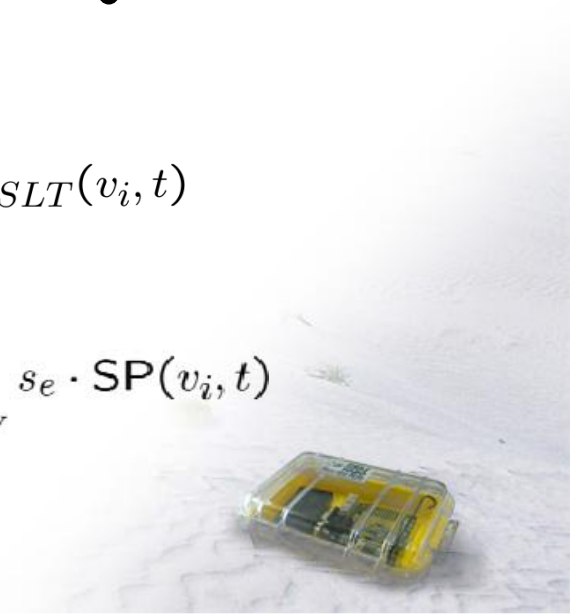


$\Rightarrow \sum_{v_i \in V} s_e \cdot d_{SLT}(v_i, t)$

$$c_{LEGA} \leq s_r \cdot (1+\sqrt{2})c(\mathsf{MST}) + (1+\sqrt{2}) \sum_{v_i \in V} s_e \cdot \mathsf{SP}(v_i, t)$$

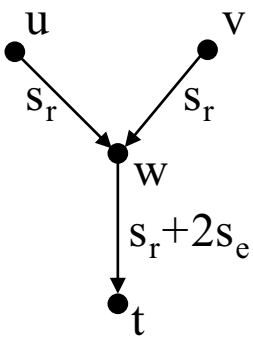
$$\leq 2(1+\sqrt{2})c_{opt}$$

Slide 6/11



Foreign coding

- MEGA (Minimum-Energy Gathering Algorithm)
  - Superposition of two tree constructions.



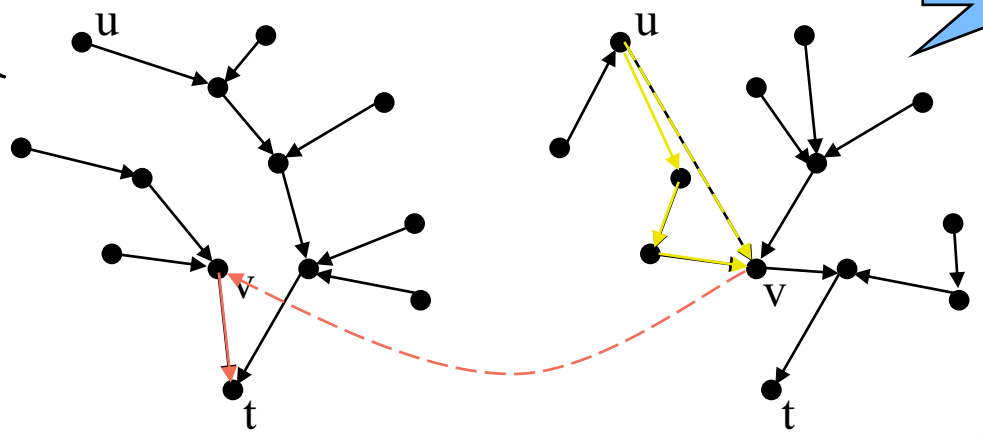
- Compute the shortest path tree (SPT) rooted at t.
- Compute a coding tree.

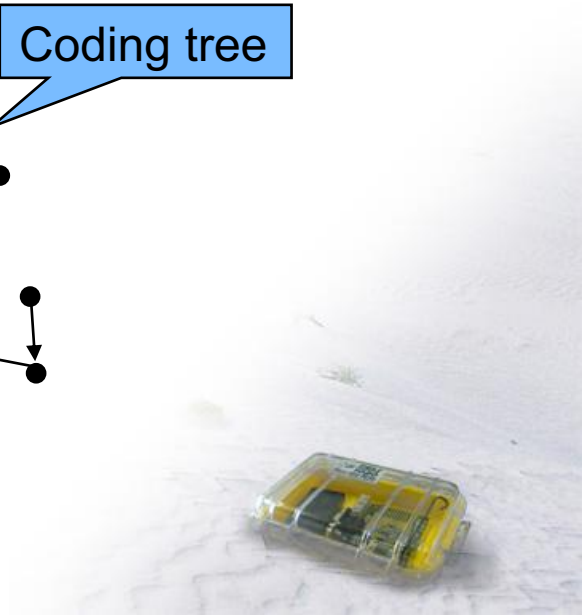
Encoding must not result in cyclic dependencies.

  - Determine for each node u a corresponding encoding node v.

SPT

Coding tree





Coding tree construction

- Build complete directed graph
- Weight of an edge $e=(v_i,v_j)$

Cost from v_j to the sink t.

$$w(e) = s_i \cdot \mathrm{SP}(v_i, v_j) + s_i^j \cdot \mathrm{SP}(v_j, t)$$

Cost from v_i to the encoding node v_j.

Number of bits when encoding v_i's info at v_j

- Compute a directed minimum spanning tree (arborescence) of this graph. (This is not trivial, but possible.)

Theorem: MEGA computes a minimum-energy data gathering topology for the given network.

All costs are summarized in the edge weights of the directed graph.

Summary

- Self-coding:
  - The problem is NP-hard [Cristescu et al, INFOCOM 2004]
  - LEGA uses the SLT and gives a $2(1+\sqrt{2})$-approximation.
  - Attention: We assumed that the raw data resp. the encoded data always needs s_r resp. s_e bits (no matter how far the encoding data is!). This is quite unrealistic as correlation is usually regional.

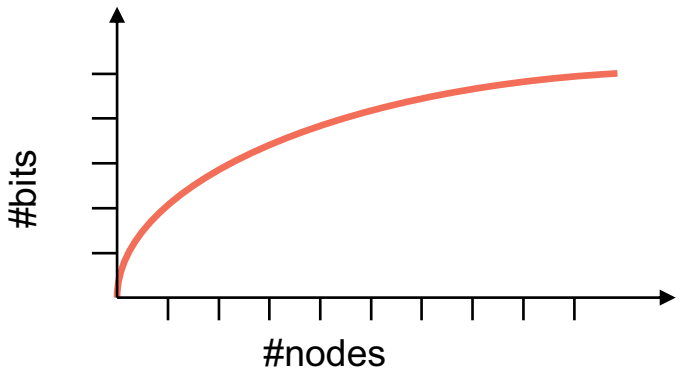
- Foreign coding
  - The problem is in P, as computed by MEGA.

- What if we allow both coding strategies at the same time?
- What if multicoding is still allowed?

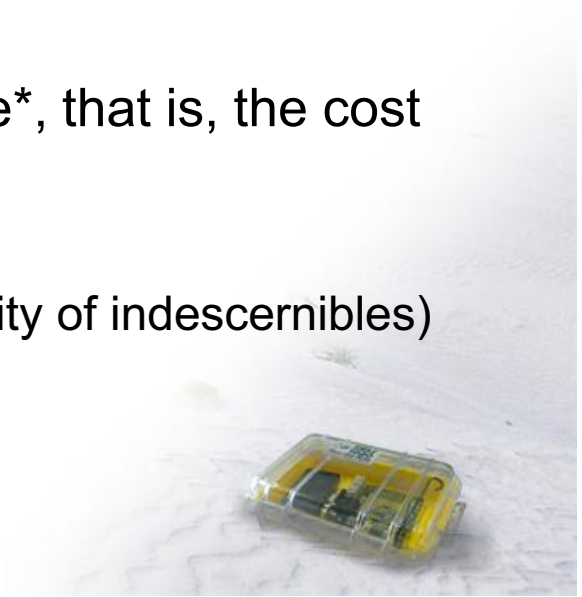
Multicoding

- Hierarchical matching algorithm [Goel & Estrin SODA 2003].

- We assume to have concave, non-decreasing aggregation functions. That is, to transmit data from k sources, we need f(k) bits with f(0)=0, $f(k) \geq f(k-1)$, and $f(k+1)/f(k) \leq f(k)/f(k-1)$.



- The nodes of the network must be a metric space*, that is, the cost of sending a bit over edge (u,v) is c(u,v), with
  - Non-negativity: $c(u,v) \geq 0$
  - Zero distance: c(u,u) = 0 (*we don't need the identity of indescernibles)
  - Symmetry: c(u,v) = c(v,u)
  - Triangle inequality: $c(u,w) \leq c(u,v) + c(v,w)$

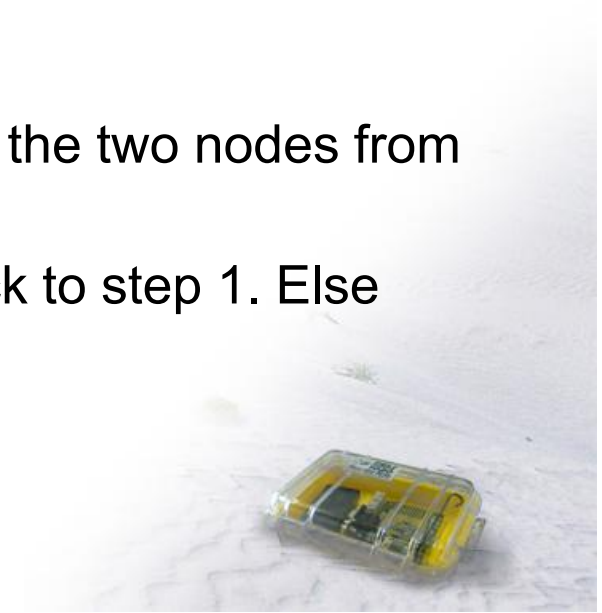


The algorithm

- Remark: If the network is not a complete graph, or does not obey the triangle inequality, we only need to use the cost of the shortest path as the distance function, and we are fine.

- Let S be the set of source nodes. Assume that S is a power of 2. (If not, simply add copies of the sink node until you hit the power of 2.) Now do the following:

1. Find a min-cost perfect matching in S.
2. For each of the matching edges, remove one of the two nodes from S (throw a regular coin to choose which node).
3. If the set S still has more than one node, go back to step 1. Else connect the last remaining node with the sink.



The result

- Theorem: For any concave, non-decreasing aggregation function f, and for [optimal] total cost C[*], the hierarchical matching algorithm guarantees

$$E\left[\max_{\forall f} \frac{C(f)}{C^*(f)}\right] \leq 1 + \log k.$$

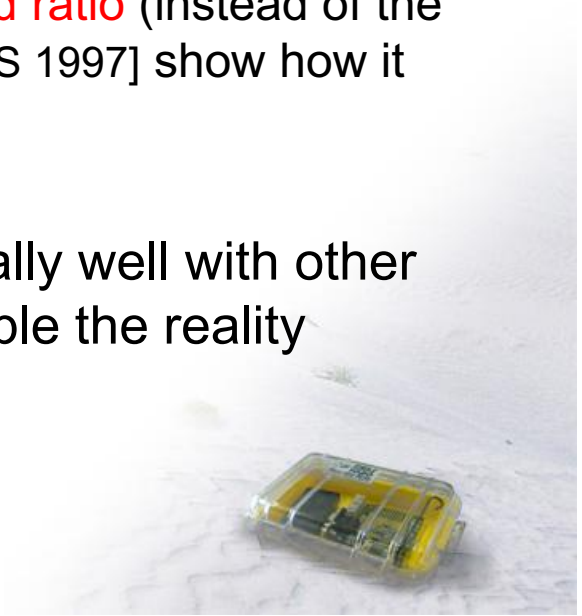
- That is, the expectation of the worst cost overhead is logarithmically bounded by the number of sources.

- Proof: Too intricate to be featured in this lecture.

Remarks

- For specific concave, non-decreasing aggregation functions, there are simpler solutions.
  - For f(x) = x the SPT is optimal.
  - For f(x) = const (with the exception of f(0) = 0), the MST is optimal.
  - For anything in between it seems that the SLT again is a good choice.
  - For any a priori known f one can use a deterministic solution by [Chekuri, Khanna, and Naor, SODA 2001]
  - If we only need to minimize the maximum expected ratio (instead of the expected maximum ratio), [Awerbuch and Azar, FOCS 1997] show how it works.

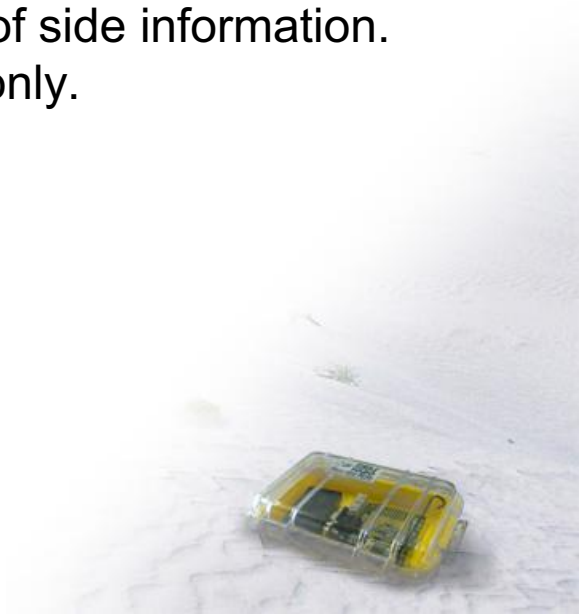
- Again, sources are considered to aggregate equally well with other sources. A correlation model is needed to resemble the reality better.



Other work using coding

- LEACH [Heinzelman et al. HICSS 2000]: randomized clustering with data aggregation at the clusterheads.
  - Heuristic and simulation only.
  - For provably good clustering (see chapter on clustering).

- Correlated data gathering [Cristescu et al. INFOCOM 2004]:
  - Coding with Slepian-Wolf
  - Distance independent correlation among nodes.
  - Encoding only at the producing node in presence of side information.
  - Same model as LEGA, but heuristic & simulation only.
  - NP-hardness proof for this model.



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>]
```

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?“

What cannot be computed using these functions?

Aggregation Model

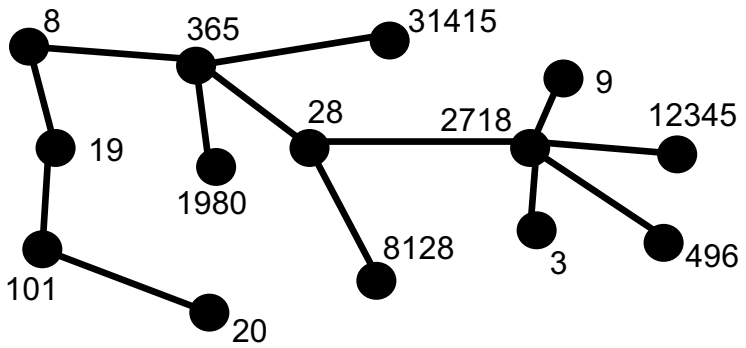
How difficult is it to compute these aggregation primitives?

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 $D \leq 2D_G$)°
- Asynchronous model of communication.
- All nodes hold a single element. °
- 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!

Slowest message arrives after 1 time unit!

→ *Distributive* (sum, count...) and *algebraic* (plus, minus...) functions are easy to compute:
Use a simple *flooding-echo* procedure → ***convergecast***!

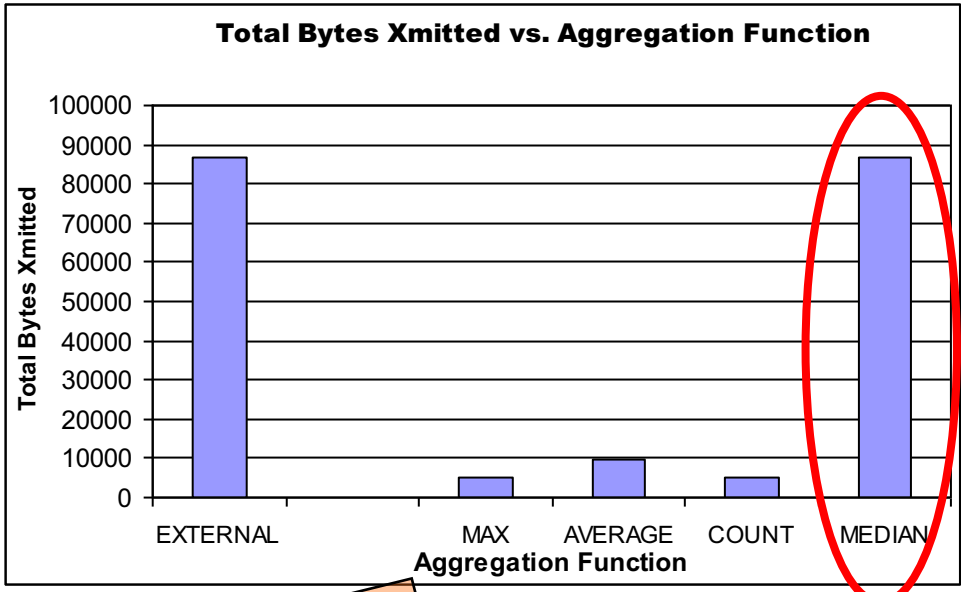
Time complexity: $\Theta(D)$

What about holistic functions (such as k-selection)???
Is it (really) harder...?
Impossible to perform in-network aggregation*?*

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

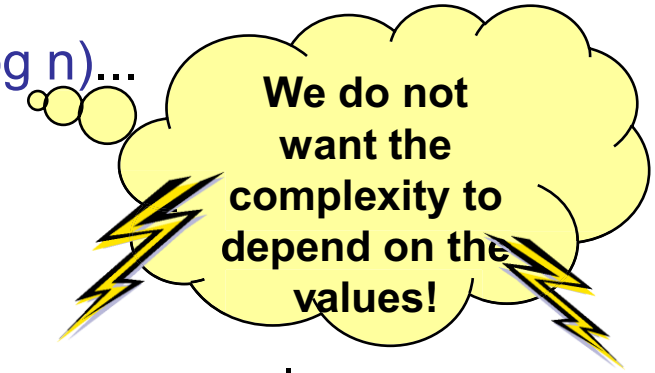
Is it difficult?

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

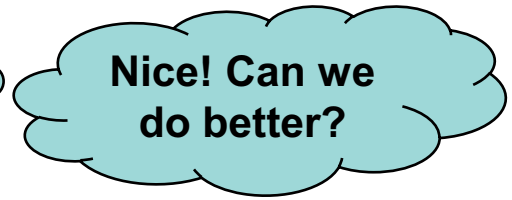
A straightforward idea: Use the sequential algorithm by Blum et al. also in a distributed setting → Time Complexity: $O(D \cdot n^{0.9114})$.

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!



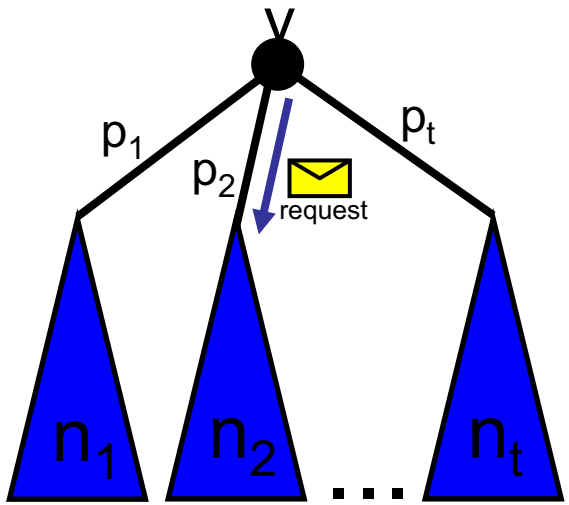
Randomized Algorithm

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

How is this done?

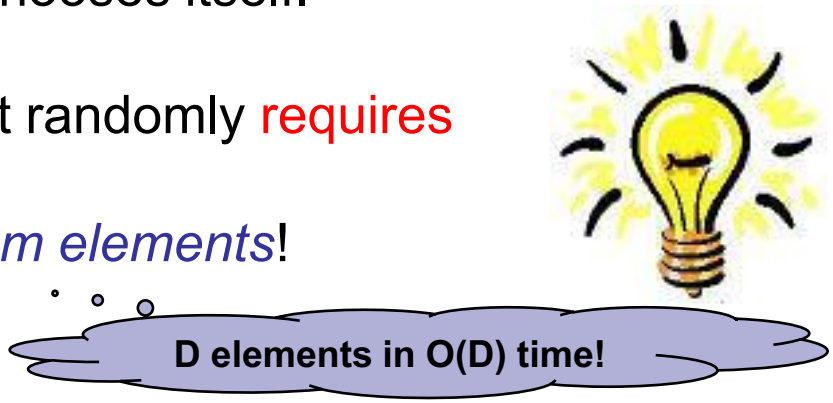
→ Assuming that all nodes know the sizes $n_1,...,n_t$ of the subtrees rooted at their children $v_1,...,v_t$, the request is forwarded to node v_i with probability:

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



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

Key observation: Choosing an element randomly requires O(D) time!
Use pipe-lining to select *several random elements*!



Randomized Algorithm

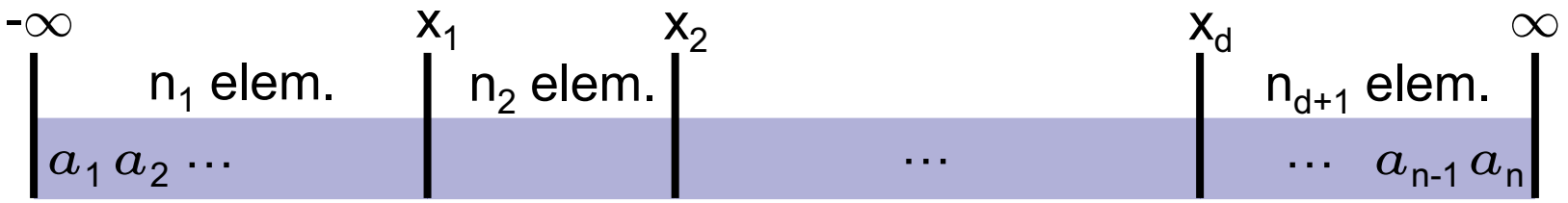
Our algorithm also operates in phases → The set of *candidates* decreases in each phase!

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 $x_1,...,x_d$ 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.

We get the following result:

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

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

More on that later...

We further proved a time lower bound of $\Omega(D \cdot \log_D n)$.
→This simple randomized algorithm is asymptotically optimal!

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

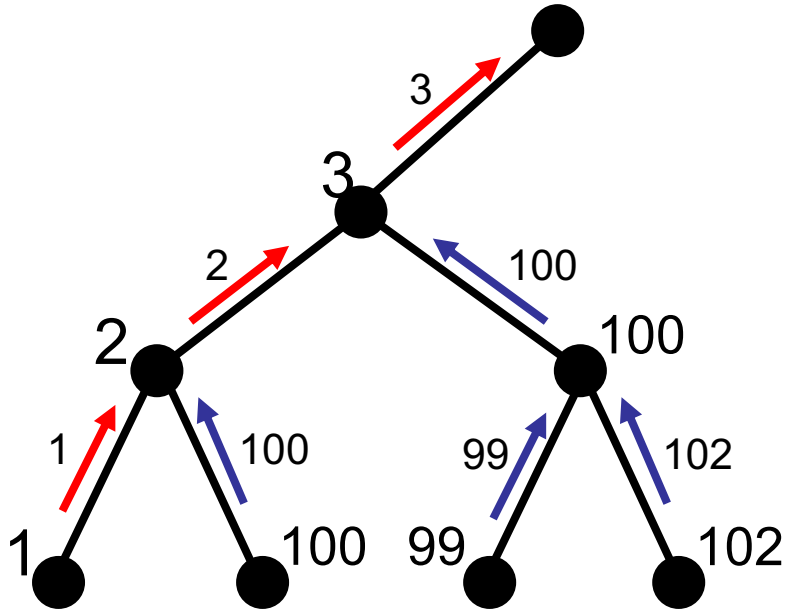
Deterministic Algorithm

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.)



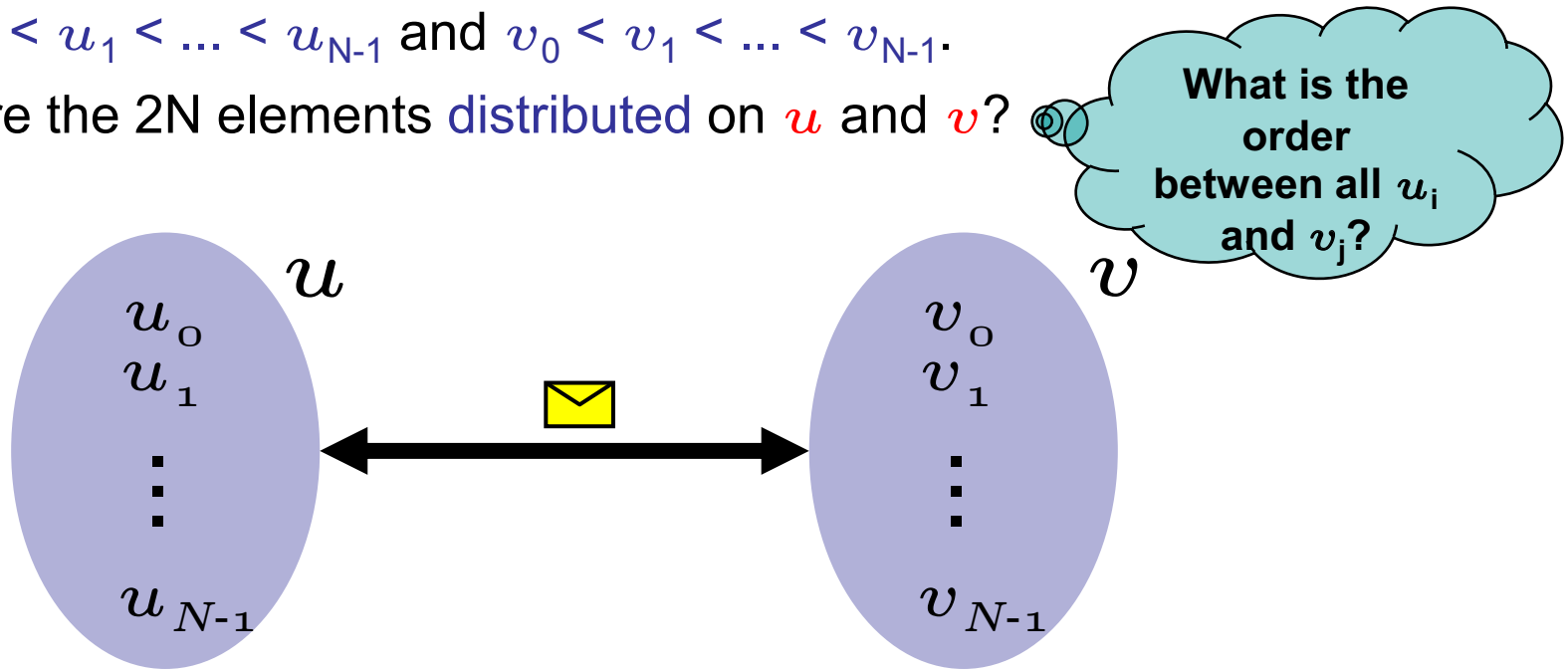
Lower Bound

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

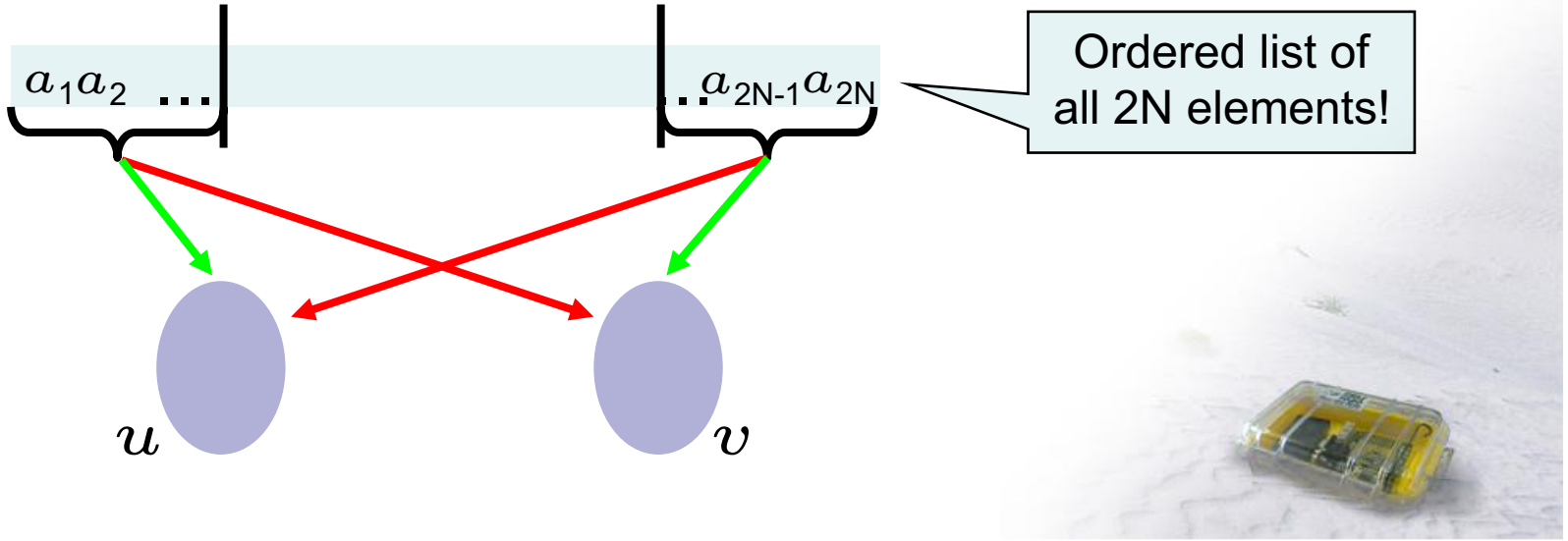
How are the 2N elements distributed on u and v?



Lower Bound

Assume $N = 2^b$. We use b independent Bernoulli variables $X_0,...,X_{b-1}$ to distribute the elements!
If $X_{b-1} = 0$ → 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.

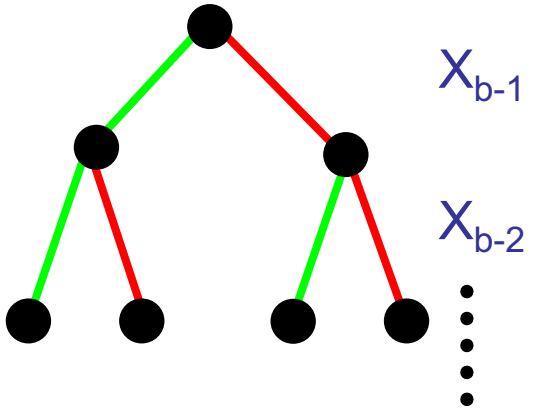
The remaining N elements are recursively distributed using the other variables $X_0,...,X_{b-2}$!



Lower Bound

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

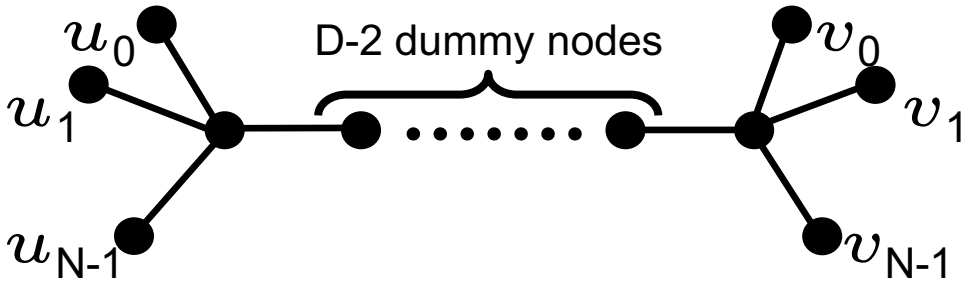
→ Determining all X_i is equivalent to finding the median!



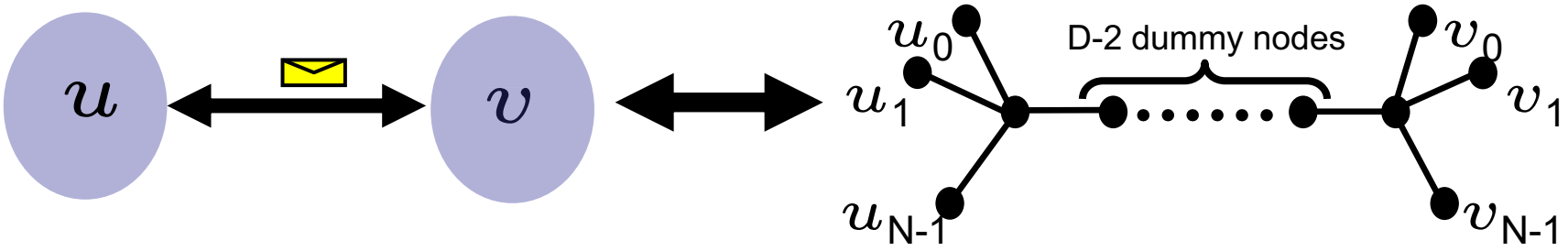
We showed that at least $\Omega(\log_{2B} n)$ rounds are required if B elements can be sent in a single round in this model!

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!

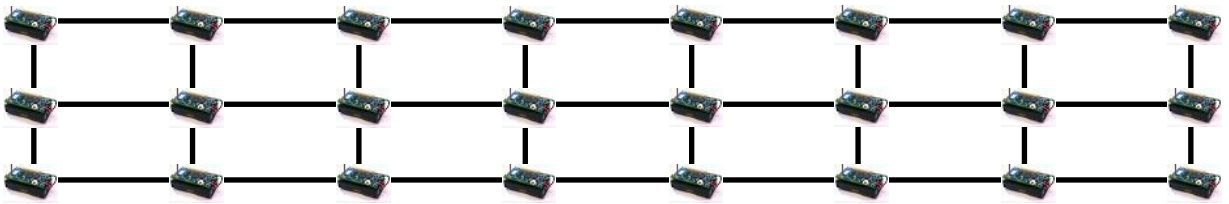
Theorem: $\Omega(D \cdot \log_D \min\{k,n-k\})$ rounds are needed to find the k^{th} smallest element.

$\Omega(D \cdot \log_D n)$ **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 $\exists c \leq 1$: $D = n^c$ → k-selection can be solved efficiently in $\Theta(D)$ time even deterministically!

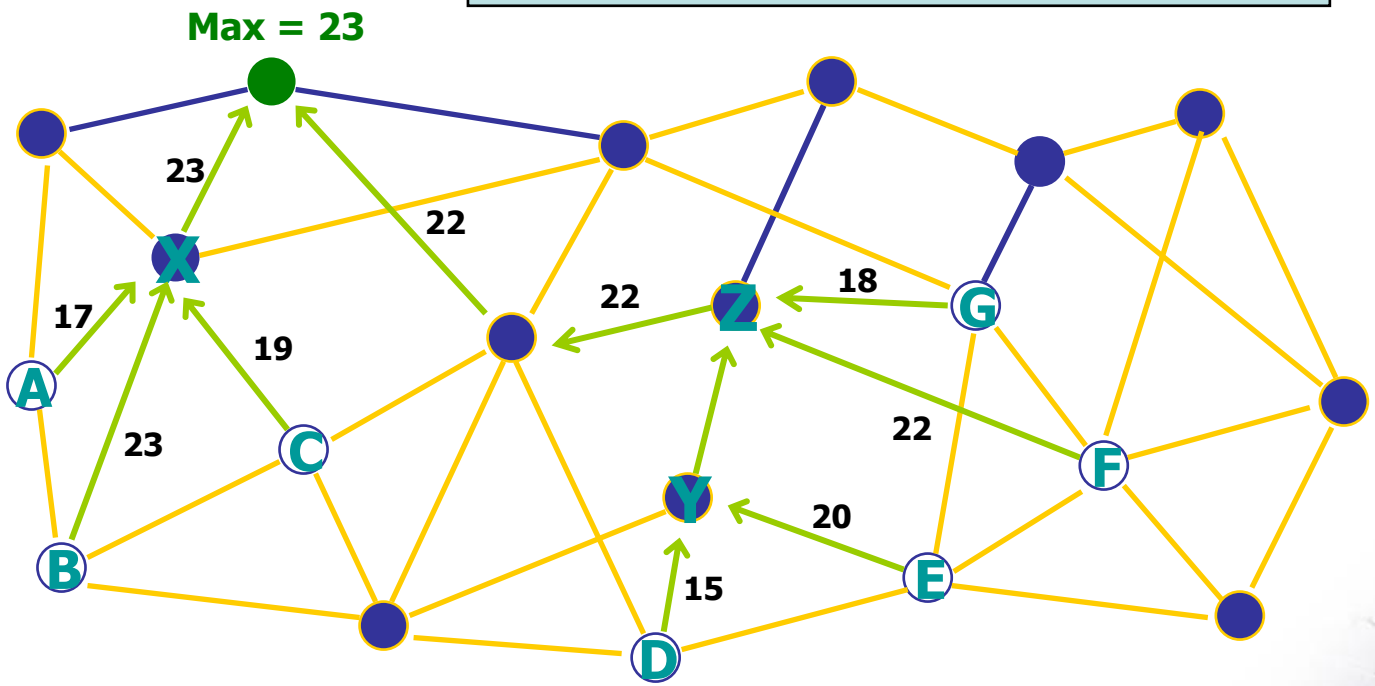
Recall the 50x50 grid used to test out TAG!



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

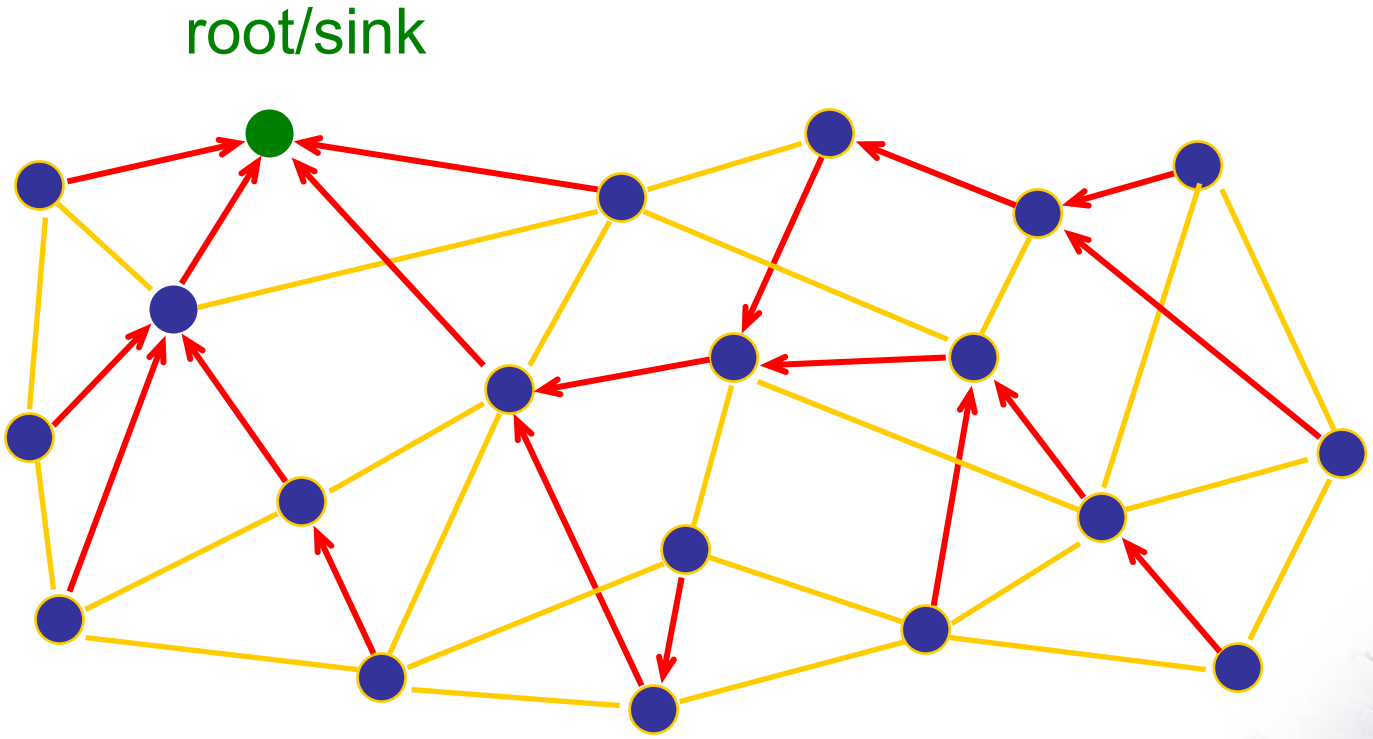
- Given
  - A set of nodes V in the Euclidean plane (or forming a metric space)
  - A root node r $\in$ V
  - Define stretch of a universal spanning tree T to be

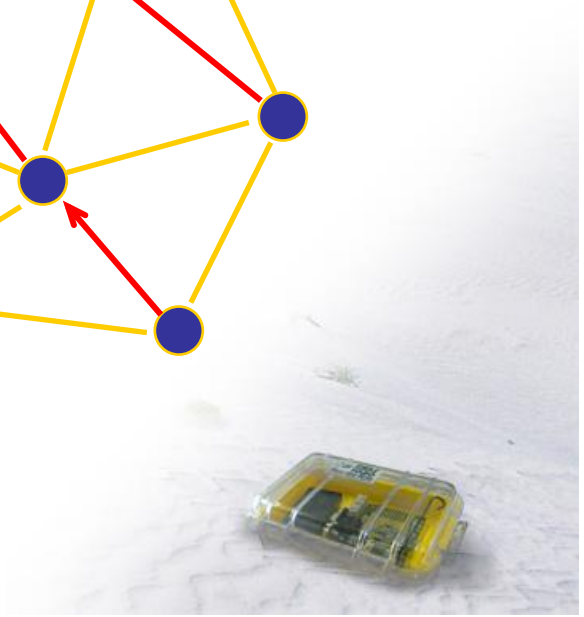
$$\max_{S \subseteq V} \frac{\text{cost(induced tree of S+r on T)}}{\text{cost(minimum Steiner tree of S+r)}}.$$

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

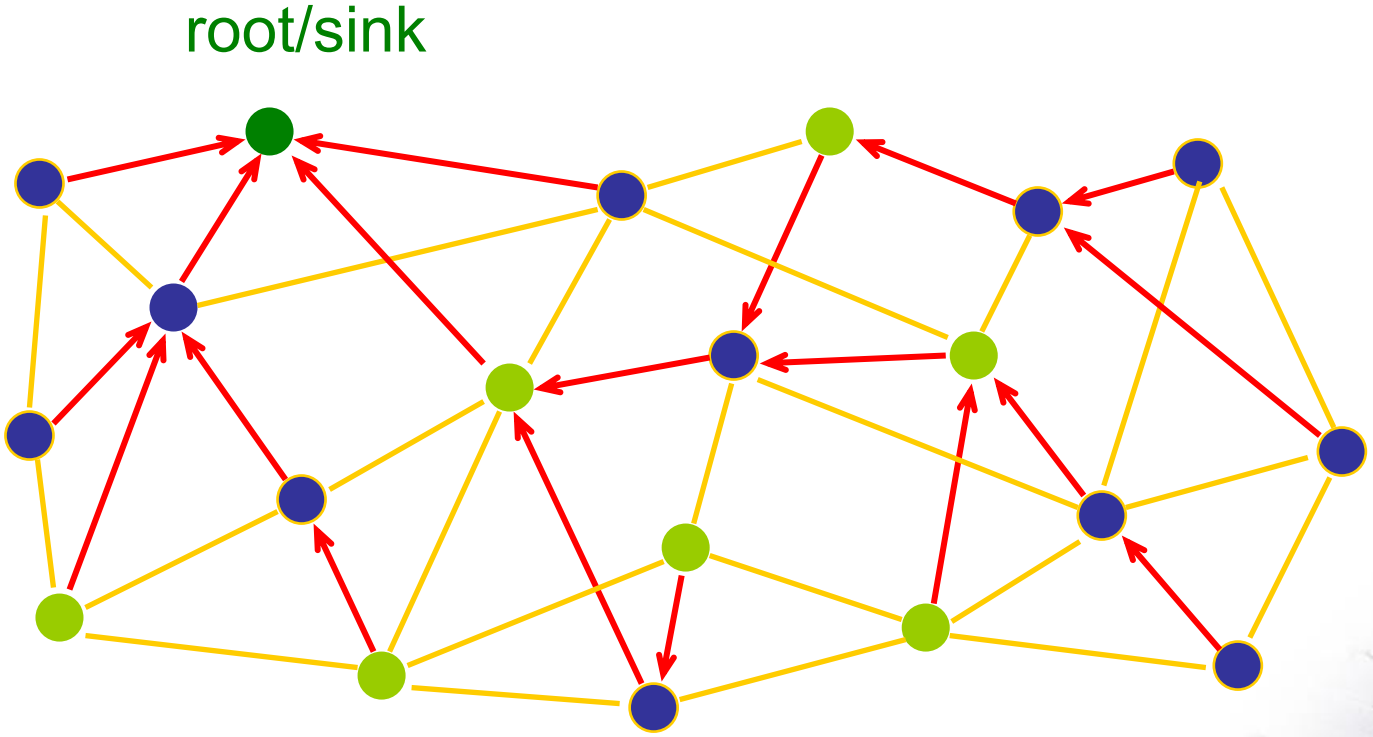
Example

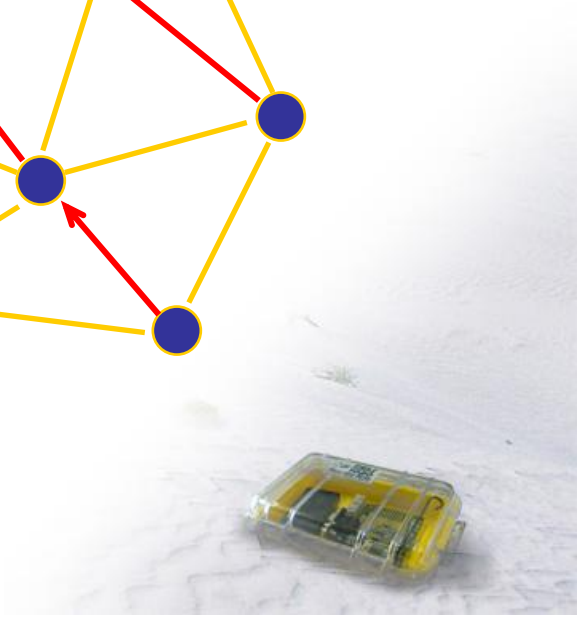
- The red tree is the universal spanning tree. All links cost 1.





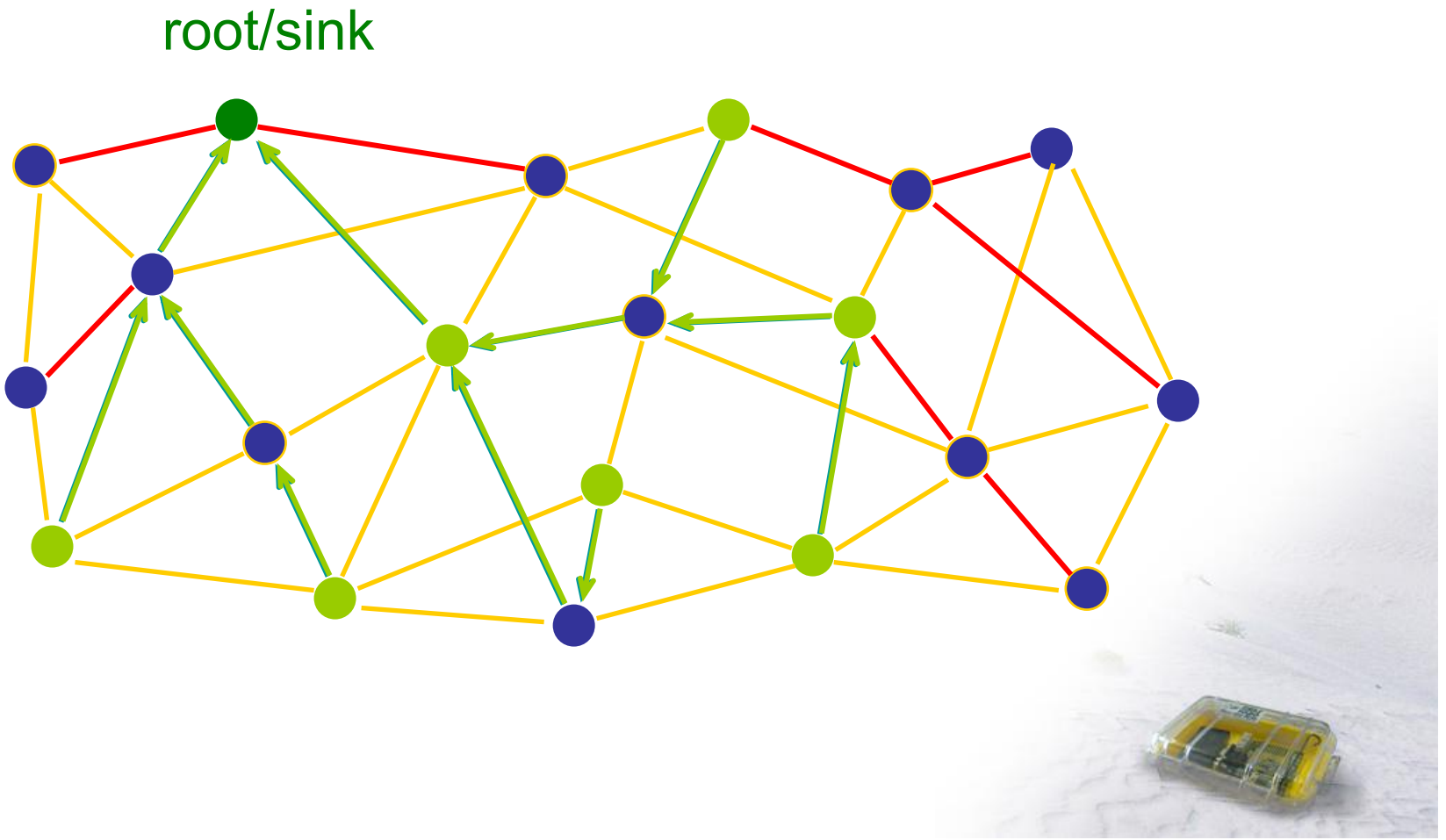
Given the lime subset…





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]

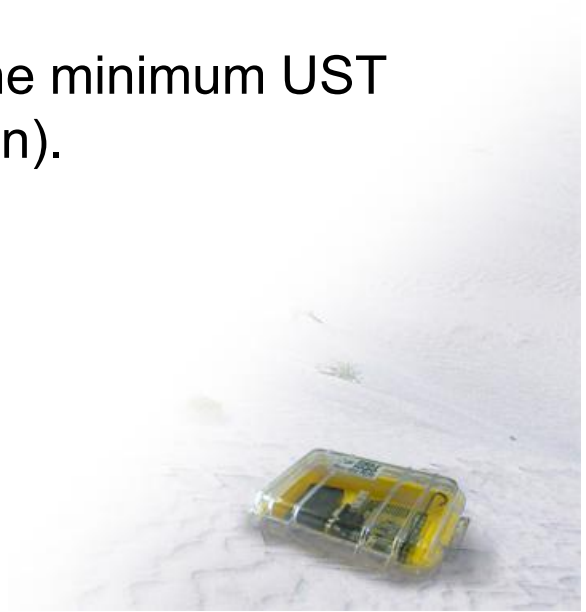
- Theorem 1: (Upper bound)

  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 Ω(log n / log log n).

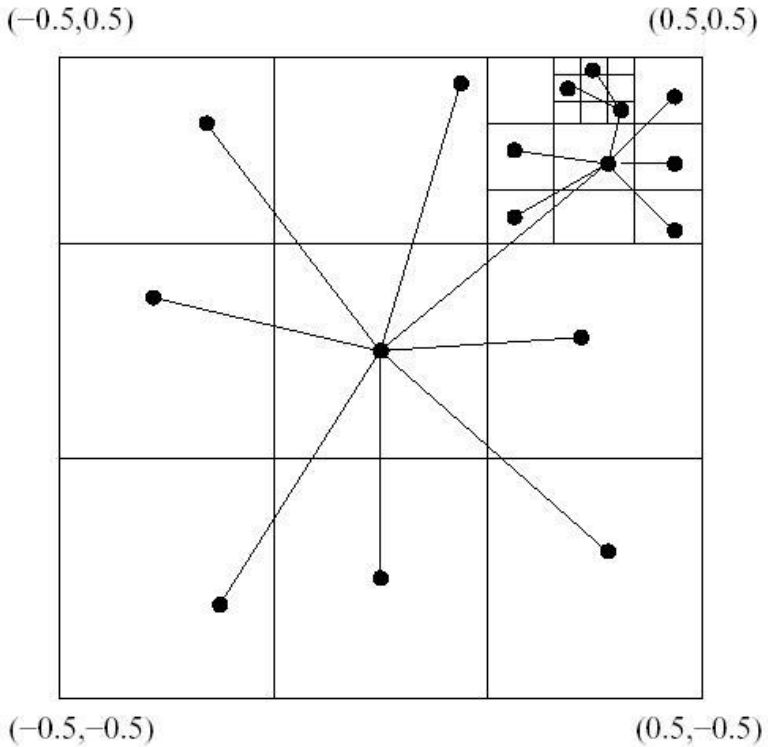
- Proofs: Not in this lecture.



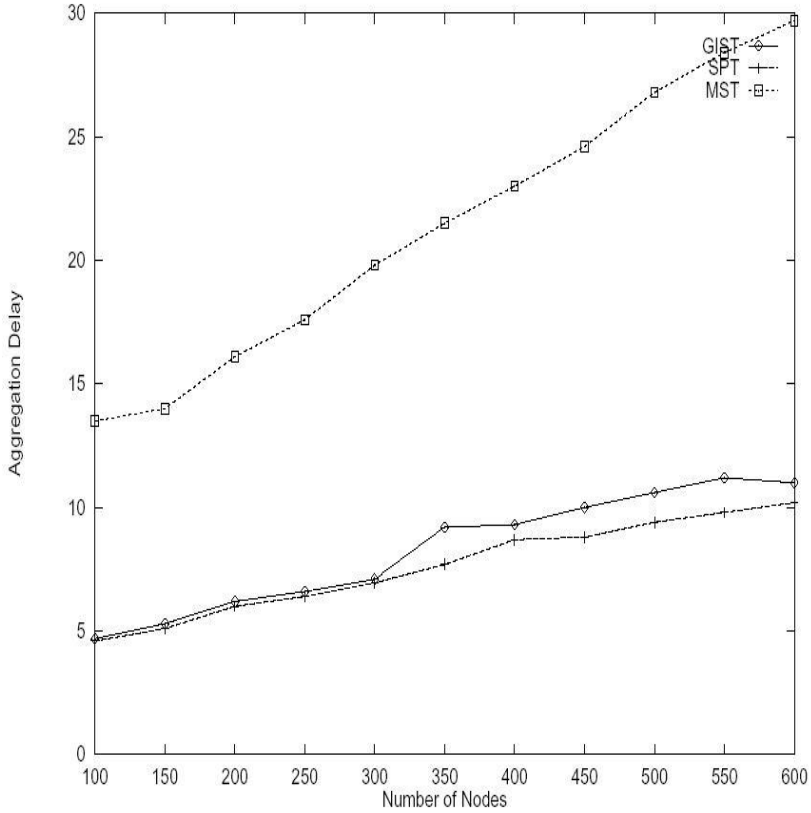
Algorithm sketch

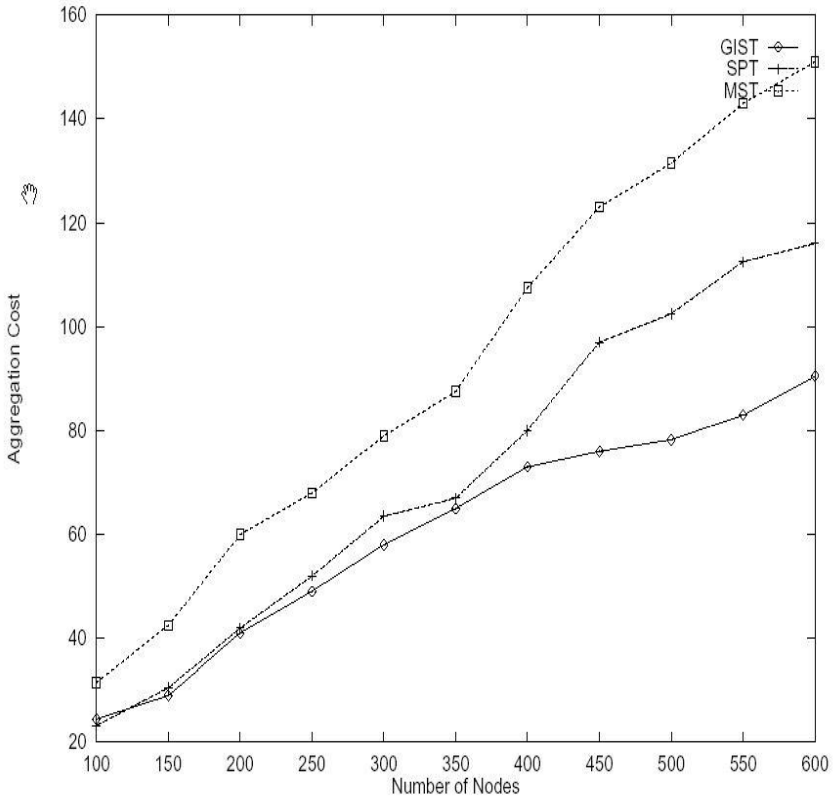
- For the simplest Euclidean case:
- Recursively divide the plane and select random node.

- Results: The induced tree has logarithmic overhead. The aggregation delay is also constant.



Simulation with random node distribution & random events





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

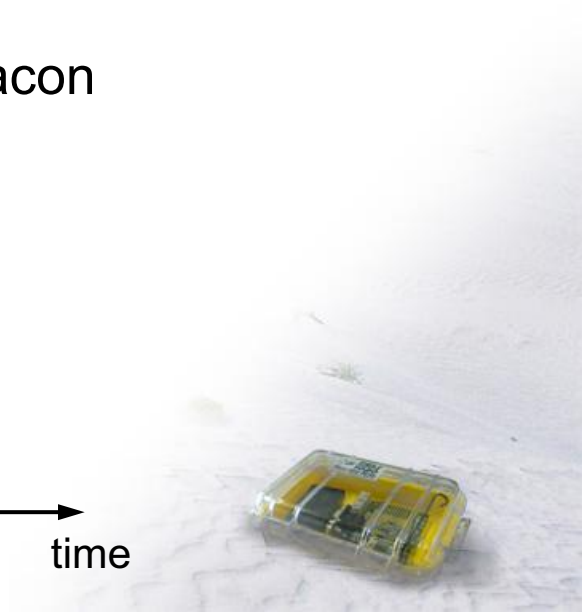
- Issue is tackled at various layers
  - MAC
  - Topology control / clustering
  - Routing

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

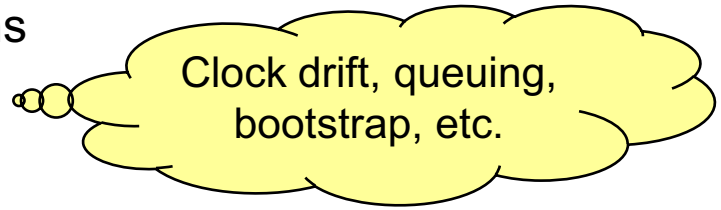
- The parent initiates each TDMA round with a beacon
  - Enables integration of disconnected nodes
  - Children tune in to their parent's schedule

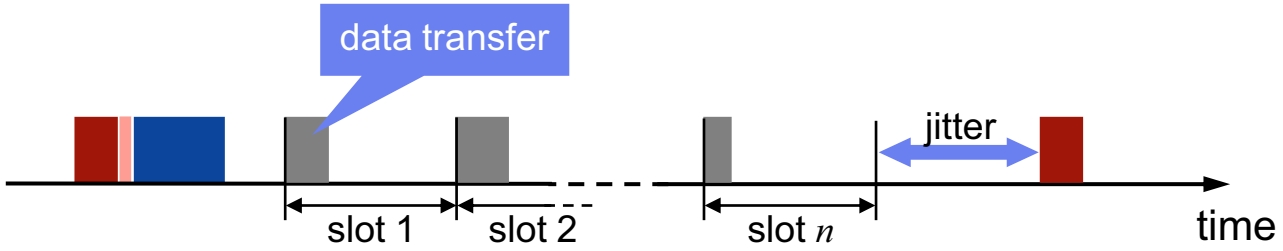


Dozer System

- 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

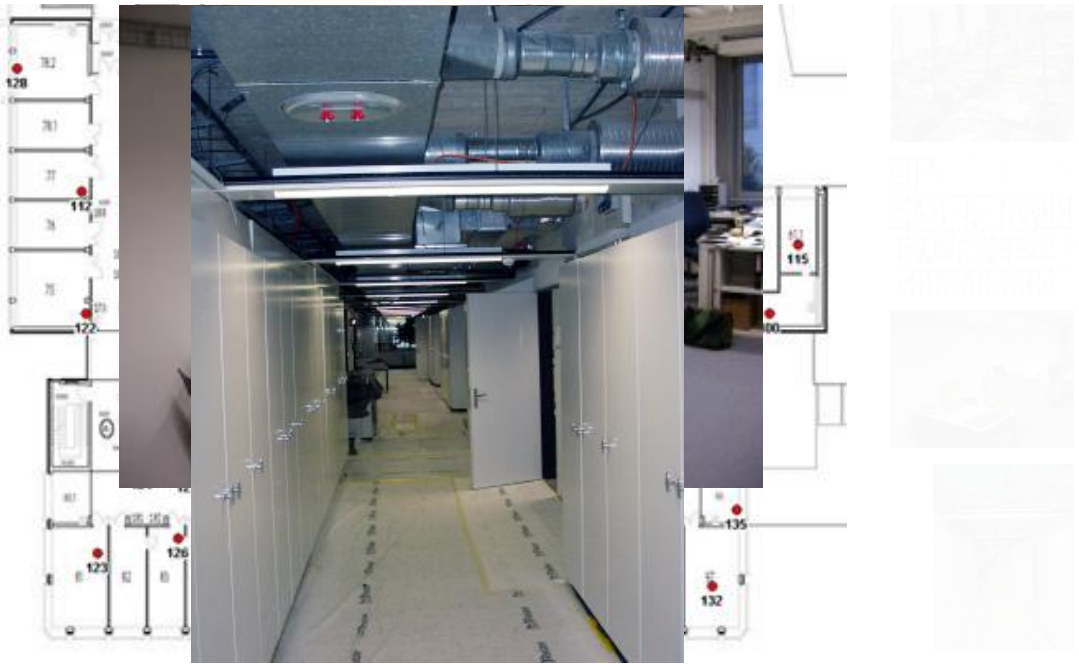
- Lightweight backchannel
  - Beacon messages comprise commands
- Bootstrap
  - Scan for a full interval
  - Suspend mode during network downtime

periodic channel activity check

- Potential parents
  - Avoid costly bootstrap mode on link failure
  - Periodic refresh the list

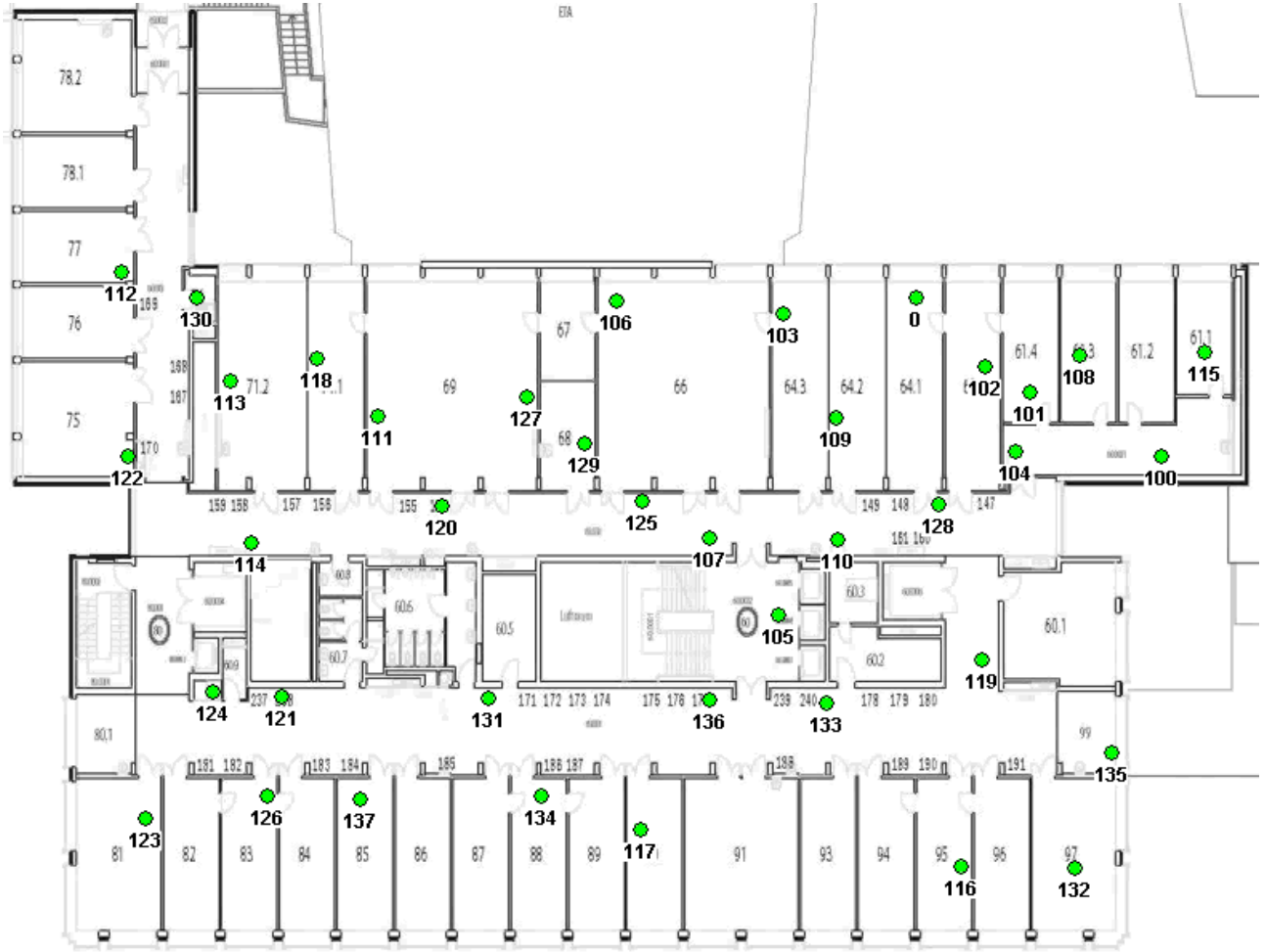
Evaluation

- Platform
  - TinyNode
    - MSP 430
    - Semtech XE1205
  - TinyOS 1.x



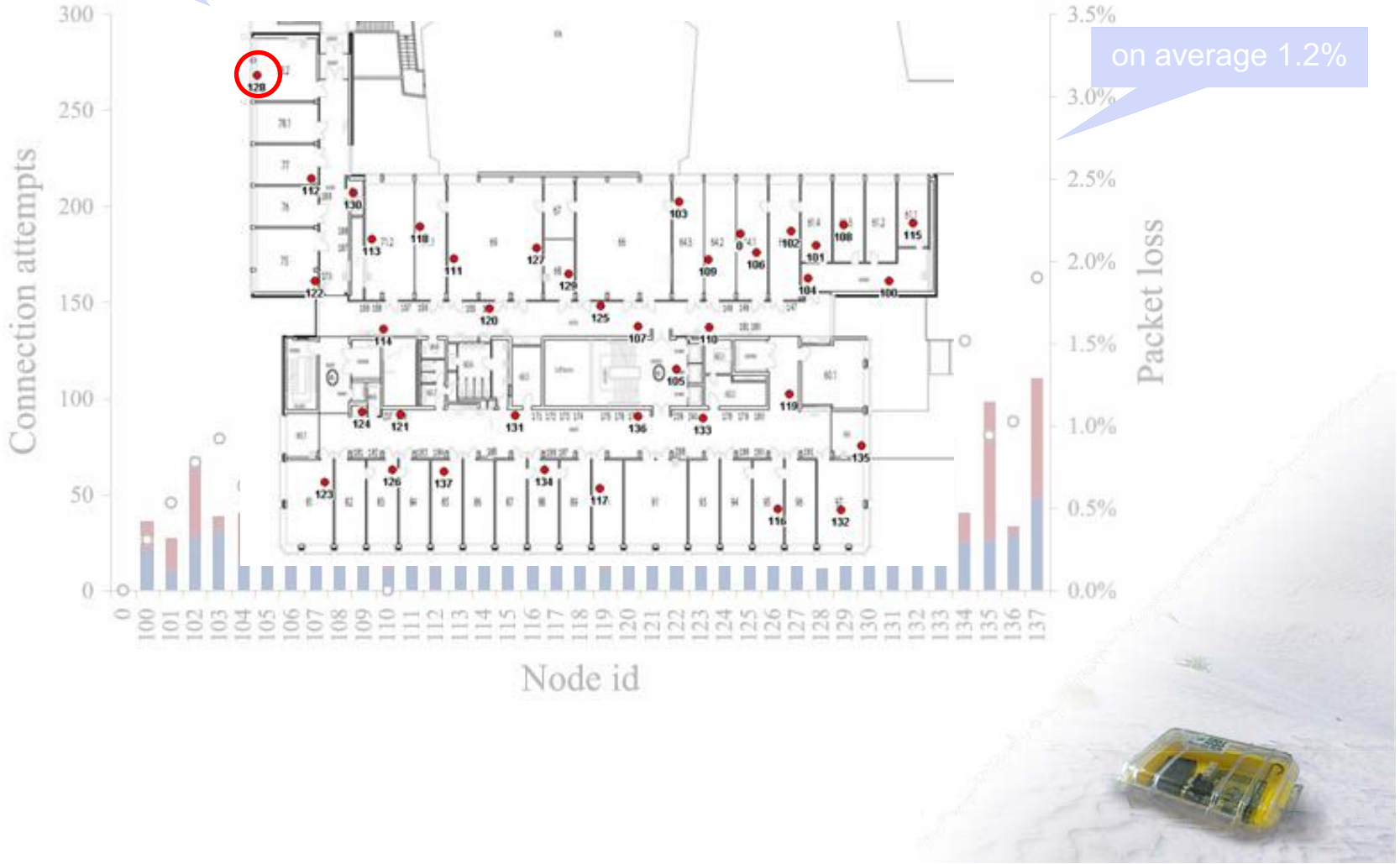
- Testbed
  - 40 Nodes
  - Indoor deployment
  - > 1 month uptime
  - 30 sec beacon interval
  - 2 min data sampling interval

Dozer in Action



Tree Maintenance

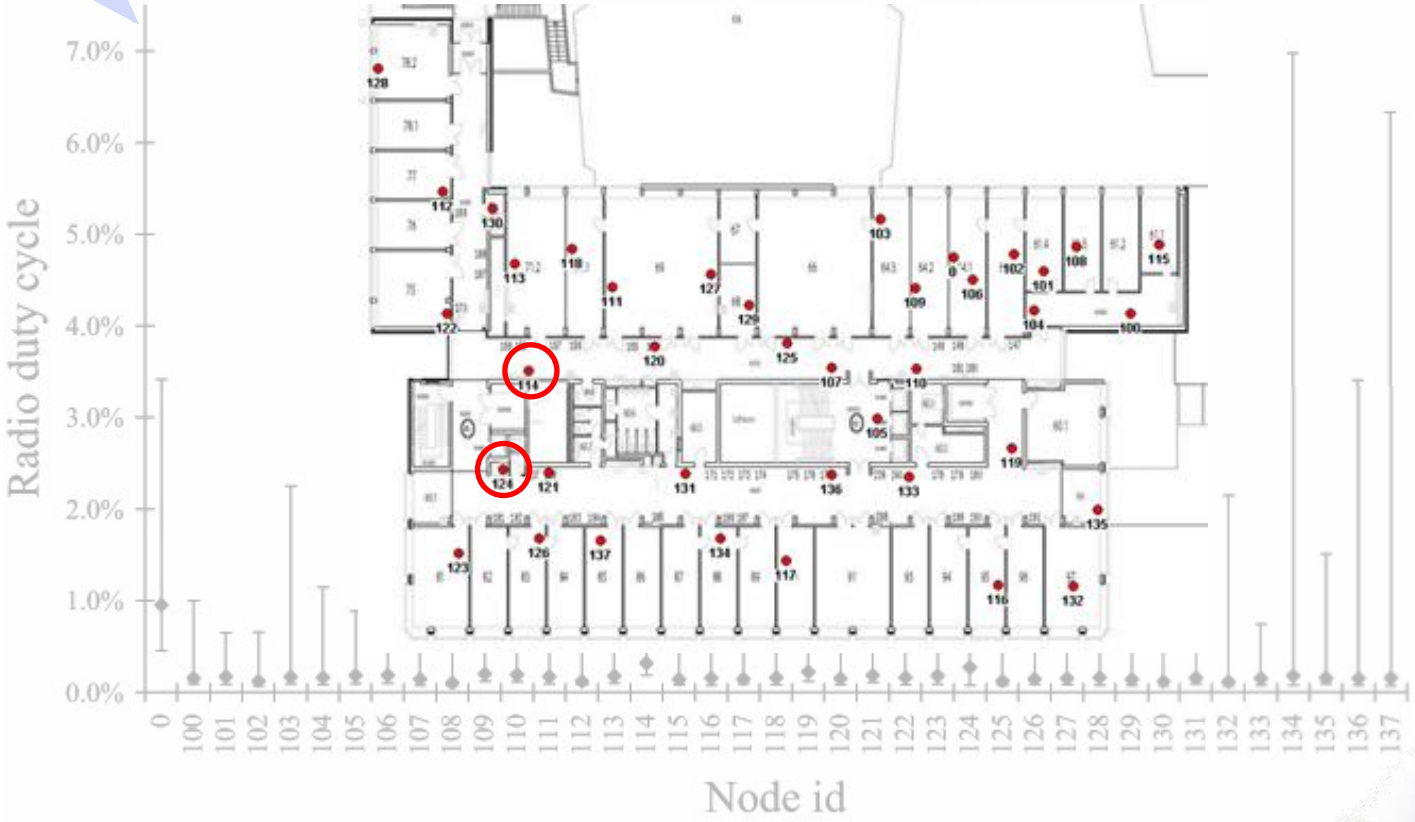
1 week of operation



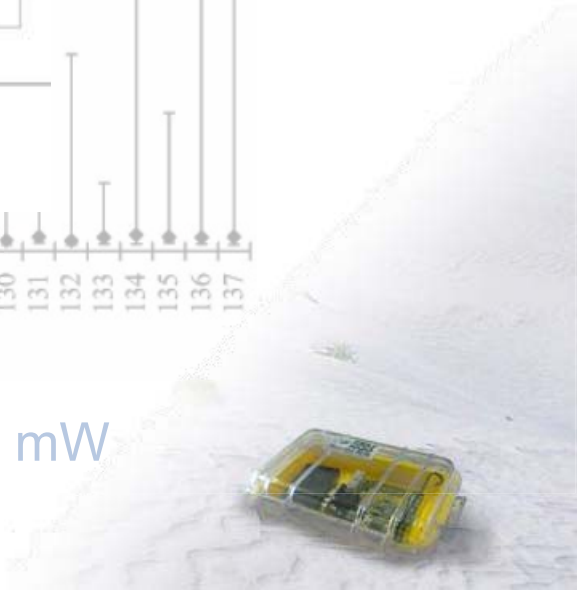
on average 1.2%

Energy Consumption

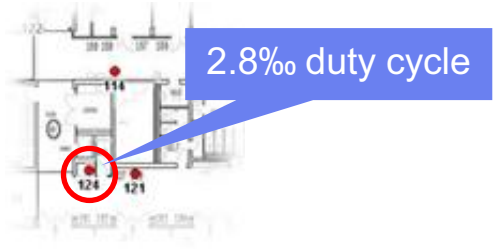
on average 1.67‰

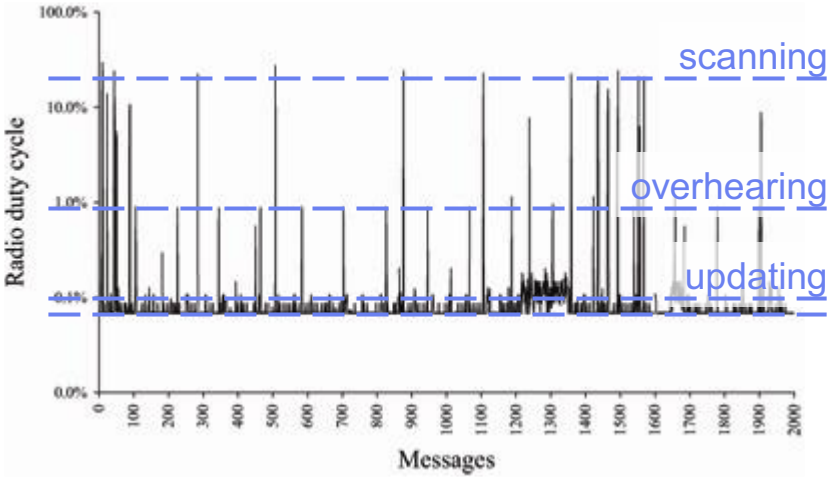


➡ Mean energy consumption of 0.082 mW

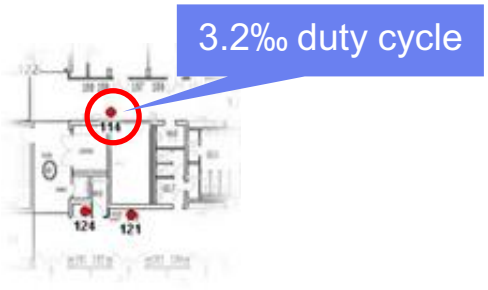


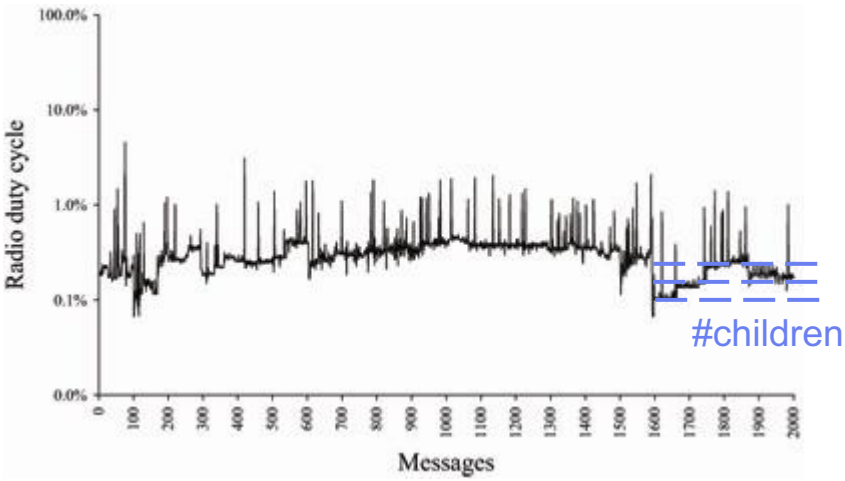
Energy Consumption





- 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)?