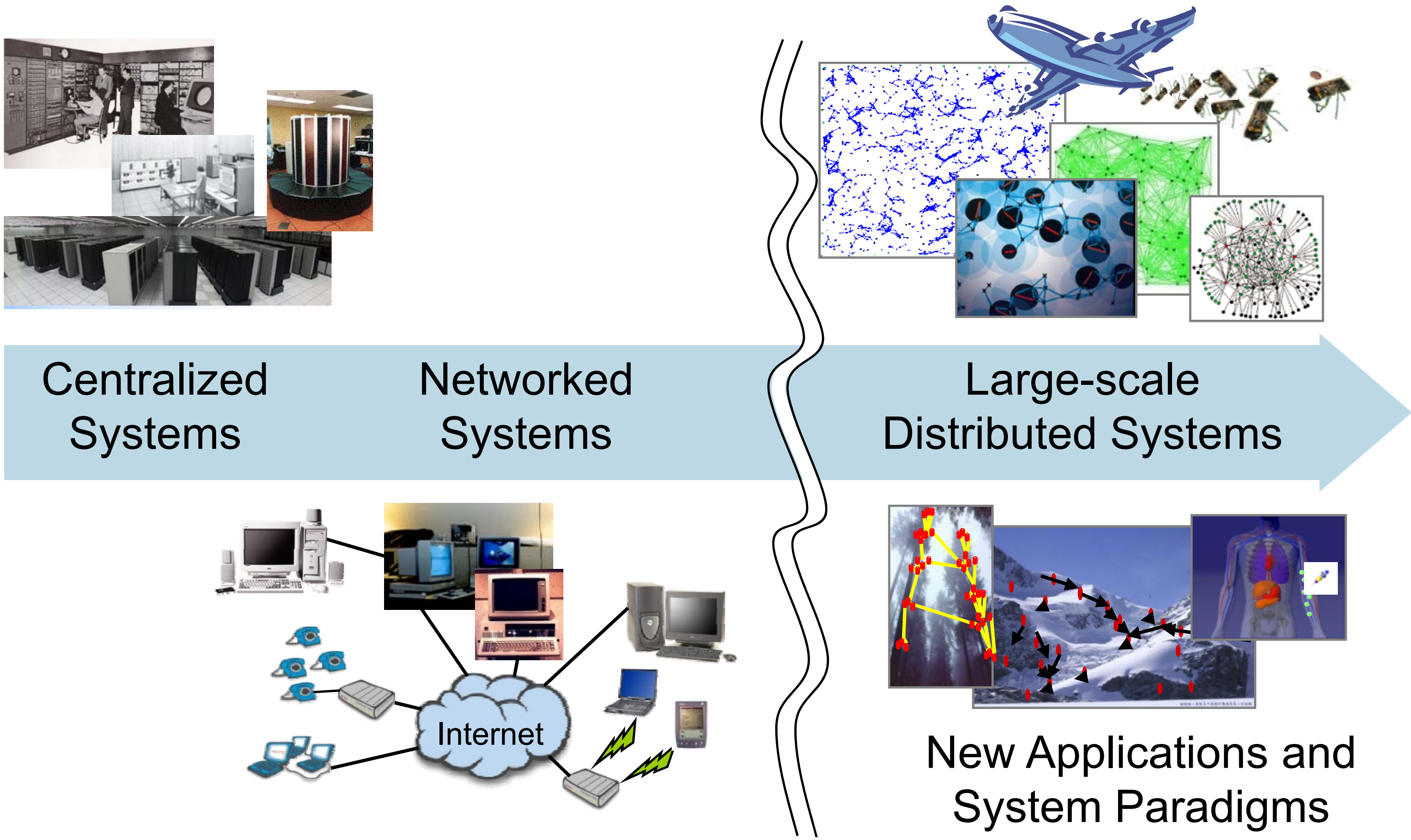
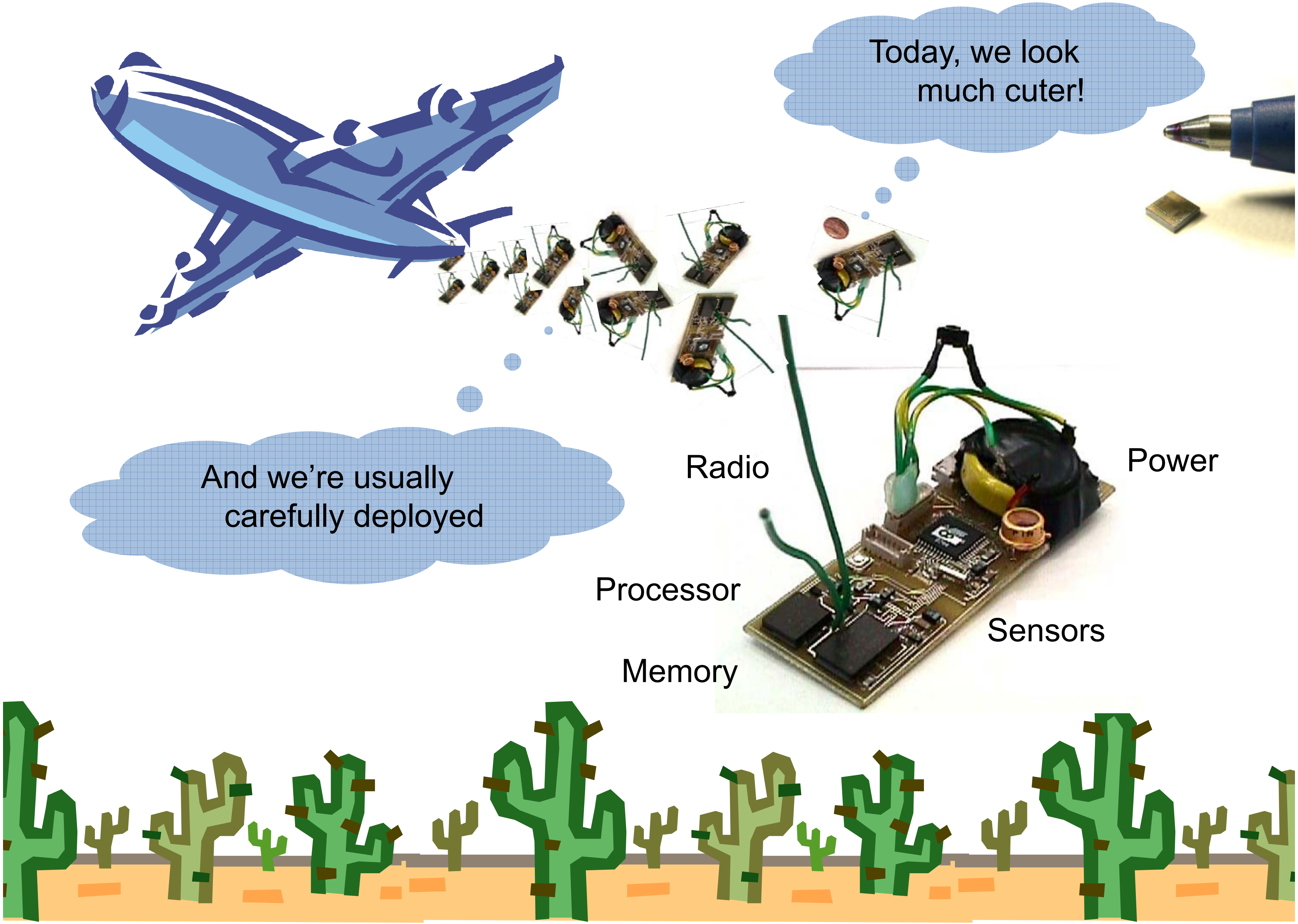
Sensor Networks

Distributed Computing and Networking Get Together to Gather Data

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

General Trend in Information Technology



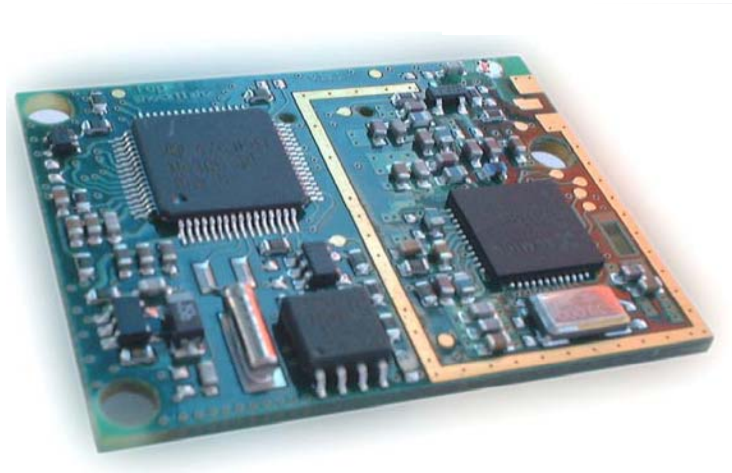

Today, we look much cuter!
And we're usually carefully deployed
Radio
Power
Processor
Sensors
Memory

A Typical Sensor Node: TinyNode 584

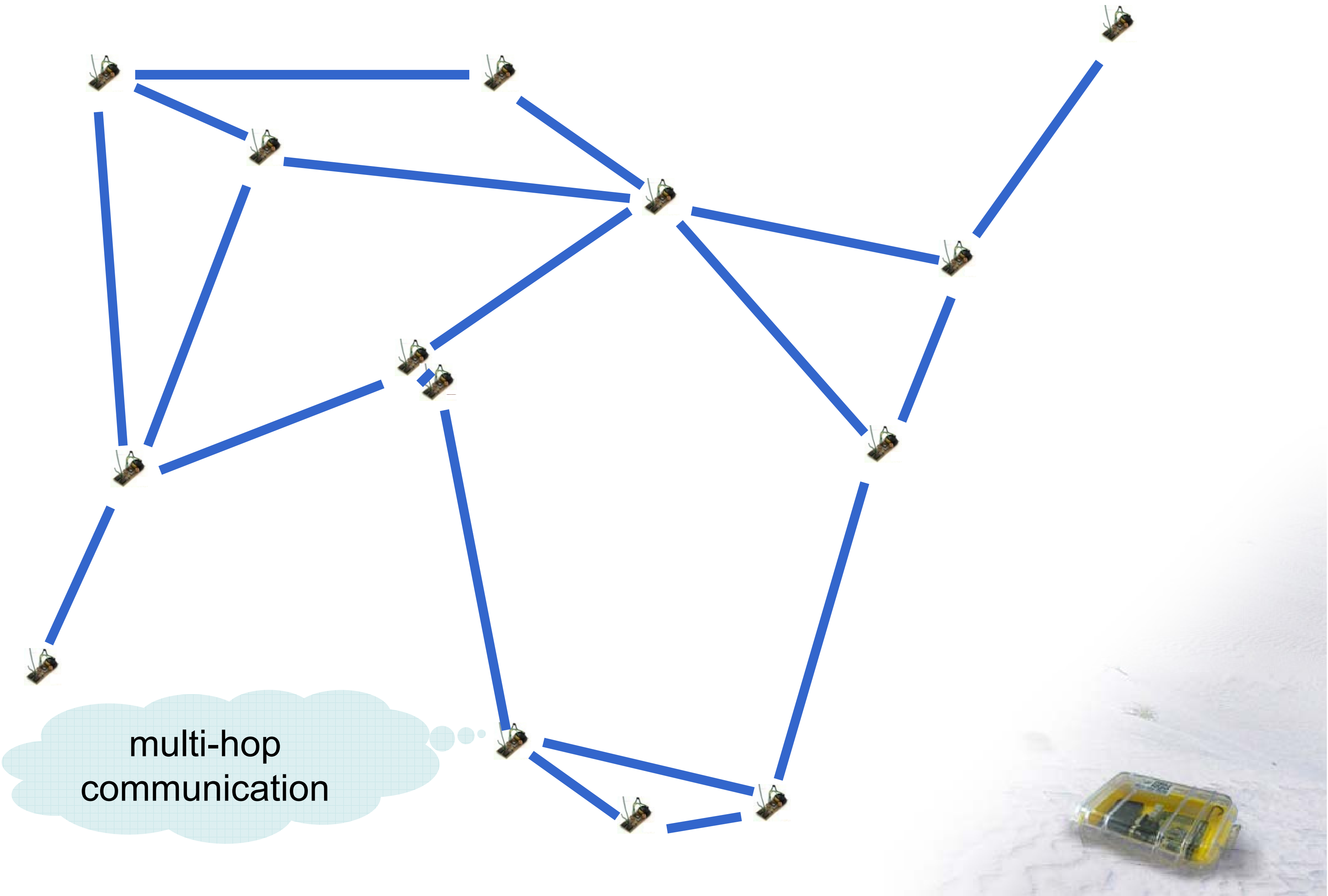
[Shockfish SA, The Sensor Network Museum]

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

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



After Deployment



Ad Hoc Networks vs. Sensor Networks

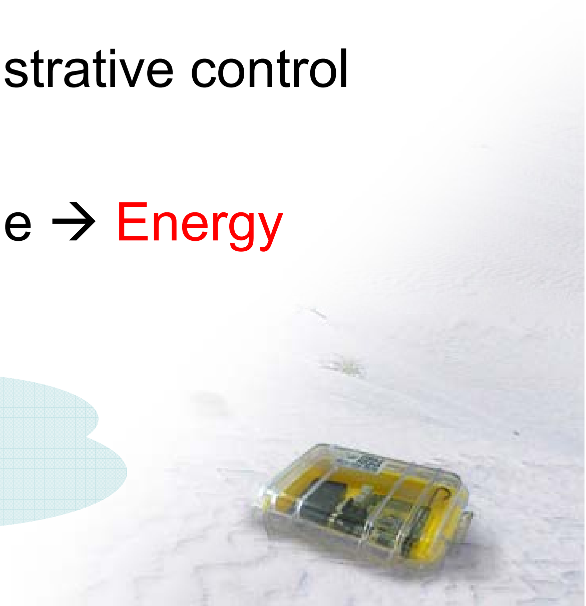
Ad Hoc Networks

- Laptops, PDA's, cars, soldiers
- All-to-all routing
- Often with mobility (MANET's)
- Trust/Security an issue
  - No central coordinator
- Maybe high bandwidth

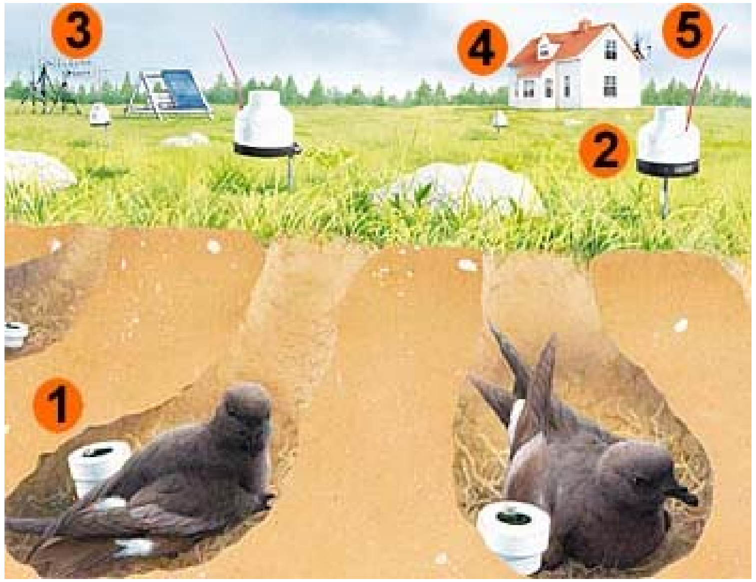
Sensor Networks

- Tiny nodes: 4 MHz, 32 kB, …
- Mostly data gathering
- Usually no mobility
  - but link failures
- One administrative control
- Long lifetime → Energy

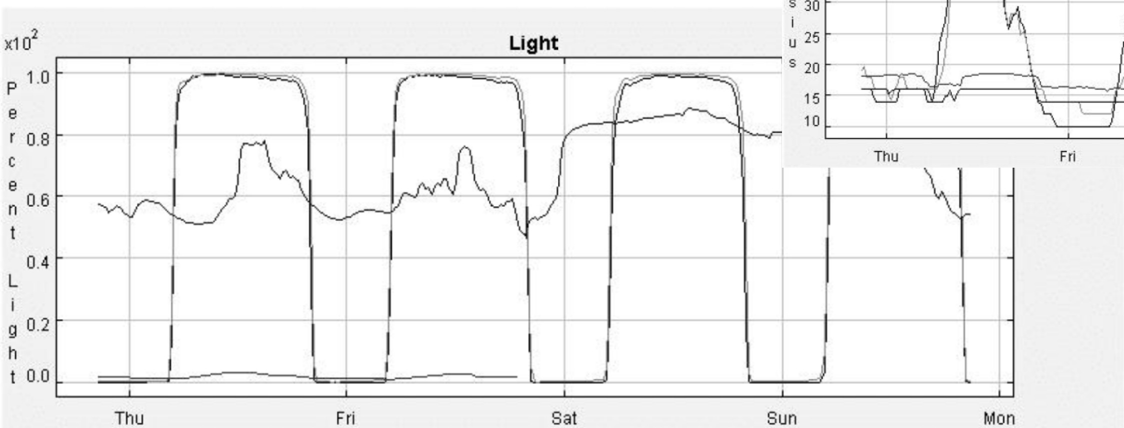
There is no strict separation; more variants such as mesh or sensor/actor networks exist

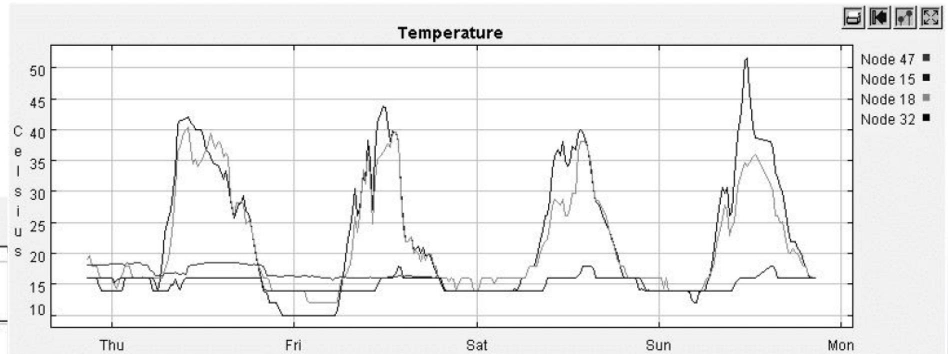


Animal Monitoring (Great Duck Island)



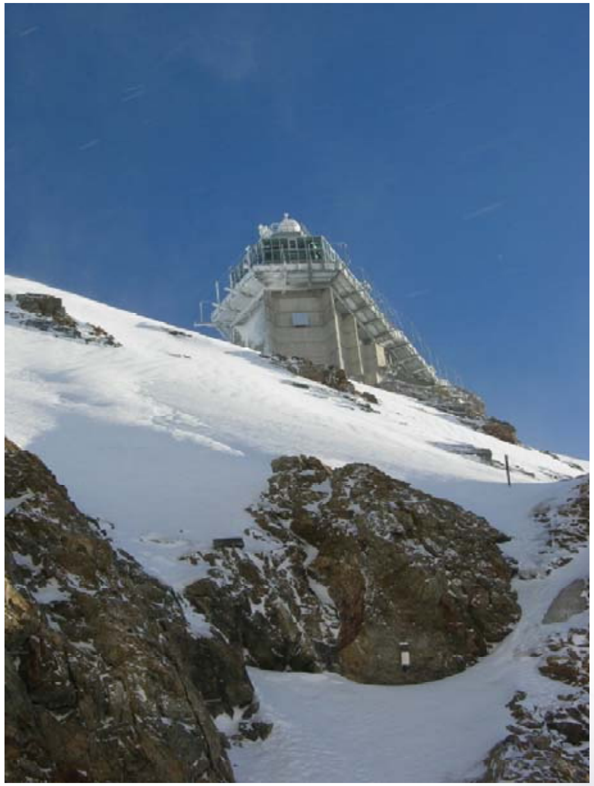
1. Biologists put sensors in underground nests of storm petrel
2. And on 10cm stilts
3. Devices record data about birds
4. Transmit to research station
5. And from there via satellite to lab





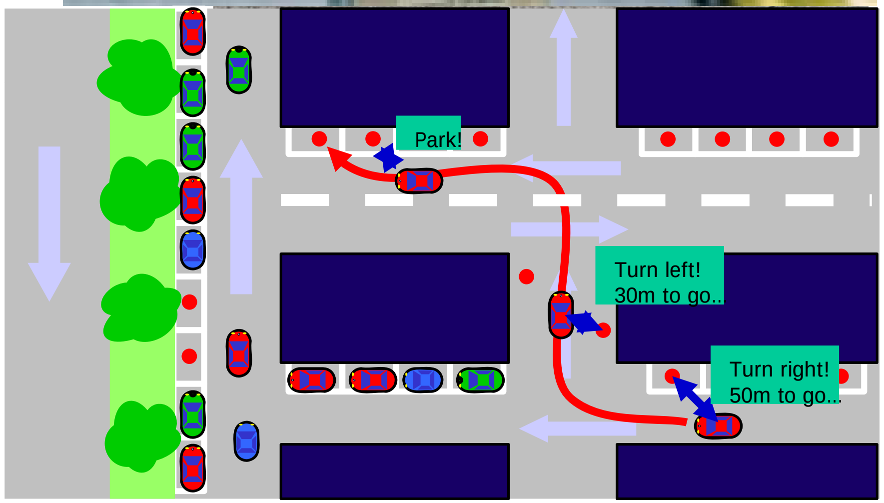
Environmental Monitoring (PermaSense)

- Understand global warming in alpine environment
- Harsh environmental conditions
- Swiss made (Basel, Zurich)



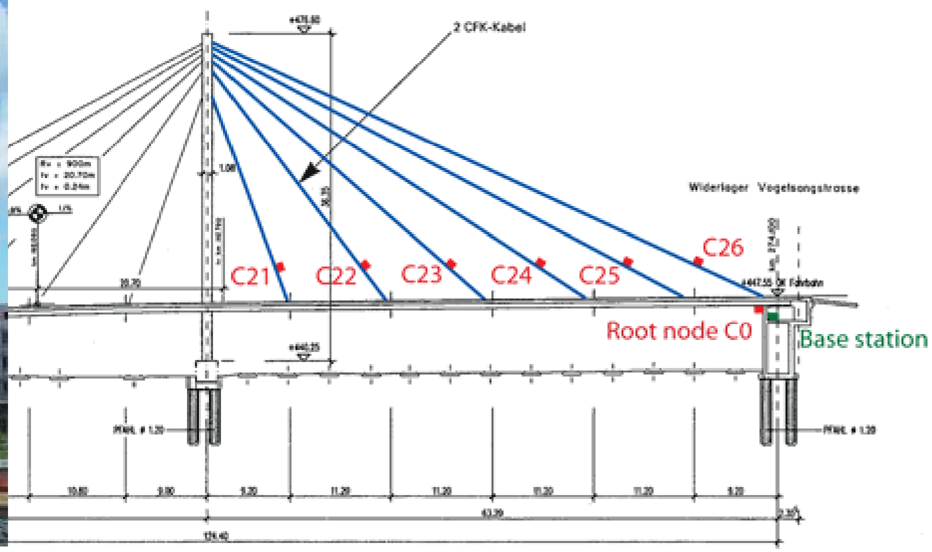
Smart Spaces (Car Parking)

- The good: Guide cars towards empty spots
- The bad: Check which cars do not have any time remaining
- The ugly: Meter running out: take picture and send fine



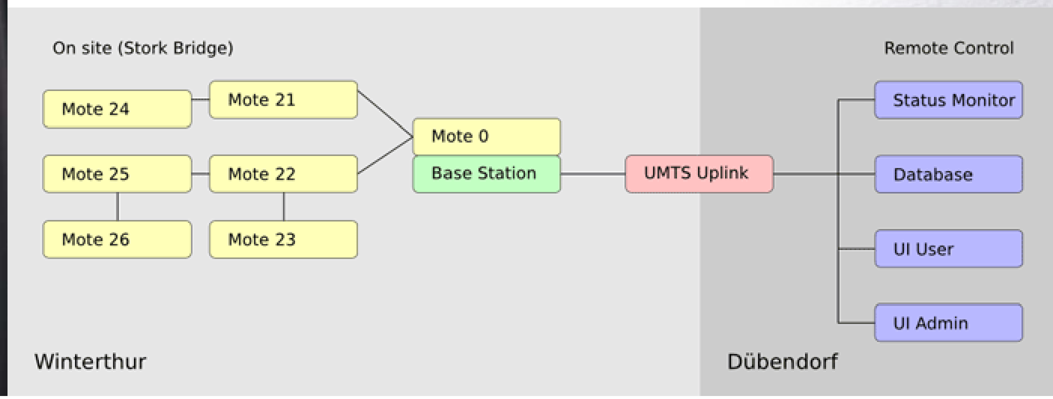
[Matthias Grossglauser, EPFL & Nokia Research]

Structural Health Monitoring (Bridge)



Swiss Made
[EMPA]

Detect structural defects, measuring temperature, humidity, vibration, etc.

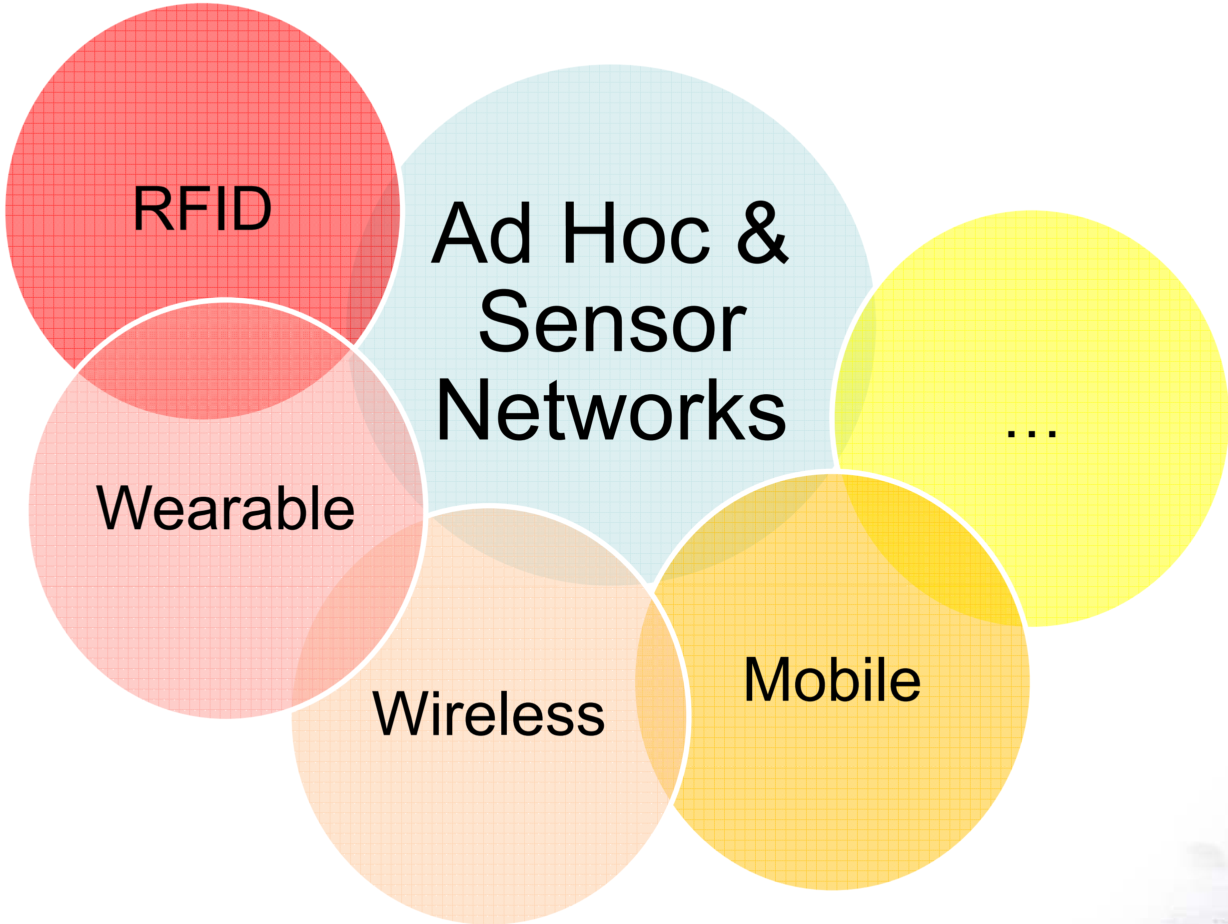


Agriculture (COMMONSense)

- Idea: Farming decision support system based on recent local environmental data.
- Irrigation, fertilization, pest control, etc. are output of function of sunlight, temperature, humidity, soil moisture, etc.
- (Actual sensors are mostly underground)

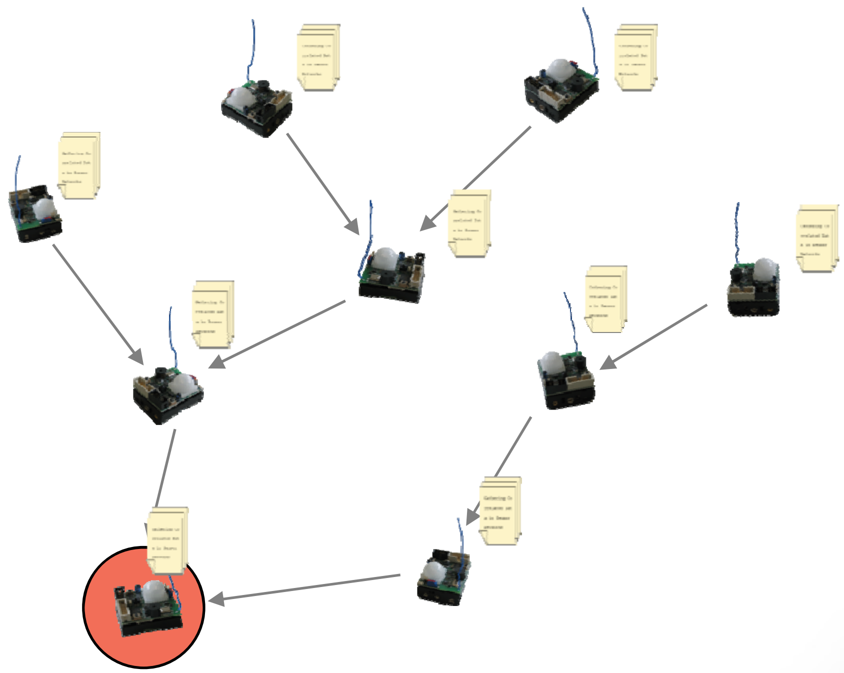
[EPFL & IIT]

Related Areas



Periodic data gathering (as in many applications)

- All nodes produce relevant information about their vicinity periodically.
- Data is conveyed to an information sink for further processing.
- Data may or may not be aggregated.
- Variation: Sense event (e.g. fire, burglar)



Data gathering with queries (e.g. TinyDB)

- 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>]
```

Overview

- Introduction
- Applications
- Data Gathering

- Minimizing Messages with Aggregation (Distributed Computing)
- Minimizing Time with Power Control
- Minimizing Energy Consumption with Sleep Schedules

- Conclusion

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

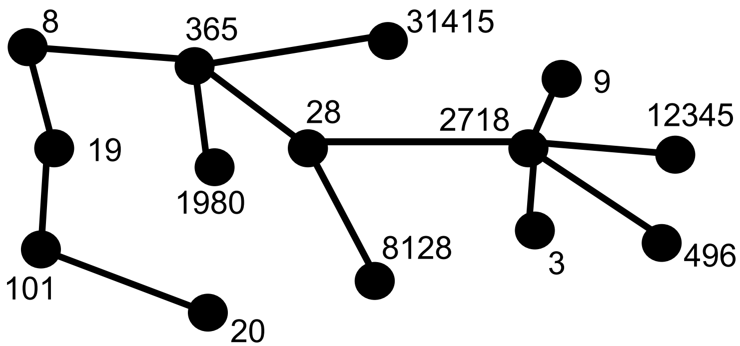
How difficult is it to compute these aggregation primitives?

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 $D \le 2D_G$)
- 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***!

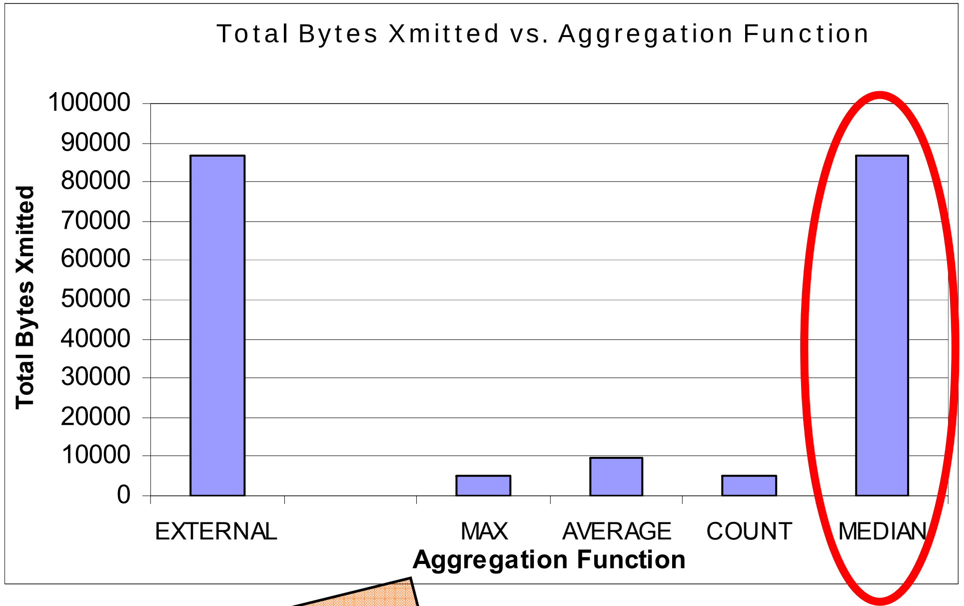
Time complexity: $\Theta(D)$, D = Diameter

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: $O(D \cdot n^{0.9114})$.

Not so great...

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 $O(D \cdot \log n)$...

We do not want the complexity to depend on the values!

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!

Nice! Can we do better?

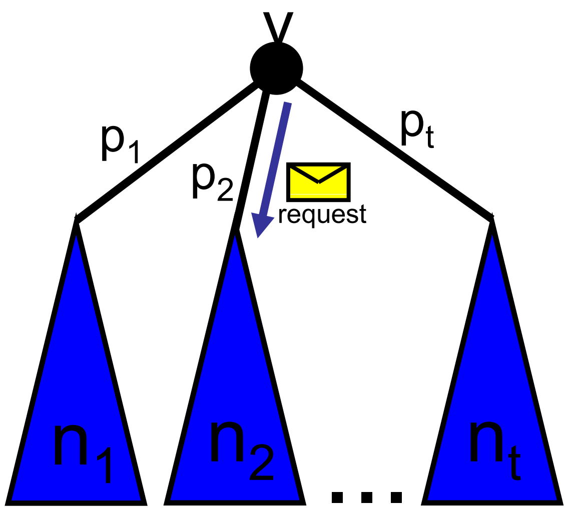
Randomized Algorithm

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

How is this done?

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

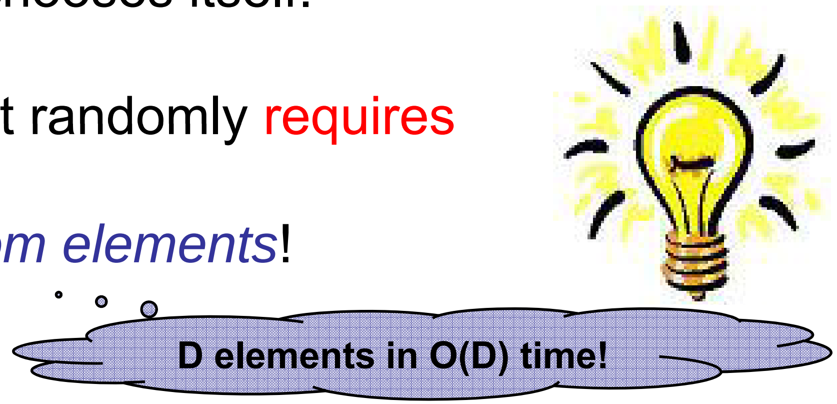
$$p_i := n_i / (1 + \Sigma_k n_k).$$



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

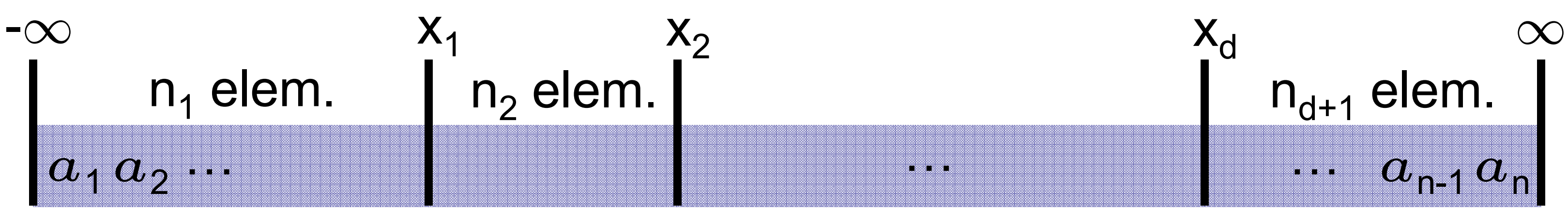
Our algorithm also operates in phases → The set of *candidates* decreases in each phase!

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 $x_1,\ldots,x_d$ 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!

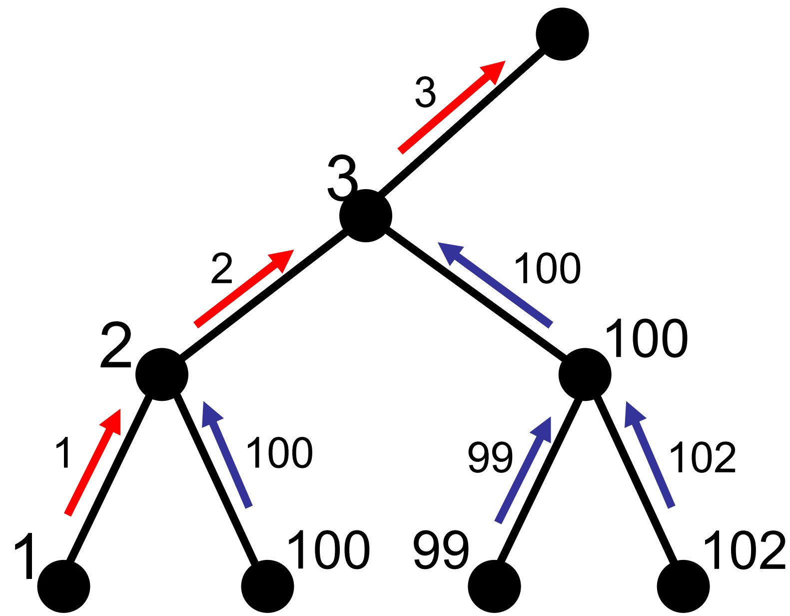
Deterministic Algorithm

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 talk)



Lower Bound

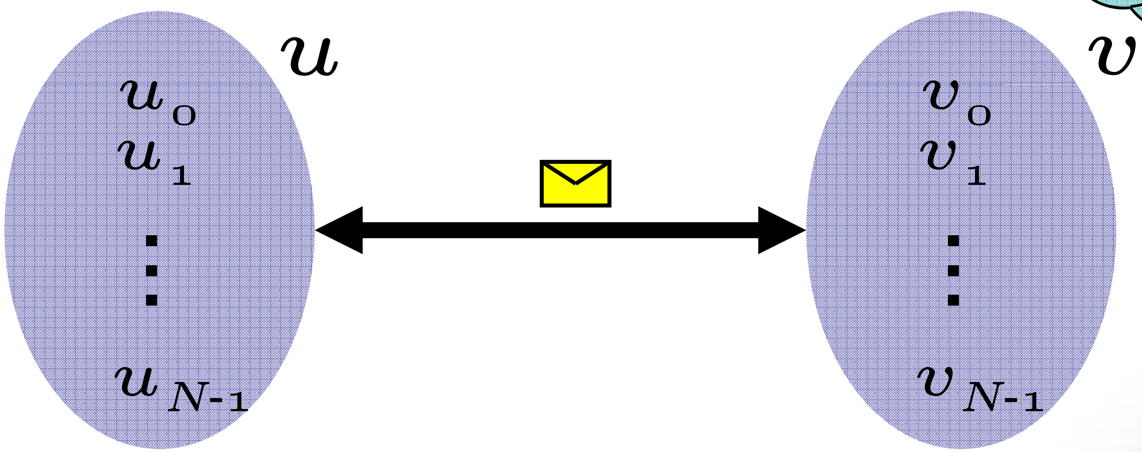
The proof of the lower bound of $\Omega(D \cdot \log_D n)$ consists of two parts:

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

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

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

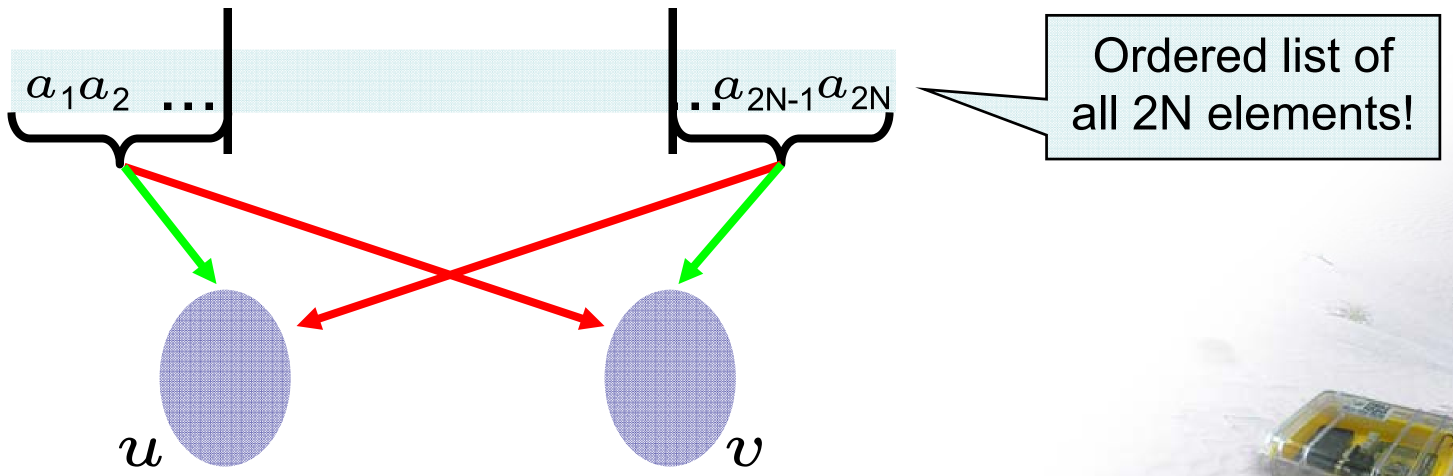
What is the order between all u_i and v_j?



Lower Bound

Assume $N = 2^b$. We use b independent Bernoulli variables $X_0,\ldots,X_{b-1}$ 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.

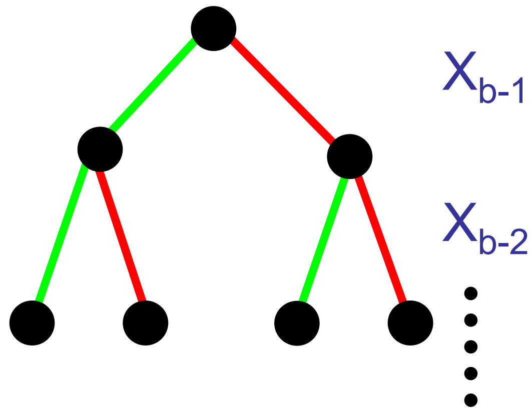
The remaining N elements are recursively distributed using the other variables $X_0,\ldots,X_{b-2}$!



Lower Bound

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

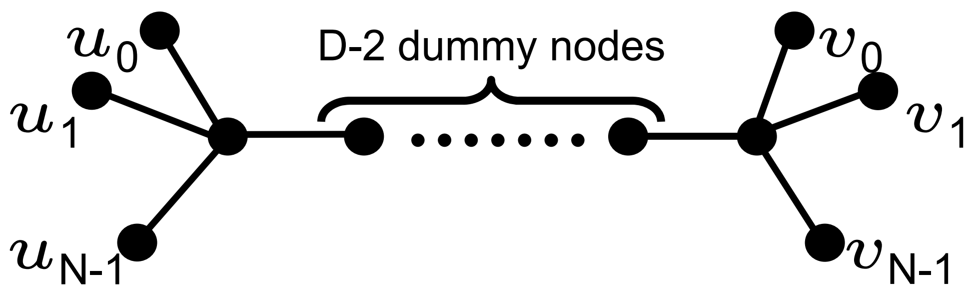
→ Determining all X_i is equivalent to finding the median!



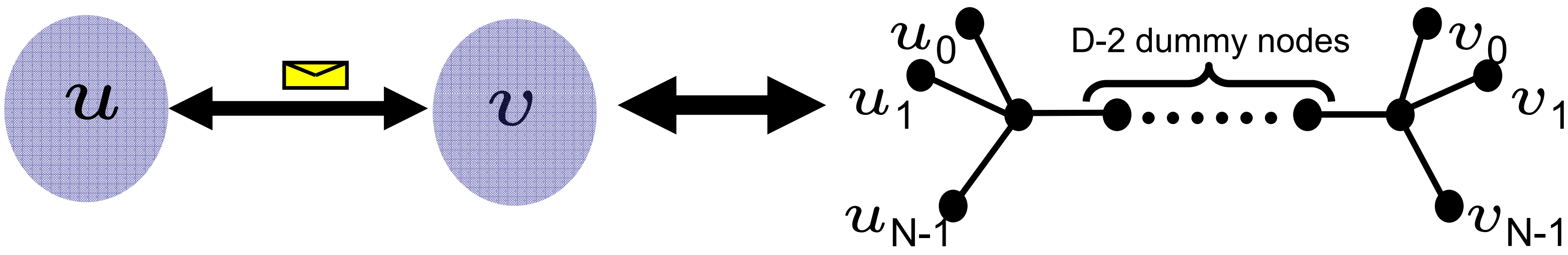
We showed that at least $\Omega(\log_{2B} N)$ rounds are required if B elements can be sent in a single round in this model!

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

Look at the following graph G of diameter D:



Lower Bound



alternative model *original model*

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

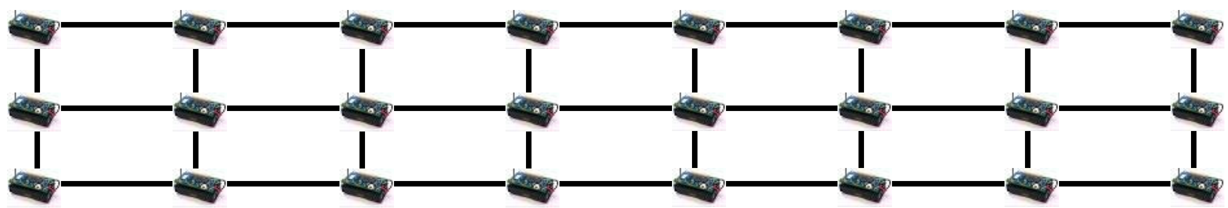
Theorem: $\Omega(D \cdot \log_D \min\{k,n-k\})$ rounds are needed to find the k^{th} smallest element.

$\Omega(D \cdot \log_D n)$ 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!



Overview

- Introduction
- Applications
- Data Gathering

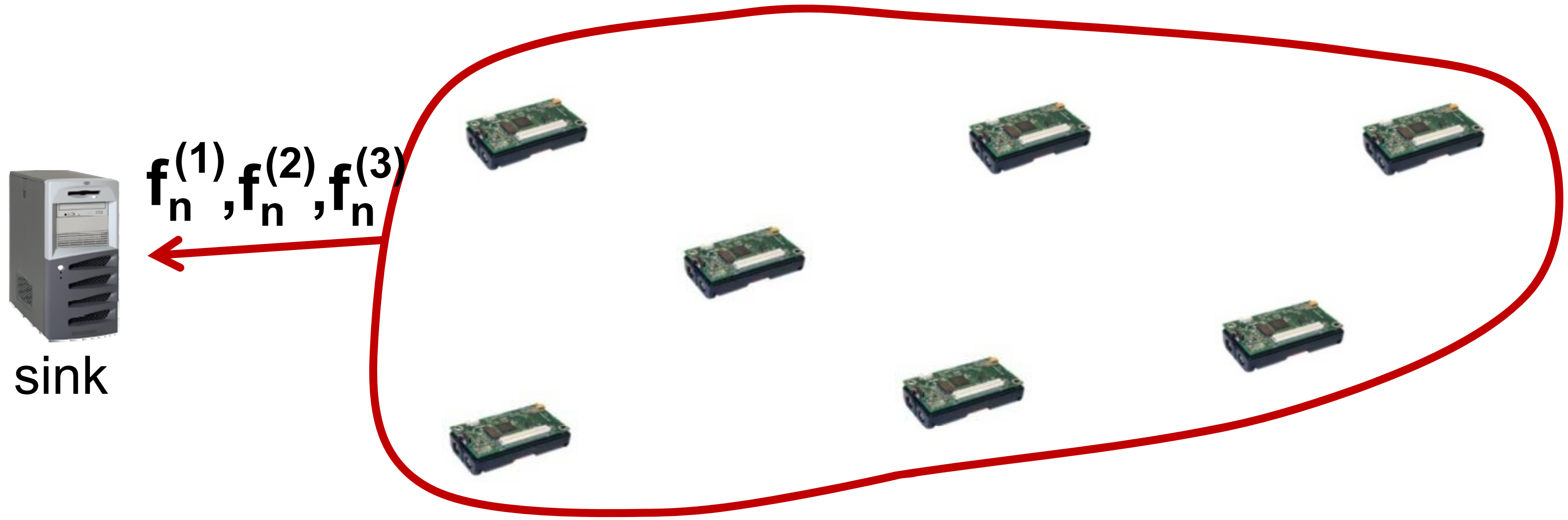
- Minimizing Messages with Aggregation
- Minimizing Time with Power Control (Networking)
- Minimizing Energy Consumption with Sleep Schedules

- Conclusion

Data Gathering in Wireless Sensor Networks

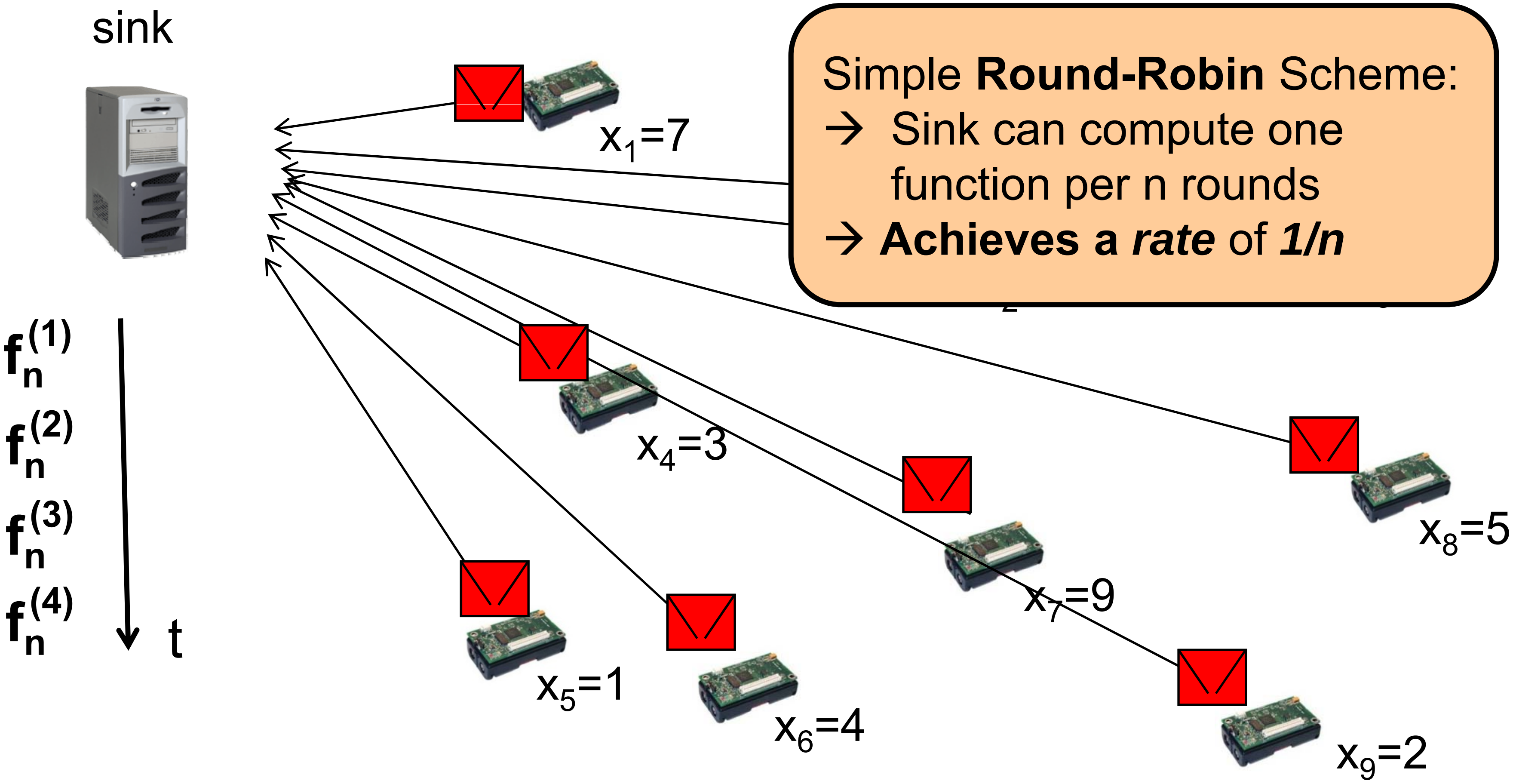
- Data gathering & aggregation
  - Classic application of sensor networks
  - Sensor nodes periodically sense environment
  - Relevant information needs to be transmitted to sink

- Functional Capacity of Sensor Networks
  - Sink peridically wants to compute a function f_n of sensor data
  - At what rate can this function be computed?



Data Gathering in Wireless Sensor Networks

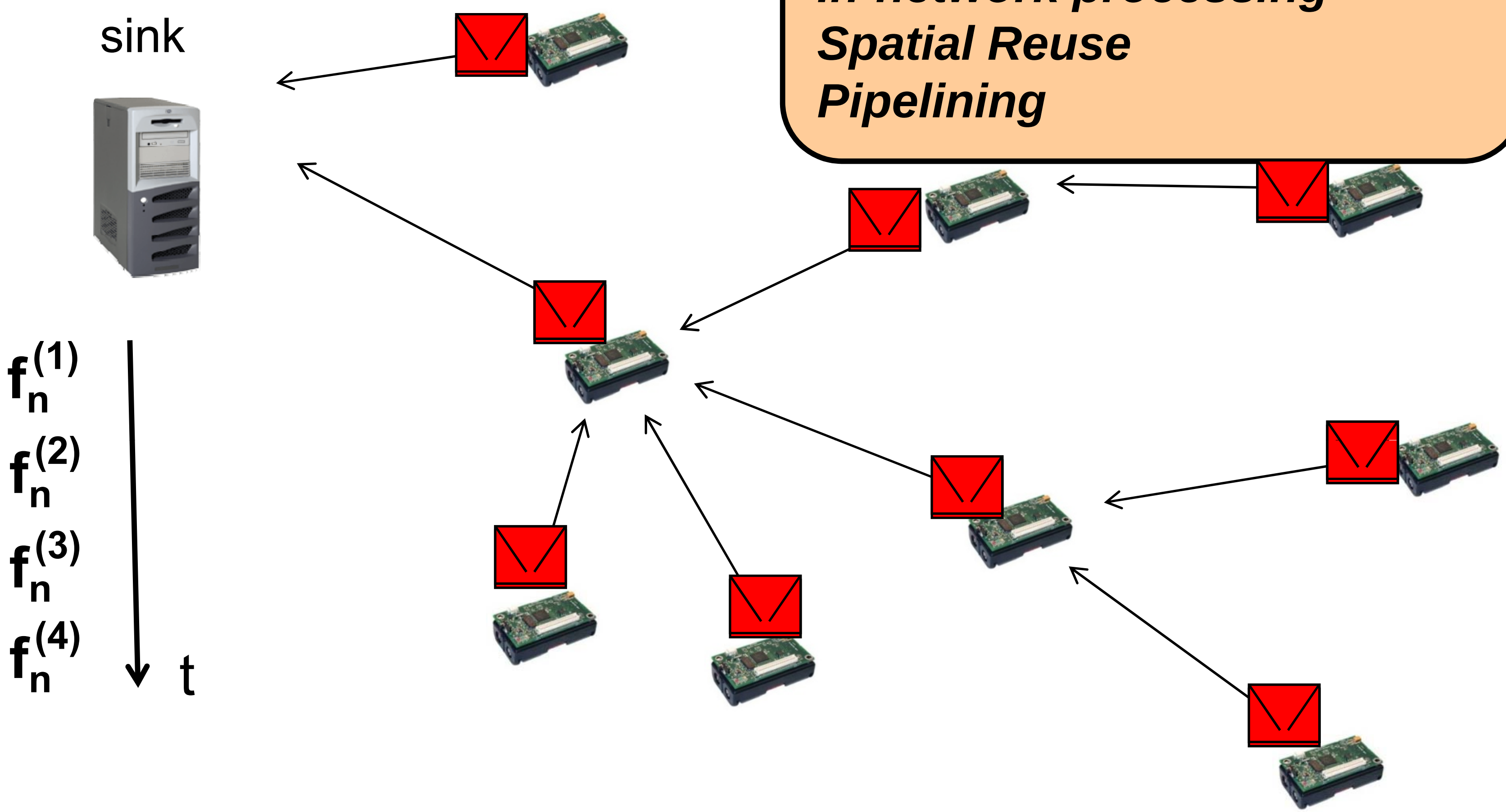
Example: simple round-robin scheme
→ Each sensor reports its results directly to the root one after another



Simple **Round-Robin** Scheme:
→ Sink can compute one function per n rounds
→ **Achieves a *rate* of *1/n***

Data Gathering in Wireless Sensor Networks

There are better schemes using
Multi-hop relaying
In-network processing
Spatial Reuse
Pipelining



Capacity in Wireless Sensor Networks

At what ***rate*** can sensors transmit data to the sink?
Scaling-laws → how does rate decrease as *n* increases…?

$\Theta(1/n)$ $\Theta(1/\sqrt{n})$ $\Theta(1/\log n)$ $\Theta(1)$

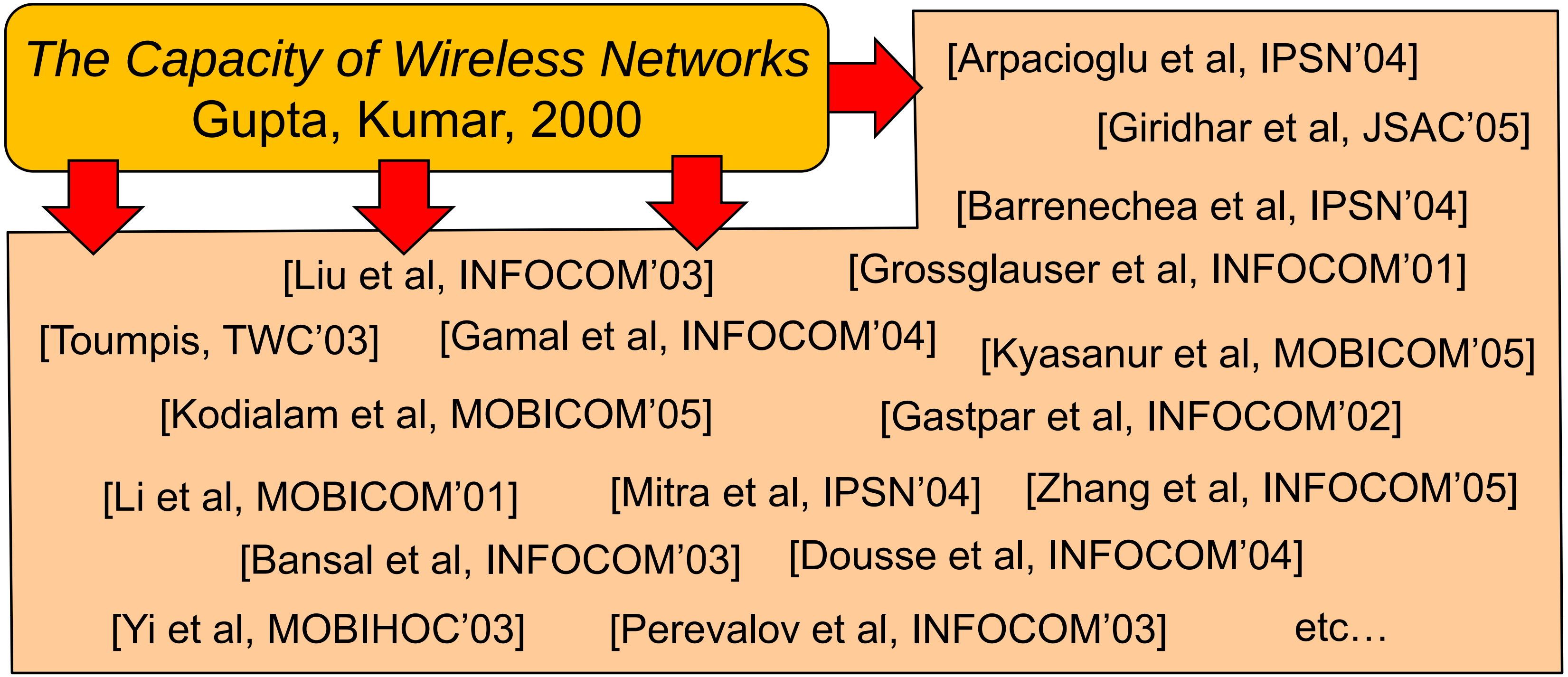
Answer depends on:

Function to be computed → Only perfectly compressible functions (max, min, avg,…)

Coding techniques → No fancy coding techniques

2. Network topology
1. Wireless communication model

“Classic” Capacity…

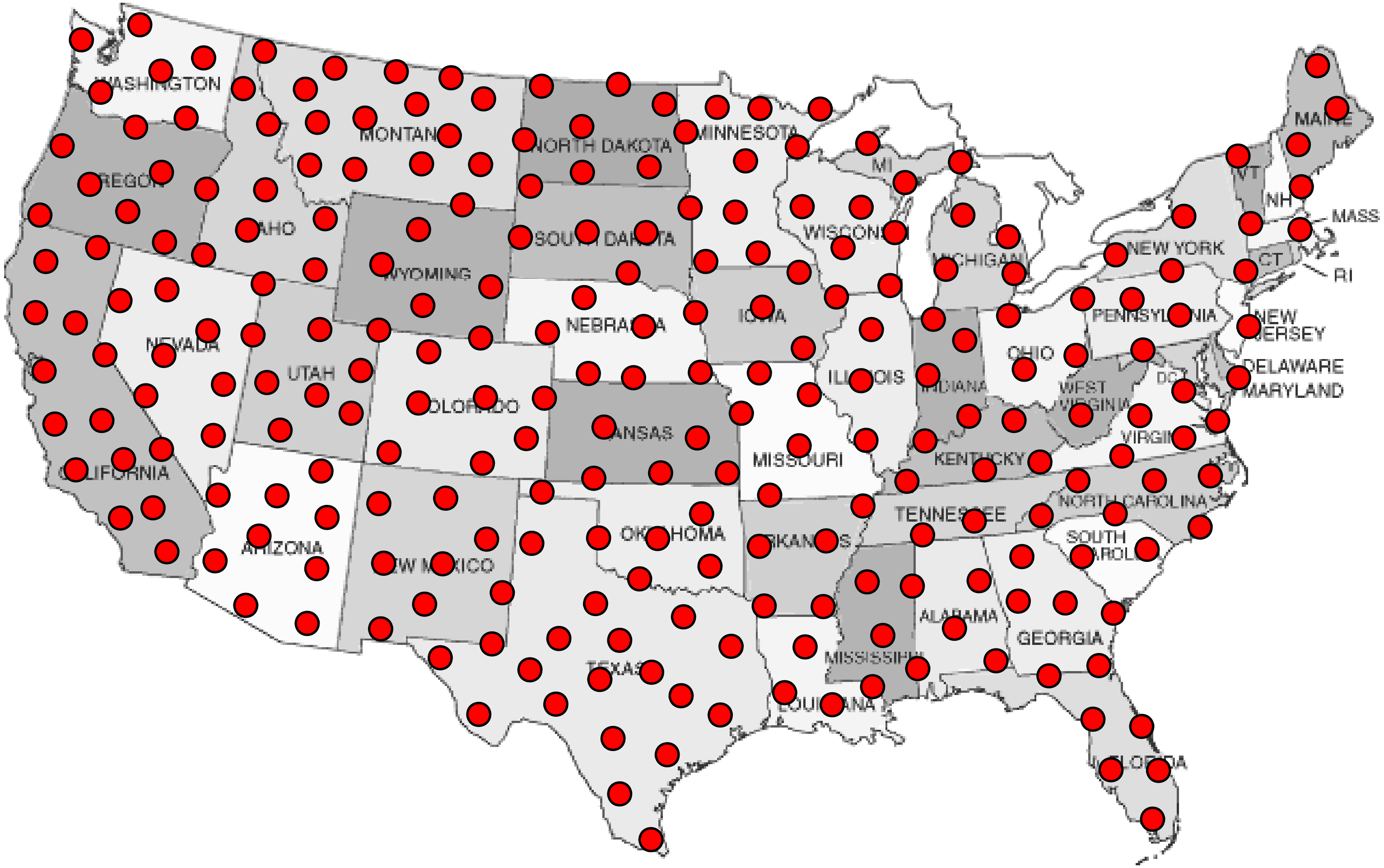


Worst-Case Capacity

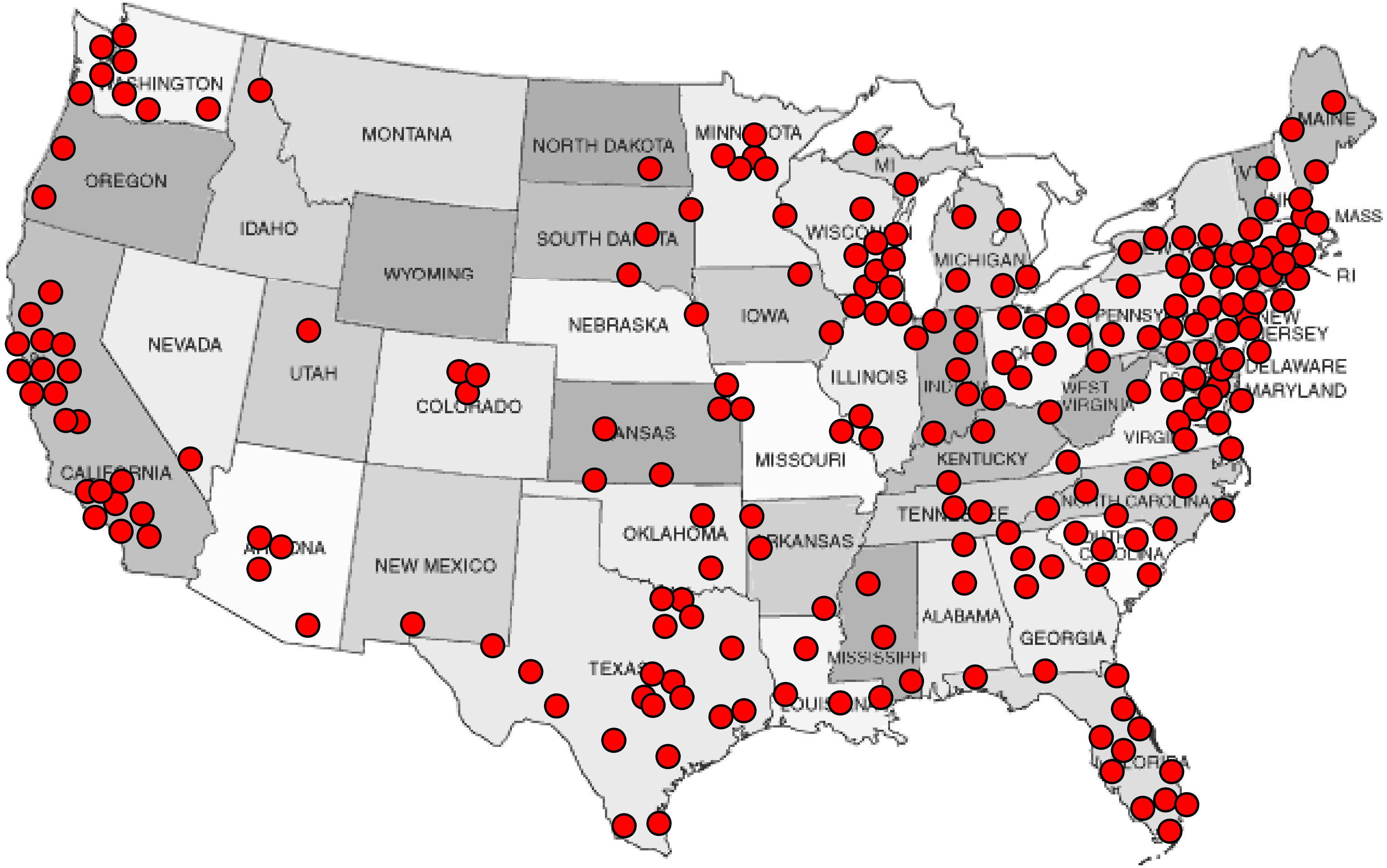
- Capacity studies so far make very strong assumptions on node deployment, topologies
  - randomly, uniformly distributed nodes
  - nodes placed on a grid
  - etc...

What if a network looks differently…?

Like this?


MONTANA
NORTH DAKOTA
MINNESOTA
MI
MAINE
VT
NH
MASS
RI
CT
NEW YORK
WISCONSIN
MICHIGAN
WYOMING
NEBRASKA
IOWA
PENNSYLVANIA
NEW JERSEY
NEVADA
UTAH
OHIO
DELAWARE
MARYLAND
DC
WEST VIRGINIA
INDIANA
KANSAS
MISSOURI
KENTUCKY
TENNESSEE
NORTH CAROLINA
SOUTH
ARIZONA
OKLAHOMA
ALABAMA
GEORGIA
MISSISSIPPI
TEXAS
LOUISIANA
FLORIDA

Or rather like this?



Worst-Case Capacity

- Capacity studies so far have made very strong assumptions on node deployment, topologies
  - randomly, uniformly distributed nodes
  - nodes placed on a grid
  - etc...

What if a network looks differently…?

We assume arbitrary node distribution

worst-case topologies

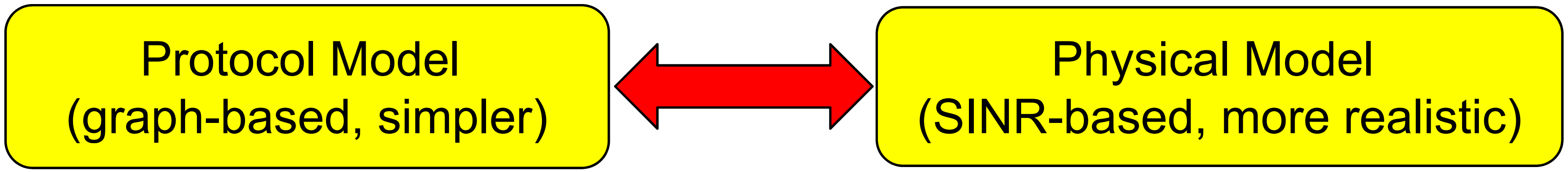
Classic Capacity ⟷ **Worst-Case Capacity**

How much information can be transmitted in nice, well-behaving networks

How much information can be Transmitted in any network

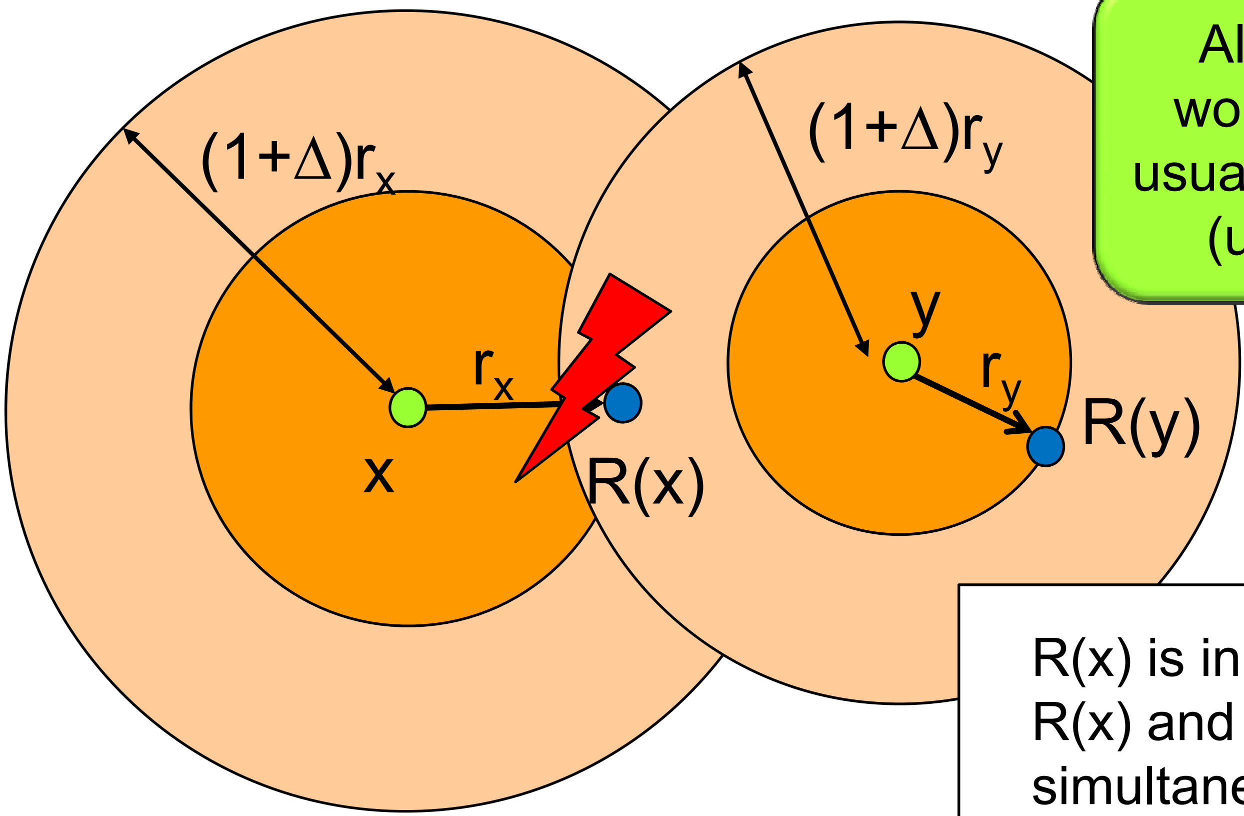
Models

- Two standard models in wireless networking



Protocol Model

- Based on graph-based notion of interference
- Transmission range and interference range



Algorithmic work on worst-case topologies usually in protocol models (unit disk graph,…)

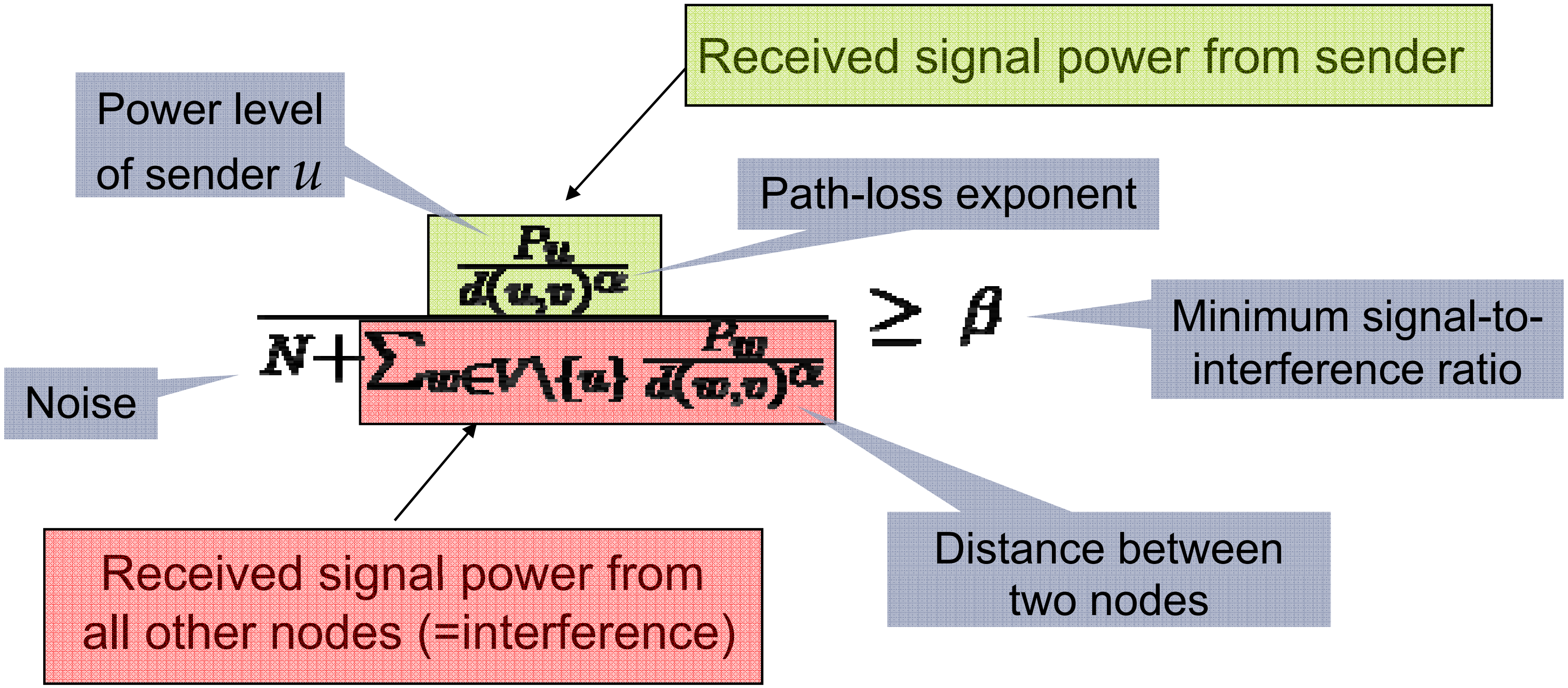
R(x) is in interference range of y
R(x) and R(y) cannot simultaneously receive!

Physical Model

- Based on signal-to-noise-plus-interference (SINR)
- Simplest case:

  → packets can be decoded if SINR is larger than β at receiver

$$\frac{\frac{P_u}{d(u,v)^\alpha}}{N + \sum_{w \in V \setminus \{u\}} \frac{P_w}{d(w,v)^\alpha}} \geq \beta$$



Models

- Two standard models of wireless communication

Protocol Model
(graph-based, simpler)

Physical Model
(SINR-based, more realistic)

- Algorithms typically designed and analyzed in protocol model

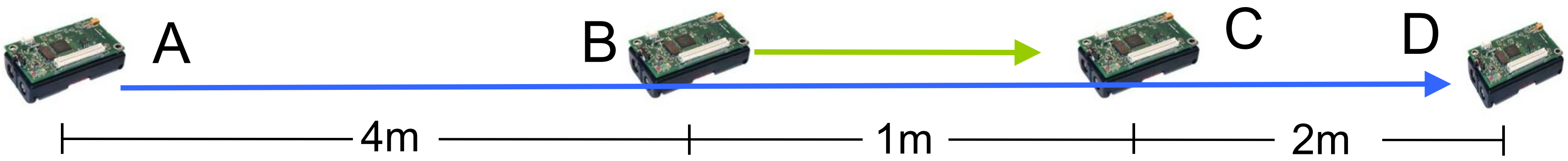
Premise: Results obtained in protocol model do not divert too much from more realistic model!

Justification:

Capacity results are typically (almost) the same in both models
(e.g., Gupta, Kumar, etc...)

Example: Protocol vs. Physical Model

A sends to D, B sends to C



Assume a single frequency (and no fancy decoding techniques!)

Is spatial reuse possible?

- NO — Protocol Model
- YES — Physical Model
  - In Reality!

Let α=3, β=3, and N=10nW

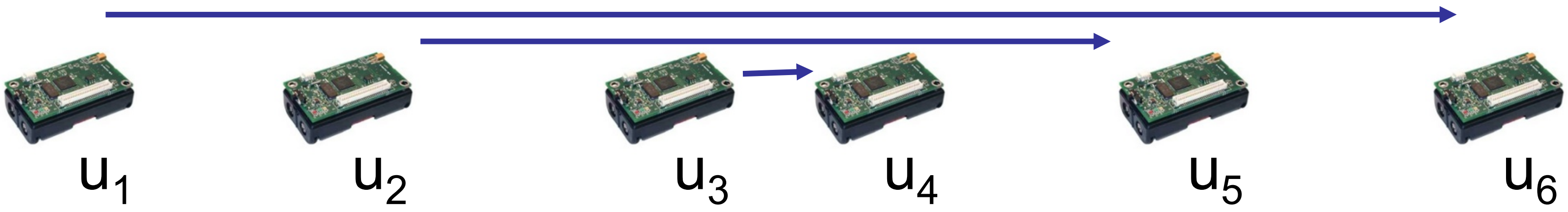
Transmission powers: P_B= -15 dBm and P_A= 1 dBm

SINR of A at D: $$\frac{1.26mW/(7m)^3}{0.01\mu W+31.6\mu W/(3m)^3} \approx 3.11 \geq \beta$$ 

SINR of B at C: $$\frac{31.6\mu W/(1m)^3}{0.01\mu W+1.26mW/(5m)^3} \approx 3.13 \geq \beta$$ 

This works in practice!

- We did measurements using standard mica2 nodes!
- Replaced standard MAC protocol by a (tailor-made) „SINR-MAC“
- Measured for instance the following deployment...



- Time for successfully transmitting 20‘000 packets:

	Time required	
	standard MAC	“SINR-MAC”
Node u_1	$721s$	$267s$
Node u_2	$778s$	$268s$
Node u_3	$780s$	$270s$

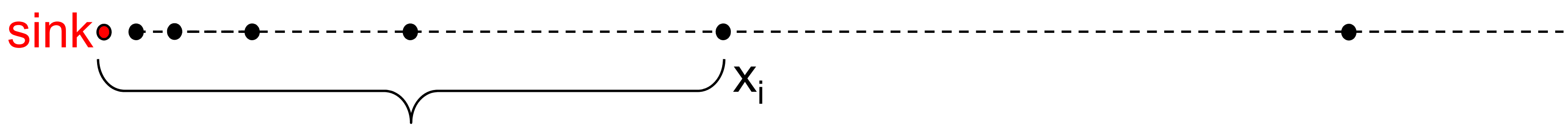
	Messages received	
	standard MAC	“SINR-MAC”
Node u_4	19999	19773
Node u_5	18784	18488
Node u_6	16519	19498

Speed-up is almost a factor 3

[Moscibroda, Wattenhofer, Weber, Hotnets’06]

Upper Bound Protocol Model

- There are networks, in which at most one node can transmit!
  → like round-robin
- Consider exponential node chain
- Assume nodes can choose arbitrary transmission power



$d(\text{sink}, x_i) = (1+1/\Delta)^{i-1}$

- Whenever a node transmits to another node
  → All nodes to its left are in its interference range!
  → Network behaves like a single-hop network

In the **protocol model**, the achievable rate is $\Theta(1/n)$.

Lower Bound Physical Model

- Much better bounds in SINR-based physical model are possible (exponential gap)
- Paper presents a scheduling algorithm that achieves a rate of $\Omega(1/\log^3 n)$

In the **physical model**, the achievable rate is $\Omega(1/\text{polylog}\, n)$.

- Algorithm is centralized, highly complex → not practical
- But it shows that high rates are possible even in worst-case networks

- Basic idea: Enable spatial reuse by exploiting SINR effects.

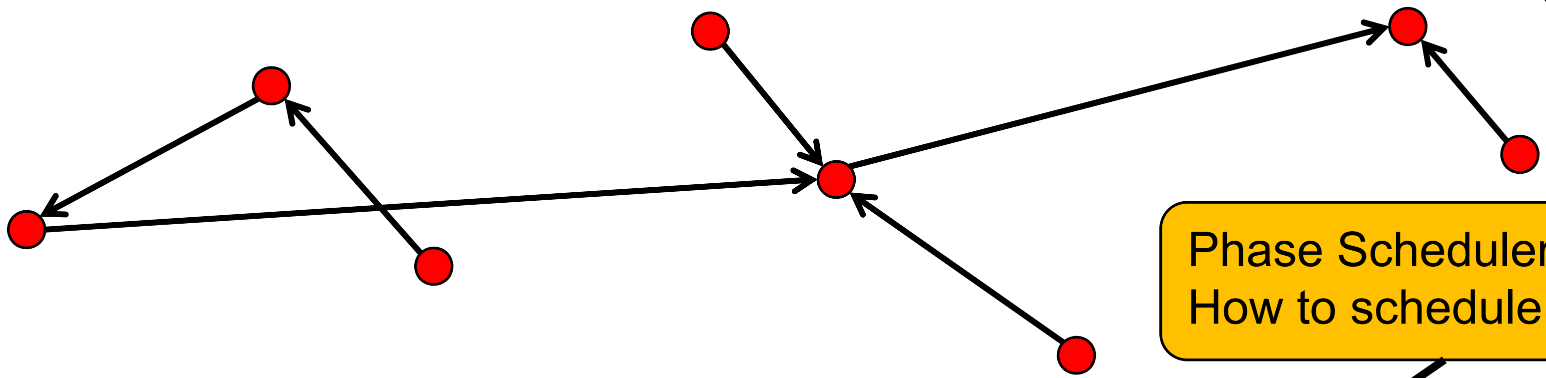
Scheduling Algorithm – High Level Procedure

- High-level idea is simple
- Construct a hierarchical tree T(X) that has desirable properties

1) Initially, each node is active
2) Each node connects to closest active node
3) Break cycles → yields forest
4) Only root of each tree remains active

loop until no active nodes

Can be adjusted if transmission power limited

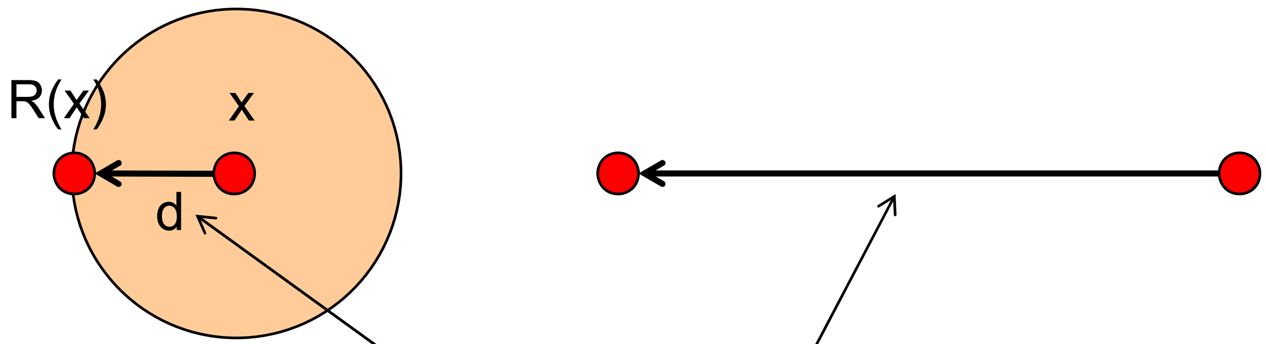


Phase Scheduler:
How to schedule T(X)?

The resulting structure has some nice properties
→ If each link of T(X) can be scheduled at least once in L(X) time-slots
→ Then, a rate of 1/L(X) can be achieved

Scheduling Algorithm – Phase Scheduler

- How to schedule T(X) efficiently
- We need to schedule links of different magnitude simultaneously!
- Only possibility:
  senders of small links must overpower their receiver!



If we want to schedule both links…

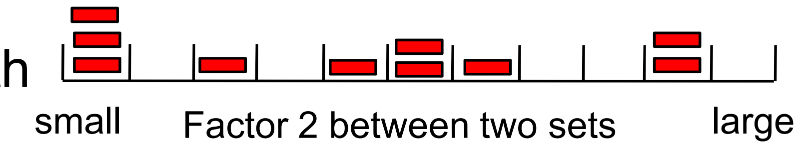
1) … R(x) must be overpowered
→ Must transmit at power more than ~d^{α}

2) If senders of small links overpower their receiver…
… their "safety radius" increases (spatial reuse smaller)

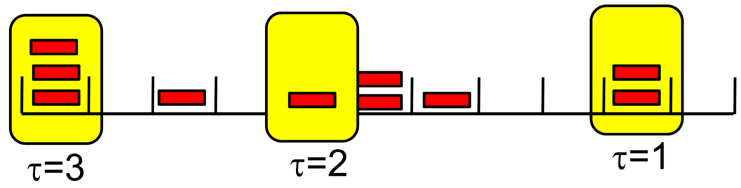
Subtle balance is needed!

Scheduling Algorithm – Phase Scheduler

1) Partition links into sets of similar length



2) Group sets such that links a and b in two sets in the same group have at least $d_a \geq (\xi\beta)^{\xi(\tau a - \tau b)} \cdot d_b$



→ Each link gets a τ_{ij} value → Small links have large τ_{ij} and vice versa

→ Schedule links in these sets in one outer-loop iteration

→ Intuition: Schedule links of similar length or very different length

3) Schedule links in a group → Consider in order of decreasing length
(I will not show details because of time constraints.)

Together with structure of T(x) → $\Omega(1/\log^3 n)$ bound

Worst-Case Capacity in Wireless Networks

Worst-Case Capacity

Traditional Capacity

Networks / Model	Max. rate in arbitrary, worst-case deployment	Max. rate in random, uniform deployment
Protocol Model	$\Theta(1/n)$	$\Theta(1/\log n)$
Physical Model	$\Omega(1/\log^3 n)$	$\Omega(1/\log n)$

[Giridhar, Kumar, 2005]

Exponential gap between protocol and physical model!

<u>The Price of Worst-Case Node Placement</u>

- Exponential in protocol model
- Polylogarithmic in physical model (almost no worst-case penalty!)

Conclusions

- Introduce worst-case capacity of sensor networks
  → How much data can periodically be sent to data sink

- Complements existing capacity studies
- Many novel insights

1) Possibilities and limitations of wireless communication
2) Fundamentals of wireless communication models
3) How to devise efficient scheduling algorithms, protocols

Sensor Networks Scale!
Efficient data gathering is possible in every (even worst-case) network!

Protocol Model Poor!
Exponential gap between protocol and physical model!

Efficient Protocols!
Must use SINR-effects and power control to achieve high rate!

Overview of results so far

- Moscibroda, Wattenhofer, Infocom 2006
  - First paper in this area, $O(\log^3 n)$ bound for connectivity, and more
  - This is essentially the paper I presented on the previous slides
- Moscibroda, Wattenhofer, Zollinger, MobiHoc 2006
  - First results beyond connectivity, namely in the topology control domain
- Moscibroda, Wattenhofer, Weber, HotNets 2006
  - Practical experiments, ideas for capacity-improving protocol
- Moscibroda, Oswald, Wattenhofer, Infocom 2007
  - Generalizion of Infocom 2006, proof that known algorithms perform poorly
- Goussevskaia, Oswald, Wattenhofer, MobiHoc 2007
  - Hardness results & constant approximation for constant power
- Chafekar, Kumar, Marathe, Parthasarathy, Srinivasan, MobiHoc 2007
  - Cross layer analysis for scheduling and routing
- Moscibroda, IPSN 2007
  - Connection to data gathering, improved $O(\log^2 n)$ result
- Locher, von Rickenbach, Wattenhofer, ICDCN 2008
  - Still some major open problems

Main open question in this area

- Most papers so far deal with special cases, essentially scheduling a number of links with special properties. The general problem is still wide open:

- A communication request consists of a source and a destination, which are arbitrary points in the Euclidean plane. Given *n* communication requests, assign a color (time slot) to each request. For all requests sharing the same color specify power levels such that each request can be handled correctly, i.e., the SINR condition is met at all destinations. The goal is to minimize the number of colors.

- E.g., for arbitrary power levels not even hardness is known…

Overview

- Introduction
- Applications
- Data Gathering

- Minimizing Messages with Aggregation
- Minimizing Time with Power Control
- Minimizing Energy Consumption with Sleep Schedules (DC & N?)

- Conclusion

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

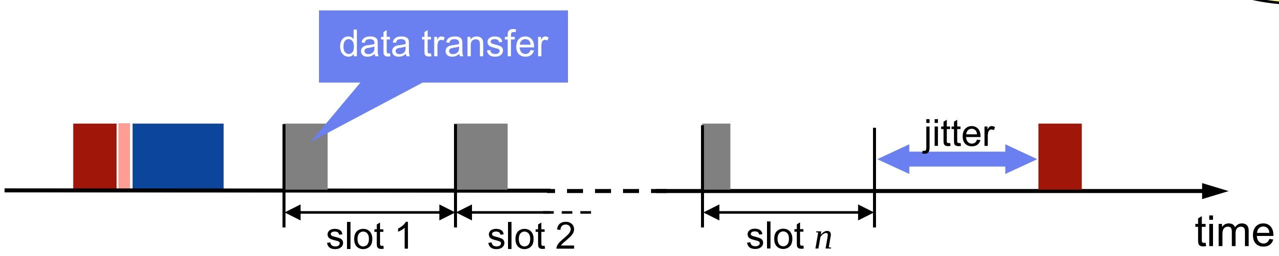
- The parent initiates each TDMA round with a beacon
  - Enables integration of disconnected nodes
  - Children tune in to their parent's schedule

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

Clock drift, queuing, bootstrap, etc.



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

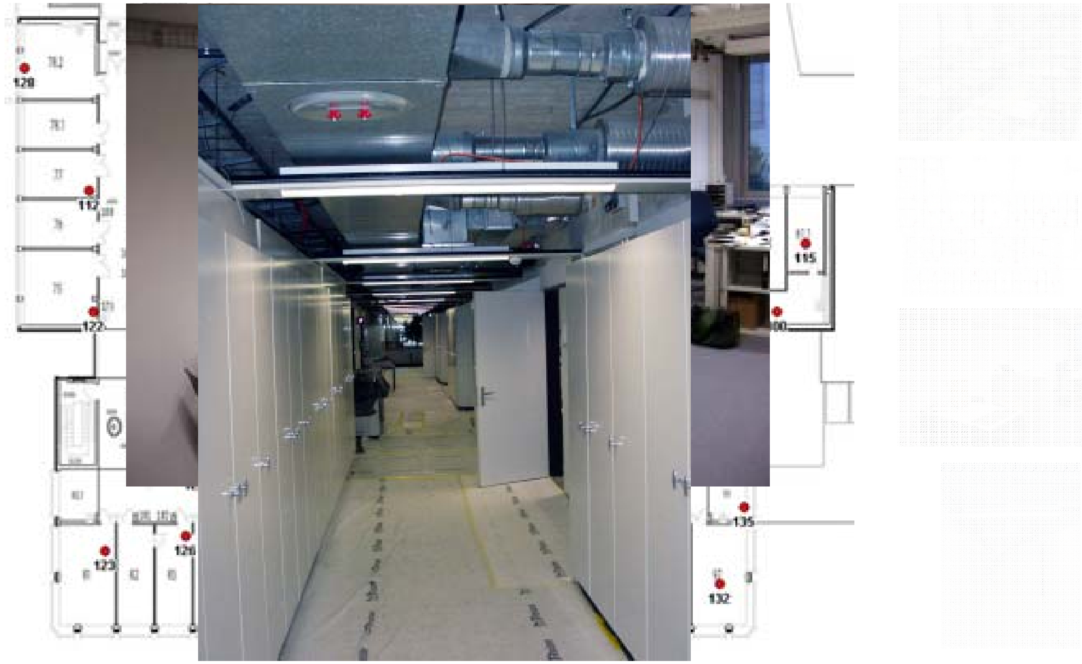
Dozer System

- Clock drift compensation
  - First fixed guard times
  - Later improved versions

- Application scheduling
  - TinyOS is single threaded and non-preemptive
  - TDMA is highly time critical
- Queuing strategy
  - Fixed size buffers

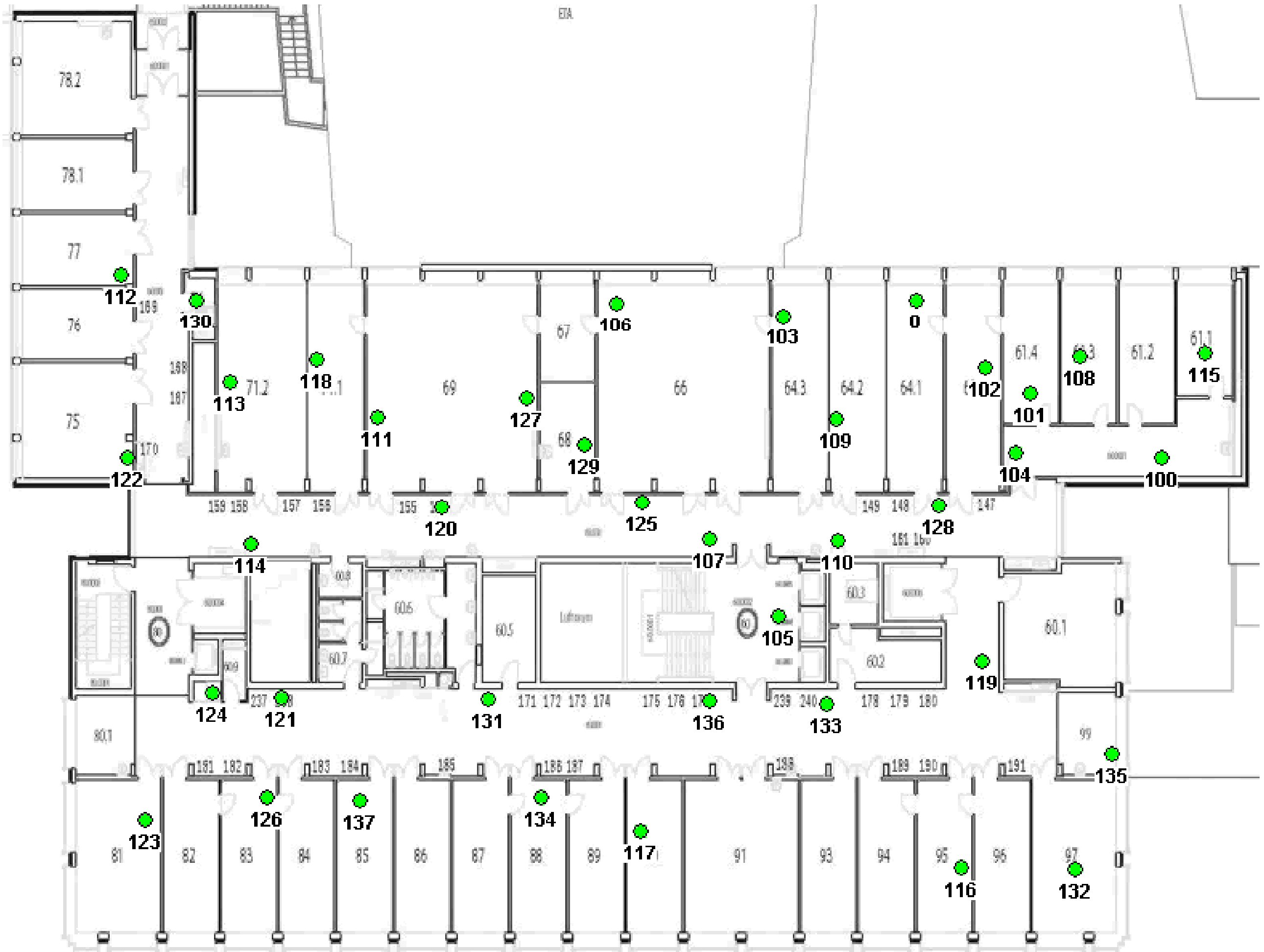
Evaluation

- 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

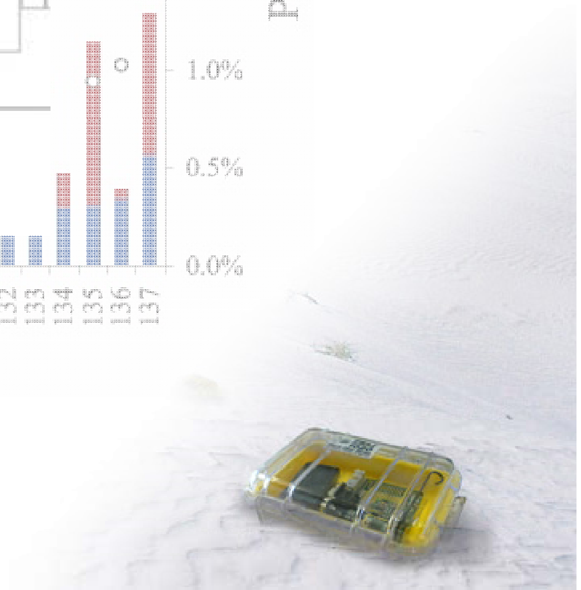
Dozer in Action



Tree Maintenance

1 week of operation

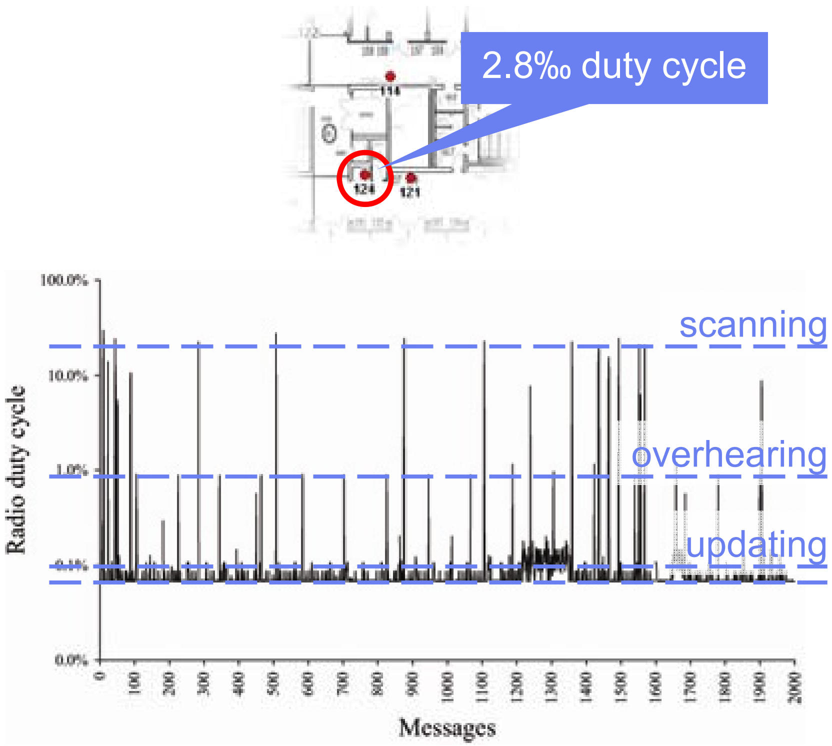
on average 1.2%

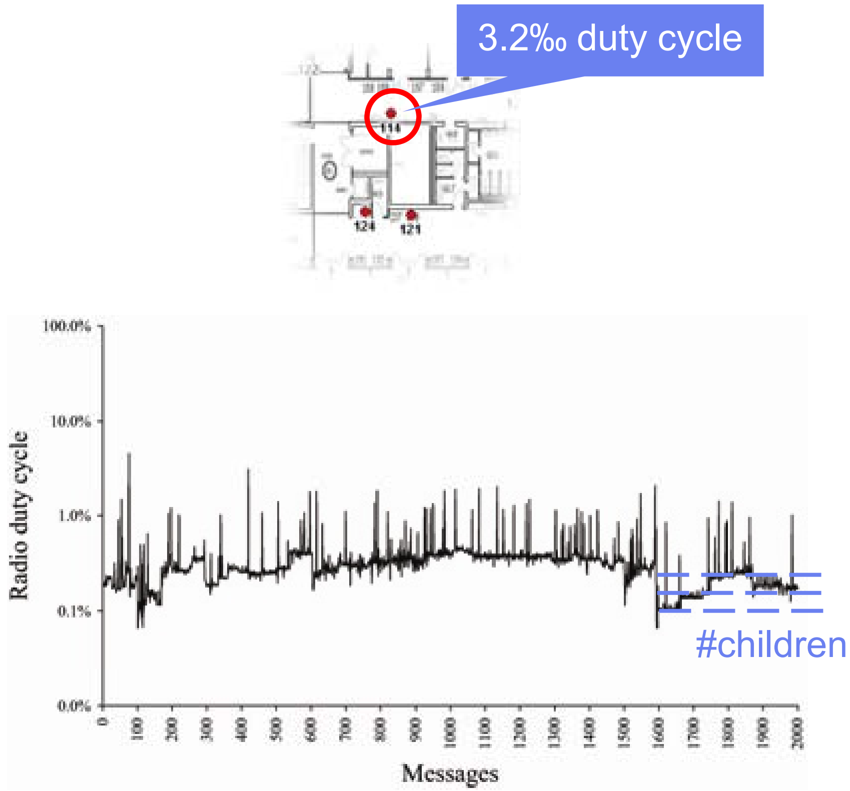


Energy Consumption

➡ Mean energy consumption of 0.082 mW

Energy Consumption





- Leaf node
- Few neighbors
- Short disruptions

- Relay node
- No scanning

Overview

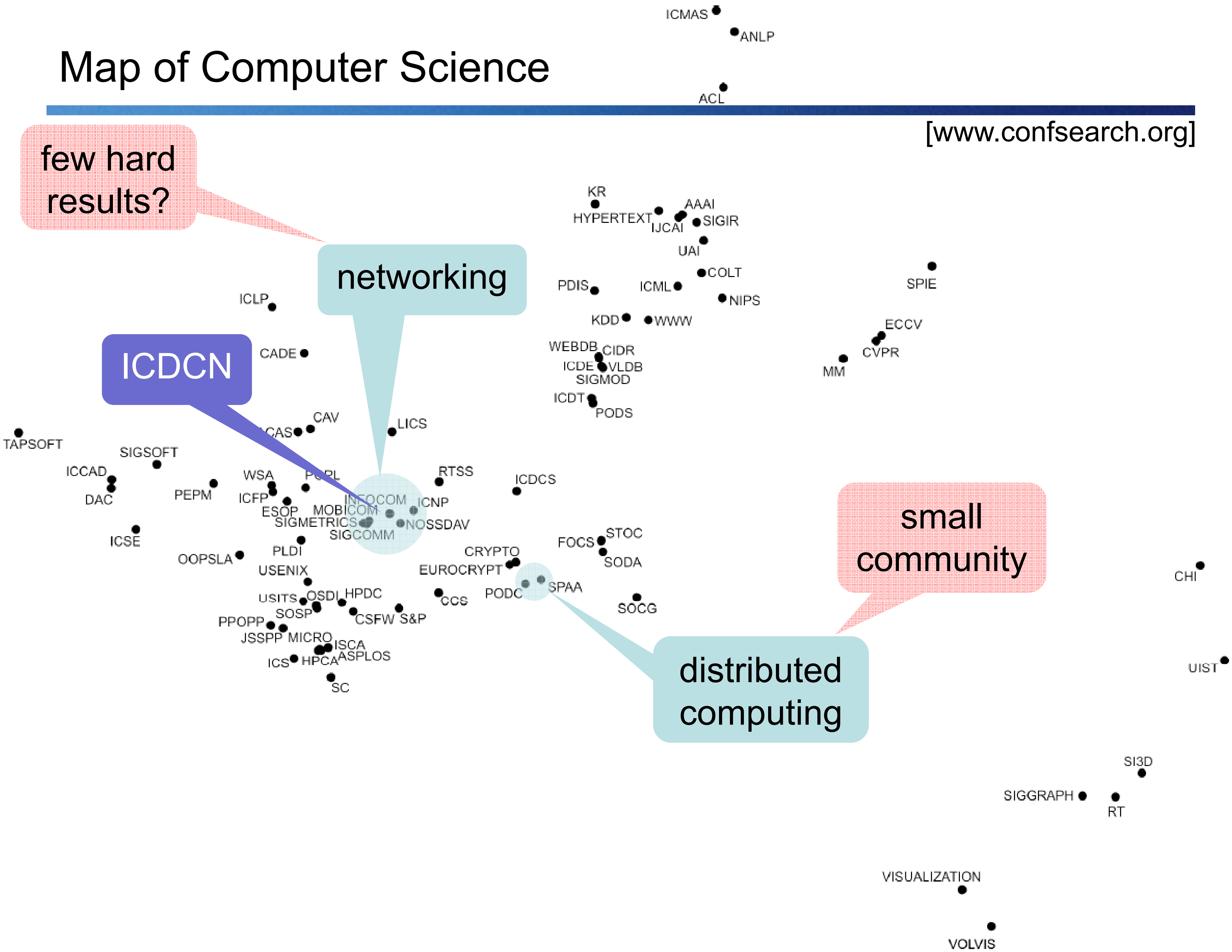
- Introduction
- Applications
- Data Gathering

- Minimizing Messages with Aggregation
- Minimizing Time with Power Control
- Minimizing Energy Consumption with Sleep Schedules (DC & N?)

- Conclusion

Map of Computer Science

[www.confsearch.org]



My Own Private View on Networking Research

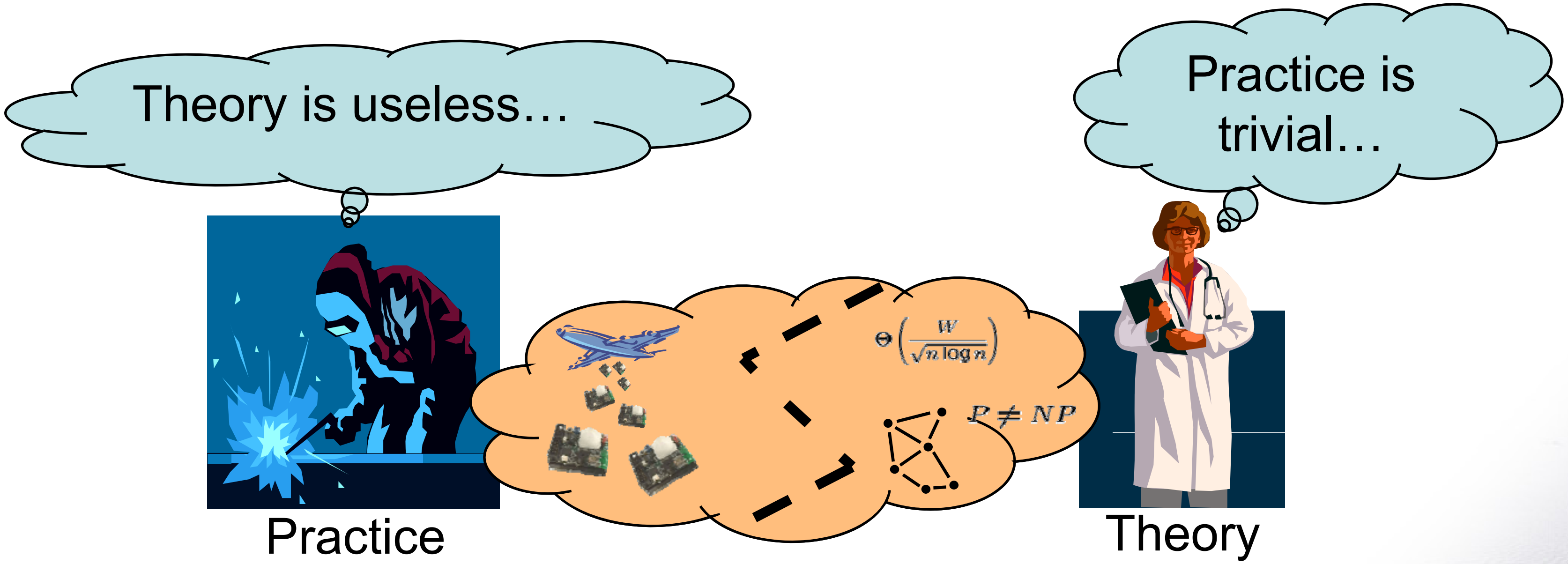
Class	Analysis	Communication model	Node distribution	Other drawbacks	Popularity
Implementation	Testbed	Reality	Reality(?)	"Too specific"	5%
Heuristic	Simulation	UDG to SINR	Random, and more	Many…! (no benchmarks)	80%
Scaling law	Theorem/ proof	SINR, and more	Random	Existential (no protocols)	10%
Algorithm	Theorem/ proof	UDG, and more	Any (worst-case)	Worst-case unusual	5%

I'm here!

In other words, I'm applying distributed computing methods to networking problems!

Conclusions

- We have seen three stories about data gathering, arguably a main task of 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 theory and practice, one shall not be surprised that the two camps largely ignore each other.

Thank You!

Questions & Comments?

Papers
Locher, Kuhn, Wattenhofer [SPAA 2007]
Moscibroda, Wattenhofer [INFOCOM 2006]
Burri, von Rickenbach, Wattenhofer [IPSN 2007]
plus a few more