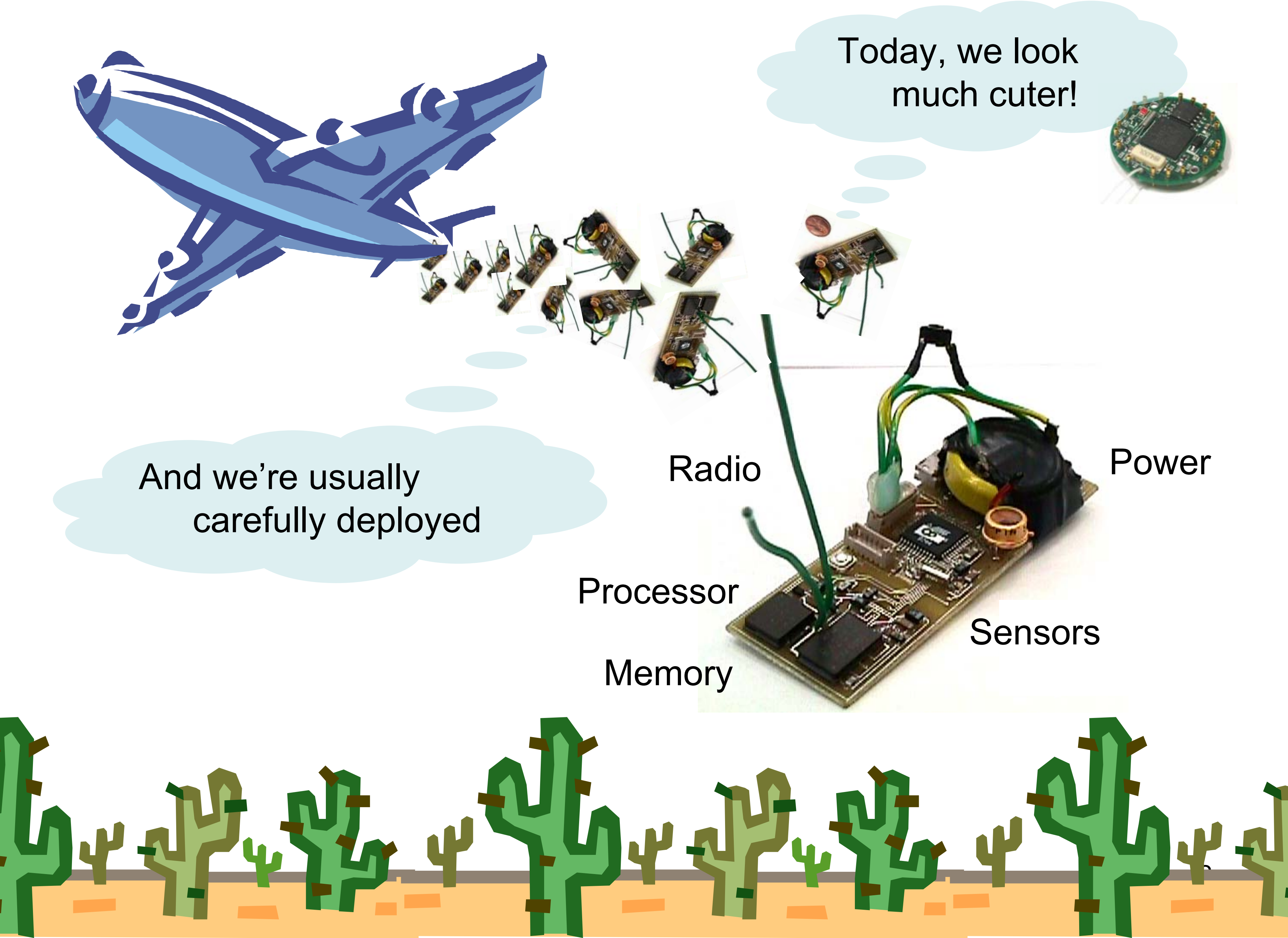
Sensor Networks

Distributed Algorithms Reloaded or *Revolutions*?

Roger Wattenhofer

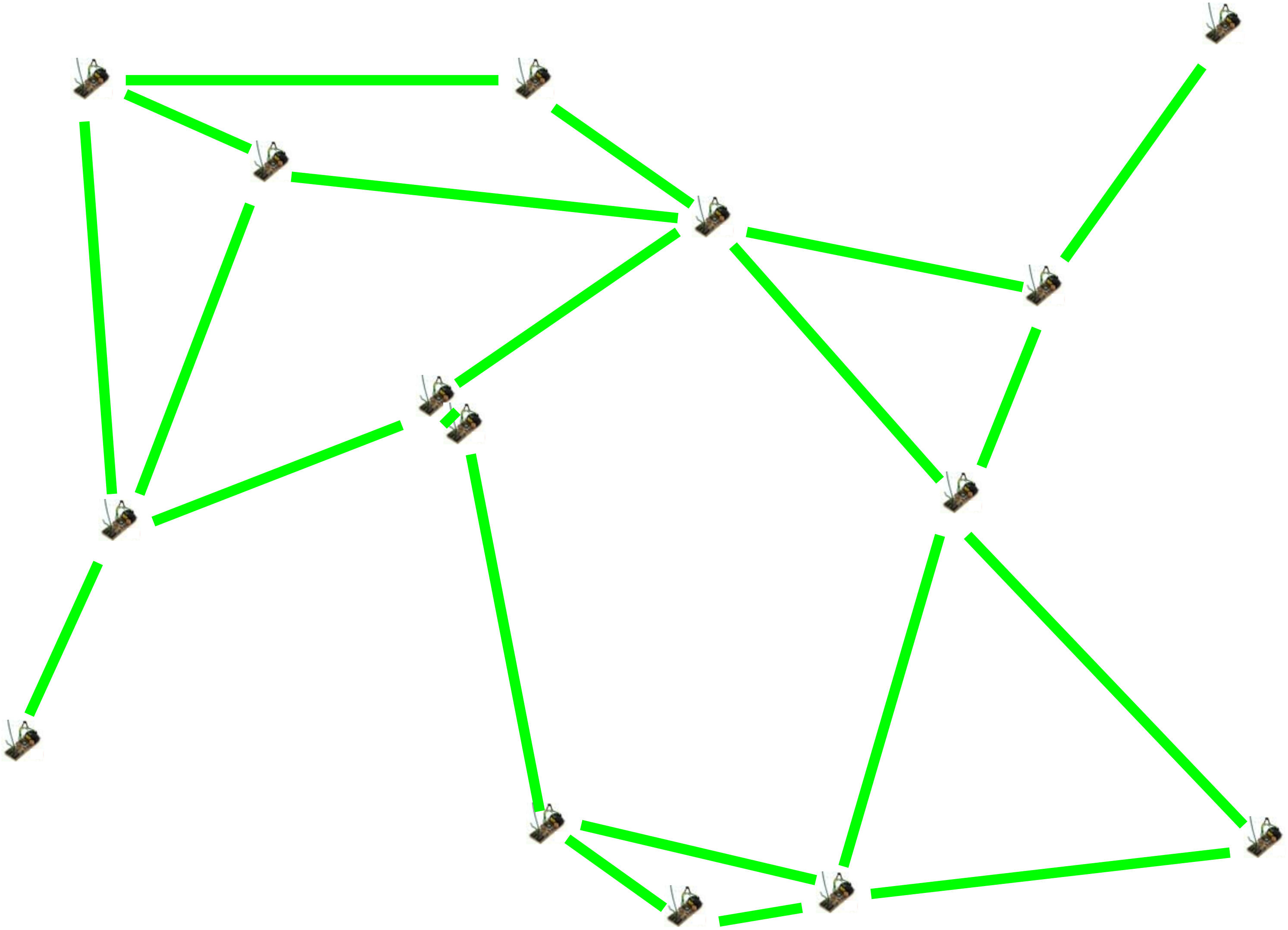
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich



Today, we look much cuter!

And we’re usually carefully deployed

Distributed (Network) Algorithms



Ad Hoc Networks	vs. Sensor Networks
Laptops, PDA's, cars, soldiers	Tiny nodes: 4 MHz, 32 kB, …
All-to-all routing	Broadcast/Echo from/to sink
Often with mobility (MANET's)	Usually no mobility – but link failures
Trust/Security an issue – No central coordinator	One administrative control
Maybe high bandwidth	Long lifetime → Energy

Reloaded or Revolutions?

- Reloaded
  - Distributed (message passing) algorithms
  - Message complexity → Support for energy efficiency
  - Time complexity → Support for dynamics

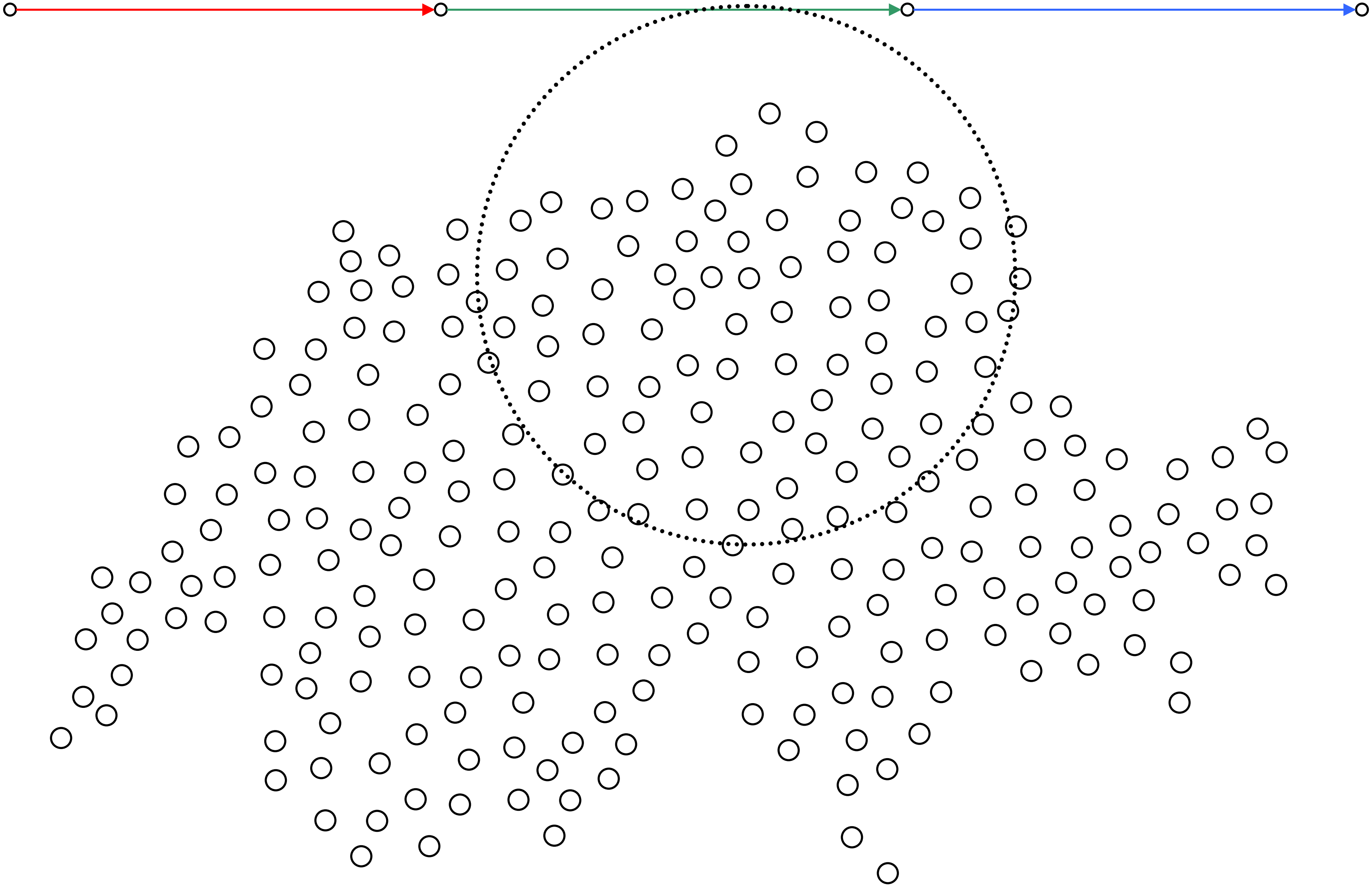
- Revolutions
  - Wireless → Interference issues → Not standard message passing, but new types of distributed algorithms
  - Wireless → New types of connectivity/interference graphs?

- Finally an application that can't live without state-of-the-art distributed algorithms?!

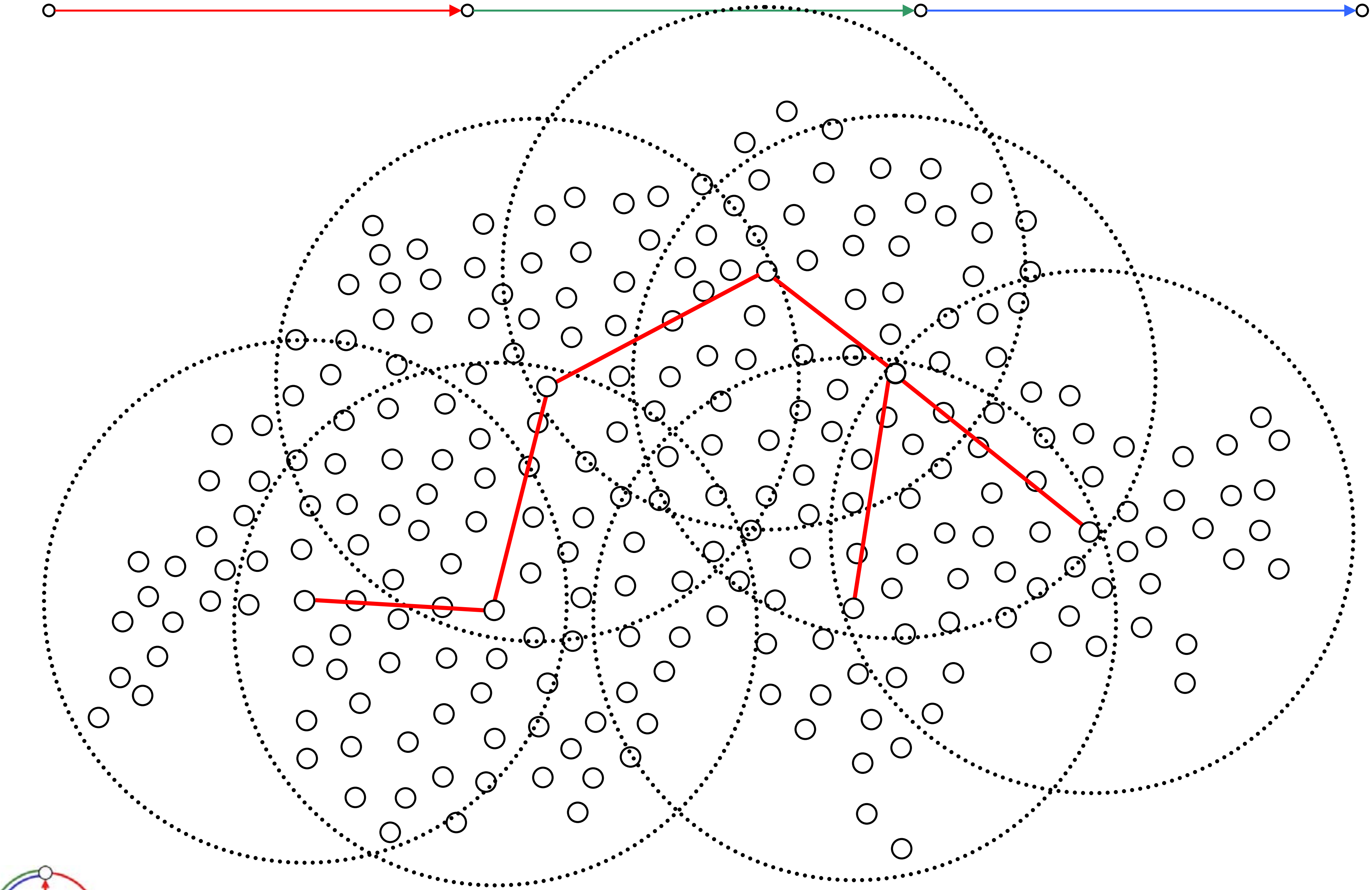
Overview

- Introduction
- Algorithmic Models: Case Study Clustering
  - Flooding vs. Dominating Sets
  - Non-Trivial Algorithm
  - Lower Bounds
  - Model Discussion
- Communication Models: Case Study Scheduling
- Conclusions

Finding a Destination by Flooding



Finding a Destination *Efficiently*

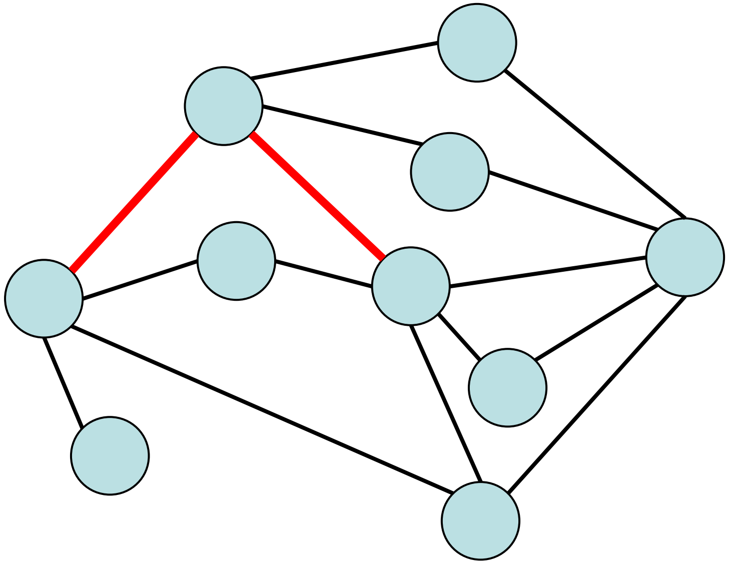


(Connected) Dominating Set

- A Dominating Set DS is a subset of nodes such that each node is either in DS or has a neighbor in DS.

- A Connected Dominating Set CDS is a connected DS, that is, there is a path between any two nodes in CDS that does not use nodes that are not in CDS.

- It might be favorable to have few nodes in the (C)DS. This is known as the Minimum (C)DS problem.



Formal Problem Definition: M(C)DS

- Input: We are given an (arbitrary) undirected graph.

- Output: Find a Minimum (Connected) Dominating Set, that is, a (C)DS with a minimum number of nodes.

- Problems
  - M(C)DS is NP-hard
  - Find a (C)DS that is “close” to minimum (approximation)
  - The solution must be local (global solutions are impractical for mobile ad-hoc network) – topology of graph “far away” should not influence decision who belongs to (C)DS

A Simple “Localized” Algorithm

- Classic greedy algorithm:

- “Always choose node with most non-dominated neighbors.”
- The solution is a log-approximation (which is asymptotically optimal, unless P $\approx$ NP).

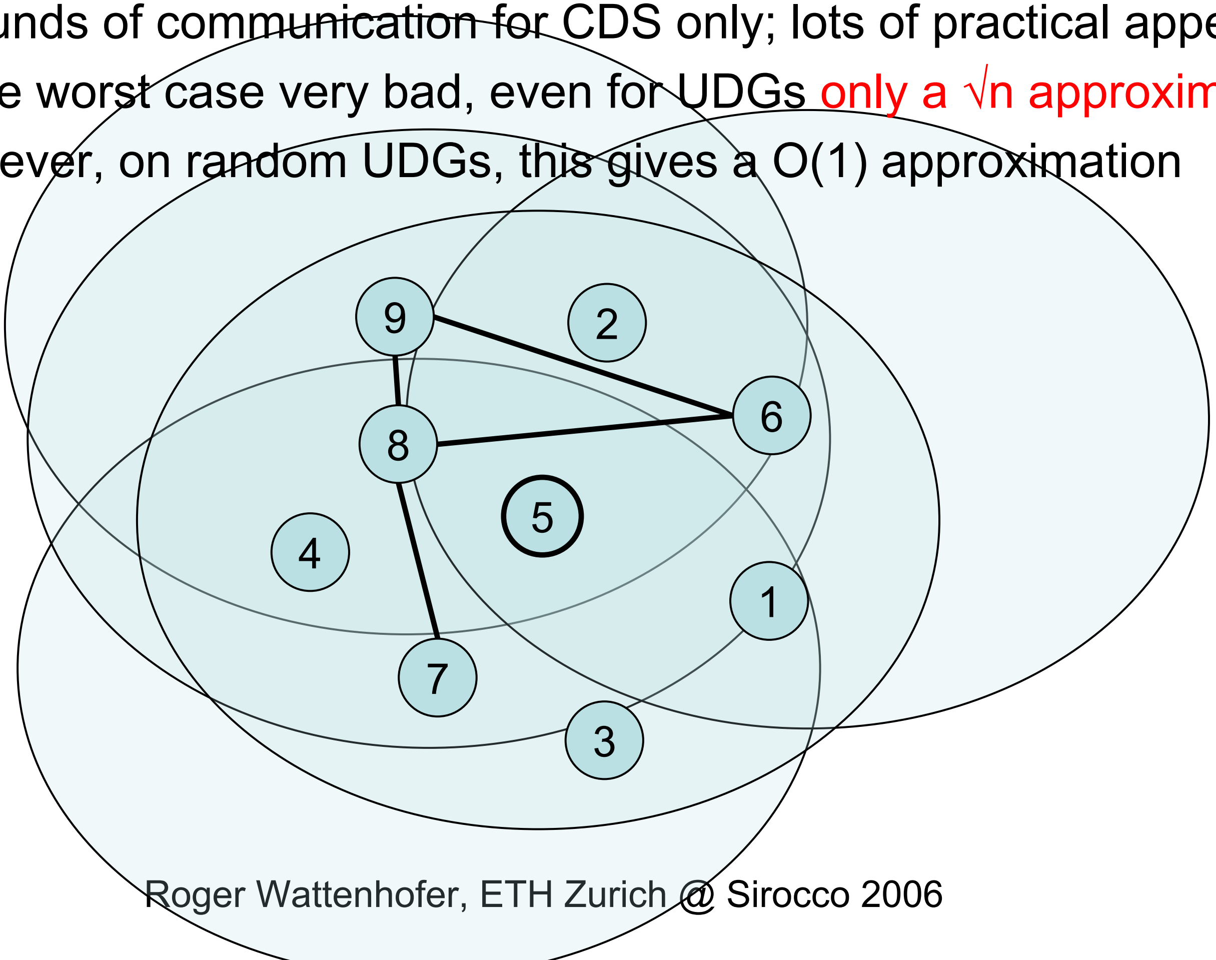
- Distributed version:

1. Wait until higher-degree (same degree: higher-ID) neighbors have decided not to join dominating set.
2. Join dominating set and tell neighbors.

- Problem: This algorithm can have a linear waiting chain. Too slow!

A Simple "Local" Algorithm

- If higher priority neighbors are connected and cover all other neighbors, then don't join CDS, else join CDS
  - This talk, inspired by an improvement of Jie Wu
  - 2 rounds of communication for CDS only; lots of practical appeal
  - In the worst case very bad, even for UDGs only a √n approximation
  - However, on random UDGs, this gives a O(1) approximation



Overview

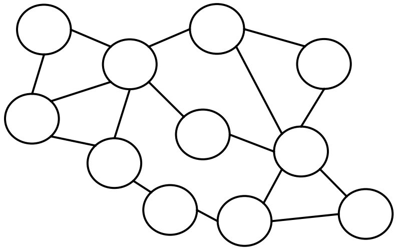
- Introduction

- Algorithmic Models: Case Study Clustering
  - Flooding vs. Dominating Sets
  - Non-Trivial Algorithm
  - Lower Bounds
  - Model Discussion

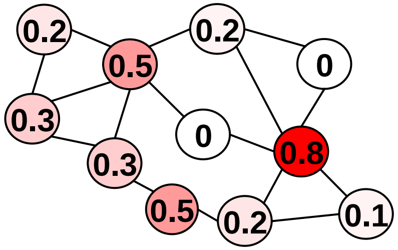
- Communication Models: Case Study Scheduling
- Conclusions

Algorithm Summary

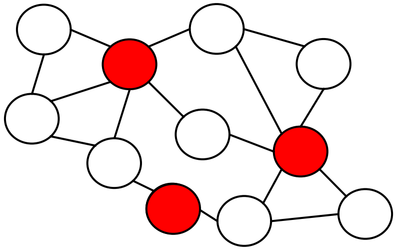
Input:
Local Graph



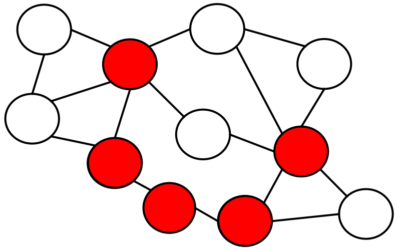
Fractional
Dominating Set



Dominating
Set



Connected
Dominating Set



Phase A:
Distributed
linear program
rel. high degree
gives high value

Phase B:
Probabilistic
algorithm

Phase C:
Connect DS
by “tree” of
“bridges”

Results

- First time/approximation tradeoff. First algorithm which achieves a non-trivial approximation ratio in constant time (even for UDG!) [Kuhn, Wattenhofer, PODC 2003]

- Improved version [Kuhn, Moscibroda, Wattenhofer, SODA 2006]
  - $O(\log^2\Delta / \varepsilon^4)$ time for a $(1+\varepsilon)$-approximation of phase A with logarithmic sized messages.
  - An improved and generalized distributed randomized rounding technique for phase B (constant time, logarithmic approximation)
  - Works for quite general linear programs.

- Is it any good…?

Overview

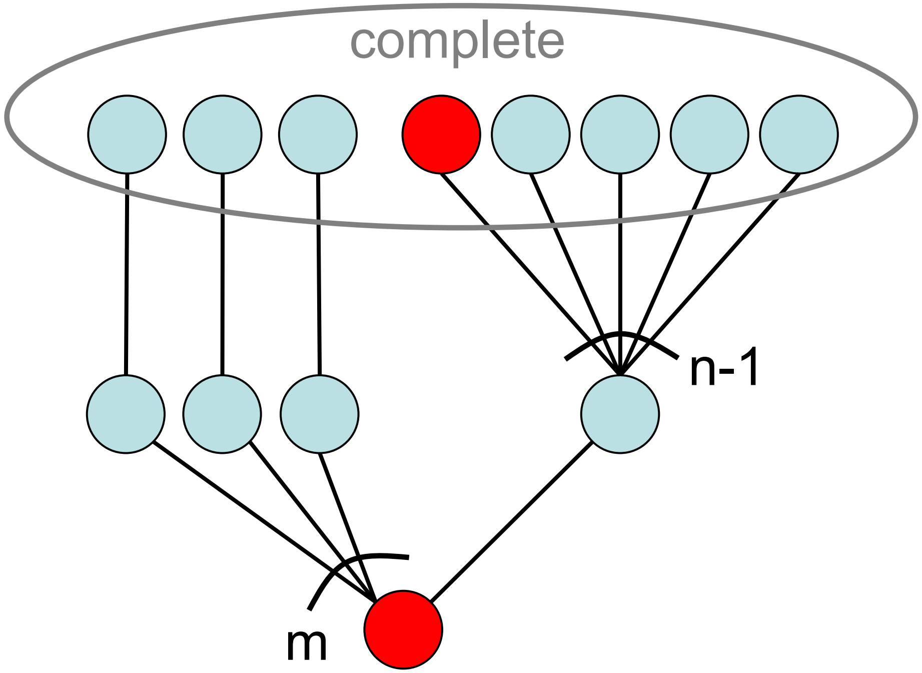
- Introduction

- Algorithmic Models: Case Study Clustering
  - Flooding vs. Dominating Sets
  - Non-Trivial Algorithm
  - Lower Bounds
  - Model Discussion

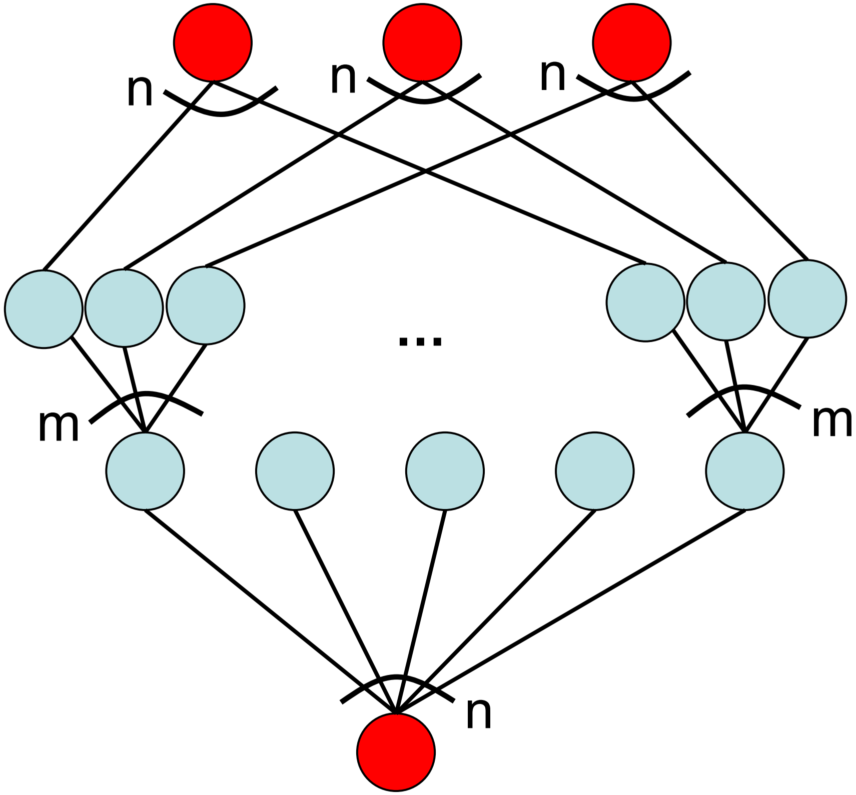
- Communication Models: Case Study Scheduling
- Conclusions

Lower Bound for Dominating Sets: Intuition…

- Two graphs (m << n). Optimal dominating sets are marked red.



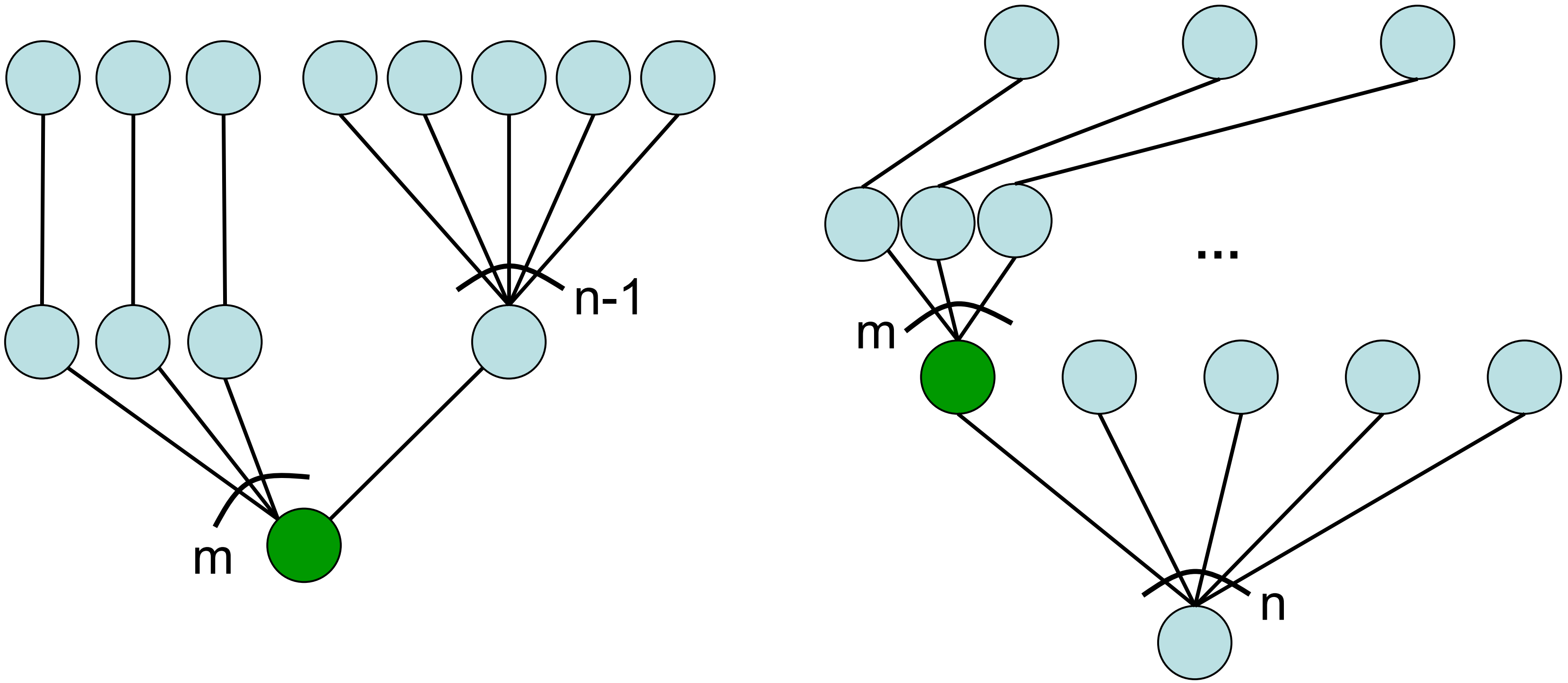
$|DS_{OPT}| = 2.$



$|DS_{OPT}| = m+1.$

Lower Bound for Dominating Sets: Intuition…

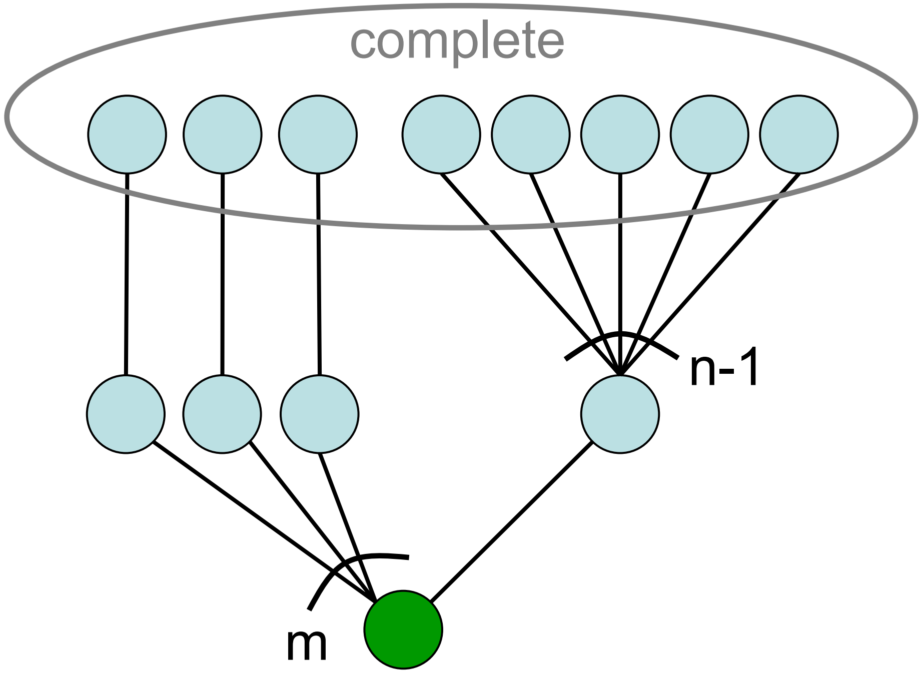
- In local algorithms, nodes must decide only using local knowledge.
- In the example green nodes see exactly the same neighborhood.

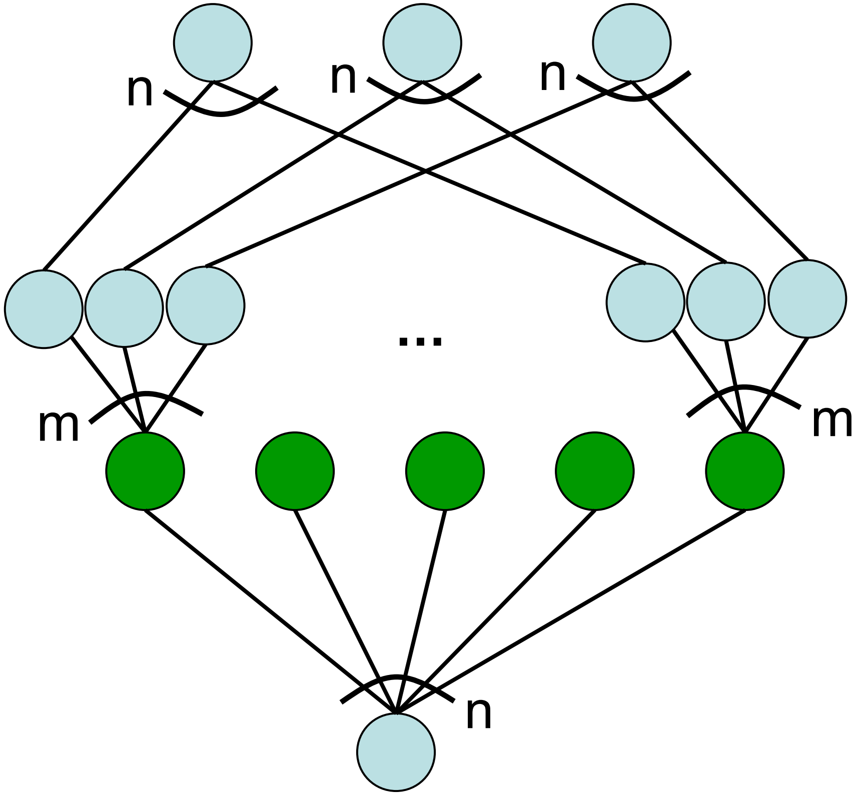


- So these green nodes must decide the same way!

Lower Bound for Dominating Sets: Intuition…

- But however they decide, one way will be devastating (with $n = m^2$)!





$|DS_{OPT}| = 2.$

$|DS_{OPT\text{ without green}}| \geq m.$

$|DS_{OPT}| = m+1.$

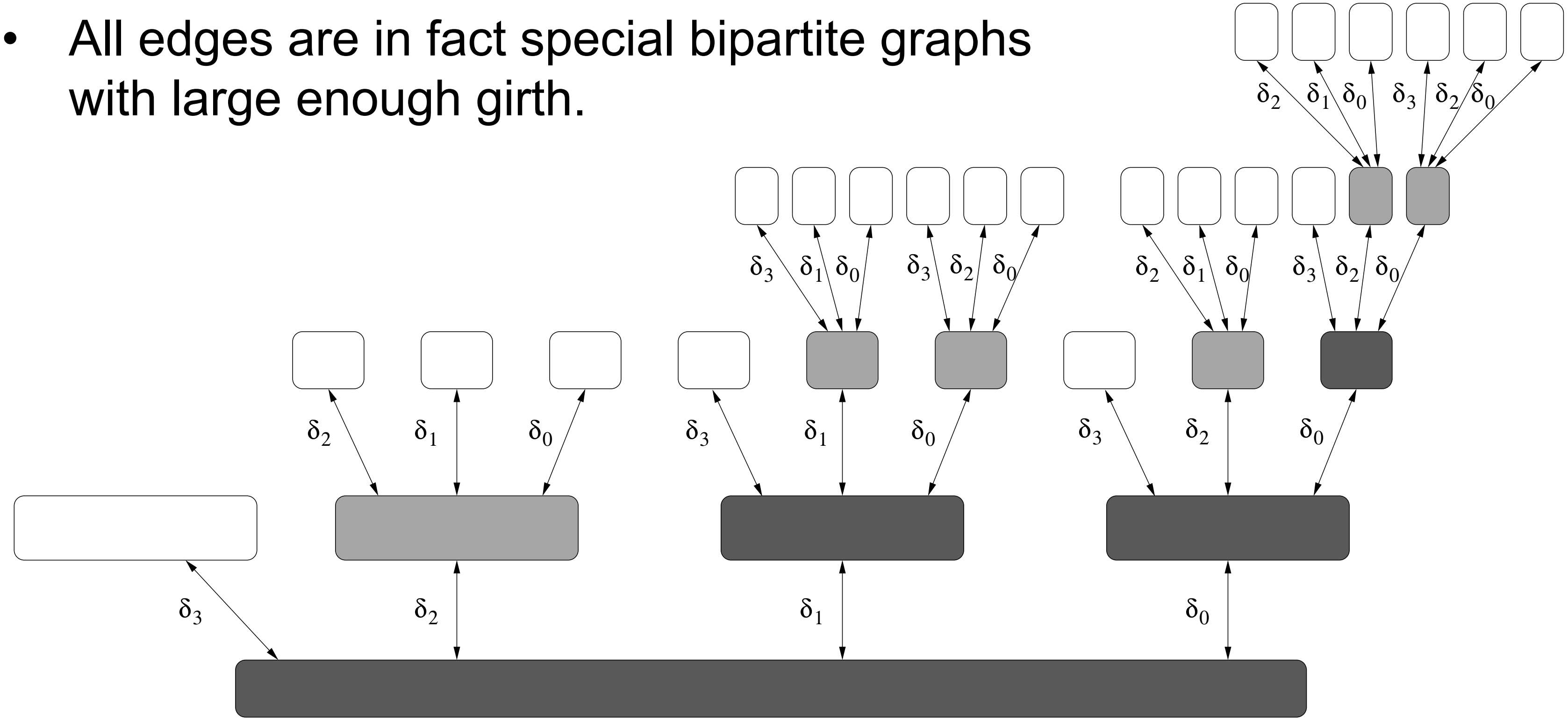
$|DS_{OPT\text{ with green}}| > n$

The Lower Bound

- Lower bounds (Kuhn, Moscibroda, Wattenhofer @ PODC 2004):
  - Model: In a network/graph G (nodes = processors), each node can exchange a message with all its neighbors for k rounds. After k rounds, node needs to decide.
  - We construct the graph such that there are nodes that see the same neighborhood up to distance k. We show that node ID's do not help, and using Yao's principle also randomization does not.
  - Results: Many problems (vertex cover, dominating set, matching, etc.) can only be approximated $\Omega(n^{c/k^2} / k)$ and/or $\Omega(\Delta^{1/k} / k)$.
  - It follows that a polylogarithmic dominating set approximation (or maximal independent set, etc.) needs at least $\Omega(\log \Delta / \log\log \Delta)$ and/or $\Omega((\log n / \log\log n)^{1/2})$ time.

Graph Used in Dominating Set Lower Bound

- The example is for k = 3.
- All edges are in fact special bipartite graphs with large enough girth.



δ_2 δ_1 δ_0 δ_3 δ_2 δ_0
δ_3 δ_1 δ_0 δ_3 δ_2 δ_0
δ_2 δ_1 δ_0 δ_3 δ_2 δ_0
δ_2 δ_1 δ_0
δ_3 δ_1 δ_0
δ_3 δ_2 δ_0
δ_3 δ_2 δ_1 δ_0

Overview

- Introduction

- Algorithmic Models: Case Study Clustering
  - Flooding vs. Dominating Sets
  - Non-Trivial Algorithm
  - Lower Bounds
  - Model Discussion

- Communication Models: Case Study Scheduling
- Conclusions

Are Localized/Local Algorithms Practical?!?

- Localized algorithm: Causality chain, butterfly effect

- Local algorithm: Synchronous communication rounds
  - Quite high demand to MAC layer
  - In reality messages get lost, due to fading, noise, and interference
  - In reality not all neighbors receive a message (hidden terminal problem)
  - In reality nodes might crash and restart (shabby power supply)

- Smells like self-stabilization
  - Messages might get lost, duplicated, or corrupted
  - Node memory/state might get corrupted (RAM only)
  - However, ROM (program, initialization, random seed) is safe

How to turn any local into a self-stabilizing algorithm

- Local algorithm:

- Initialize (local) variables
- Phase
  - Compute message from variables
  - Transmit message
  - Receive messages from neighbors
  - Recompute variables
  - Decision? If not → go to next phase

Receive	Variables	Transmit
-	Init	Out0
In1	Phase1	Out1
In2	Phase2	Out2
…	…	…

- Self-stabilizing algorithm:

- Simply keep transmitting <Out0,Out1,Out2, …> in one single message. (For many local algorithms, this message can be encoded to save space.)

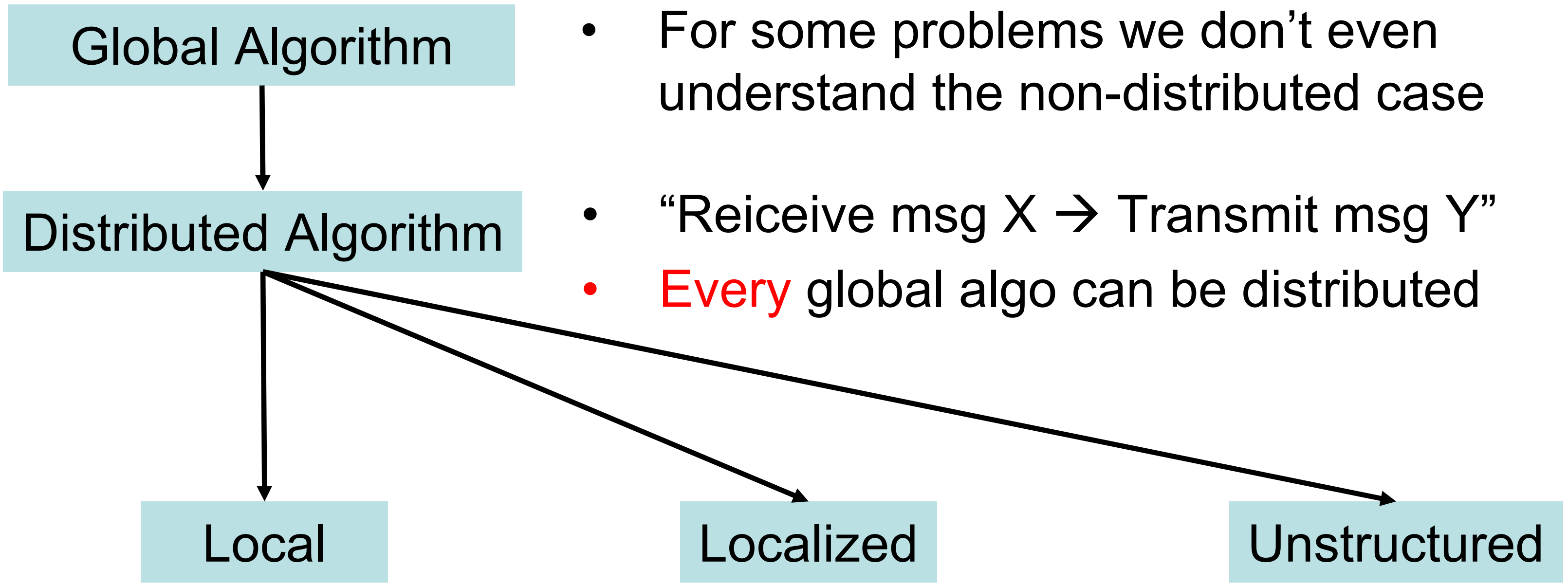
- And keep checking whether your memory is still ok.

- It works! Adversarial memory corruptions are local only.

- [Awerbuch, Varghese, FOCS 91]

Algorithm Classes

- For some problems we don't even understand the non-distributed case
- "Reiceive msg X → Transmit msg Y"
- Every global algo can be distributed



Local

+ Node can only communicate with neighbors k times.

+ Strict time bounds

– Synchronous model

Localized

+ Often simple

– Nodes can wait for neighbor actions

– Often linear chain of causality

Unstructured

+ Implement MAC layer yourself; you control everything

– Often complicated

– Argumentation overhead

Clustering for Unstructured Radio Networks

- “Big Bang” (deployment) of a sensor and/or ad-hoc network:
  - Nodes wake up asynchronously (very late, maybe)
  - Neighbors unknown
  - Hidden terminal problem
  - No global clock
  - No established MAC protocol
  - No reliable collision detection
  - Limited knowledge of the number of nodes or degree of network.

- We have randomized algorithms that compute DS (or MIS) in polylog(n) time even under these harsh circumstances, where n is an upper bound on the number of nodes in the system.

- [Kuhn, Moscibroda, Wattenhofer @ MobiCom 2004]

Example: Comparison of Two Algorithms for Dominating Set

Algorithm 1

- Algorithm c S
- $k^2+O(1$ sions/node
- O(Δ approximation
- nplex!
- nance OK

General Graph!

No Position Information!

Algorithm 2

- Algorith s DS
- 1 tr n/node
- O ximation
- formance great!

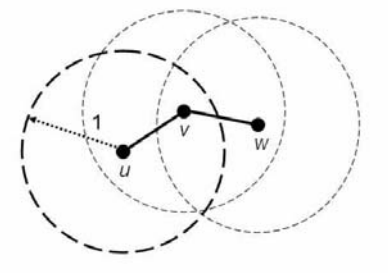
Unit Disk Graph Only!

Requires GPS Device!

The model determines the distributed complexity of a problem

Connectivity Models

General Graph



UDG

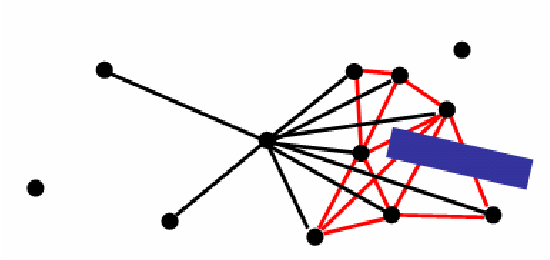
too pessimistic

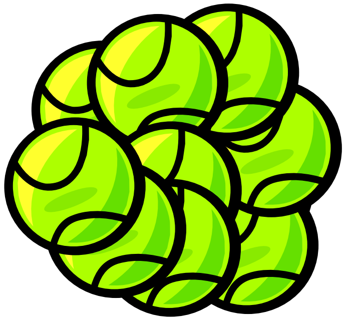
too optimistic

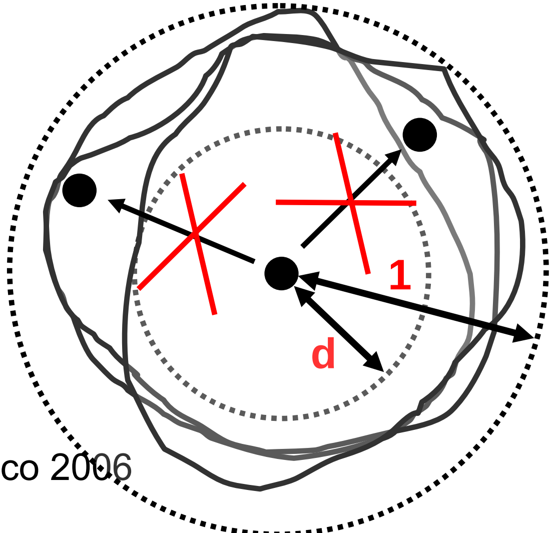
Bounded Independence

Unit Ball Graph

Quasi UDG

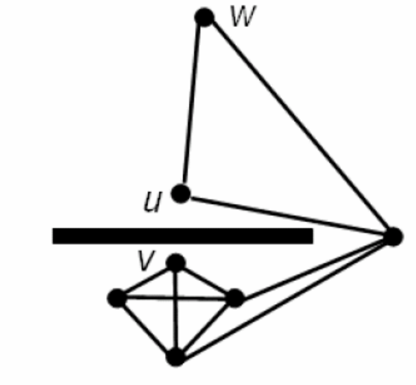






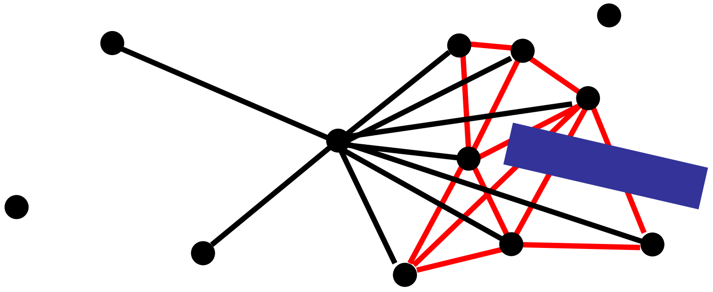
Connectivity: Bounded Independence Graph (BIG)

- How realistic is QUDG?
  - u and v can be close but not adjacent
  - model requires very small d in obstructed environments (walls)



- However: in practice, neighbors are often also neighboring

- Solution: BIG Model
  - Bounded independence graph
  - Size of any independent set grows polynomially with hop distance *r*
  - e.g. $O(r^2)$ or $O(r^3)$



Connectivity: Unit Ball Graph (UBG)

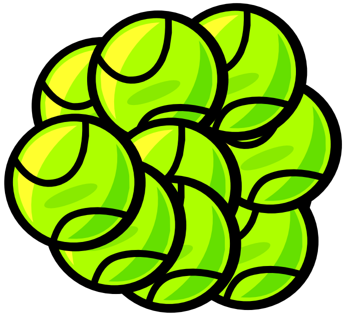
- $\exists$ metric (V,d) describing distances between nodes $u,v \in V$

such that:

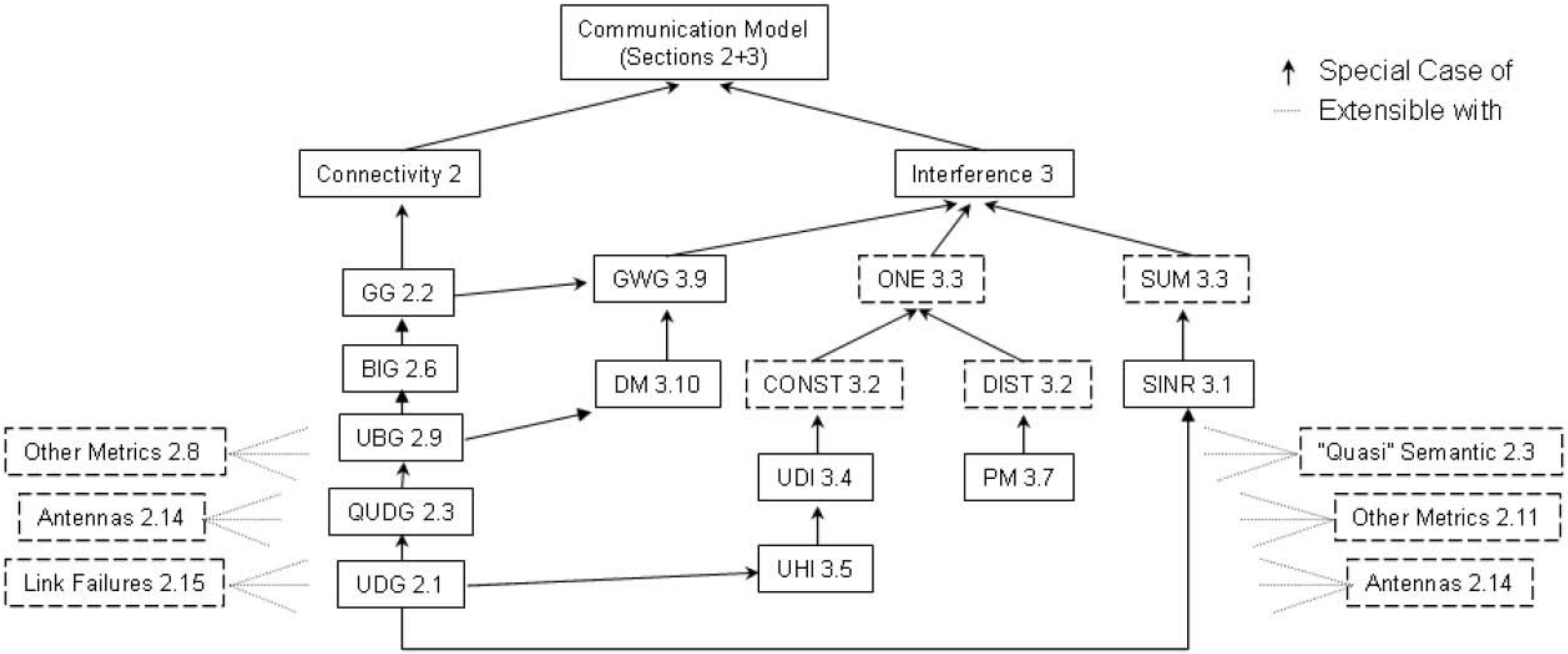
$$\boxed{\begin{array}{l} d(u,v) \leq 1 : (u,v) \in E \\ d(u,v) > 1 \ : (u,v) \notin E \end{array}}$$

- Assume that doubling dimension of metric is constant
  - Doubling dimension: log(#balls of radius r/2 to cover ball of radius r)

UBG based on underlying doubling metric.



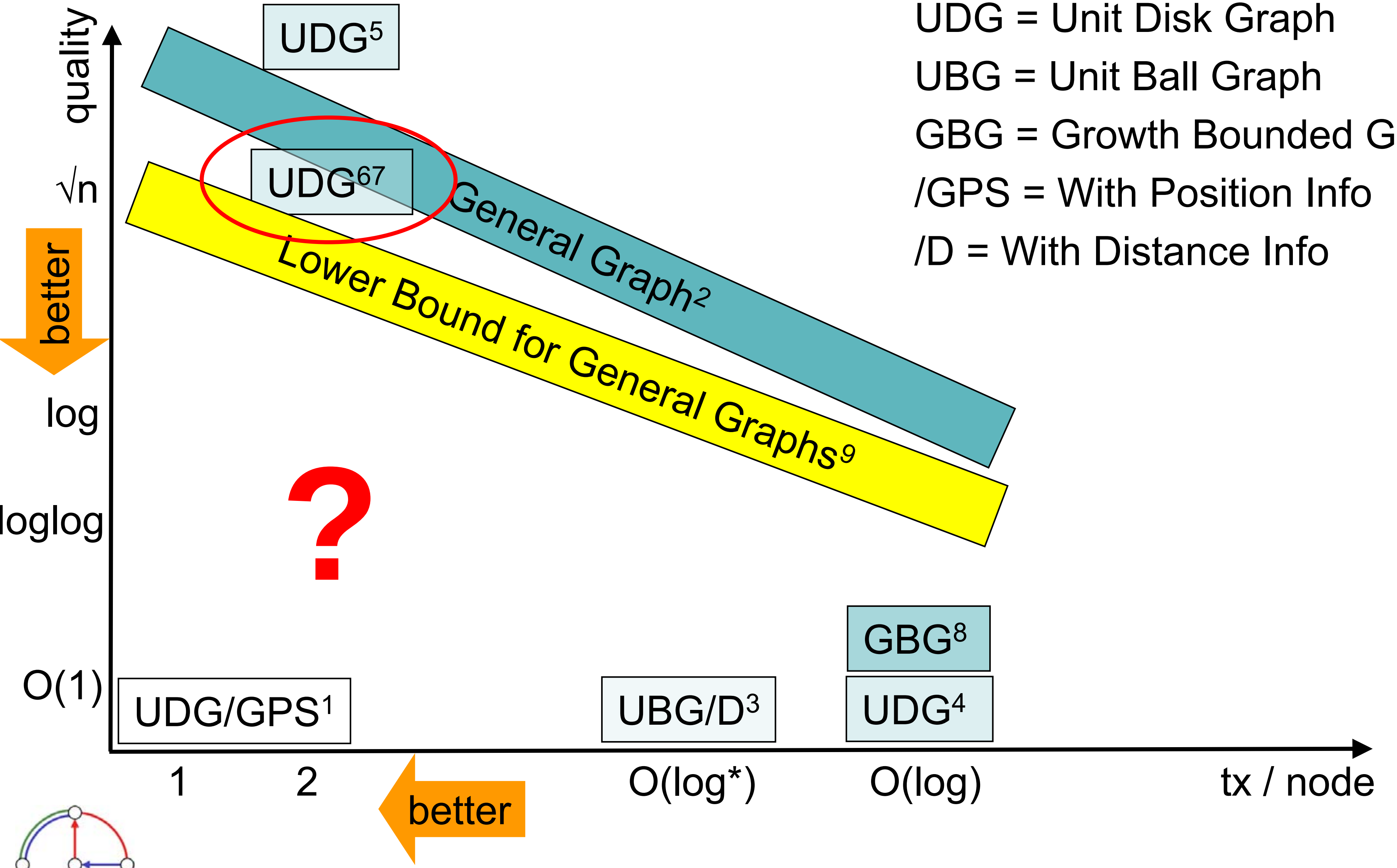
Models can be put in relation



- Try to proof correctness in an as “high” as possible model
- For efficiency, a more optimistic (“lower”) model might be fine

[Schmid, Wattenhofer, WPDRTS 2006]

The model determines the complexity



UDG = Unit Disk Graph

UBG = Unit Ball Graph

GBG = Growth Bounded G.

/GPS = With Position Info

/D = With Distance Info

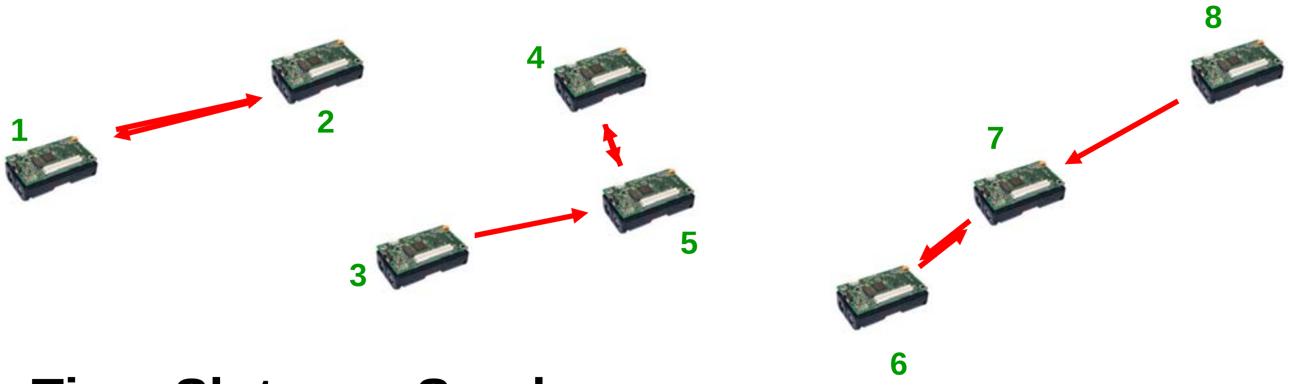
References

1. Folk theorem, e.g. Kuhn, Wattenhofer, Zhang, Zollinger, PODC 2003
2. Kuhn, Wattenhofer, PODC 2003
   - Improved: Kuhn, Moscibroda, Wattenhofer, SODA 2006
   - CDS by Dubhashi et al, SODA 2003
3. Kuhn, Moscibroda, Wattenhofer, PODC 2005
4. Alzoubi, Wan, Frieder, MobiHoc 2002
5. Wu and Li, DIALM 1999
6. Gao, Guibas, Hershberger, Zhang, Zhu, SCG 2001
7. Wattenhofer, MedHocNet 2005 talk, Improving on Wu and Li
8. Kuhn, Moscibroda, Nieberg, Wattenhofer, DISC 2005
9. Kuhn, Moscibroda, Wattenhofer, PODC 2004

Overview

- Introduction
- Algorithmic Models: Case Study Clustering
- Communication Models: Case Study Scheduling
  - Introduction
  - Intuition & Results
  - Lower Bound Example
  - From Theory to Practice
- Conclusions

Spatial Reuse (with Scheduling)



Time-Slot	Senders:
t_1:	v_1, v_4, v_7
t_2:	v_2, v_3, v_6
t_3:	v_5, v_8

This example uses 3 time slots!

Schedule a set of given links in as few as possible time slots

Physical Model

- Let us look at the signal-to-noise-plus-interference (SINR) ratio!
- Message arrives if SINR is larger than β at receiver

Power level of sender u

Path-loss exponent

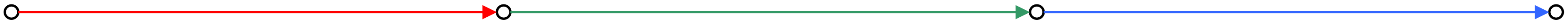
$$\frac{\frac{P_u}{d(u,v)^\alpha}}{N+\sum_{w\in V\setminus\{u\}}\frac{P_w}{d(w,v)^\alpha}} \geq \beta$$

Noise

Distance between two nodes

Minimum signal-to-interference ratio

A Simple Scheduling Problem



Consider the following simple scheduling task Ψ:

Ψ: **Every node can send *one* message successfully!**

Receivers can be choosen optimally! (e.g. nearest neighbor)

How many time-slots are required so every node can send at least once?

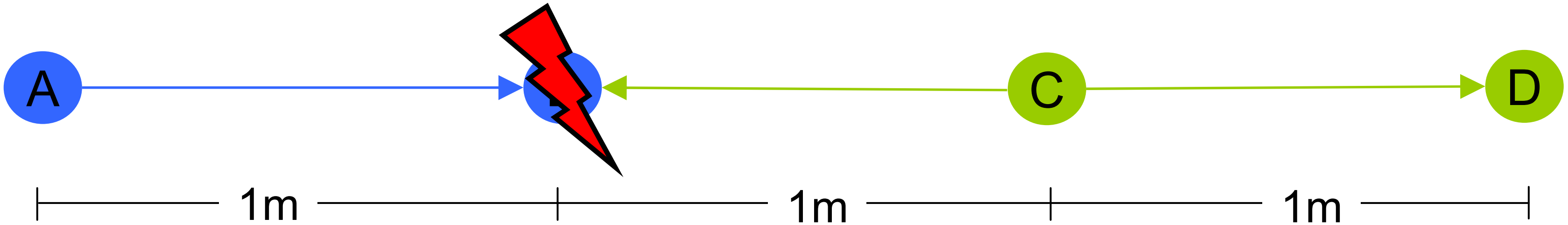
„The Scheduling Complexity in Wireless Networks“

Overview

- Introduction
- Algorithmic Models: Case Study Clustering
- Communication Models: Case Study Scheduling
  - Introduction
  - Intuition & Results
  - Lower Bound Example
  - From Theory to Practice
- Conclusions

Can we schedule links concurrently…?

A wants to sent to B, C wants to send to D

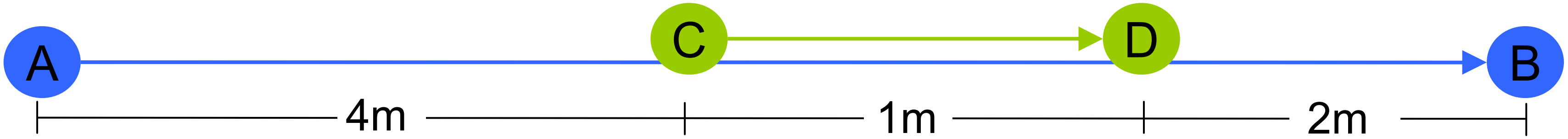


- Let α=3, β=3, and N=10nW (realistic values!)
- Set the transmission powers as follows P_C= -7 dBm and P_A= 0 dBm
- SINR at D is: $$\frac{1mW/(1m)^3}{0.01\mu W+200\mu W/(1m)^3} \approx 5.0 \geq \beta$$
- SINR at B is: $$\frac{200\mu W/(1m)^3}{0.01\mu W+1mW/(3m)^3} \approx 5.4 \geq \beta$$

Simultaneous transmission *is* possible !

Let’s make it tougher!

A wants to sent to B, C wants to send to D



- Let α=3, β=3, and N=10nW
- Set the transmission powers as follows P_C= -15 dBm and P_A= 1 dBm
- SINR at D is: $$\frac{1.26mW/(7m)^3}{0.01\mu W + 31.6\mu W/(3m)^3} \approx 3.11 \geq \beta$$
- SINR at B is: $$\frac{31.6\mu W/(1m)^3}{0.01\mu W + 1.26mW/(5m)^3} \approx 3.13 \geq \beta$$

Simultaneous transmission *is* possible !

The Scheduling Complexity of Wireless Networks

- This is possibly the simplest possible scheduling problem!
  - n nodes in 2D Euclidean plane (nodes in arbitrary position)
  - Nodes can choose power levels
  - Message successfully received if SINR at receiver sufficient
  - Ψ: Each node‘s transmission is successfully received by someone

Scheduling Complexity $S(\Psi)$

The minimal number of time-slots required until every node can successfully transmit at least once (in any network with n nodes)!

Clearly, $S(\Psi) \leq n$

Results

[Moscibroda, Wattenhofer, Infocom 2006]

- The trivial protocol (scheduling each node individually) requires n time slots.

  $\mathbf{S(\Psi) \leq n}$

- Any protocol with uniform power assignment requires Ω(n) time slots.

  $\mathbf{S(\Psi) \in \Omega(n)}$

- Any protocol with $P \sim O(d^{\alpha})$ power assignment requires Ω(n) time slots.

  $\mathbf{S(\Psi) \in \Omega(n)}$

Exponential gap!

- If done right, scheduling complexity of Ψ is $\mathbf{S(\Psi) \in O(\log^3 n)}$

- In any network, a strongly-connected topology can be scheduled in time

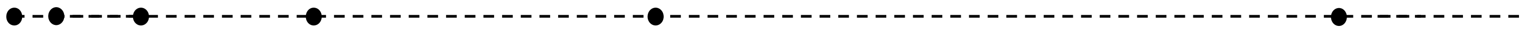
  $\mathbf{S(connected) \in O(\log^4 n)}$

Overview

- Introduction
- Algorithmic Models: Case Study Clustering
- Communication Models: Case Study Scheduling
  - Introduction
  - Intuition & Results
  - Lower Bound Example
  - From Theory to Practice
- Conclusions

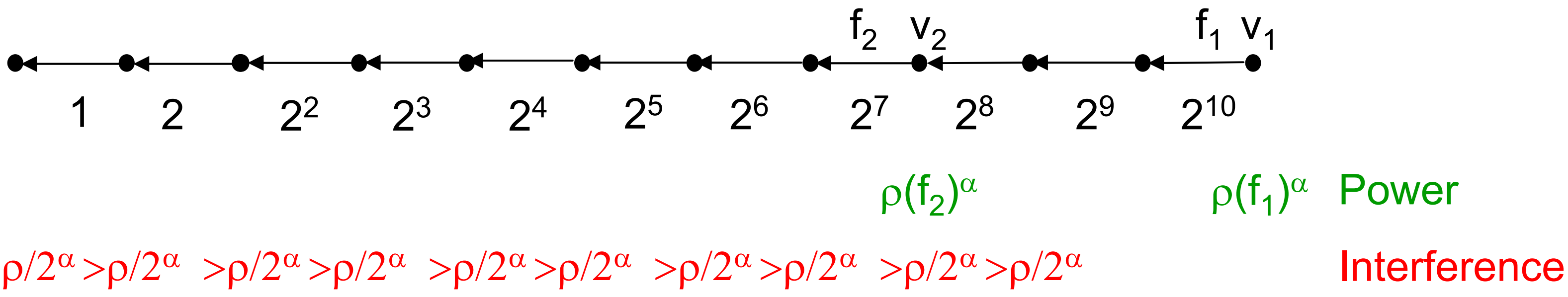
Lower Bound for $P \sim O(d^\alpha)$ Power Assignment

- Consider the exponential chain:



Lower Bound for $P \sim O(d^\alpha)$ Power Assignment

- Consider the exponential chain:



f_2 v_2 f_1 v_1

1 2 2^2 2^3 2^4 2^5 2^6 2^7 2^8 2^9 2^{10}

$\rho(f_2)^\alpha$ $\rho(f_1)^\alpha$ Power

$>\rho/2^\alpha$ $>\rho/2^\alpha$ $>\rho/2^\alpha$ $>\rho/2^\alpha$ $>\rho/2^\alpha$ $>\rho/2^\alpha$ $>\rho/2^\alpha$ $>\rho/2^\alpha$ $>\rho/2^\alpha$ $>\rho/2^\alpha$ Interference

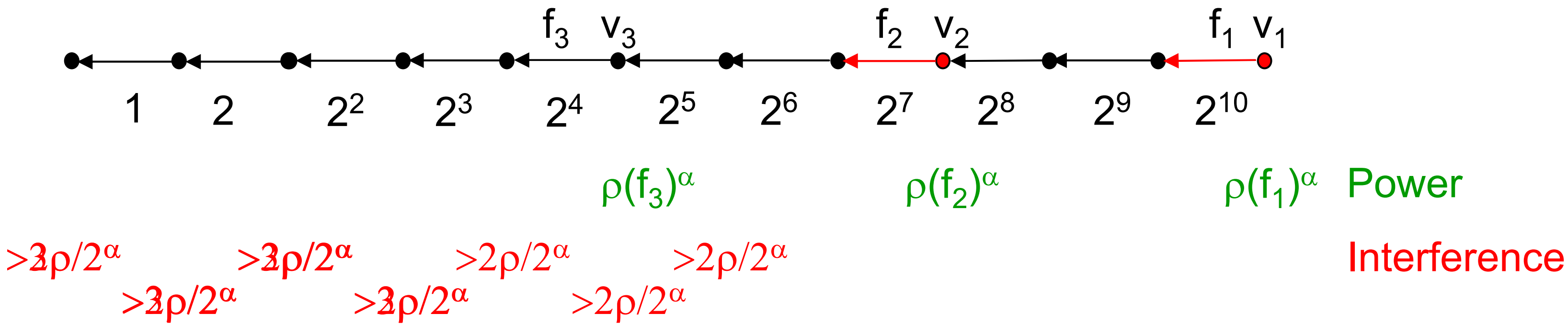
Not trivial…

- Exponential chain can be scheduled in O(1) time!
- How many links can we schedule simultaneously with $P \sim O(d^\alpha)$?
- Consider first node v_1...

  → its power is $P_1=\rho 2^{10\alpha}$ for some constant ρ
- This creates interference of at least $\rho/2^\alpha$ at every other node!
- The second node v_2 also sends with power $P_2=\rho 2^{7\alpha}$
- Again, this creates an additional interference of at least $\rho/2^\alpha$ at every other node!

Lower Bound for $P \sim O(d^{\alpha})$ Power Assignment

- Consider the exponential chain:



Not trivial…

- Exponential chain can be scheduled in O(1) time!
- How many links can we schedule simultaneously?
- Let us start with the first node v_1...

  → its power is $P_1 = \rho 2^{10\alpha}$ for some constant ρ
- This creates interference of at least $\rho/2^{\alpha}$ at every other node!
- The second node v_2 also sends with power $P_2 = \rho 2^{7\alpha}$
- Again, this creates an additional interference of at least $\rho/2^{\alpha}$ at every other node!

And so on…

Lower Bound for $P \sim O(d^\alpha)$ Power Assignment

- Assume we can schedule X nodes in parallel.
- The left-most receiver x_r faces an interference of at least $X\cdot\rho/2^\alpha$

  → yet, x_r receives the message, say from x_s.
- How large can X be?
- The SINR at x_r must be at least β, and hence

$$\frac{2^\alpha}{X} \approx \frac{\frac{\cancel{\rho}\cdot \cancel{d(x_s,x_r)^\alpha}}{\cancel{d(x_s,x_r)^\alpha}}}{\cancel{N} + X\cdot\frac{\cancel{\rho}}{2^\alpha}} \overset{!}{\geq} \beta$$

- From this, it follows that X is at most $2^\alpha/\beta$!

Any $P \sim O(d^\alpha)$ power assignment algorithm has scheduling complexity: **$S(\Psi) \in \Omega(n)$**

Overview

- Introduction
- Algorithmic Models: Case Study Clustering
- Communication Models: Case Study Scheduling
  - Introduction
  - Intuition & Results
  - Lower Bound Example
  - From Theory to Practice
- Conclusions

Observations and Implications

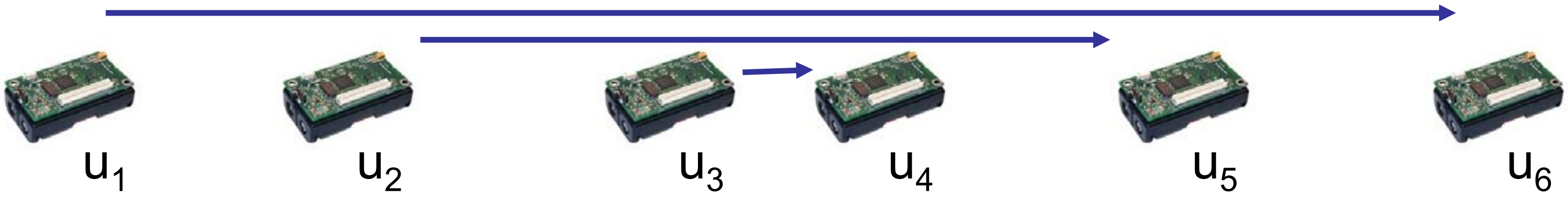
- All MAC layer protocols we are aware of use either uniform or d^α power assignment.
  - Thus, the theoretical performance of current MAC layer protocols almost as bad as scheduling every single node individually!

- In contrast: faster polylogarithmic scheduling (faster MAC protocols) are theoretically possible in all (even worst-case) networks, if nodes choose power carefully.
  - Theoretically, there is no fundamental scaling problem with scheduling (in contrast to capacity).
  - Theoretically efficient MAC protocols must use non-trivial power levels!

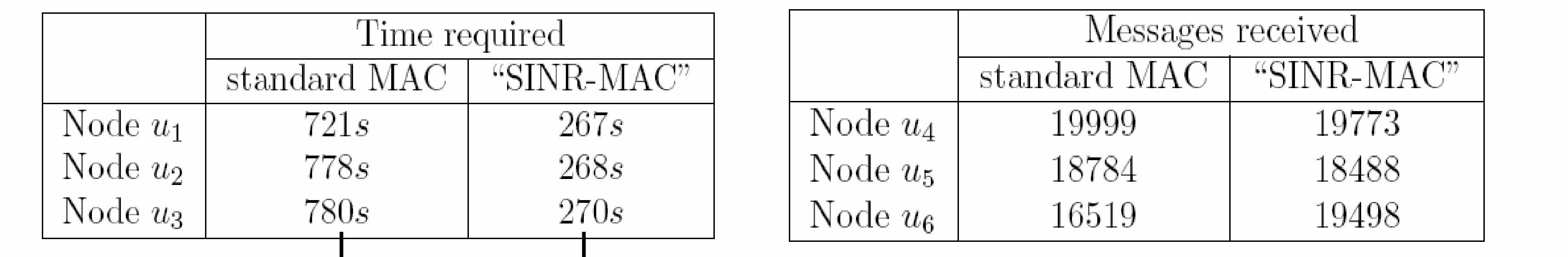
- Well, the word theory appeared in every line...

From Theory to 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:



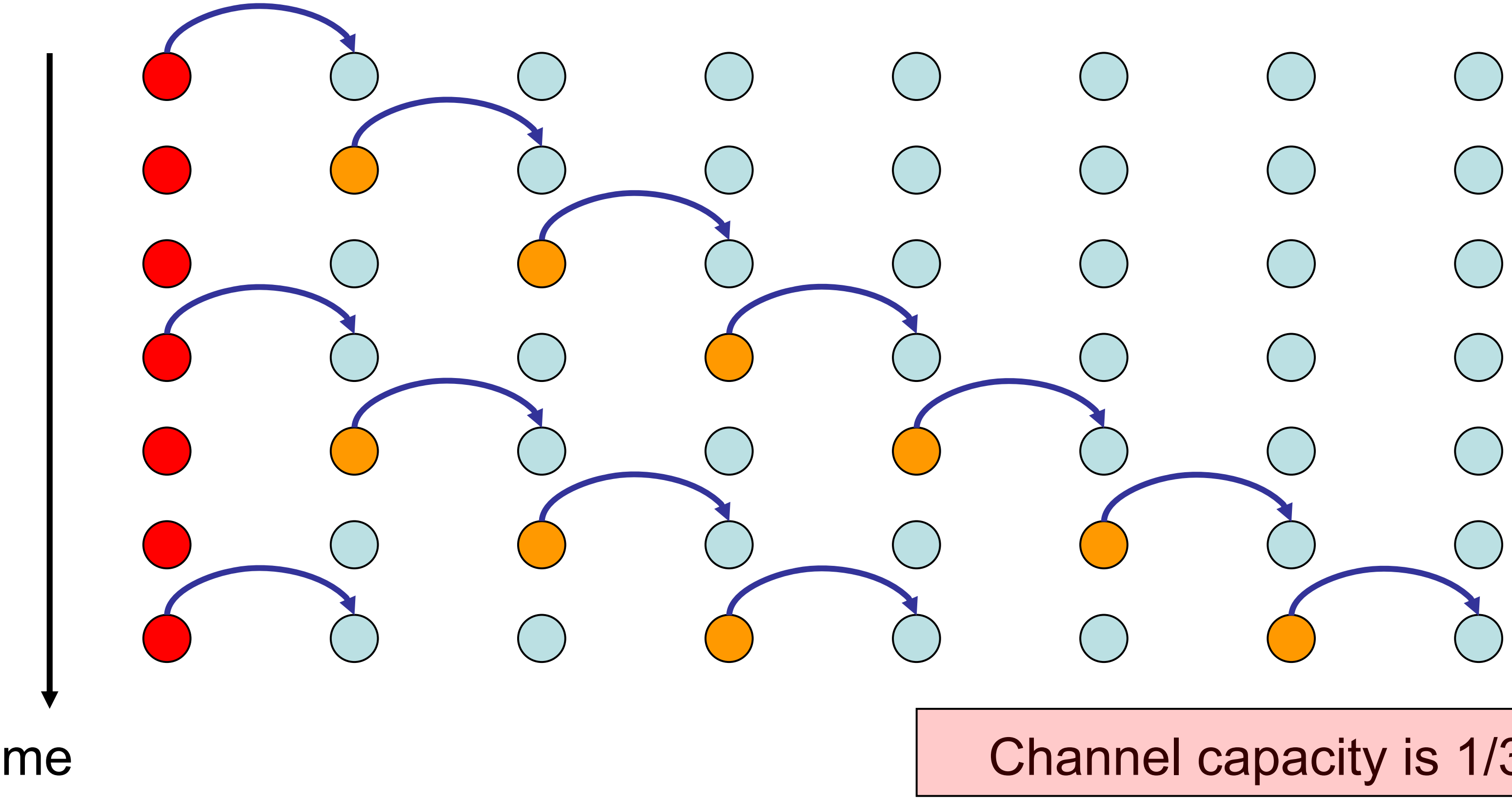
	Time required	
	standard MAC	“SINR-MAC”
Node u_1	$721s$	$267s$
Node u_2	$778s$	$268s$
Node u_3	$780s$	$270s$

	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

Possible Applications – Improved “Channel Capacity”

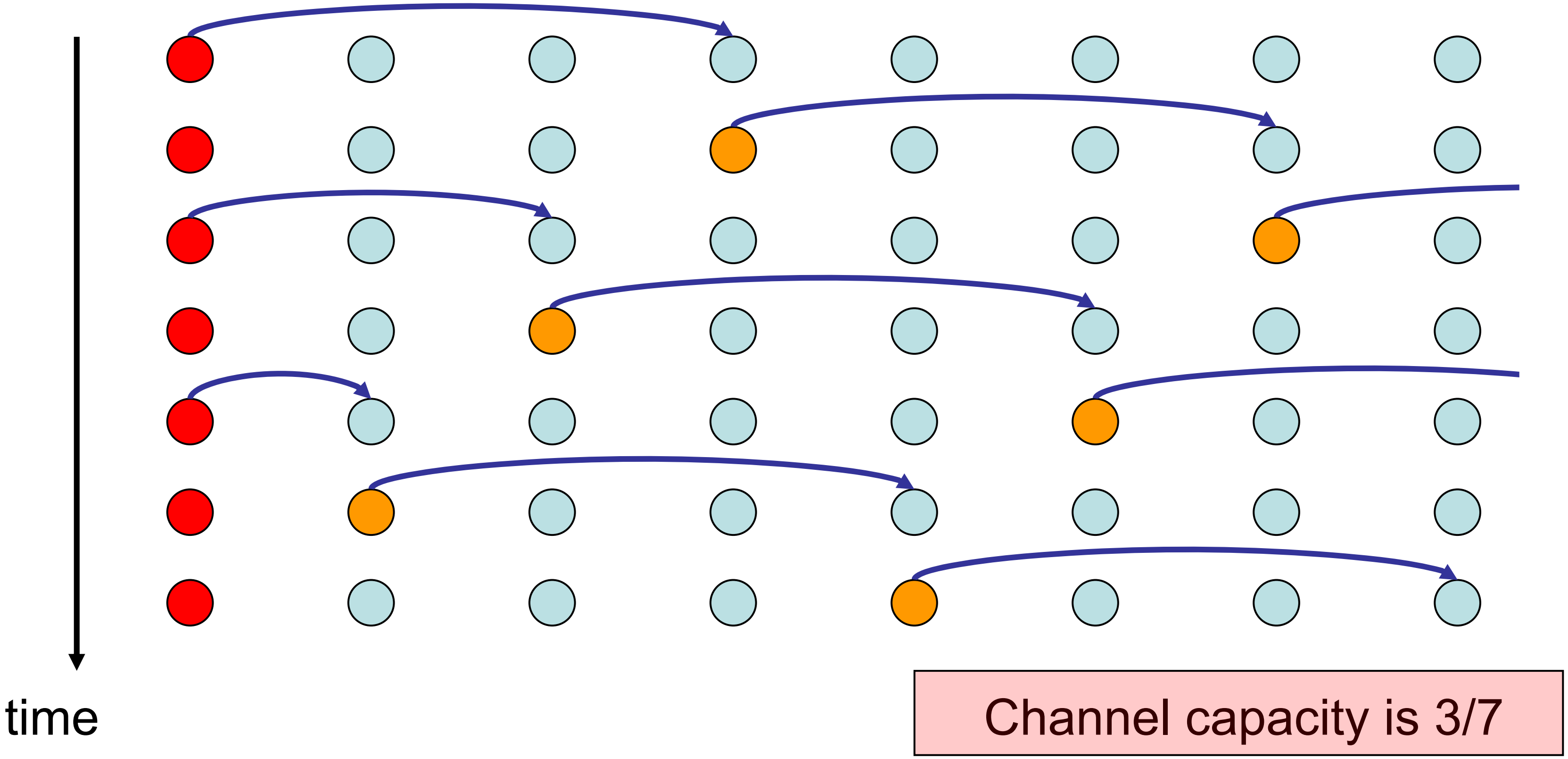
- Consider a channel consisting of wireless sensor nodes
- What throughput-capacity of this channel...?



Channel capacity is 1/3

Possible Applications – Improved "Channel Capacity"

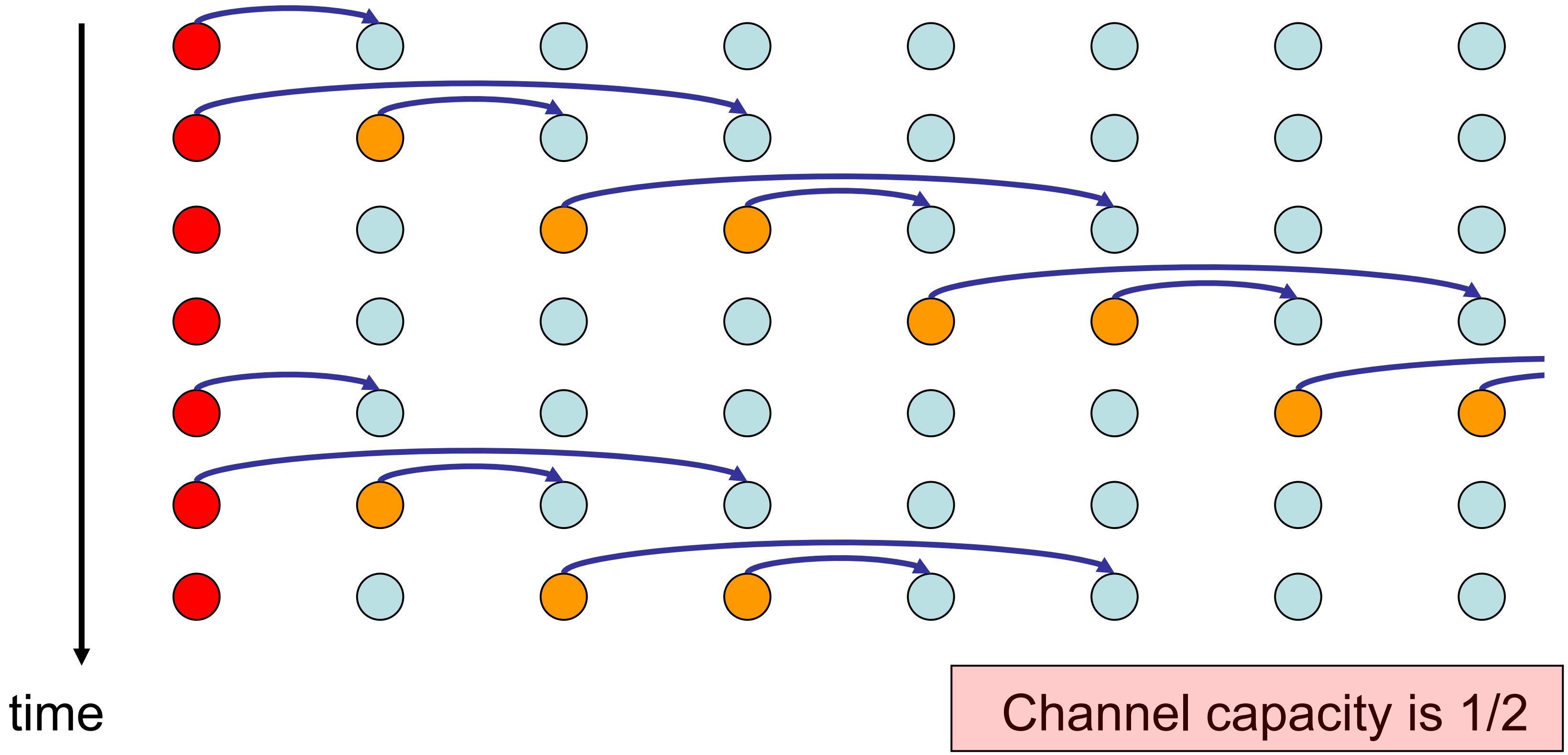
- A better strategy...
- Assume node can reach 3-hop neighbor



Channel capacity is 3/7

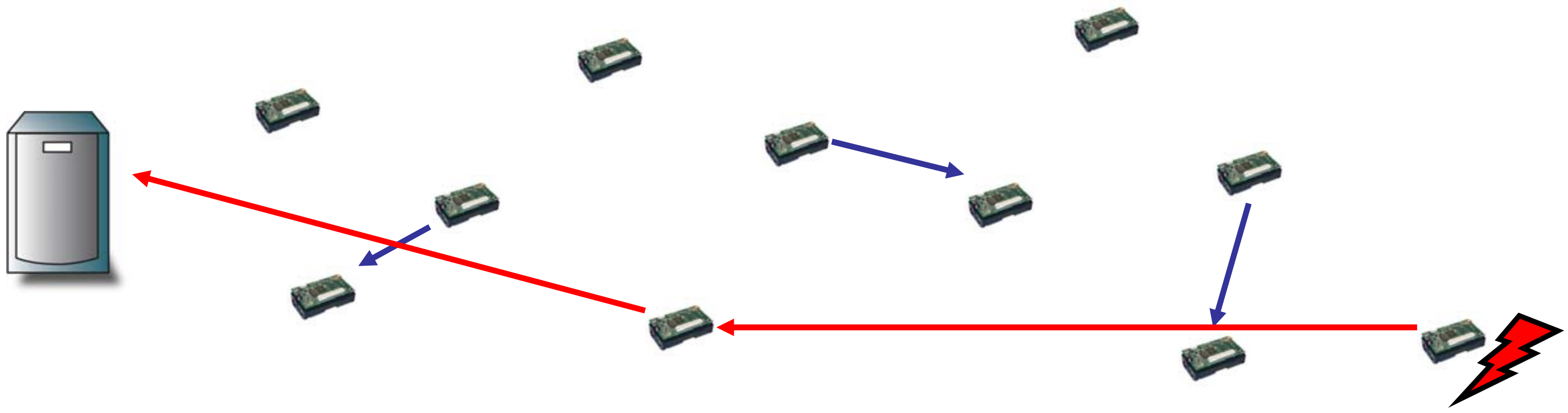
Possible Applications – Improved “Channel Capacity”

- All such (graph-based) strategies have capacity strictly less than 1/2!
- For certain α and β, the following strategy is better!



Channel capacity is 1/2

Possible Applications – Data Gathering



- Neighboring nodes must communicate periodically (for time synchronisation, neighborhood detection, etc…)
- Sending data to base station may be time critical → use long links
- Employing clever power control may reduce delay & reduce coordination overhead!

→ From theory (scheduling) to practice (protocol design)…?

Overview

- Introduction
- Algorithmic Models: Case Study Clustering
- Communication Models: Case Study Scheduling

- Conclusions

More…

- Clustering (Dominating Sets, etc.)
- Interference and Signal-to-Noise-Ratio
- MAC Layer and Coloring
- Topology and Power Control
- Deployment (Unstructured Radio Networks)
- New Routing Paradigms (e.g. Link Reversal)
- Geo-Routing
- Broadcast and Multicast
- Data Gathering
- Location Services and Positioning
- Time Synchronization
- Modeling and Mobility
- Lower Bounds for Message Passing
- Selfish Agents, Economic Aspects, Security

Link Layer

Network Layer

Services

Theory/Models

Summary

- Sensor networks are an excellent application for distributed algorithms

- We need to study new network topologies
  - Network models between geometry and graph theory (BIG, UBG)
  - Interference models such as SINR

- We need to study new algorithmic paradigms
  - Distributed → Localized → Local → Self-Stabilizing → Unstructured

Thank You!

Questions? Comments?