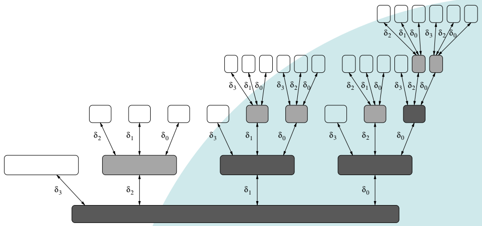
P2P

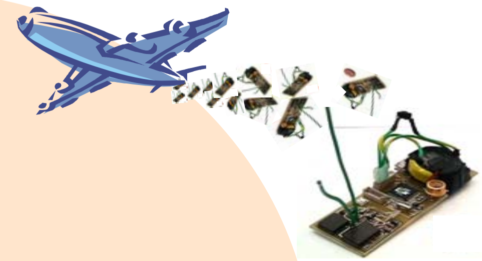
Past 2 Present

***D**istributed*
***C**omputing*
***G**roup*
Roger Wattenhofer

P2P 2005

My Research





Distributed Computing Theory

Wireless Networking (Ad Hoc, Sensor)

P2P

Distributed Systems

Disclaimer:
I'm a P2P ignorant giving a P2P talk

P2P vs. Ad Hoc/Sensor Networking

- Often considered to be “similar”
  - Without infrastructure, without servers, etc.
  - Routing is essential
  - Both feature some sort of topology control (“What are the neighbors?”)

- Major differences
  - Internet vs. wireless (interference, MAC layer, etc.)
  - Graph theory vs. geometry (…really?!?)
  - Churn vs. mobility
  - Completely different applications

P2P vs. Distributed Computing/Systems

Ignorant's Lemma 1: P2P research is the child of successful file sharing applications a la Napster and the distributed computing/systems community

Ignorant's Corollary 2: A child should learn from his/her parents

- Let's first try to "prove" Lemma 1
- … and then convince you about Corollary 2

Overview

- Introduction

- Past
  - What is the first P2P paper/system?
  - Really?

- Present

The Four P2P Evangelists

- If you read your average P2P paper, there are (almost) always four papers cited who “invented” efficient P2P in 2001:

Chord | CAN | Pastry | Tapestry

- These papers are somewhat similar, with the exception of CAN (which is not really efficient)

- So what’s the „Dead Sea Scrolls of P2P”?

“Dead Sea Scrolls of P2P”

„Accessing Nearby Copies of Replicated Objects in a Distributed Environment“, by Greg Plaxton, Rajmohan Rajaraman, and Andrea Richa, at SPAA 1997.

- Basically, the paper proposes an efficient search routine (similar to the evangelist papers). In particular search, insert, delete, storage costs are all logarithmic, the base of the logarithm is a parameter.
- However, it‘s a theory paper, so that alone would be too simple...
- So the paper takes into account latency; in particular it is assumed that nodes are living in a metric, and that the graph is of „bounded growth“ (meaning that node densities do not change abruptly).

Genealogy of P2P

The parents of Plaxton et al.?

WWW, POTS, etc.

Plaxton et al.

1997

1998

1999 Napster

2000 Gnutella

Chord CAN Pastry Tapestry | 2001 | eDonkey Kazaa

Viceroy P-Grid Kademlia | 2002 | Gnutella-2 BitTorrent

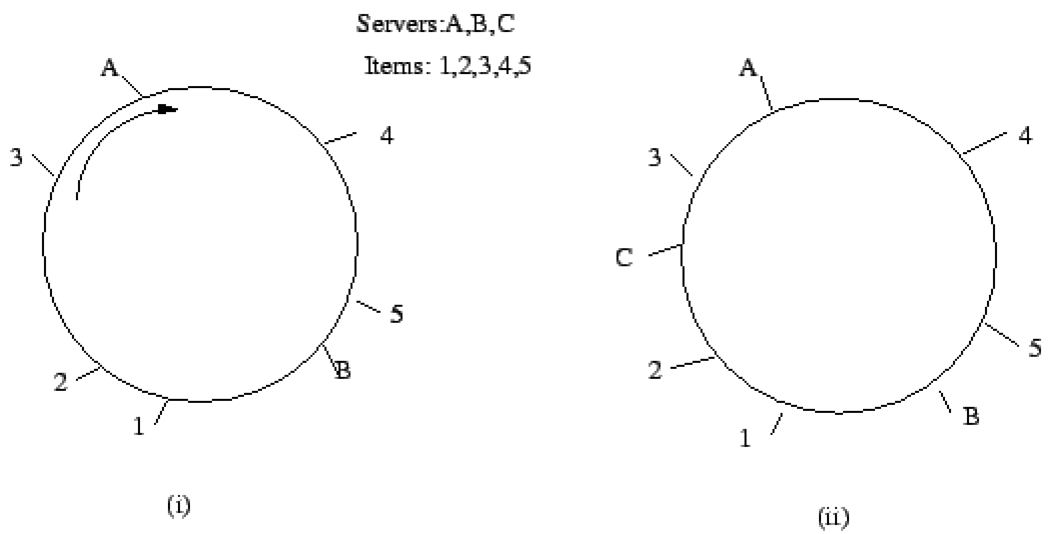
Koorde SkipGraph SkipNet | 2003 | Skype Steam PS3

Overview

- Introduction
- Past
  - What is the first P2P paper/system?
  - Really?
- Present

Consistent Hashing

"Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web." David Karger, Eric Lehman, Tom Leighton, Matthew Levine, Daniel Lewin and Rina Panigrahy, at STOC 1997.



- Big difference: still a client/server paradigm.

Locating Shared Objects

- “Sparse Partitions”. Baruch Awerbuch and David Peleg, at FOCS 1990.
- “Concurrent Online Tracking of Mobile Users”. Baruch Awerbuch and David Peleg, at SIGCOMM 1991.
- “Locating Nearby Copies of Replicated Internet Servers”. James Guyton and Michael Schwartz, at SIGCOMM 1995.
- “A Model for Worldwide Tracking of Distributed Objects”. Marteen van Steen, Franz Hauck, Andrew Tanenbaum, at TINA 1996.
- Maintaining a distributed directory.

Compact Routing

"A trade-off between space and efficiency for routing tables". David Peleg and Eli Upfal, at STOC 1988.

- Trade-off routing table memory space vs. stretch (quality of routes)
- Name-independent vs. labeled routing
  - Name-independent: the node names are fixed (like in a regular network)
  - Labeled: a designer can choose names (P2P)
- In particular interesting if latency *does* matter.

Hypercubic Topologies

- In my lecture *Distributed Computing* I teach six topologies:
  - Hypercube — Plaxton et al. | Chord | Kademlia
  - Butterfly / Benes Network — Viceroy
  - DeBruijn Graph — Koorde
  - Skip List — SkipGraph | SkipNet
  - Pancake Graph — Kuhn et al.
  - Cube-Connected-Cycles — Your-name-here

Overview

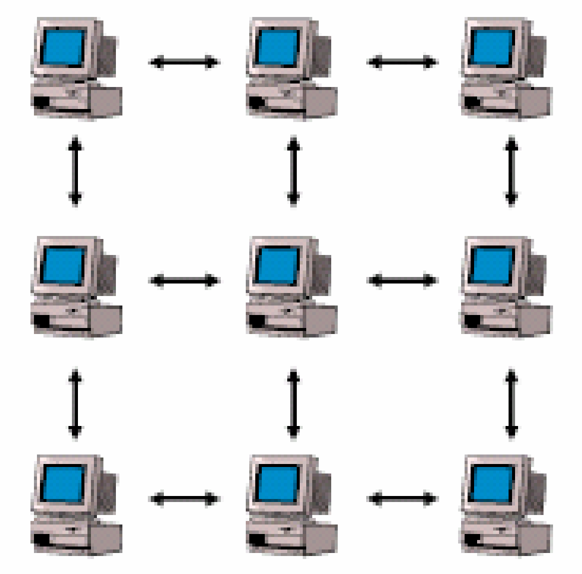
- Introduction
- Past
- Present*
  - Dynamic systems & mobility
  - Fault-tolerance (crash failures)
  - Security (Byzantine failures)
  - Selfish agents & computational economy
  - Simple and implementable algorithms
  - Local algorithms
  - Geometry, metrics, bounded growth, etc.
  - Applications

*current hot topics in distributed computing

Dynamic Peer-to-Peer Systems

"A Self-Repairing Peer-to-Peer System Resilient to Dynamic Adversarial Churn". Fabian Kuhn, Stefan Schmid, Roger Wattenhofer, at IPTPS 2005.

- Properties compared to centralized client/server approach
  - Availability, Reliability, Efficiency
- However, P2P systems are
  - composed of unreliable desktop machines
  - under control of individual users

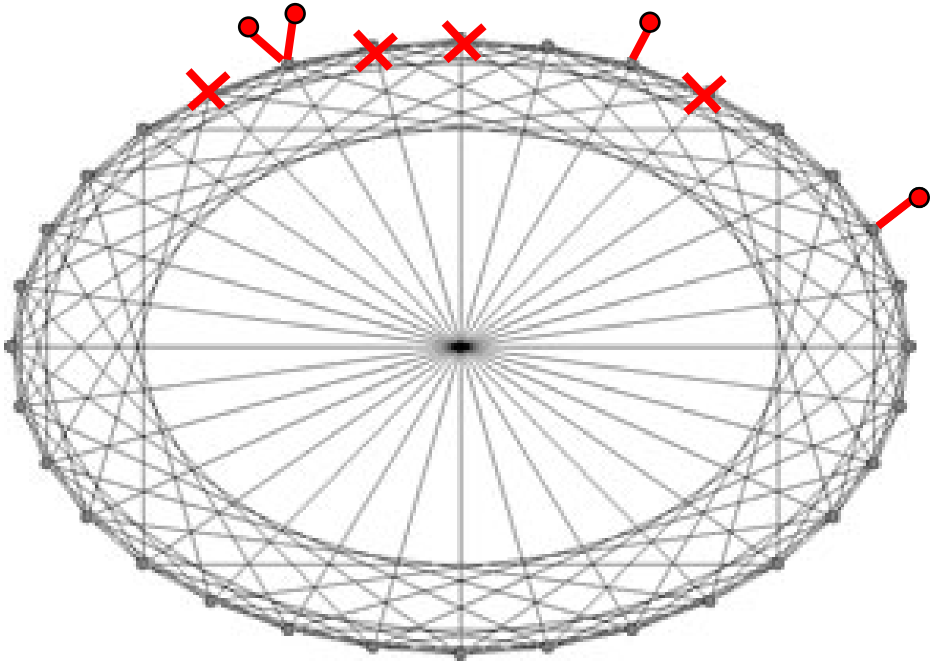


→ Peers may join and leave the network at any time!

Churn (permanent joins and leaves)

How to maintain desirable properties such as

– Connectivity,
– Network diameter,
– Peer degree?



Motivation

- Why permanent churn?

  Saroiu et al.: „A Measurement Study of P2P File Sharing Systems"
  Peers join system for one hour on average

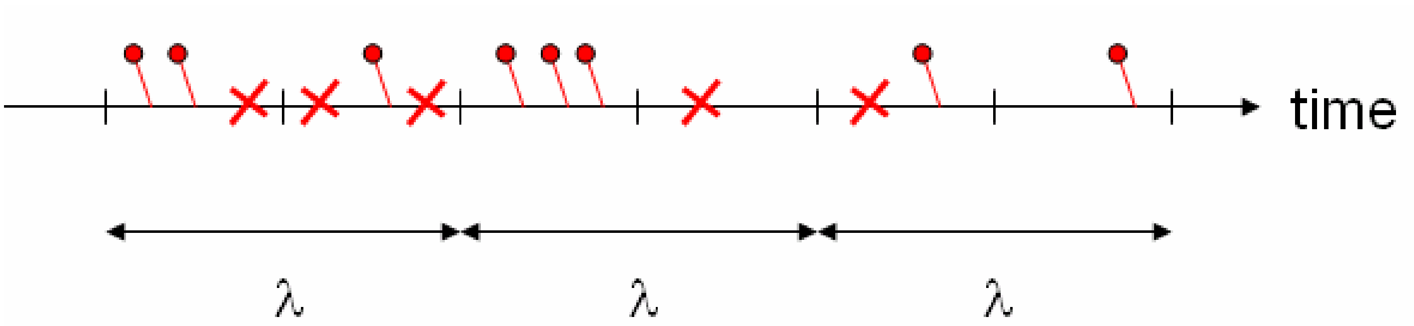
→ Hundreds of changes per second with millions of peers in system!

- Why adversarial (worst-case) churn?

  E.g., a crawler takes down neighboring machines rather than randomly chosen peers!

The Adversary

- Model worst-case faults with an adversary *ADV(J,L,λ)*
- *ADV(J,L,λ)* has complete visibility of the entire state of the system
- May add at most *J* and remove at most *L* peers in any time period of length λ



- Note: Adversary is *not* Byzantine!

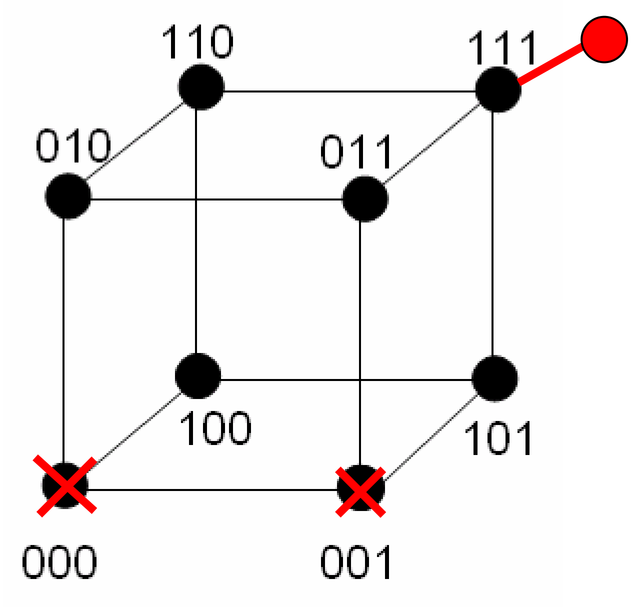
Synchronous Model

- Our system is synchronous, i.e., our algorithms run in rounds. One round:
  - receive messages,
  - local computation,
  - send messages
- However: Real distributed systems are asynchronous!
- But: Notion of time necessary to bound the adversary

A First Approach

- Fault-tolerant hypercube?

- What if number of peers is not 2^i?
- How to prevent degeneration?
- Where to store data?



- Idea: Simulate the hypercube

Simulated Hypercube System

- Simulation: Each node consists of several peers

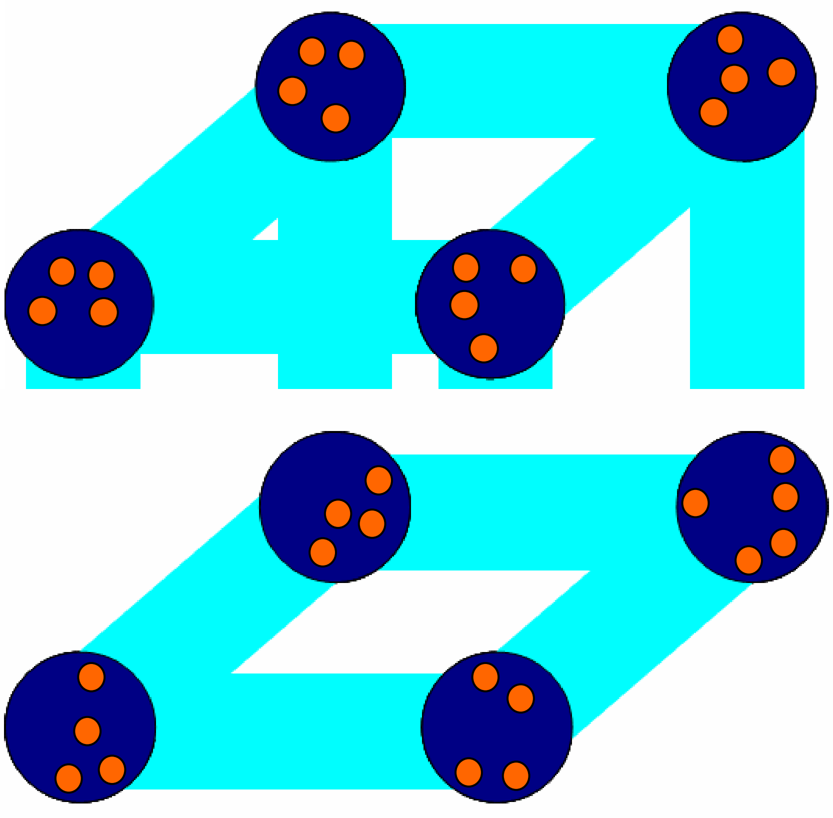
Basic components:

- Route peers to sparse areas

  Token distribution

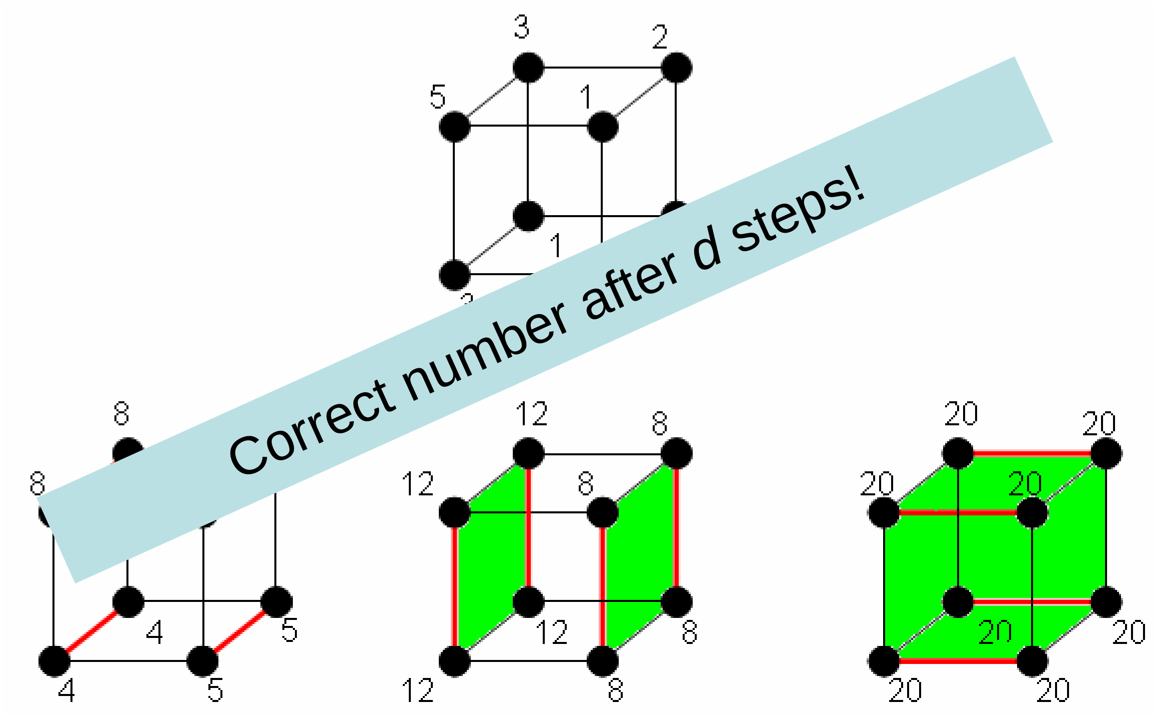
- Adapt dimension

  Information aggregation



Example: Information Aggregation

- Algorithm: Count peers in every sub-cube by exchange with corresponding neighbor



Correct number after *d* steps!

Results

- All our algorithms (token distribution and data aggregation) consistently run in the background.
- We can tolerate an adversary who can insert/delete O(log n) peers per maximum message delay.
- Our system is never fully repaired, but always fully functional.
- In detail, we have in spite of *ADV(O(log n),O(log n),1)*:
  - always at least one peer per node,
  - at most *O(log n)* peers per node,
  - network diameter *O(log n)*,
  - peer degree *O(log n)*.

Overview

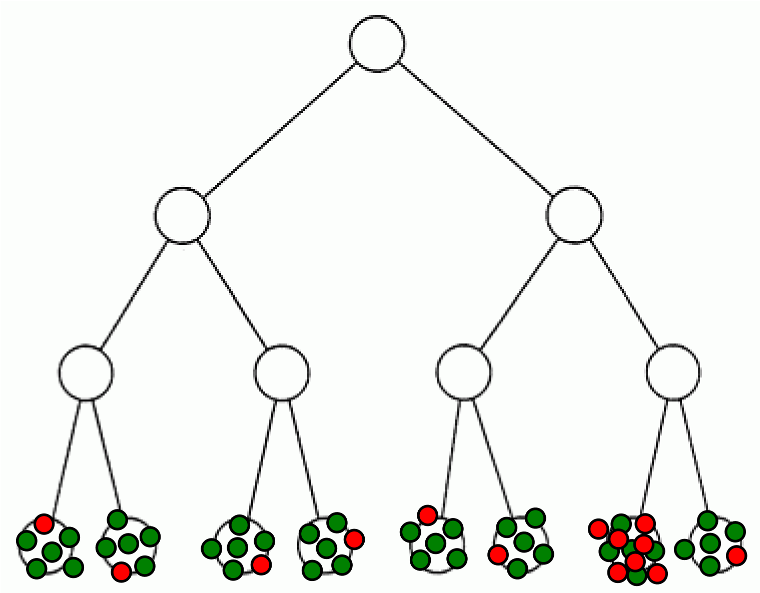
- Introduction
- Past
- Present
  - Dynamic systems & mobility
  - Fault-tolerance (crash failures)
  - Security (Byzantine failures)
  - Selfish agents & computational economy
  - Simple and implementable algorithms
  - Local algorithms
  - Geometry, metrics, bounded growth, etc.
  - Applications

Byzantine Failures

- If adversary controls more and more corrupted nodes and then crashes all of them at the same time ("sleepers"), we stand no chance.

- "Robust Distributed Name Service". Baruch Awerbuch and Christian Scheideler, at IPTPS 2004.

- Idea: Assume that the Byzantine peers are the minority. If the corrupted nodes are the majority in a specific part of the system, they can be detected (because of their unusual high density).



Overview

- Introduction
- Past
- Present
  - Dynamic systems & mobility
  - Fault-tolerance (crash failures)
  - Security (Byzantine failures)
  - Selfish agents & computational economy
  - Simple and implementable algorithms
  - Local algorithms
  - Geometry, metrics, bounded growth, etc.
  - Applications

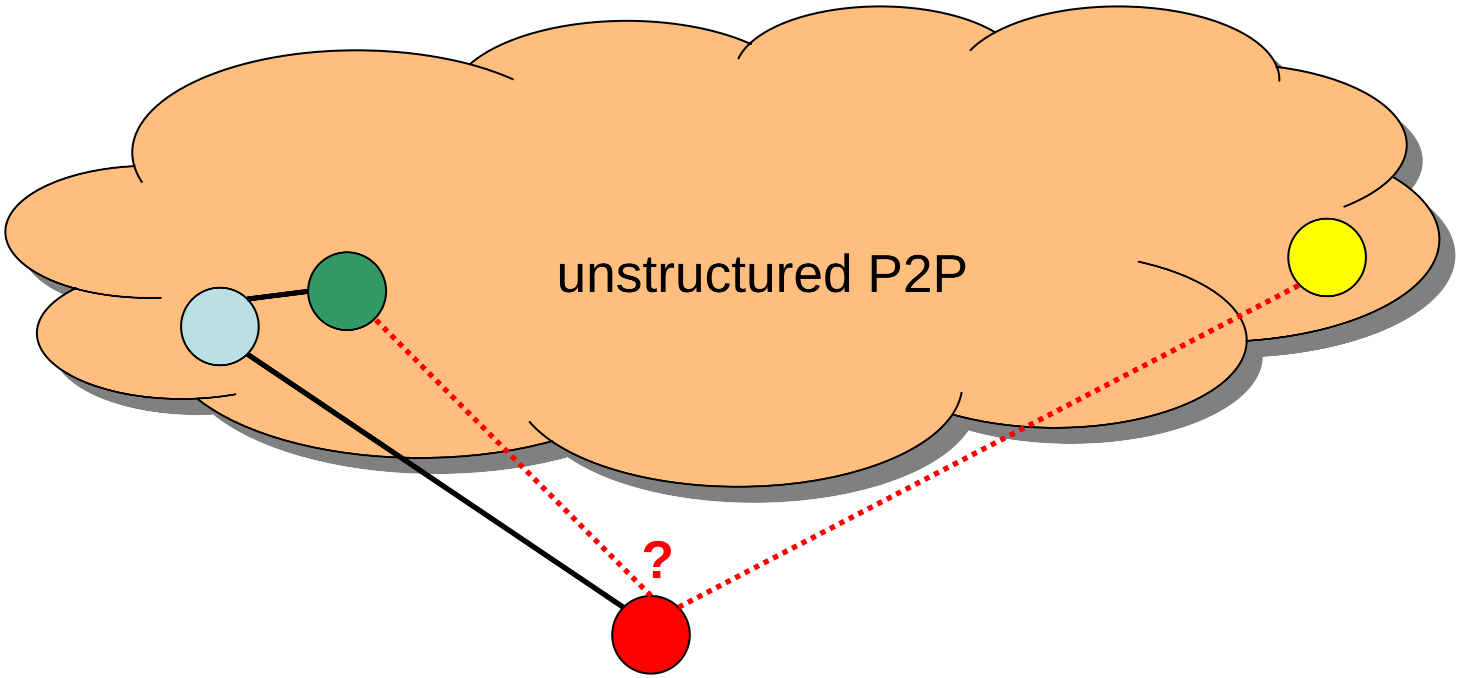
Selfish Agents

- Freeloading…How to generalize BitTorrent's "tit4tat" mechanism?
- But also: In unstructured P2P systems: Who should I connect to?
  – I want to be highly connected since this improves my searches
  – I want to have few neighbors only (forward too many searches)
  – Hypercubic networks probably are a "socially efficient" solution, however, if every node acts selfishly, do we end up with a hypercubic network?!?
- "On a network creation game". Alex Fabrikant, Ankur Luthra, Elitza Maneva, Christos H. Papadimitriou, Scott Shenker, at PODC 2003

Overview

- Introduction
- Past
- Present
  - Dynamic systems & mobility
  - Fault-tolerance (crash failures)
  - Security (Byzantine failures)
  - Selfish agents & computational economy
  - Simple and implementable algorithms
  - Local algorithms
  - Geometry, metrics, bounded growth, etc.
  - Applications

Unstructured P2P: Who should I connect to?



- How do I figure out that the yellow node is farther away?
  Idea: Cluster the network using a generalized MIS (ε-net).

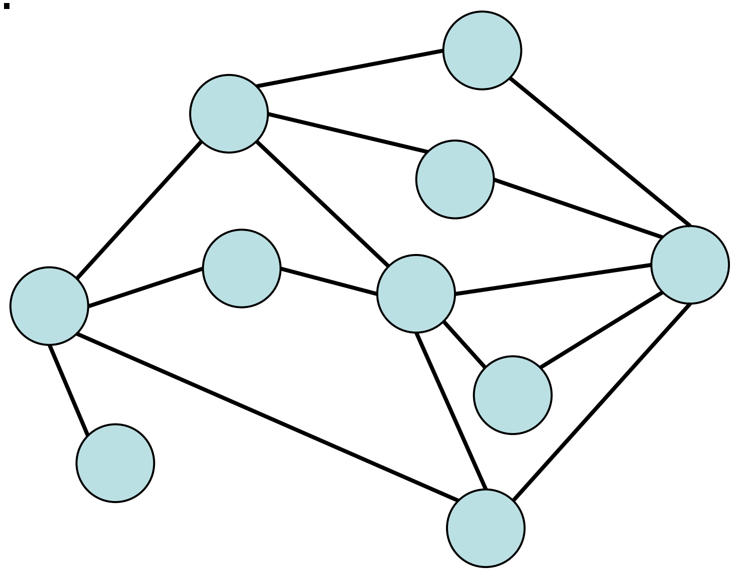
- "Structuring Unstructured P2P Networks". Stefan Schmid, Roger Wattenhofer, in submission.

Overview

- Introduction
- Past
- Present
  - Dynamic systems & mobility
  - Fault-tolerance (crash failures)
  - Security (Byzantine failures)
  - Selfish agents & computational economy
  - Simple and implementable algorithms
  - Local algorithms
  - Geometry, metrics, bounded growth, etc.
  - Applications

Local Algorithms

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



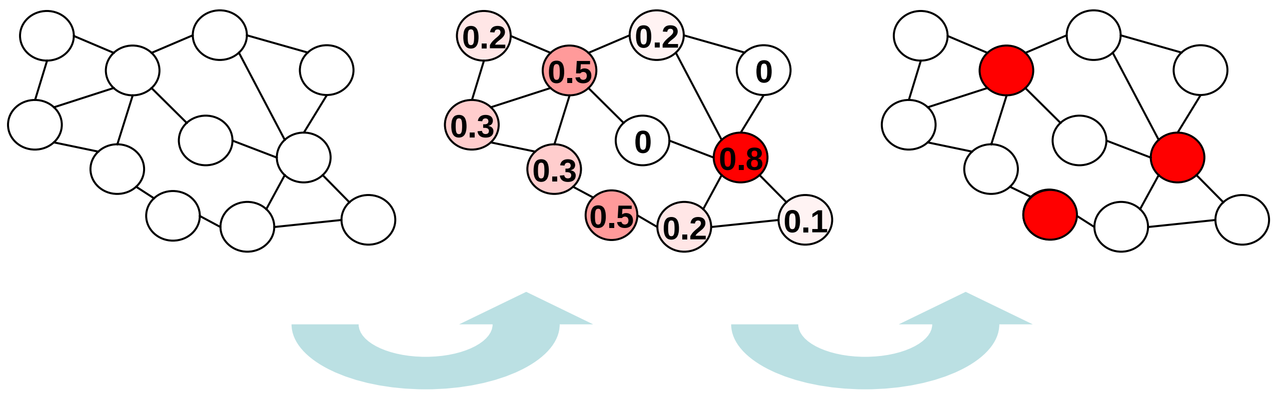
- It might be favorable to have few nodes in the DS. This is known as the Minimum DS problem.

- This by itself is a hard problem, however, the solution must be local (global solutions are impractical in dynamic P2P networks) – topology of graph "far away" should not influence a local decision.

- "Constant-Time Distributed Dominating Set Approximation". Fabian Kuhn, Roger Wattenhofer, at PODC 2003.

Algorithm Overview

Input: Local Graph	Fractional Dominating Set	Dominating Set



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

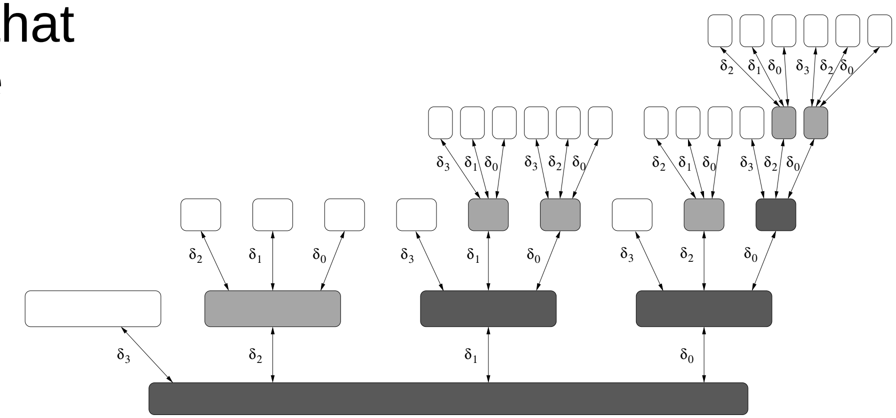
Phase B:
Probabilistic
algorithm

A Lower Bound

- “What cannot be computed locally!” Fabian Kuhn, Thomas Moscibroda, Roger Wattenhofer, at PODC 2004.

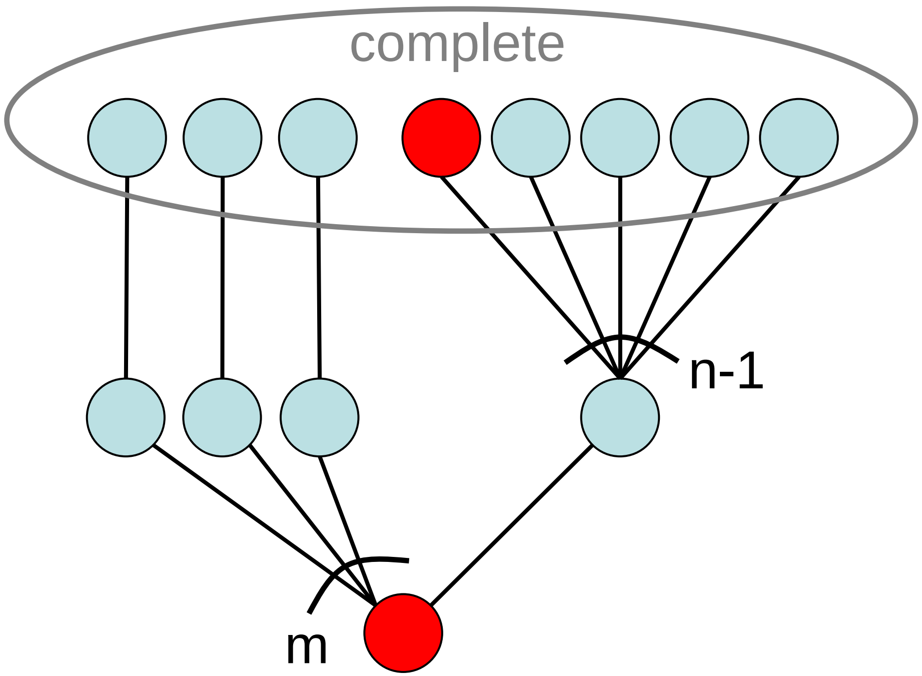
- Model: In a network/graph G (nodes = processors), each node can exchange a message with all its neighbors for k rounds. After k rounds, the node needs to decide.

- We construct a 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.

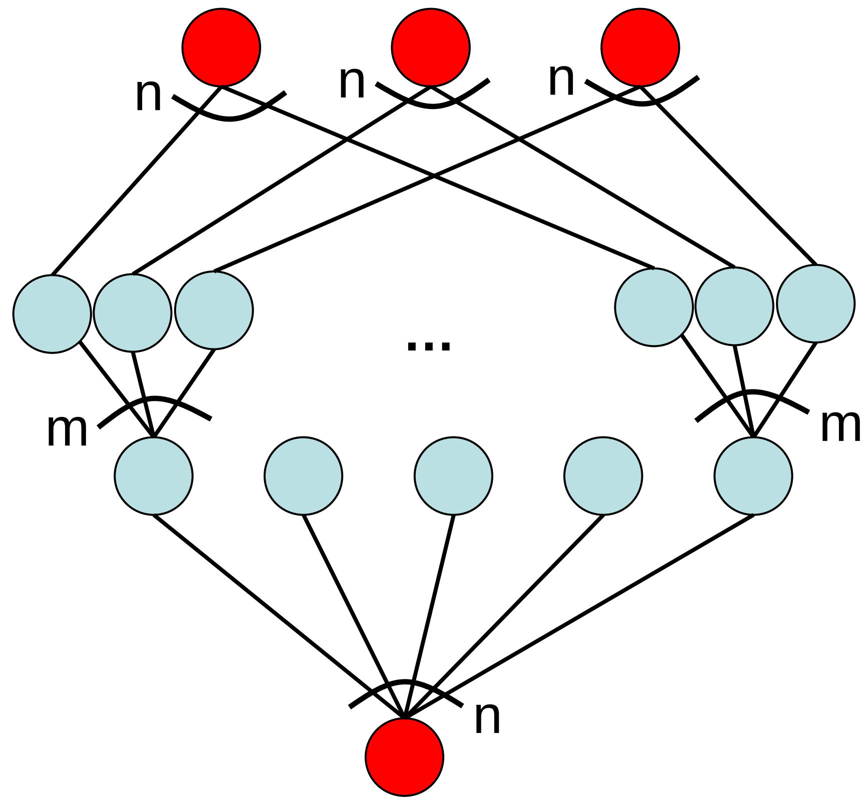


Lower Bound for Dominating Sets: Intuition…

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



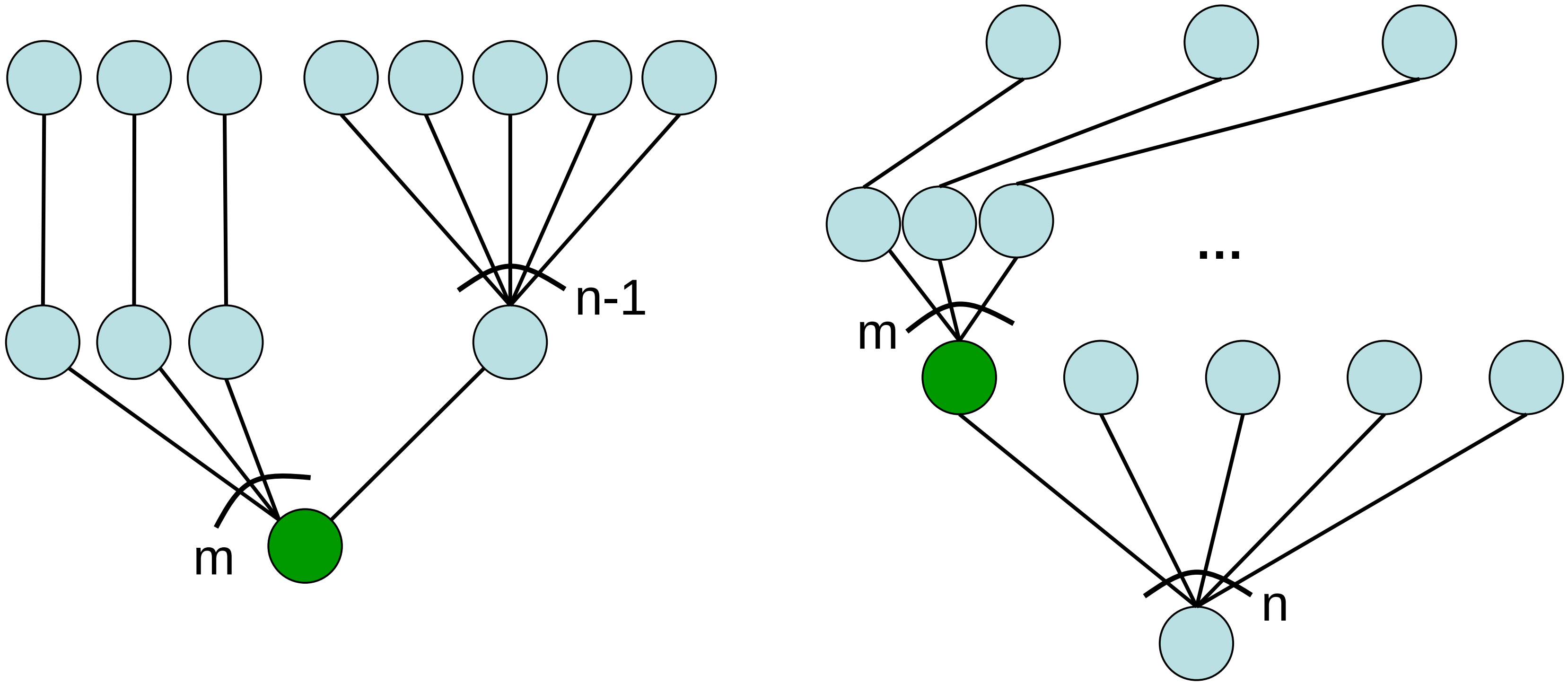
$|DS_{OPT}| = 2.$



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

Lower Bound for Dominating Sets: Intuition…

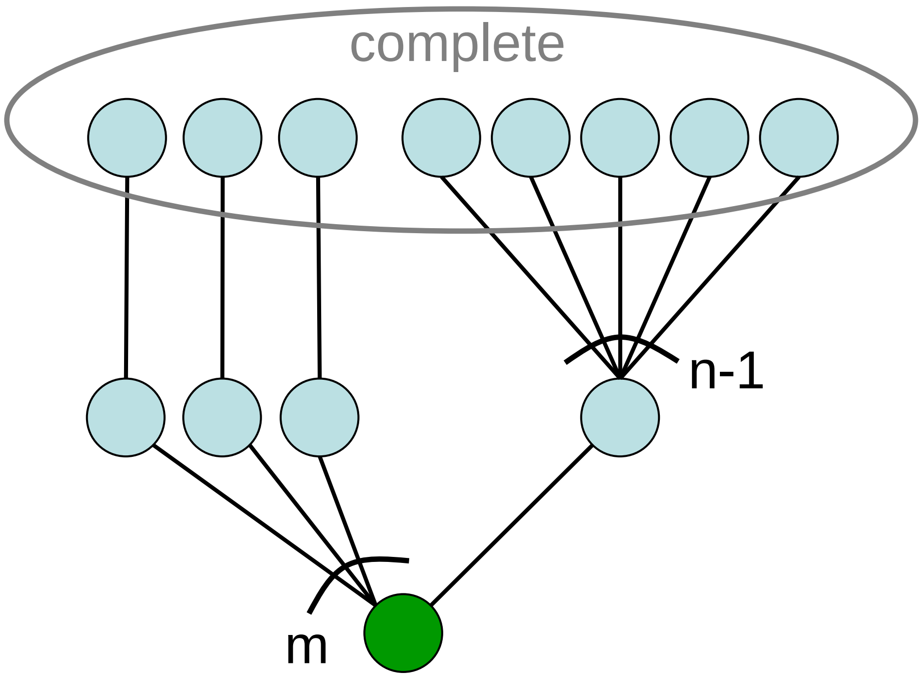
- In local algorithms, nodes must decide only using local knowledge.
- In the example green nodes see exactly the same neighborhood.

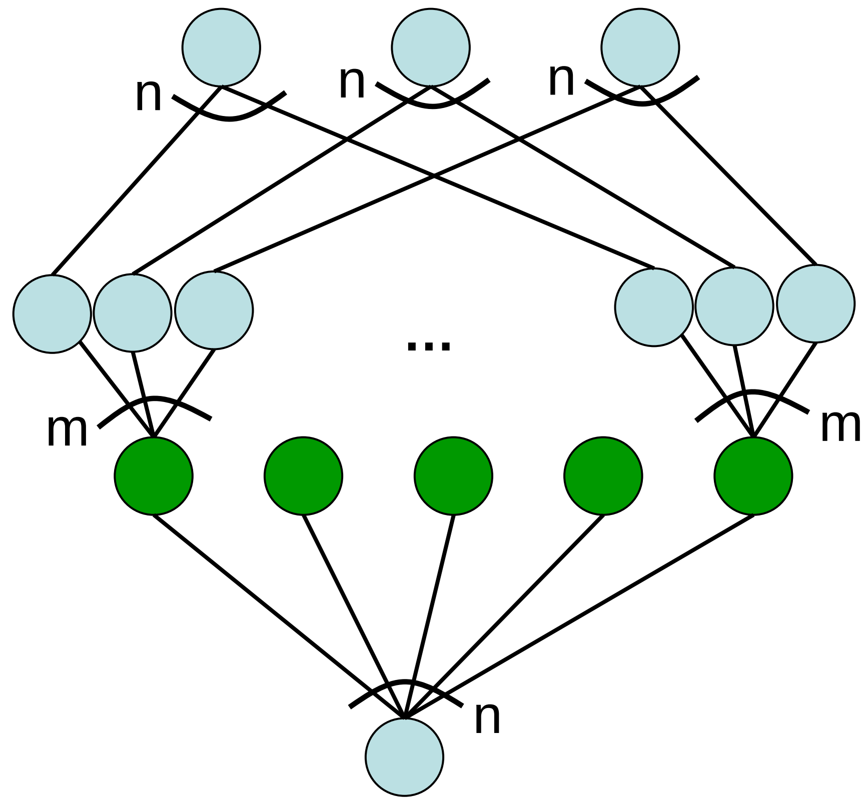


- So these green nodes must decide the same way!

Lower Bound for Dominating Sets: Intuition…

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





$|DS_{OPT}| = 2.$

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

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

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

Results

- Many problems (vertex cover, dominating set, matching, independent set, ε-net, etc.) cannot be approximated better than $\Omega(n^{c/k^2} / k)$ and/or $\Omega(\Delta^{1/k} / k)$.

- It follows that a polylogarithmic approximation of many standard problems needs at least $\Omega(\log \Delta / \log\log \Delta)$ and/or $\Omega((\log n / \log\log n)^{1/2})$ time.

- For some (exotic) problems this is tight.

Overview

- Introduction
- Past
- Present
  - Dynamic systems & mobility
  - Fault-tolerance (crash failures)
  - Security (Byzantine failures)
  - Selfish agents & computational economy
  - Simple and implementable algorithms
  - Local algorithms
  - Geometry, metrics, bounded growth, etc.
  - Applications

Geometry strikes back!

- Having a logarithmic number of hops is nice, however, hopping back and forth over continents is a major nuisance.

- So instead of placing joining nodes randomly into the structured P2P system, one might think of placing nodes such that the total latency of a search is small. In other words, geographically close nodes should also be close in the topology.

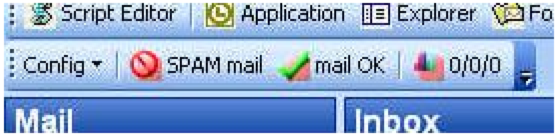
- In fact, this was already the topic of the Plaxton et al. paper, but it's certainly coming back. These days people have new models for the Internet graph ("almost metric") which allow for new exciting results.

- "Competitive Algorithms for Distributed Data Management". Yair Bartal, Amos Fiat, and Yuval Rabani, at STOC 1992.

Overview

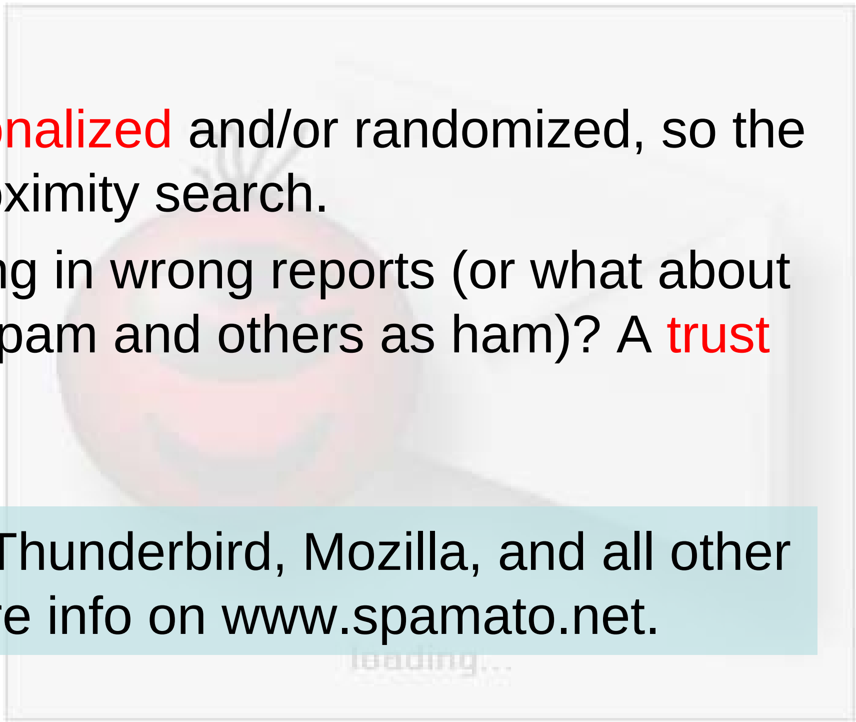
- Introduction
- Past
- Present
  - Dynamic systems & mobility
  - Fault-tolerance (crash failures)
  - Security (Byzantine failures)
  - Selfish agents & computational economy
  - Simple and implementable algorithms
  - Local algorithms
  - Geometry, metrics, bounded growth, etc.
  - Applications

SP.a.M/\TØ – An Extendable Spam Filter System

- Collaborative spam filter, users report spam: 

- Principle Idea: Reported spam is stored in DHT repository.

- Problems:
  - These days spams are personalized and/or randomized, so the DHT needs some form of proximity search.
  - What about Mr. Bad Guy filling in wrong reports (or what about email that some classify as spam and others as ham)? A trust system is needed.



- Available for Windows/Outlook, Thunderbird, Mozilla, and all other mail clients through a proxy. More info on www.spamato.net.

Conclusions

- The most exciting years of P2P still to come!
- On the file-sharing side we see the first structured systems (Kad)
- On the research/theory side there are a bunch of stimulating areas:
  - Dynamic systems & mobility
  - Fault-tolerance (crash failures)
  - Security (Byzantine failures)
  - Selfish agents & computational economy
  - Simple (implementable) algorithms
  - Local algorithms
  - Geometry, metrics, bounded growth, etc.
- Last not least applications beyond file sharing are emerging!

Questions?
Comments?

Distributed
Computing
Group
Roger Wattenhofer