Distributed Computing over
Communication Networks:

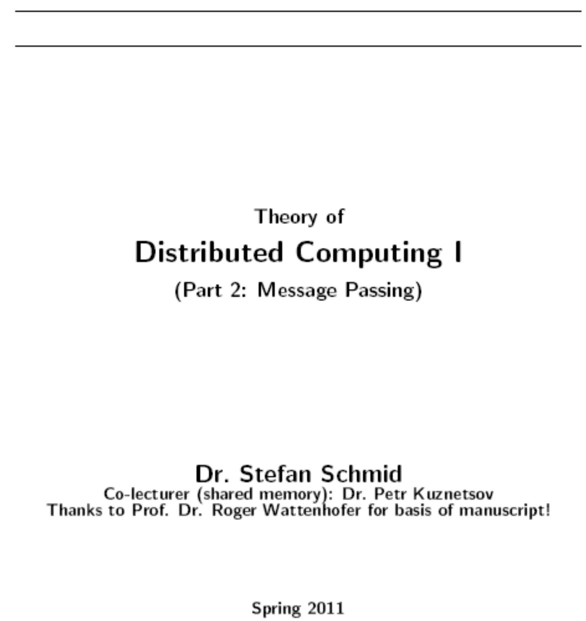
Topology

(with an excursion to P2P)

Some administrative comments...

There will be a „**Skript**“ for this part of the lecture. (Same as slides, except for today... ☺)

Will be online together with the **slides** after the lecture (or during...).



Theory of

Distributed Computing I

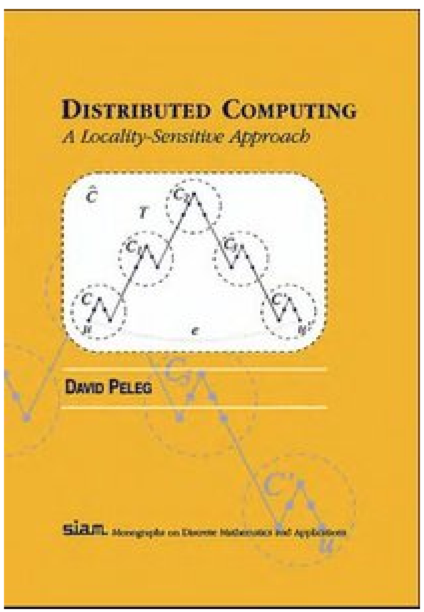
(Part 2: Message Passing)

Dr. Stefan Schmid

Co-lecturer (shared memory): Dr. Petr Kuznetsov

Thanks to Prof. Dr. Roger Wattenhofer for basis of manuscript!

Spring 2011



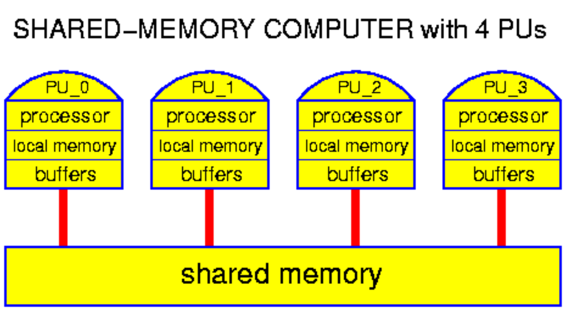
Moreover, the course follows the **cool book by Peleg** (but only first, simple chapters are covered)

Shared Memory vs Message Passing?

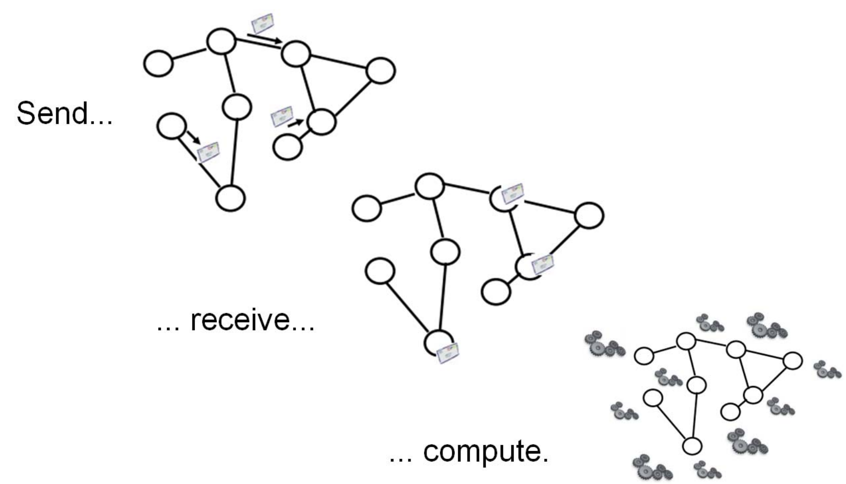
Same same but different?

.... different:

- communication over networks
- focus on message or communication (bit-) complexity
- decoupling / synchronicity / ...: not necessarily



vs



Shared Memory vs Message Passing?

Same same but different?

.... same?

not in this course...

Sharing Memory Robustly in Message-Passing Systems

HAGIT ATTIYA

The Technion, Haifa, Israel

AMOTZ BAR-NOY

IBM T. J. Watson Research Center, Yorktown Center, Yorktown Heights, New York

AND

DANNY DOLEV

IBM Almaden Research Center, San Jose, California and Hebrew University, Jerusalem, Israel

Abstract. Emulators that translate algorithms from the shared-memory model to two different message-passing models are presented. Both are achieved by implementing a wait-free, atomic, single-writer multi-reader register in unreliable, asynchronous networks. The two message-passing models considered are a complete network with processor failures and an arbitrary network with dynamic link failures.

These results make it possible to view the shared-memory model as a higher-level language for designing algorithms in asynchronous distributed systems. Any wait-free algorithm based on atomic, single-writer multi-reader registers can be automatically emulated in message-passing systems, provided that at least a majority of the processors are not faulty and remain connected. The overhead introduced by these emulations is polynomial in the number of processors in the

What you will learn!

Topology: What (communication) network is good?

The basics: leader election, tree algorithms, ...

Classical TCS reloaded: Maximal independent sets computed distributedly?

Distributed lower bounds?

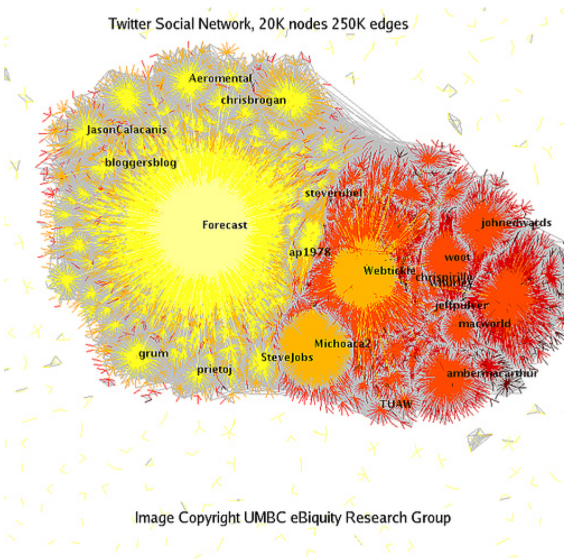
Graph coloring

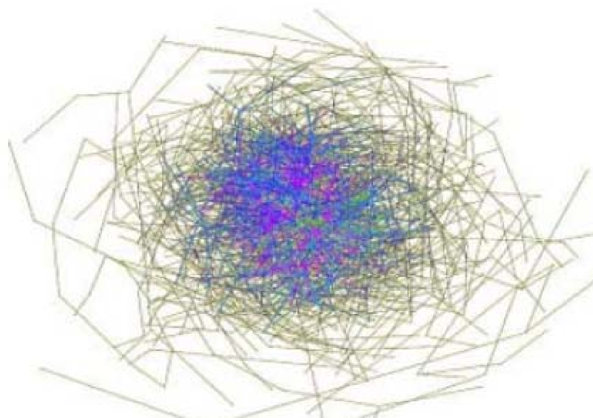
maybe: social networks or game theory

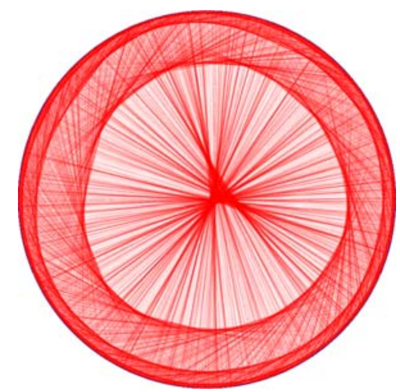
Good Topologies?

Topology („network graph“)

- sometimes *given* (e.g., social networks)
- sometimes chaotic / semi-structured / „organically growing“ (e.g., unstructured peer-to-peer networks)
- sometimes subject to design and optimization (e.g., parallel computer architectures, structured peer-to-peer networks, etc.)




Gnutella 2001


Chord DHT

What is a „good topology“?!

Good Topologies?

What is a „good topology“? It depends...

- How to interconnect the cities of a country with an efficient railroad infrastructure?
- How to to interconnect components of a parallel computer?
- How to interconnect peers of a peer-to-peer system?
- Or even: how to control the „topology“ of a wireless network?! (E.g., setting the „transmission radii“ in a smart manner may save energy and increase the throughput due to less interference, etc.)

Possible criteria?!

Criteria?

Simple and efficient routing: implication for topology?

e.g., „short“ paths and low diameter (wrt #hops, latency, energy, ...?), no state needed at „routers“ (destination address defines next hop), good expansion (for flooding), etc.

Scalability: implication for topology?

e.g., small number of neighbors to store (and maintain?), low degree, large maxflow, redundant paths / no bottleneck links, ...

Robustness (random or worst-case failures?): implication for topology?

e.g., „symmetric“ structure, no single point of failure, redundant paths, good expansion, large mincut, k-connectivity, ...

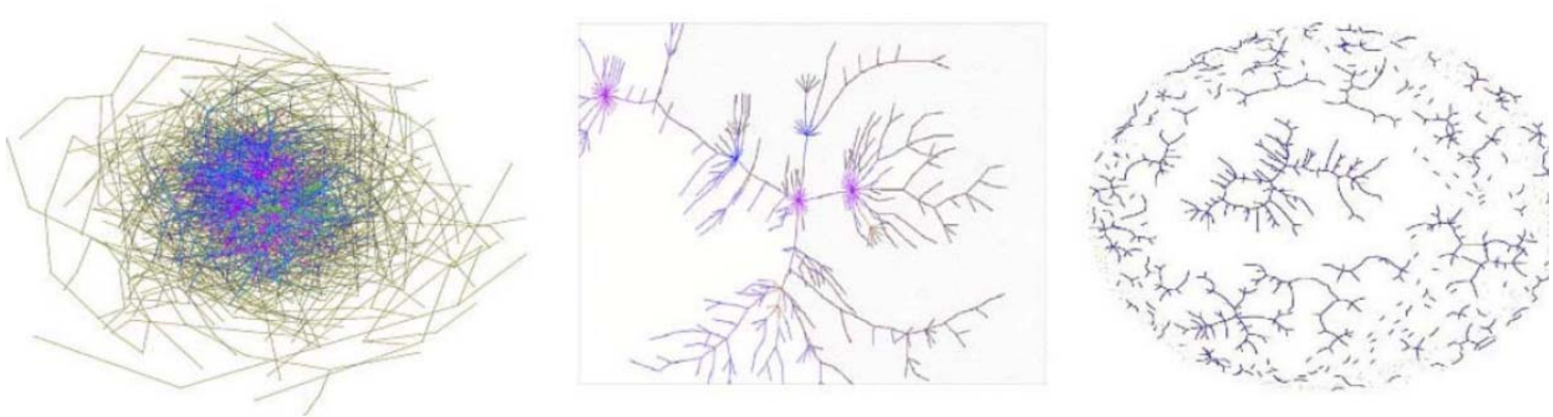
...

Does the Gnutella P2P network have a robust topology?

Not very much... Gnutella topology and also the protocol does not scale well: Gnutella went down when Napster was „unplugged“...

Criteria?

Example: Robustness (e.g., Gnutella)



Measurement study 2001 with ~2000 peers: [Saroiu et al. 2002]

Left: all connections

Middle: 30% random peers removed: still mostly connected („giant component“), robust to random failures / leaves

Right: 4% highest degree peers removed: many disconnected components, not robust

Can we design the topology of a wireless network?!
No notion of „wires“, only disks!

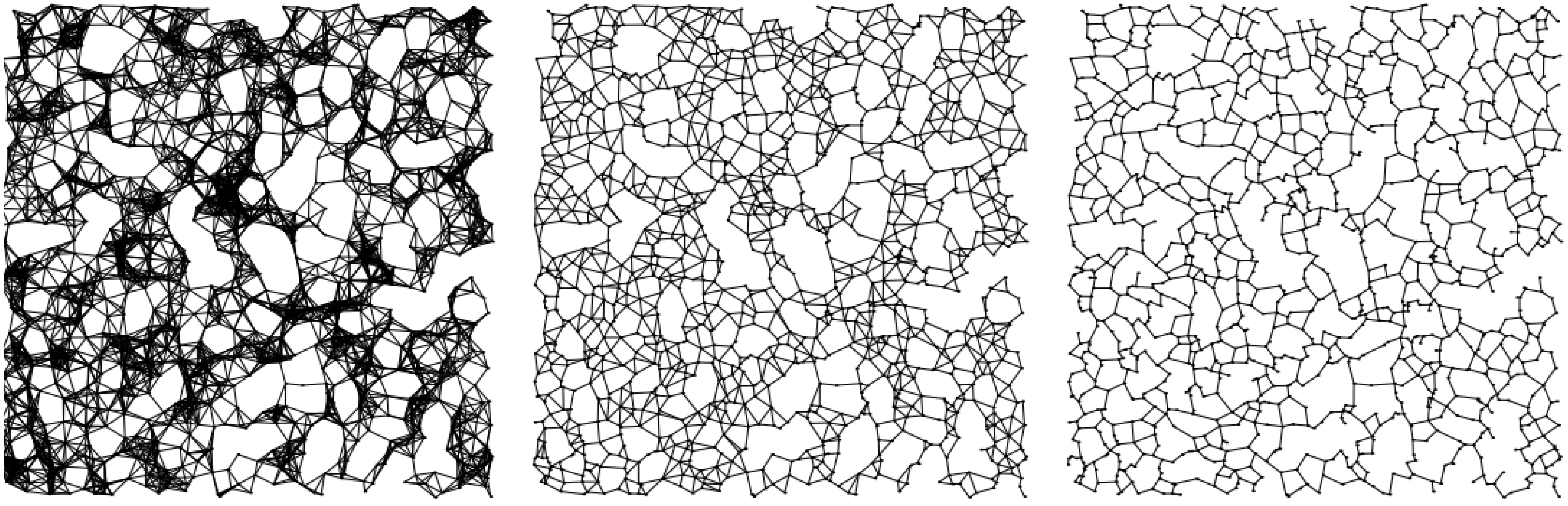
Yes, even if node positions are given!
E.g., by adjusting transmission power! Or by using only a subset of the neighbors to forward things.

Interesting field of **topology control** in wireless networks!

What could be purpose?

Reduce interference, increase throughput, ...
... while maintaining shortest paths or minimal energy paths!
Key words: Gabriel graphs, Delaunay graphs, etc.

Example: XTC Topology Control



Left: Unit Disk Graph (connected to all nodes at distance at most 1)

Middle: Gabriel Graph (subset of links only)

Right: XTC Graph (subset of links can be locally computed)

Short Excursion: Peer-to-Peer Networks

Napster:

centralized, „no topology"

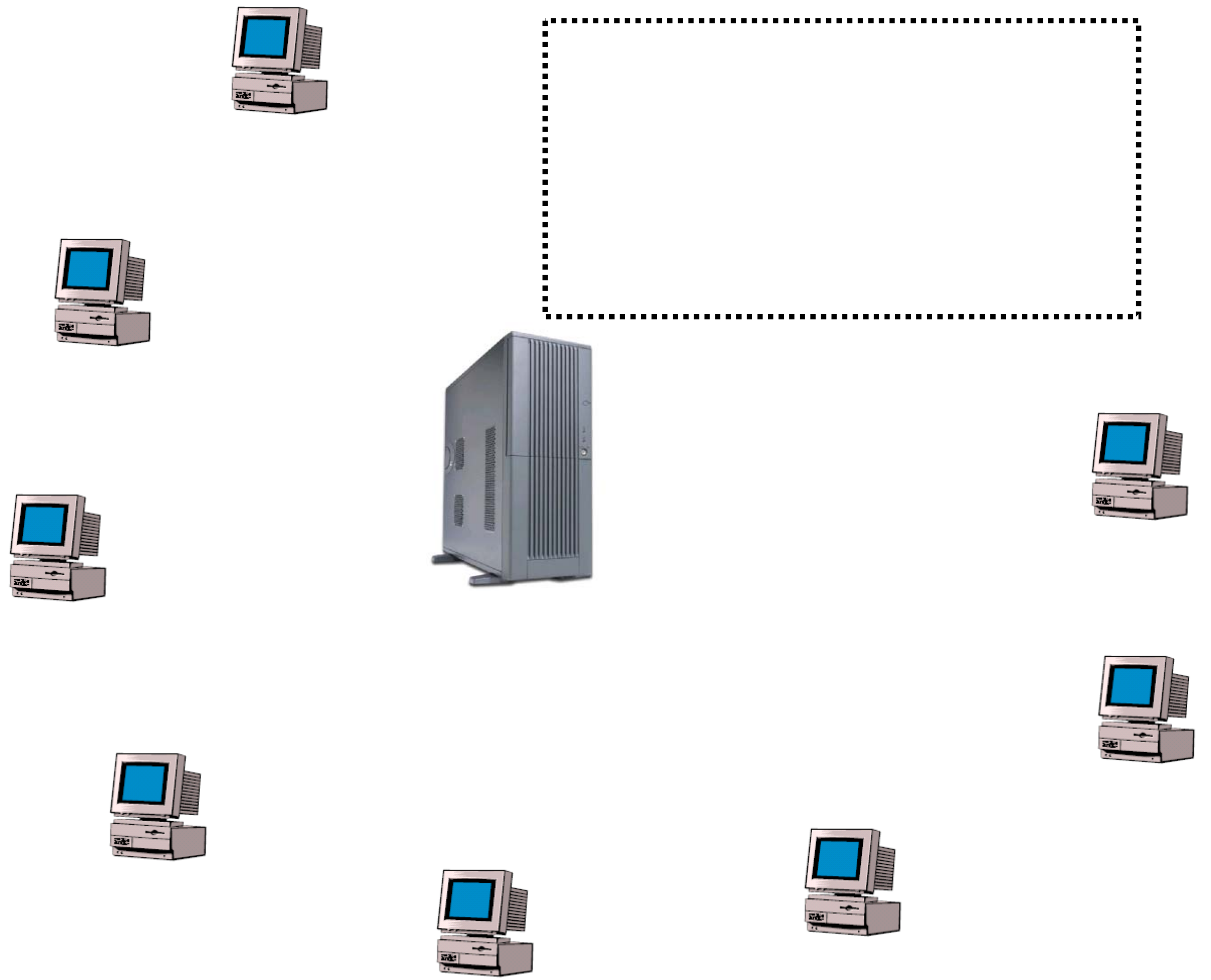
Gnutella:

fully decentralized, „random topology"

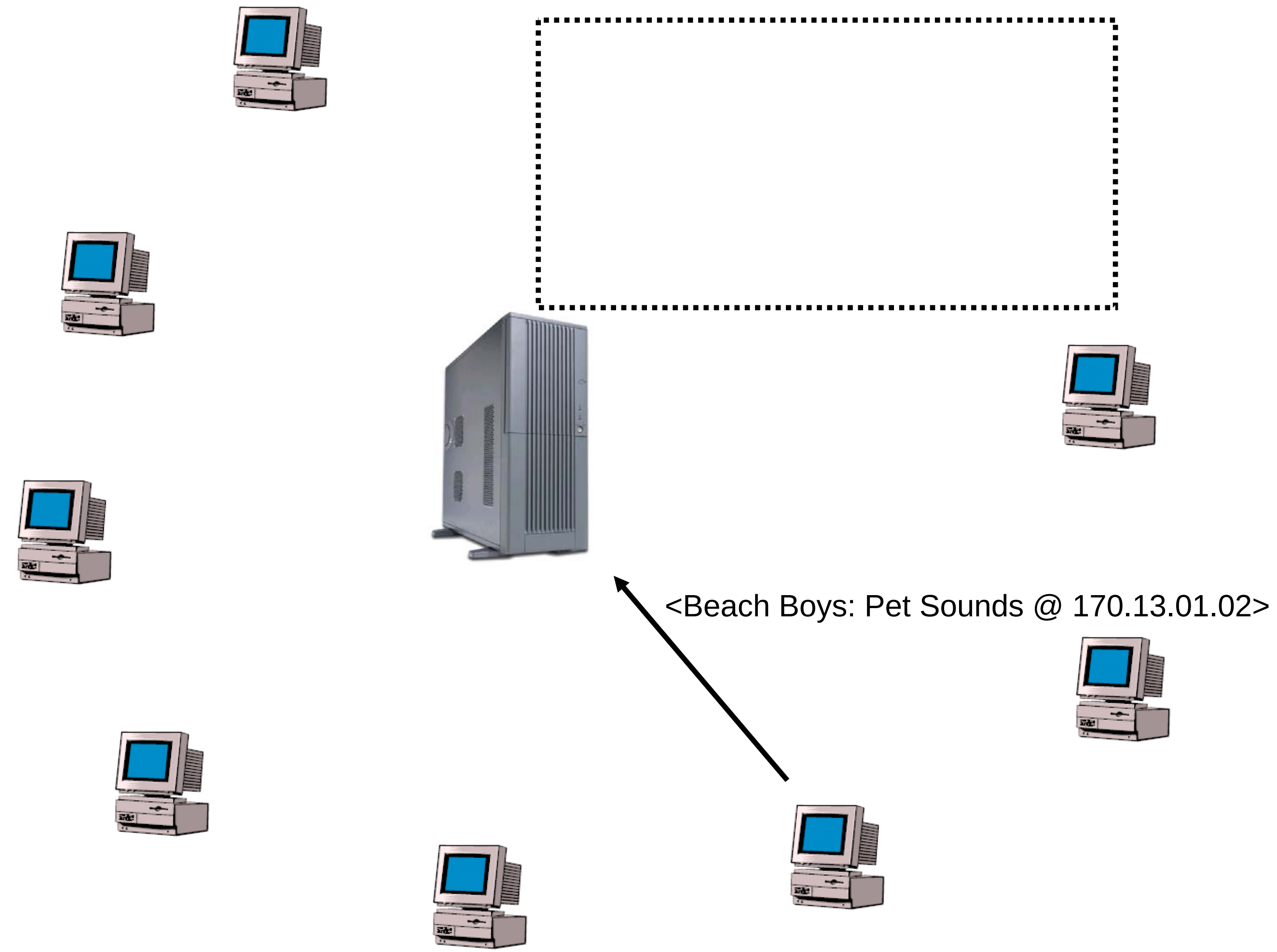
DHT:

„structured", often hypercubic topology (why?)

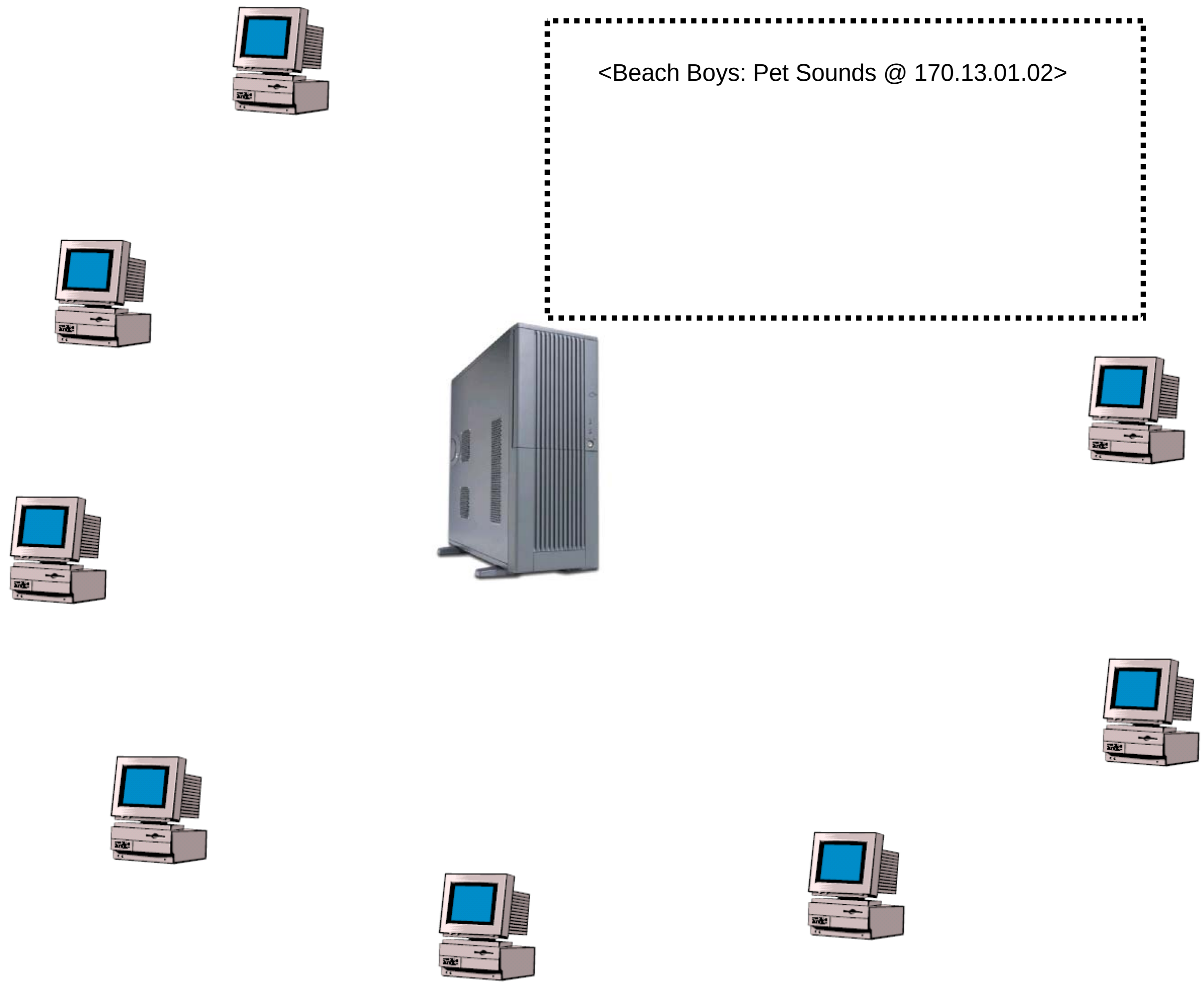
Napster: Centralized index



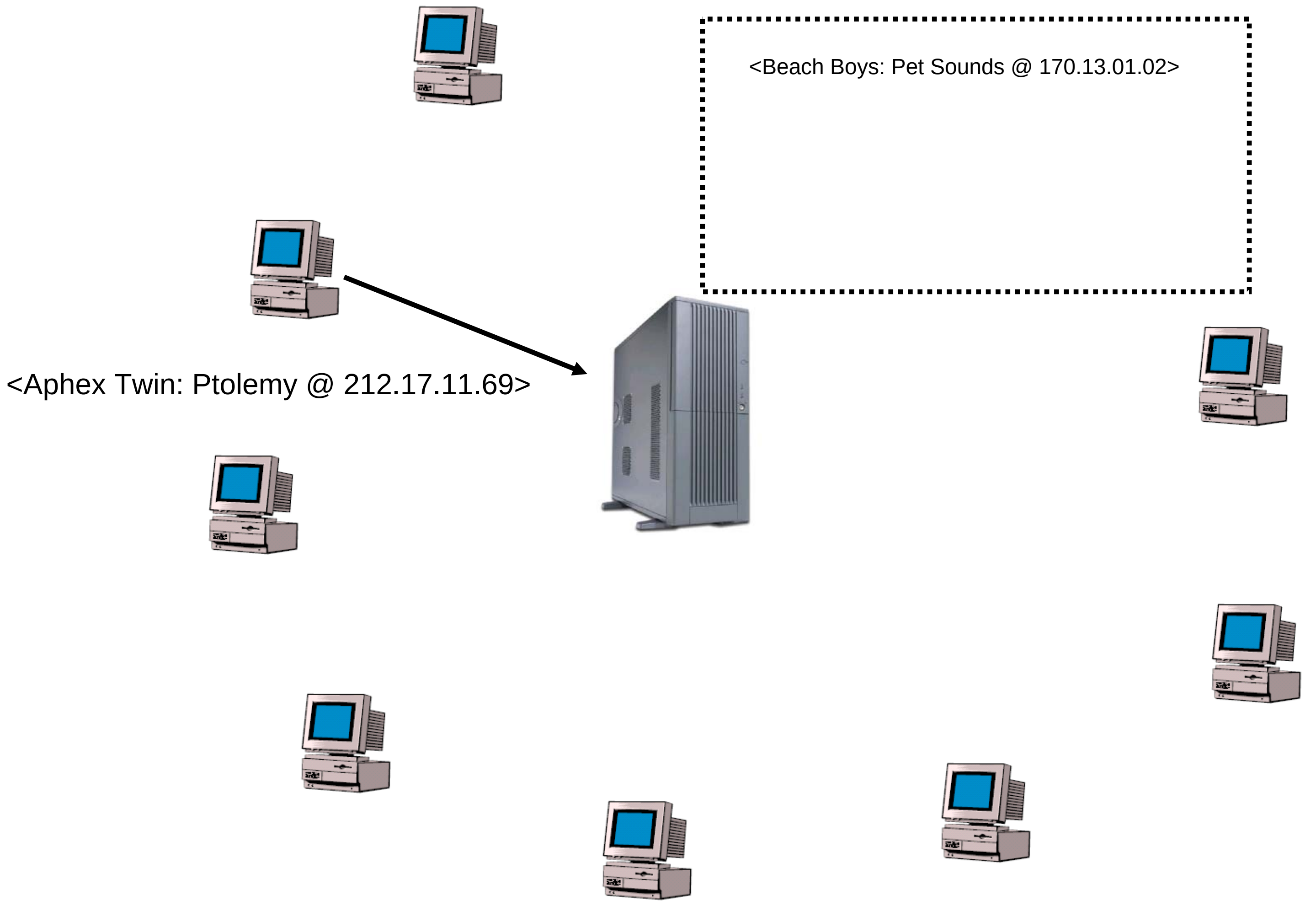
Napster



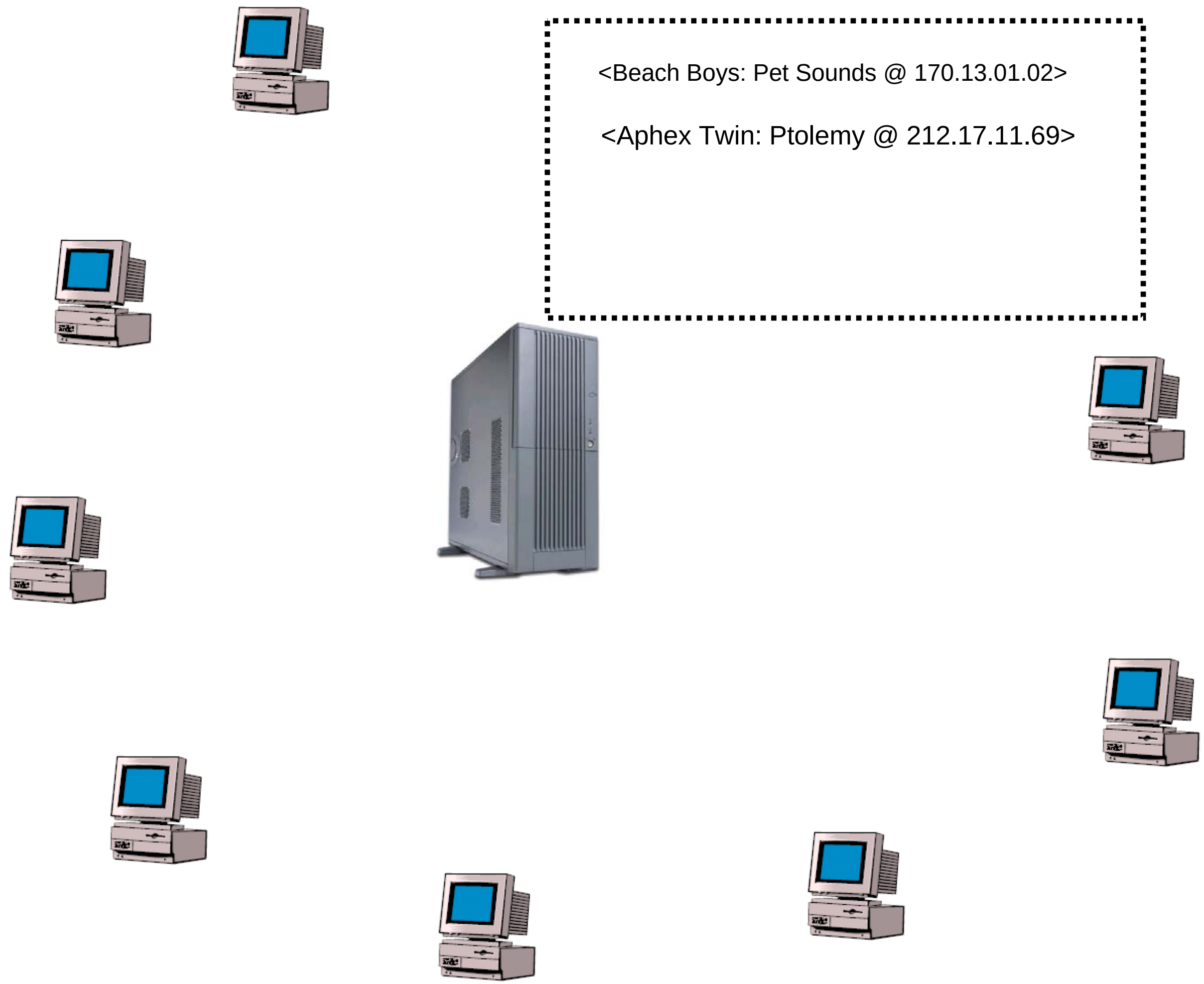
Napster



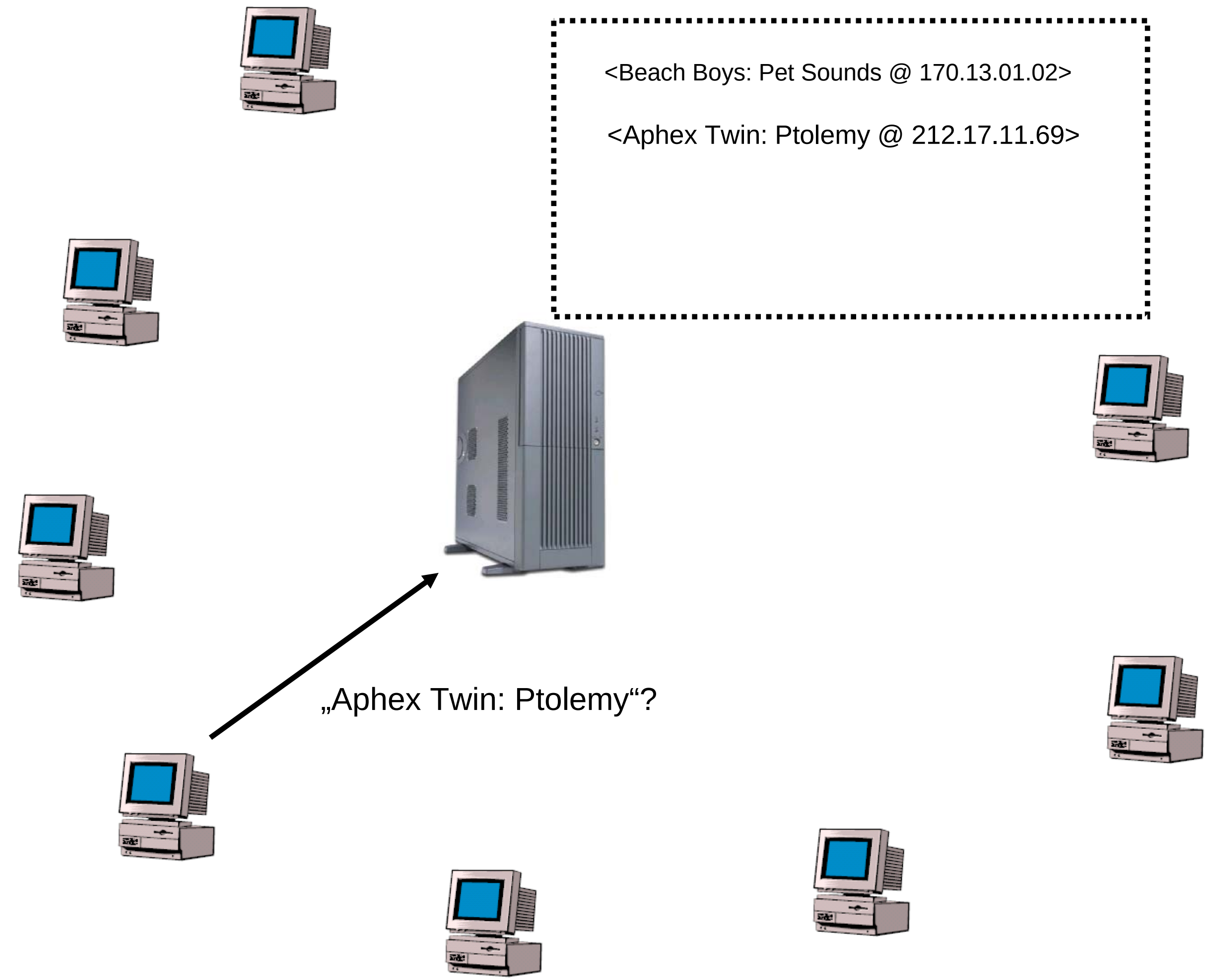
Napster


<Beach Boys: Pet Sounds @ 170.13.01.02>
<Aphex Twin: Ptolemy @ 212.17.11.69>

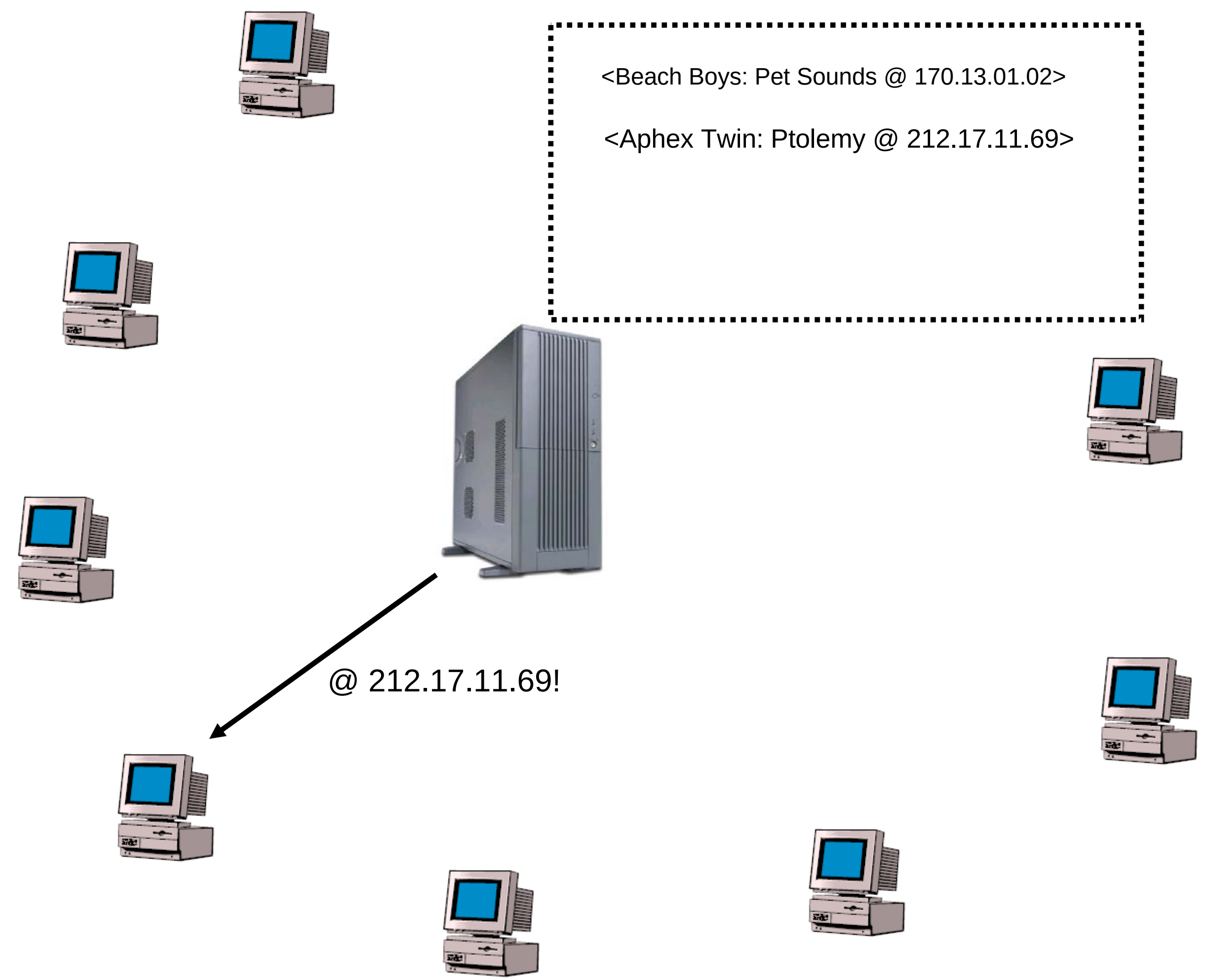
Napster


<Beach Boys: Pet Sounds @ 170.13.01.02>
<Aphex Twin: Ptolemy @ 212.17.11.69>

Napster

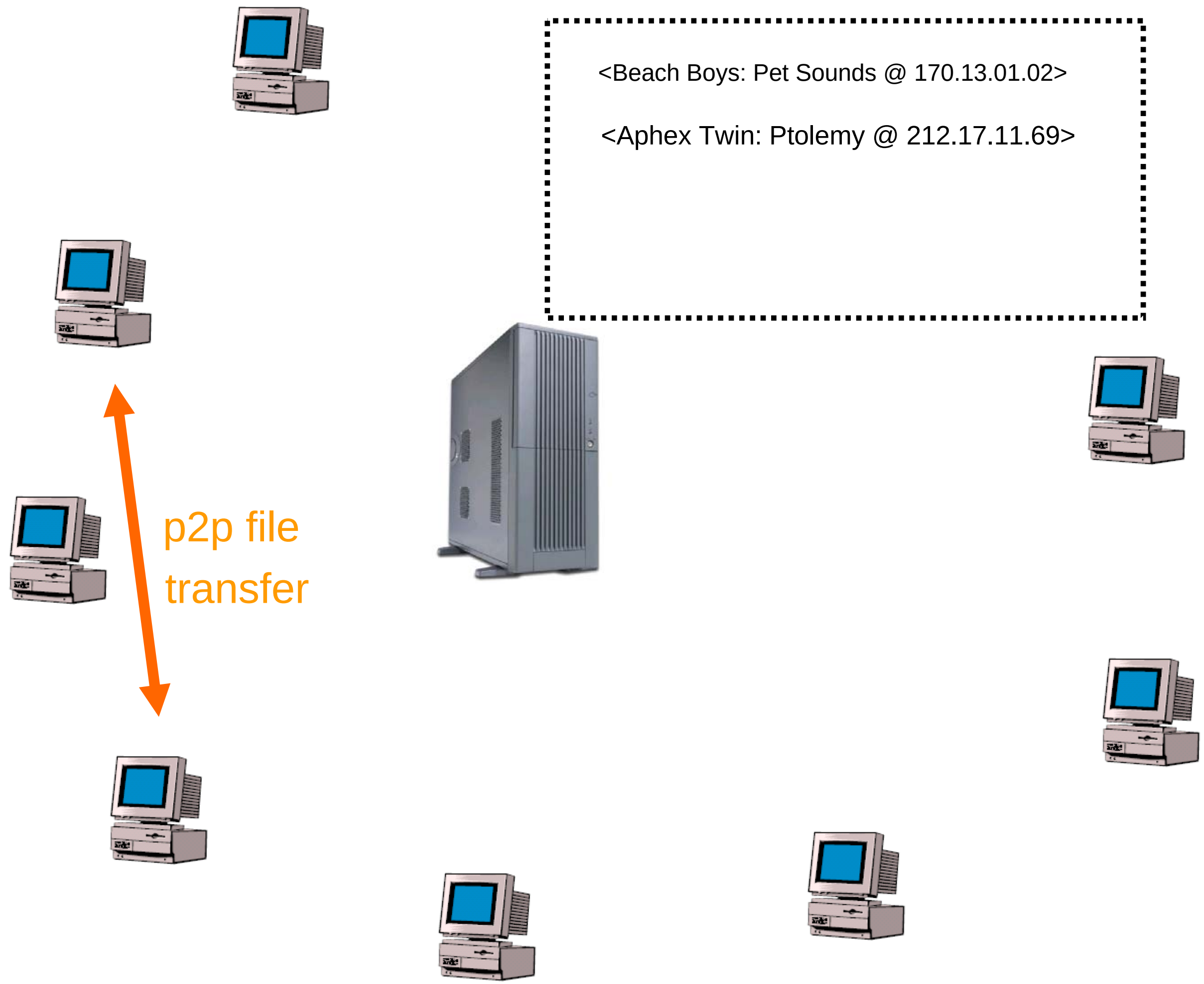

<Beach Boys: Pet Sounds @ 170.13.01.02>
<Aphex Twin: Ptolemy @ 212.17.11.69>
„Aphex Twin: Ptolemy“?

Napster



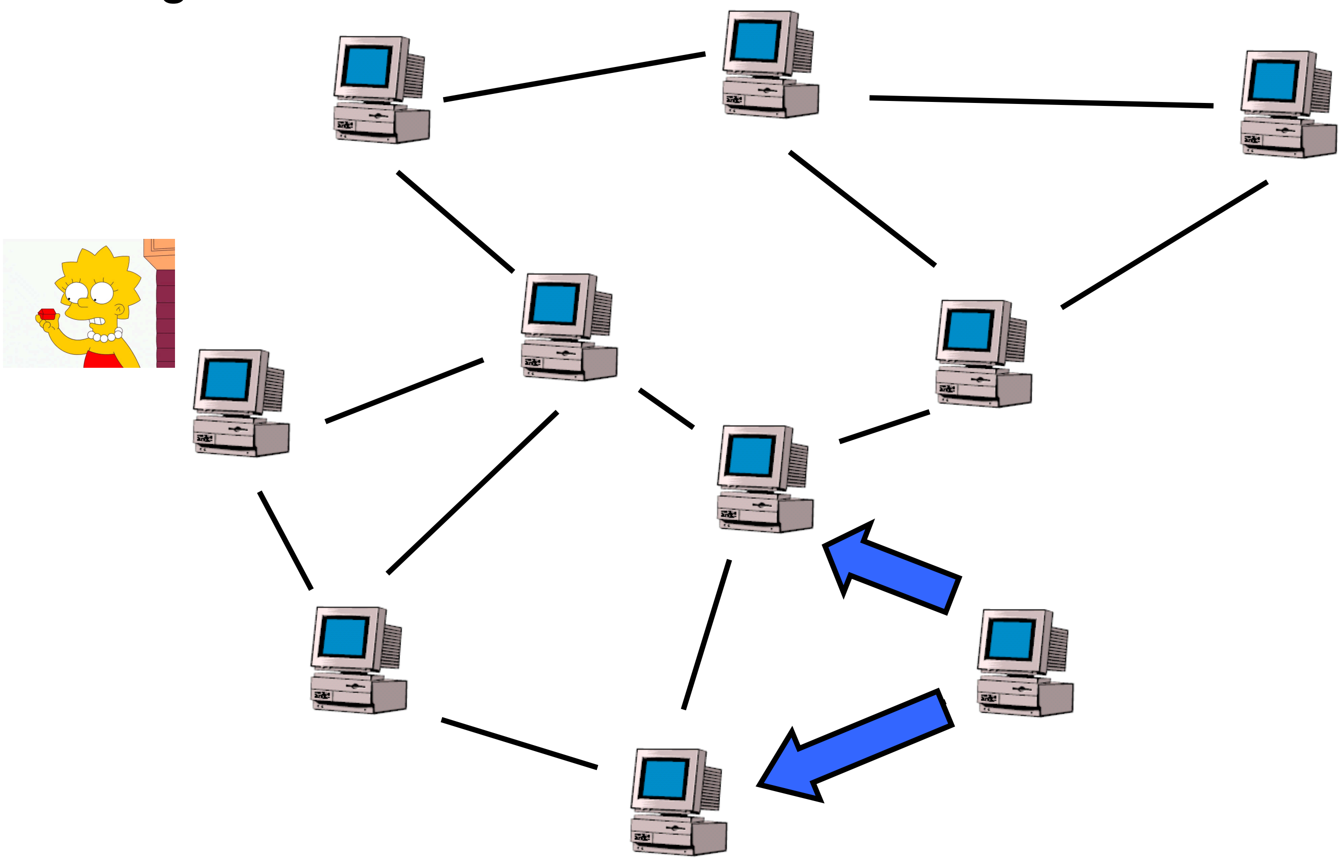
<Beach Boys: Pet Sounds @ 170.13.01.02>
<Aphex Twin: Ptolemy @ 212.17.11.69>
@ 212.17.11.69!

Napster

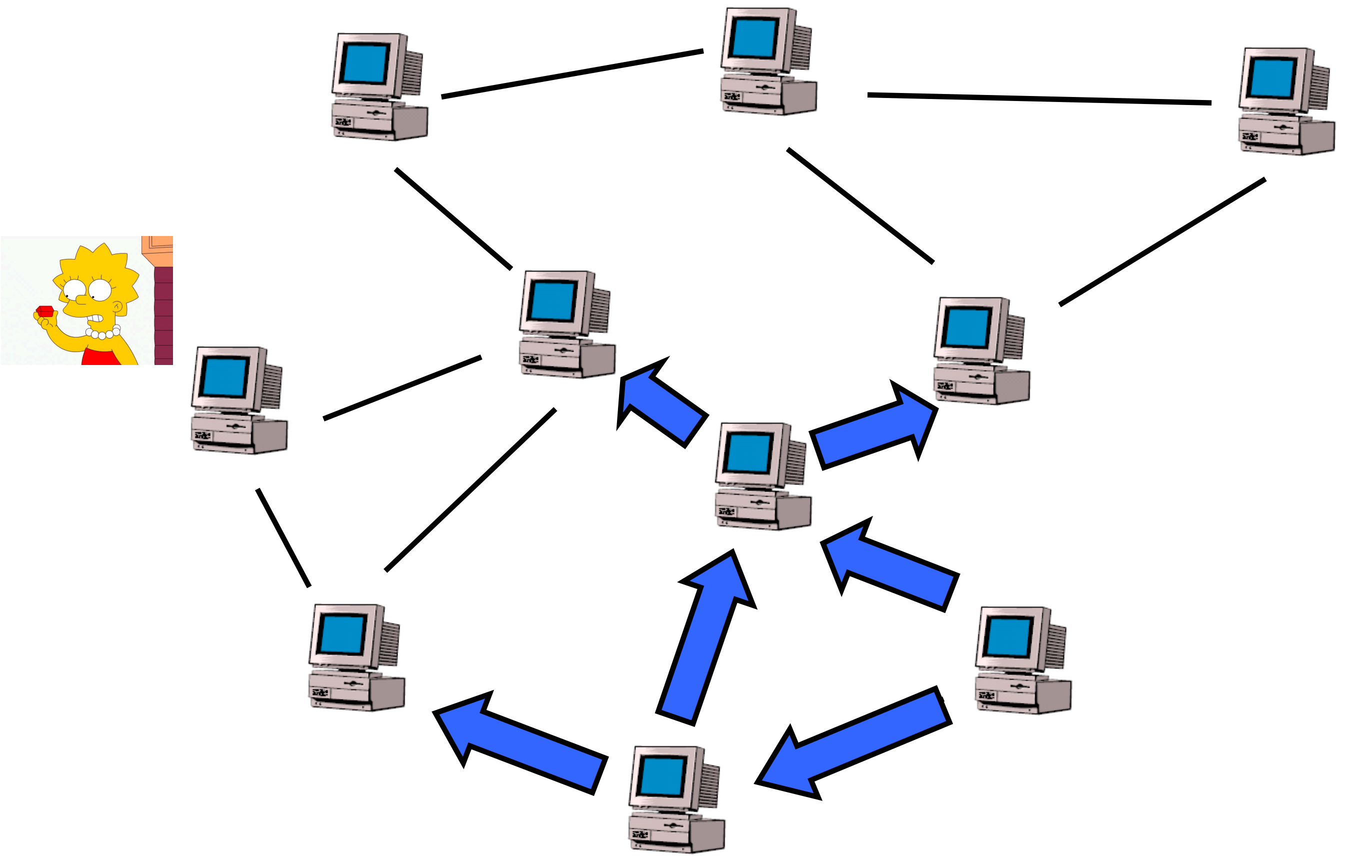

<Beach Boys: Pet Sounds @ 170.13.01.02>
<Aphex Twin: Ptolemy @ 212.17.11.69>
p2p file transfer

Gnutella: Unstructured network & flooding

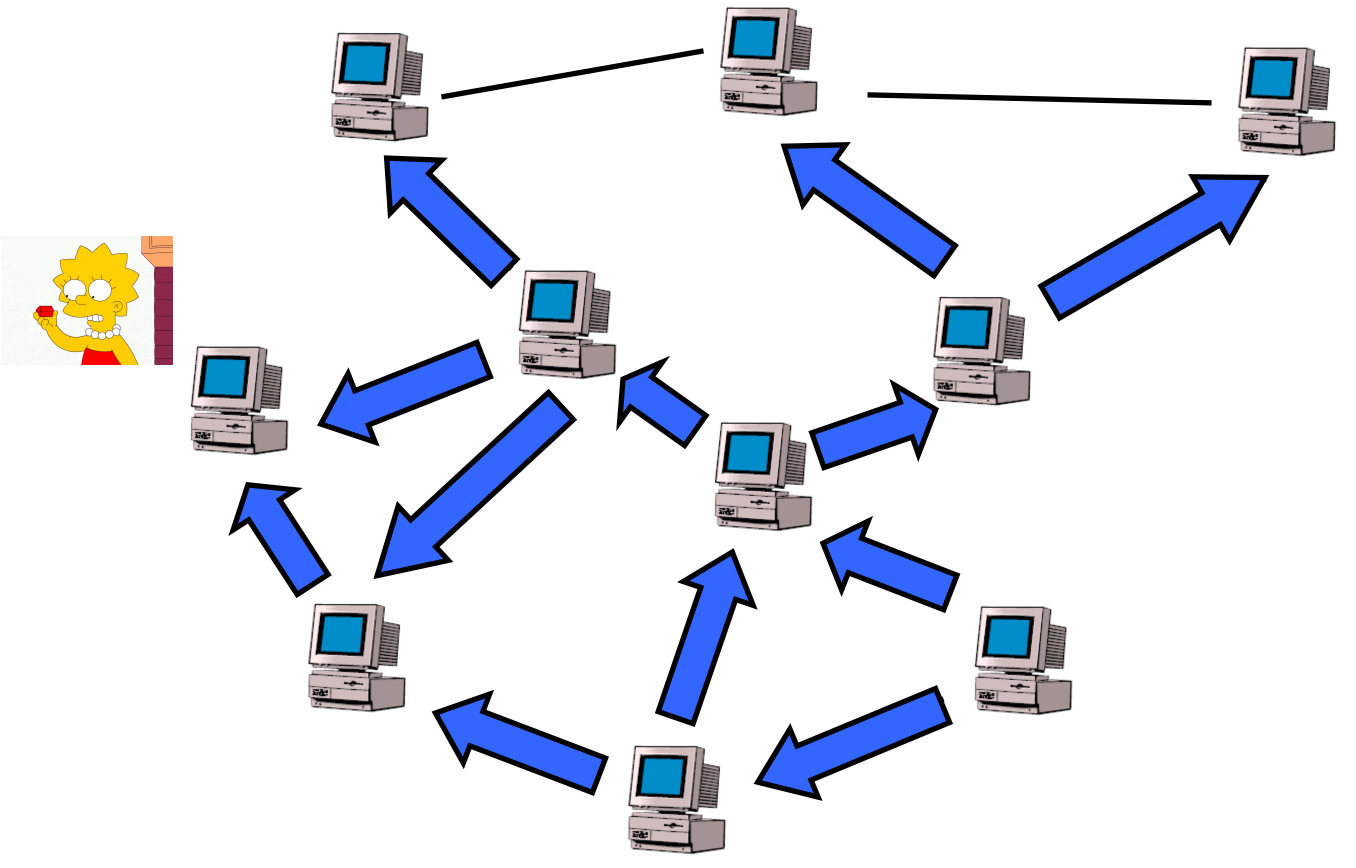
Peers basically connect to neighbors of neighbors:
high clustering...



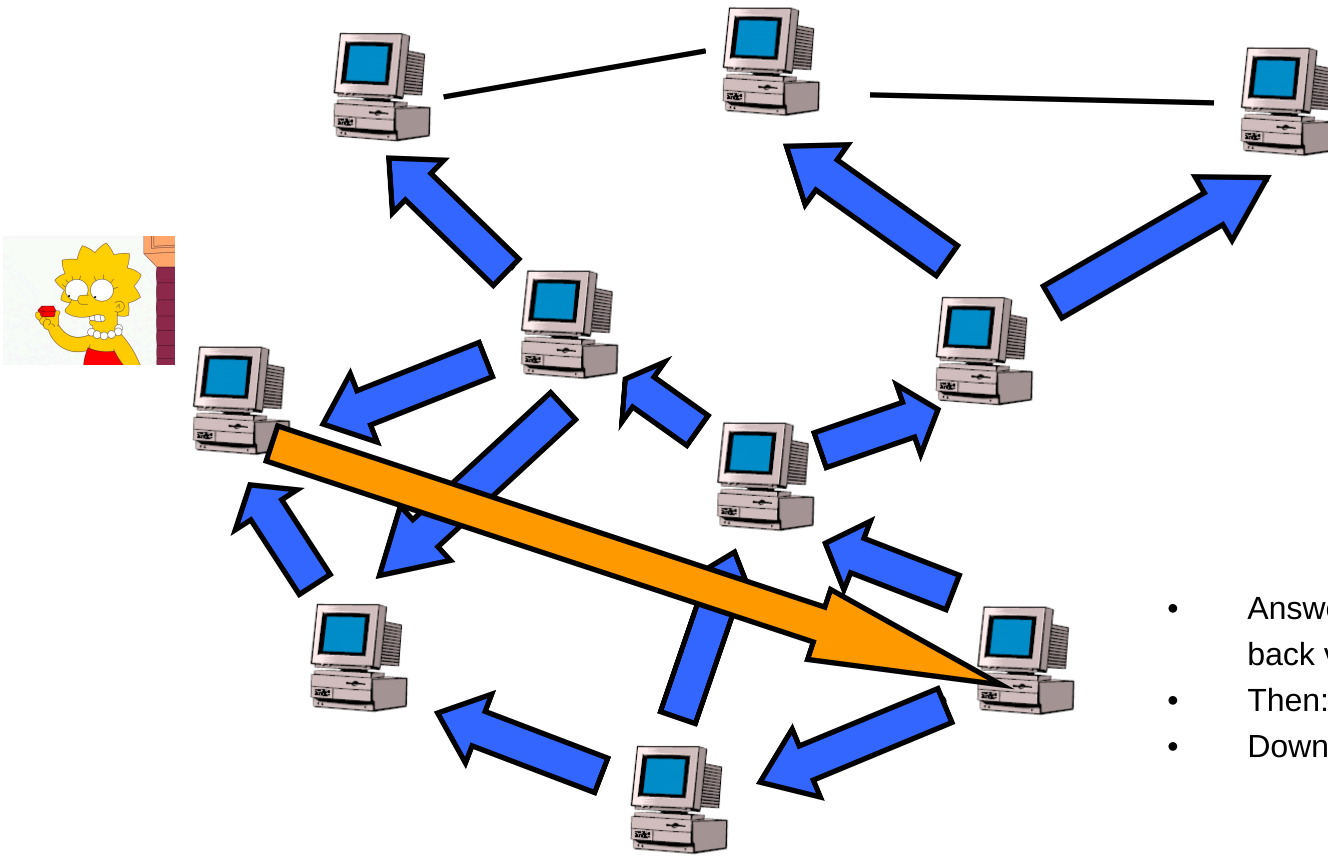
Gnutella



Gnutella



Gnutella



- Answers come back via multihop
- Then: direct download
- Download from one source

Distributed Hash Tables (DHTs)

DHTs: decentralized peer-to-peer systems with routing wrt to *keys*

Oversimplifying:

1. The topology of DHTs is often hypercubic (easy routing, good degree and diameter, robustness, ...)

2. Which peers should store which data?

   Concept of consistent hashing:

   map both peers and files/data onto a 1-dimensional virtual ring [0,1)

   - Peers have random ID

   - Files (e.g., contents or file names) are hashed to [0,1) too

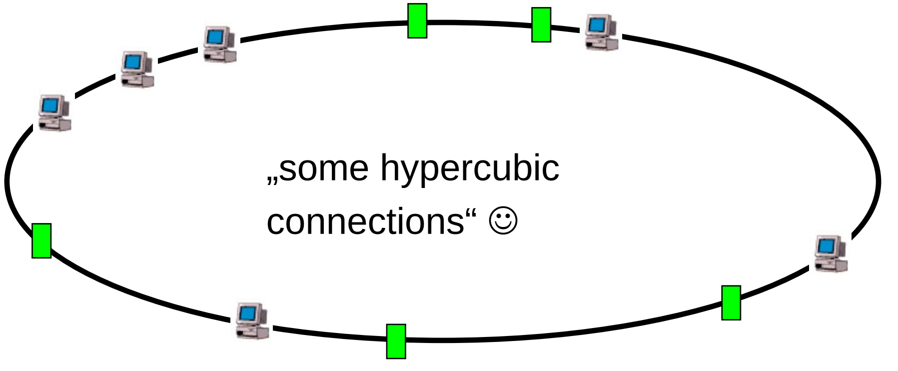
   => defines how peers are connected

   => peer closest to file is responsible for storing (pointer to) data

Distributed Hash Tables (DHTs)

DHTs: decentralized peer-to-peer systems with routing wrt to *keys*

Basic idea:

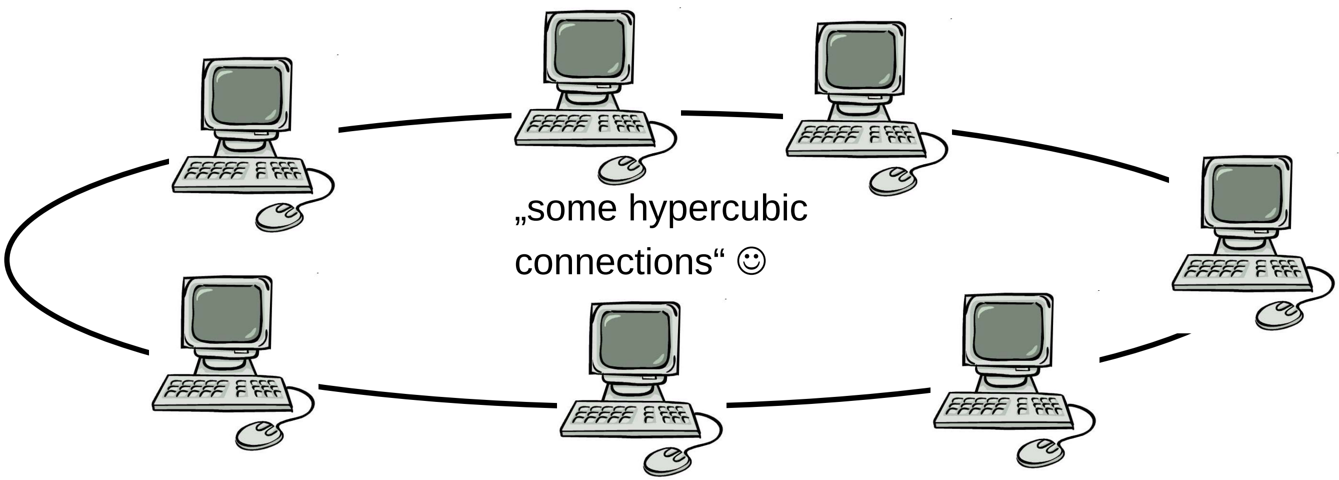


So we have to move all files to the corresponding peers??

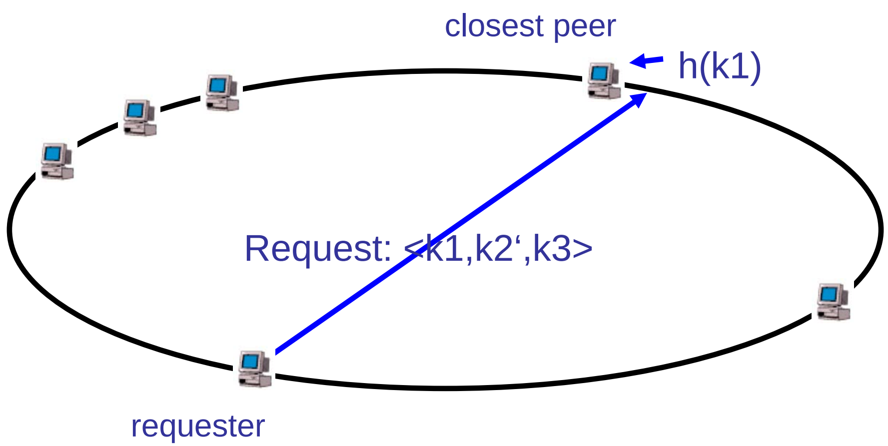
No! Idea: leave files at peers which already store them, and only store pointers to these files in the DHT! (**1st indirection**!)

Kad (Simplified!)

The Kad system: DHT accessed by eMule client

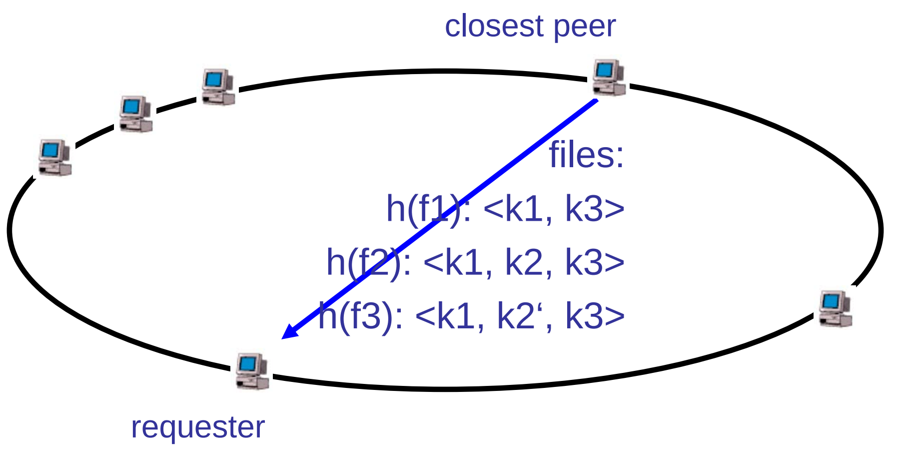


Background: Kad Keyword Request



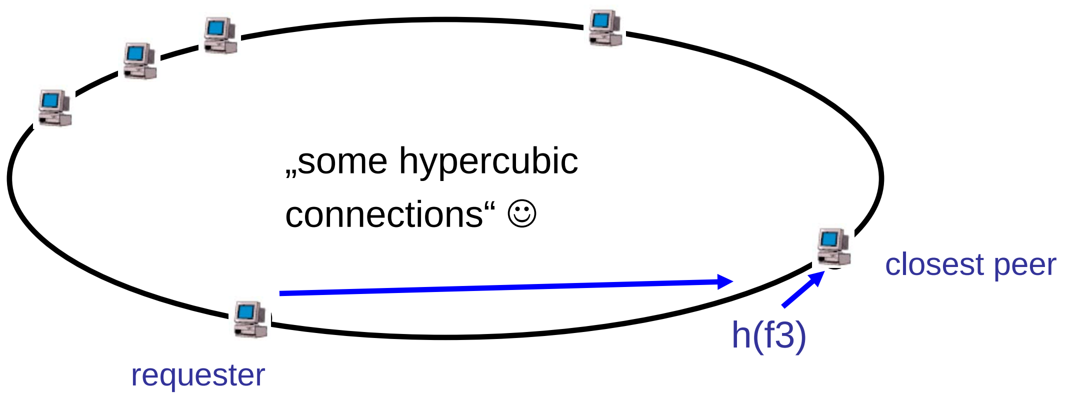
Lookup only with first keyword in list. Key is hash function on this keyword, will be routed to peer with Kad ID closest to this hash value.
(**2nd indirection**!)

Background: Kad Keyword Request



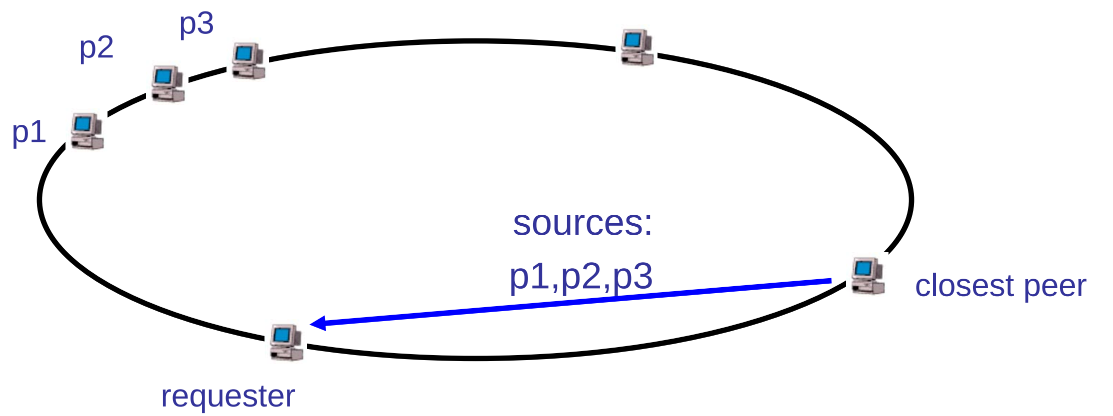
Peer responsible for this keyword returns different sources together with keywords.

Background: Kad Source Request



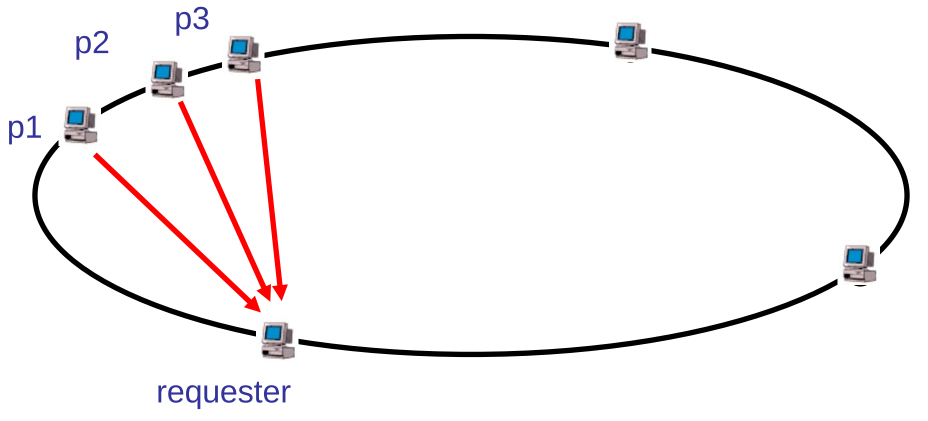
Peer can use this hash to find peer responsible for the file (possibly many with same content / same hash)

Background: Kad Source Request



Peer provides requester with a list of peers storing a copy of the file.

Background: Kad Download



Eventually, the requester can download the data from these peers.

Back to Topologies: Graph Theory

Network topologies are often described as graphs!

Graph G=(V,E): V = set of nodes/peers/..., E= set of edges/links/...

d(.,.): distance between two nodes (shortest path), e.g. d(A,D)=?

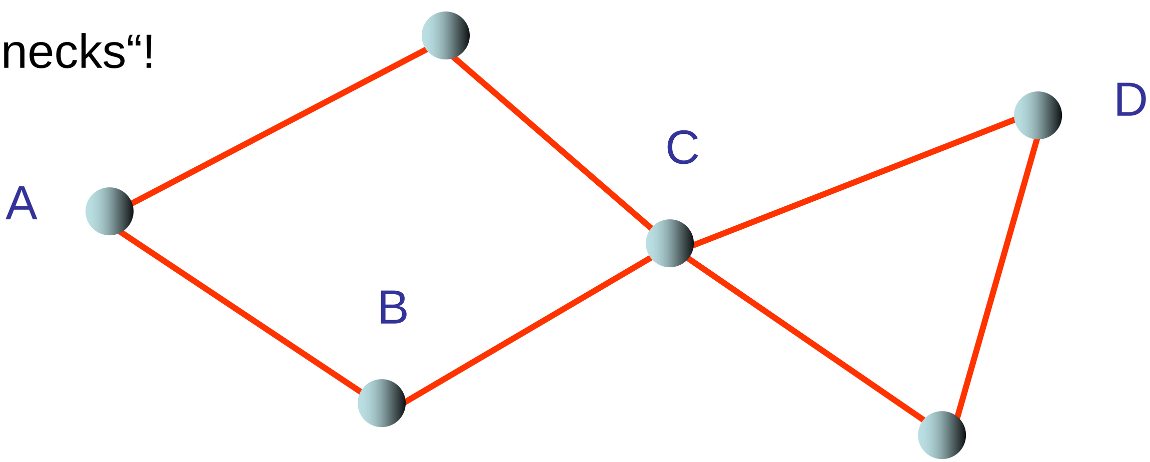
D(G): diameter (D(G)=$\max_{u,v}$ d(u,v)), e.g. D(G)=?

Γ**(U):** neighbor set of nodes U

α**(U)** = $|\Gamma(U)| / |U|$ (size of neighbor set compared to size of U)

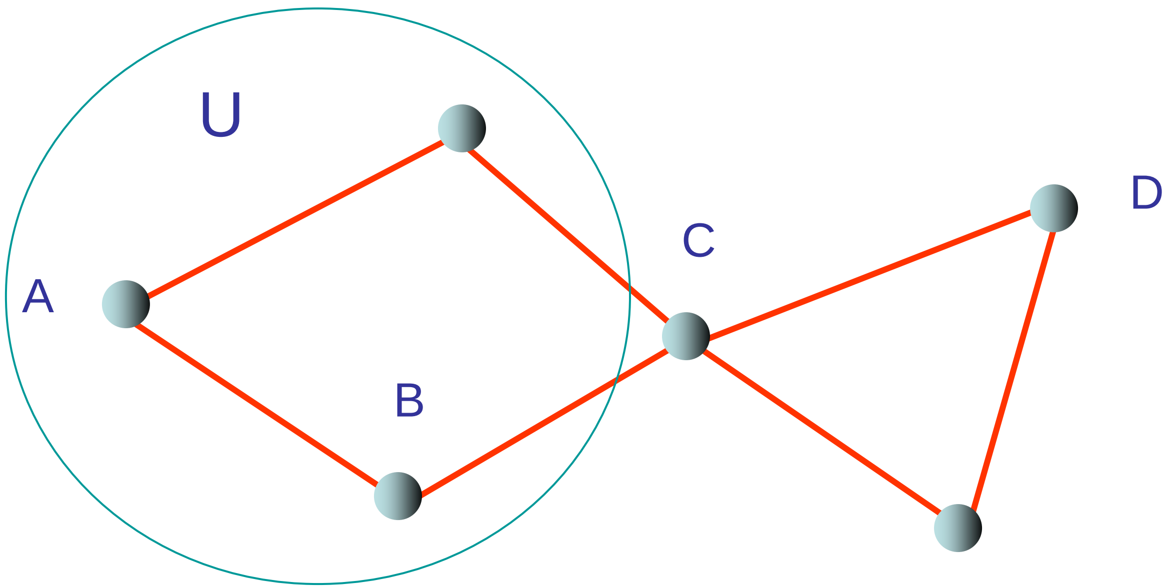
α**(G)** = $\min_{U,\ |U| \leq V/2} \alpha(U)$: expansion of G (meaning?)

Expansion captures „bottlenecks“!



Graph Theory

Explanation: Γ(U), α(U)?

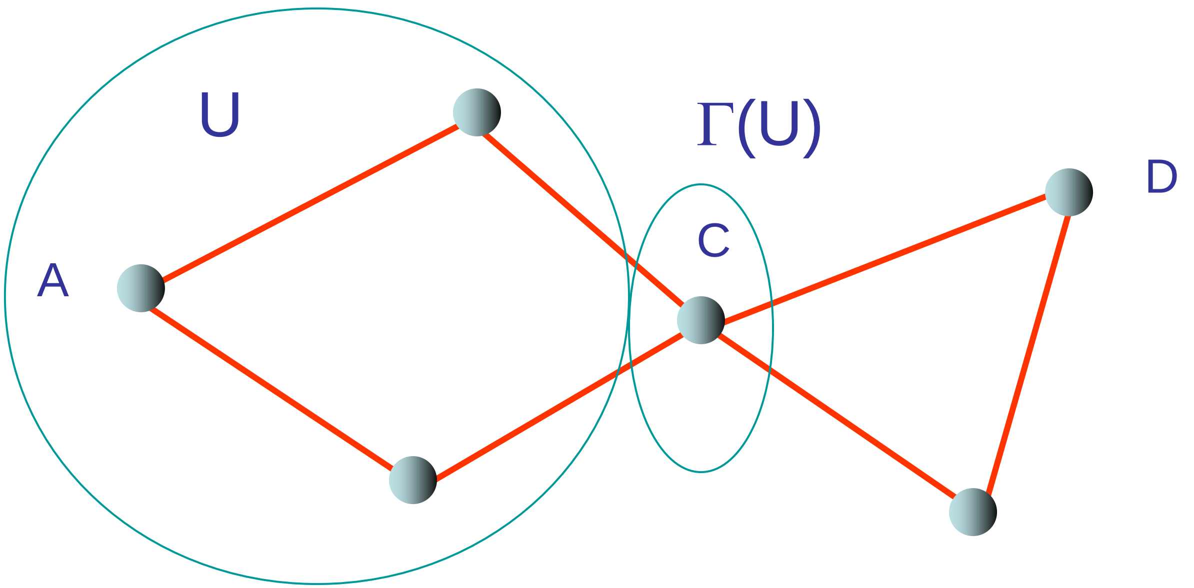


Neighborhood is just C, so...

... α=1/3.

Graph Theory

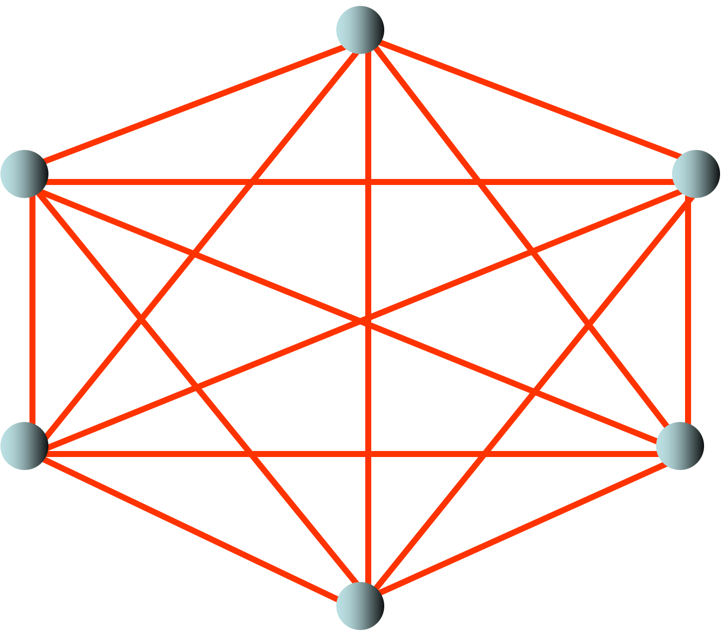
Explanation: Γ(U), α(U)?



α(U)=1/3 (bottleneck!)

What is a good topology?

Complete network: pro and cons?



Pro: robust, easy and fast routing, small diameter...

Cons: does not scale! (degree?, number of edges?, ...)

Good Topologies?

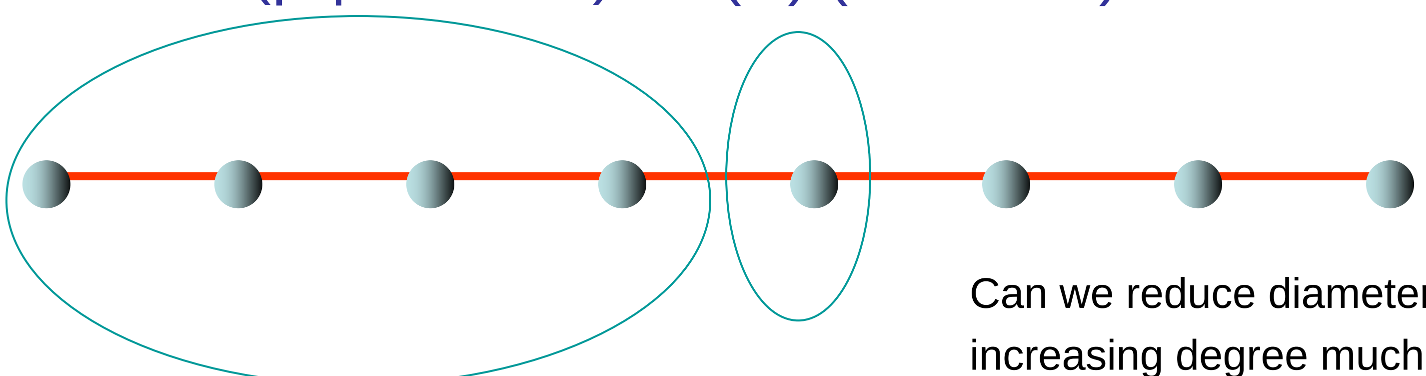
Line network: pro and cons?

Degree? Diameter? Expansion?

Pro: easy and fast routing (tree = unique paths!), small degree (2)...

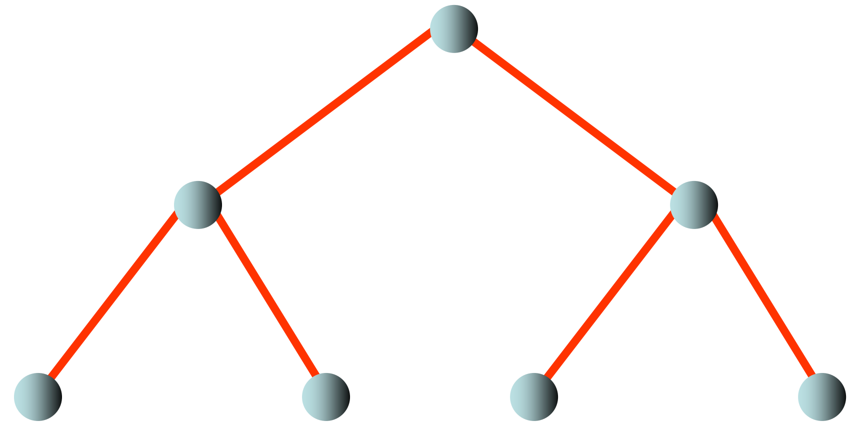
Cons: does not scale! (diameter = n-1, expansion = 2/n, ...)

Expansion: U (|V|/2 nodes) Γ(U) (= 1 node)



Can we reduce diameter without increasing degree much?

Good Topologies?



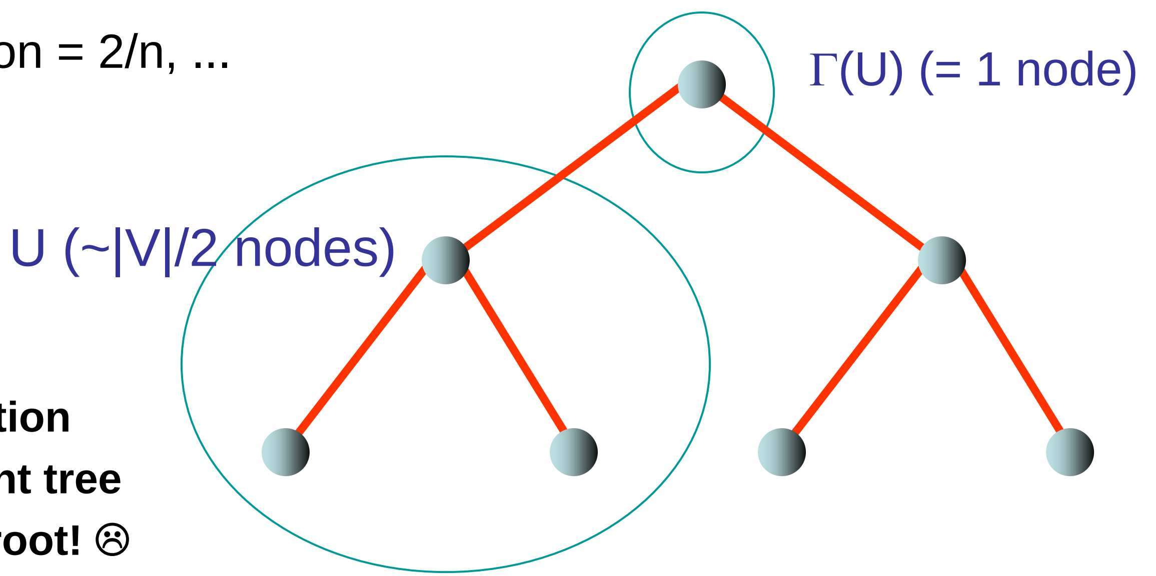
Binary tree network: pro and cons?

Degree? Diameter? Expansion?

Pro: easy and fast routing (tree = unique paths!), small degree (3), log diameter...

Cons: bad expansion = 2/n, ...

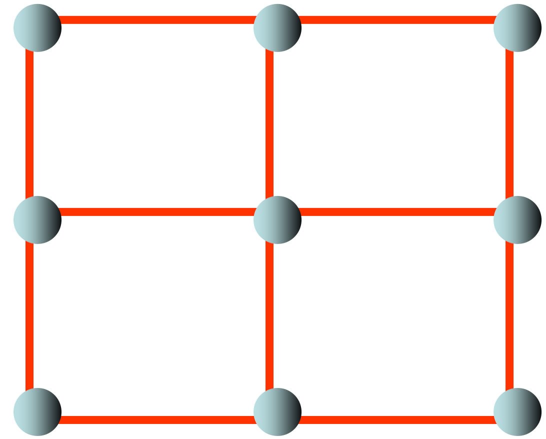
Expansion:



All communication from left to right tree goes through root! ☹

Good Topologies?

2d Mesh: pro and cons?

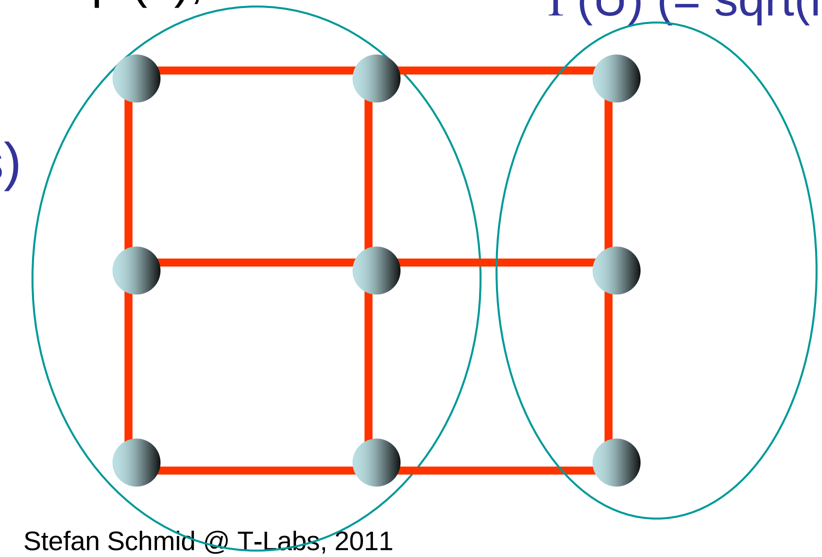


Degree? Diameter? Expansion?

Pro: easy and fast routing (coordinates!), small degree (4), <2 sqrt(n) diameter...

Cons: diameter?, expansion = ~2/sqrt(n), ...

Expansion:



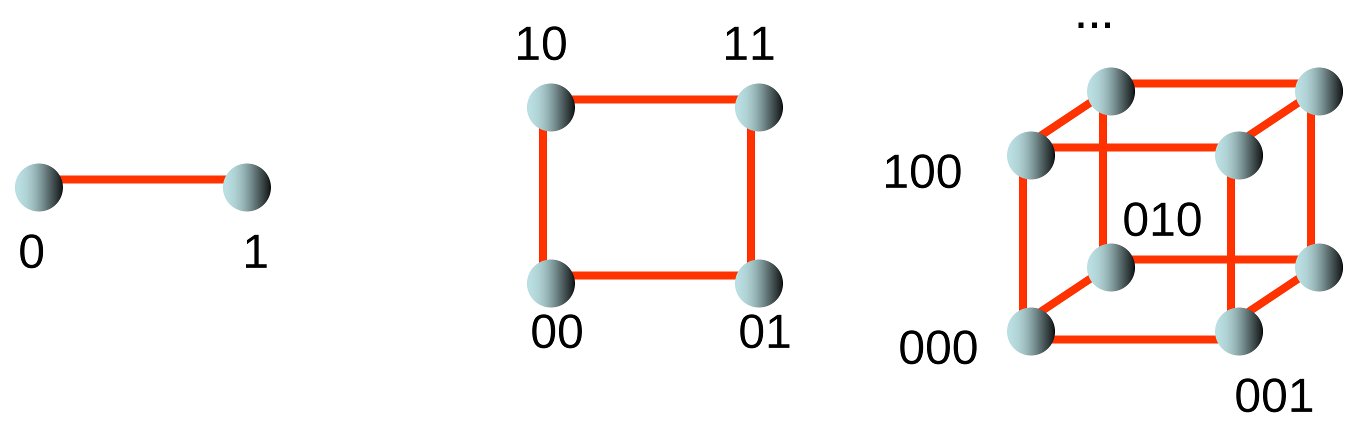
Good Topologies?

d-dim Hypercube: Formalization?

Nodes V = {$(b_1,...,b_d)$, $b \in \{0,1\}$} (nodes are bitstrings!)

Edges E = for all i: $(b_1,..., b_i, ..., b_d)$

connected to $(b_1, ..., 1-b_i, ..., b_d)$



Degree? Diameter? Expansion? How to get from (100101) to (011110)?

2^d = n nodes => d = log(n): degree

Diameter: fix one bit after another => log(n) too

Good Topologies?

d-dim Hypercube:

Nodes $V = \{(b_d,\ldots,b_1),\ b \in \{0,1\}\}$

Edges E = for all i: $(b_d,\ldots,\ b_i,\ \ldots,\ b_1)$

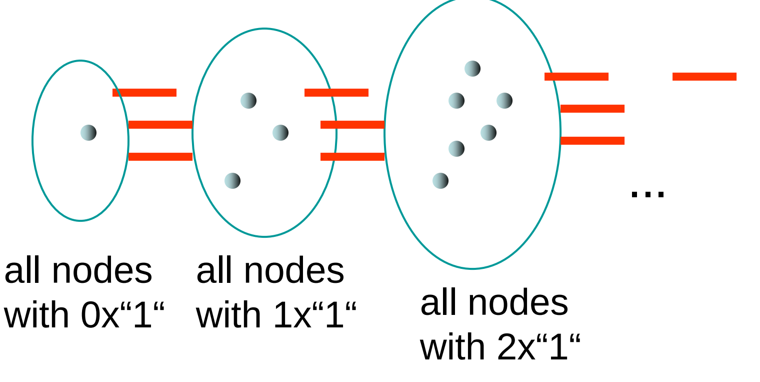
connected to $(b_d,\ \ldots,\ 1-b_i,\ \ldots,\ b_1)$

Expansion? Find small neighborhood!

1/sqrt(d)=1/sqrt(log n)

Idea: nodes with ix“1“ are connected to which nodes?

To nodes with (i-1)x“1“ and (i+1)x“1“...:



Good Topologies?

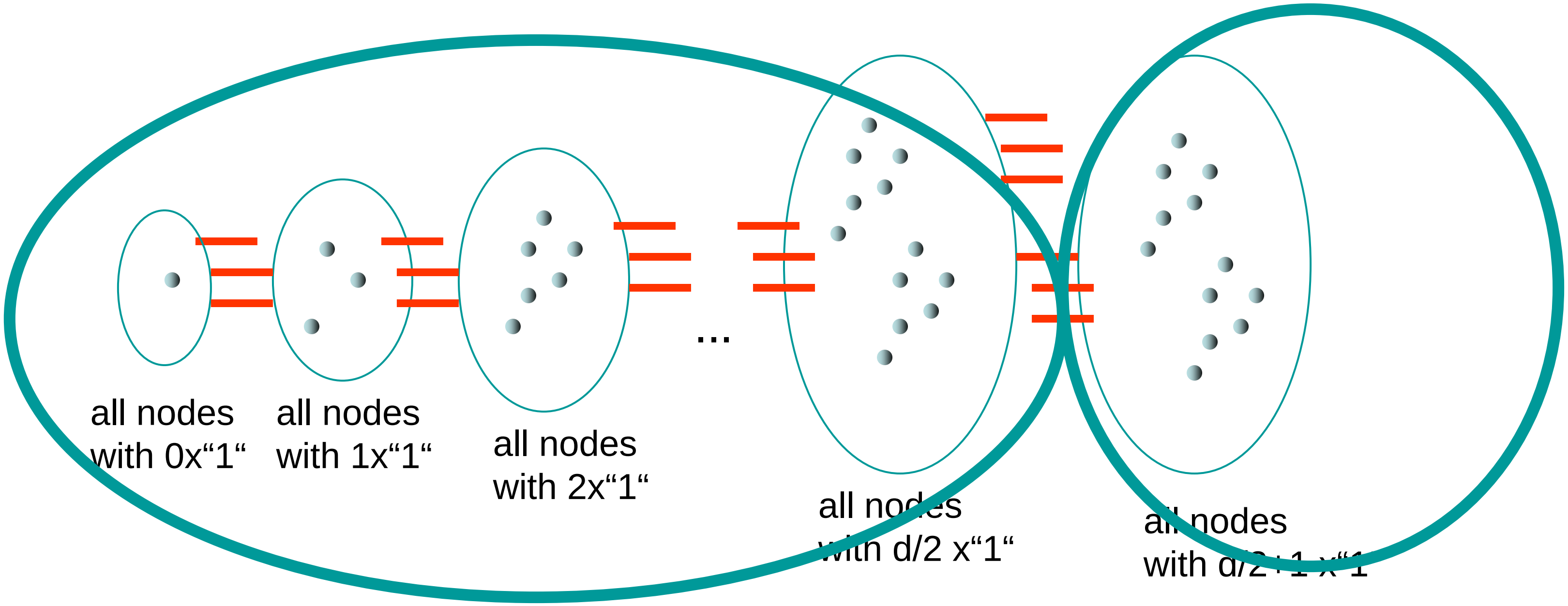
Idea:

How many nodes?

U (~n/2 nodes)

Γ(U) (= ?)

= binomial(d,d/2+1)



Expansion then follows from computing the ratio...

Many networks are hypercubic!

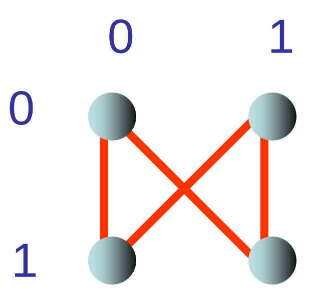
Butterfly graph: (known? e.g., for parallel architectures)

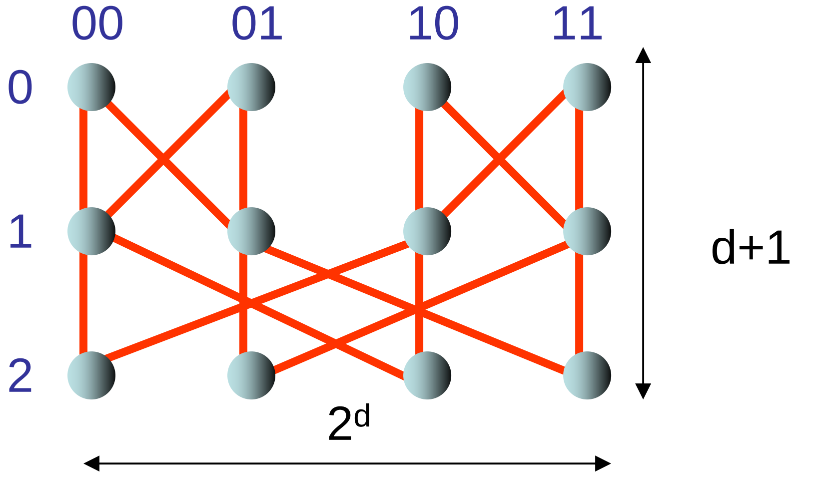
Nodes V = $\{(k, b_1...b_d) \in \{0,...,d\} \times \{0,1\}^d\}$ (2-dimensional: „number + bitstring“)

Edges E = for all i: $(k\text{-}1, b_1...b_k...b_d)$

connected to $(k, b_1...b_k...b_d)$ and $(k, b_1...1\text{-}b_k...b_d)$

Essentially a rolled-out hypercube! Diam, Deg, Exp? How many nodes in total?





Degree 4, Diameter 2d (e.g., go to corresponding „bottom“, then up)

Many networks are hypercubic!

Butterfly graph:

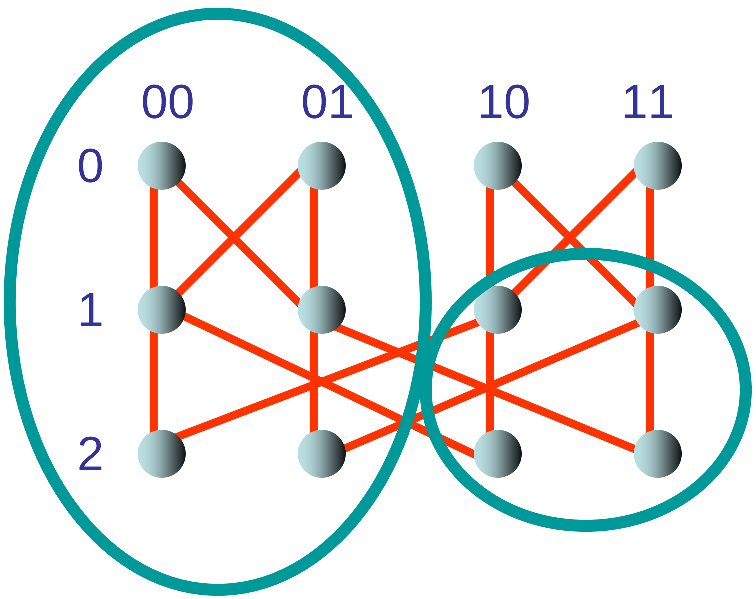
Nodes V = $\{(k, b_1 \ldots b_d) \in \{0,\ldots,d\} \times \{0,1\}^d\}$

Edges E = for all i: $(k-1, b_1 \ldots b_k \ldots b_d)$

connected to $(k, b_1 \ldots b_k \ldots b_d)$ and $(k, b_1 \ldots 1-b_k \ldots b_d)$

Expansion:

U (~n/2 nodes)



Γ(U) (= ?)

~ n/d

Expansion roughly 1/d.

Many networks are hypercubic!

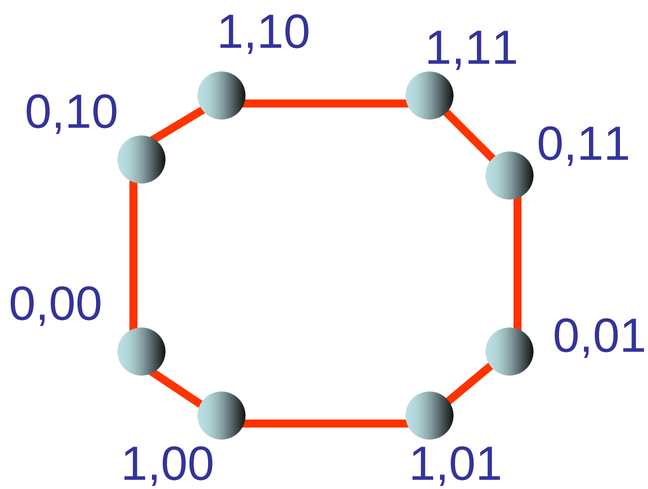
Cube-Connected Cycles: Hypercube with „replaced corners“

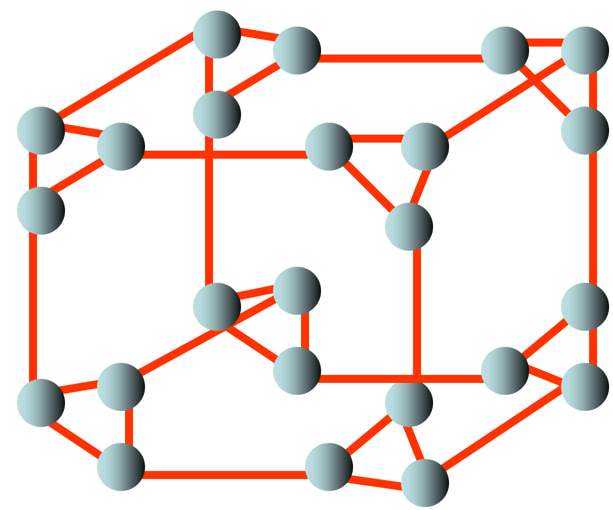
Nodes $V = \{(k, b_1 \ldots b_d) \in \{0,\ldots,d-1\} \times \{0,1\}^d\}$

Edges E = for all i: $(k-1, b_1 \ldots b_k \ldots b_d)$

connected to $(k-1, b_1 \ldots b_k \ldots b_d)$, $(k+1, b_1 \ldots b_k \ldots b_d)$ and $(k, b_1 \ldots 1-b_k \ldots b_d)$

Example:





Many networks are hypercubic!

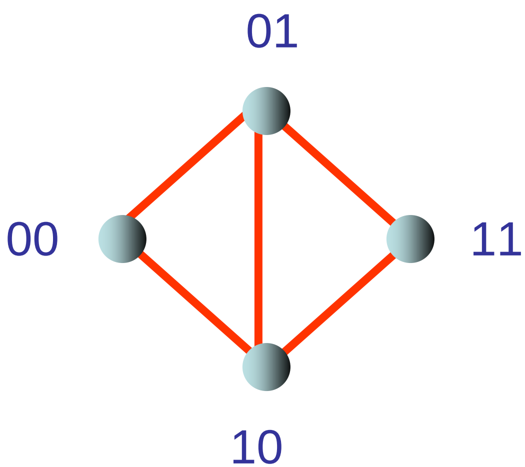
De Bruijn Graph:

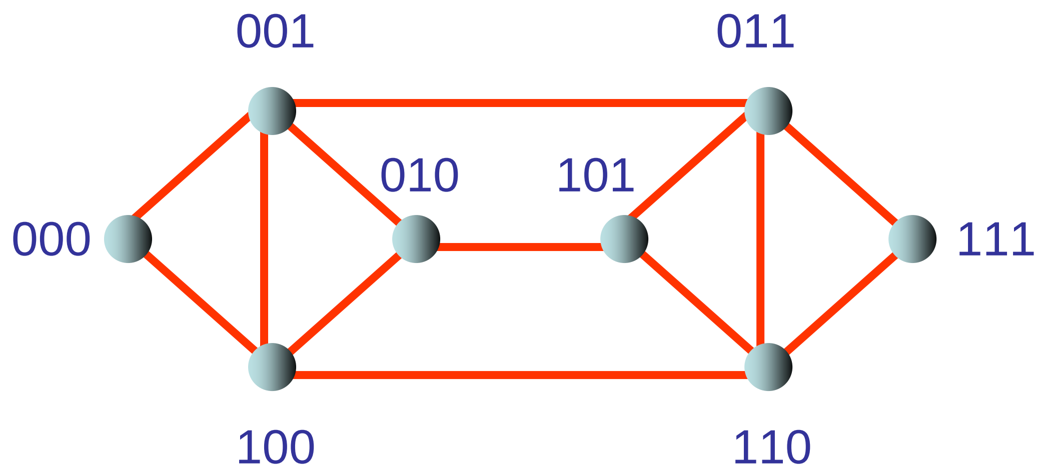
Nodes V = $\{(b_1 \ldots b_d) \in \{0,1\}^d\}$ (bitstrings...)

Edges E = for all i: $(b_1 \ldots b_k \ldots b_d)$

connected to $(b_2 \ldots b_d 0)$ and $(b_2 \ldots b_d 1)$ („shift left and add 0 and 1“)

Example:





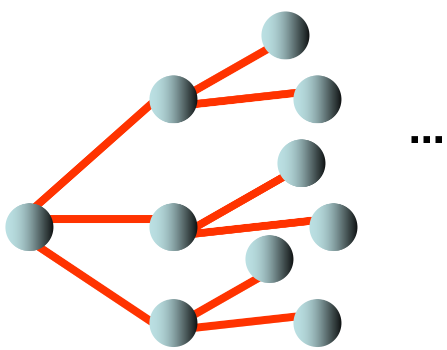
How to route on this topology?

Fill in bits from the back...

What is the degree-diameter tradeoff? Idea? Proof?

Theorem

Each network with n nodes and max degree d>2 must have a diameter of at least log(n)/log(d-1)-1.



1 d d-1 ...

In two steps, at most

d (d-1)

additional nodes can be reached!

So in *k* steps at most:

$$1+\sum_{i=0}^{k-1} d\cdot(d-1)^i = 1 + d\cdot\frac{(d-1)^k-1}{(d-1)-1} \le \frac{d\cdot(d-1)^k}{d-2}$$

To ensure it is connected this must be at least n, so:

$$(d-1)^k \ge \frac{(d-2)\cdot n}{d} \quad\Leftrightarrow\quad k\ge \log_{d-1}\left(\frac{(d-2)\cdot n}{d}\right) \quad\Leftrightarrow\quad k \ge \log_{d-1} n + \log_{d-1}\left(\frac{d-2}{d}\right)$$

Reformulating yields the claim... ☺

Example: Pancake Graphs

Graph which minimizes max(degree, diameter)?

Solution: Pancake graph gives log n / log log n

Example: d-dim Pancake graph

Nodes = permutations of {1,...,d}

Edges = prefix reversals

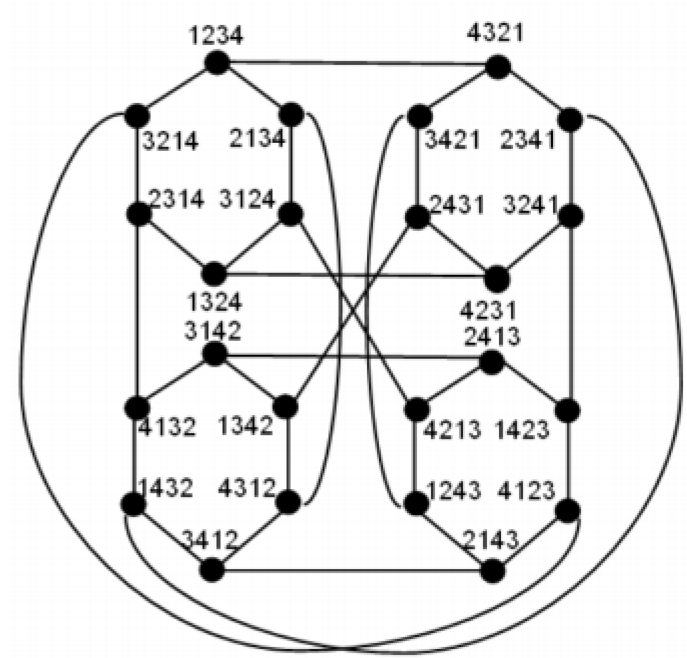
nodes? degree?

d! many nodes and degree (d-1).

Routing?

E.g., from (3412) to (1243)?

Fix bits at the back, one after the other, in two steps, so diameter also log n / log log n.



So we know:

hypercube graphs, de Bruijn graphs, ...

But how but if number of nodes/peers is not a power of two or so?

And how to join and leave a network without much disruptions and „local state changes“ / few messages?

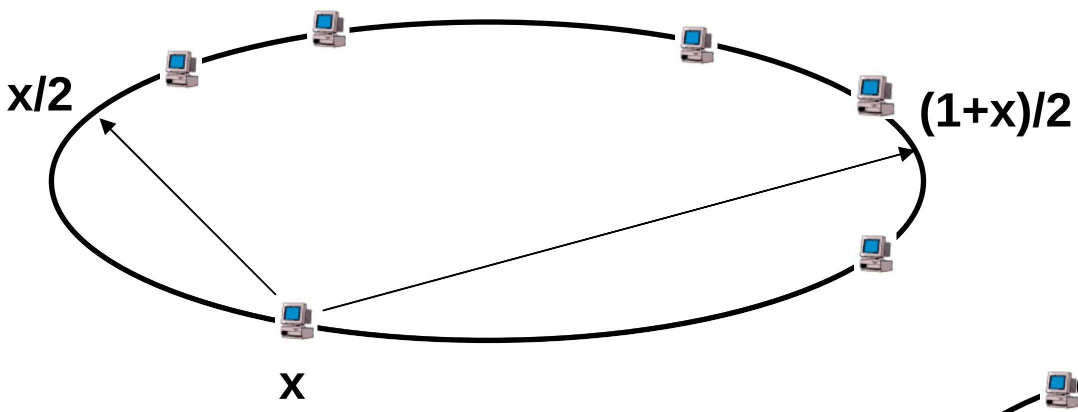
We sketch to ideas...:

1. Continuous-discrete approach
2. Graph simulation

Continuous-Discrete Approach (Naor & Wieder)

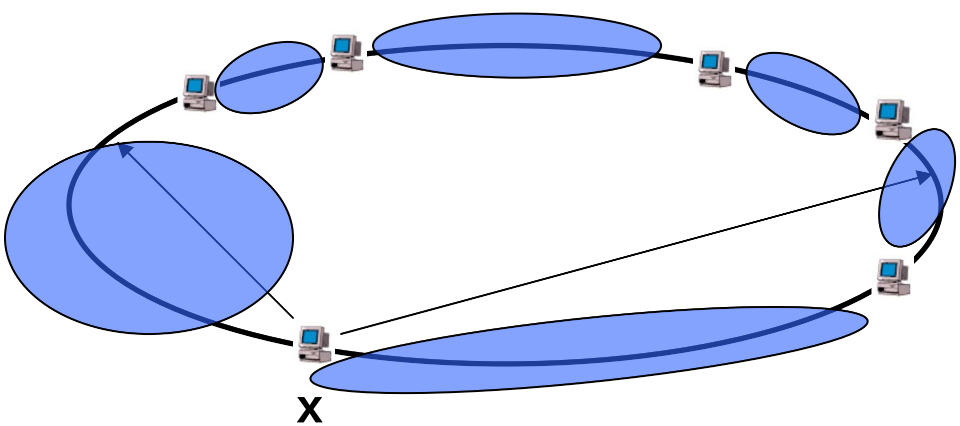
Idea:

1. Map peers to a virtual ring [0,1), at uniform random positions
2. Define „continuous graph“: to which „points“ should nodes connect
   (and find routing algorithms on continous graph etc.)
3. „Discretize graph“: nodes are responsible for the links in their neighborhood (routing adapted easily)



Continuous graph: e.g., node at position x connects to points x/2 and (1+x)/2

Discrete graph: responsibility zones...



It turns out: for x/2 and (1+x)/2 we get a de Bruijn graph! And we can build also hypercubes etc.! ☺

Other idea: Simulate the desired topology!

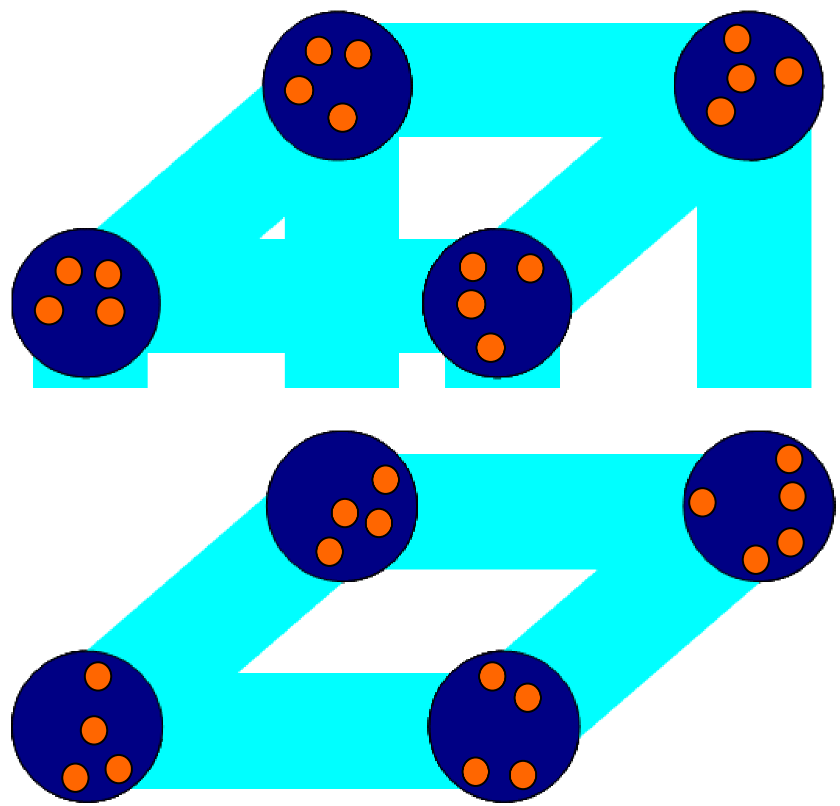
1. Take a graph with desirable properties
2. Simulate the graph by representing each vertex by a set of peers
3. Find a token distribution algorithm on this graph to balance peers
4. Find an algorithm to estimate the total number of peers in the system
5. Find an algorithm to adapt the graph‘s dimension

Example: Hypercube

How to connect peers

- in vertex?
- between vertices?

How many joins and leaves
per time unit can be tolerated?



Further reading:

Theory of

Distributed Computing I

(Part 2: Message Passing)

Dr. Stefan Schmid

Co-lecturer (shared memory): Dr. Petr Kuznetsov

Thanks to Prof. Dr. Roger Wattenhofer for basis of manuscript!

Spring 2011

Novel Architectures for P2P Applications: the Continuous-Discrete Approach*†

MONI NAOR‡
The Weizmann Institute of Science
moni.naor@weizmann.ac.il

UDI WIEDER§
Microsoft Research
uwieder@microsoft.com

October 4, 2006

Abstract

We propose a new approach for constructing P2P networks based on a dynamic decomposition of a continuous space into cells corresponding to servers. We demonstrate the power of this approach by suggesting two new P2P architectures and various algorithms for them. The first serves as a DHT (Distributed Hash Table) and the other is a dynamic expander network. The DHT network, which we call Distance Halving, allows logarithmic routing and load, while preserving constant degrees. It offers an optimal tradeoff between the degree and the path length in the sense that degree d guarantees a path length of $O(\log_d n)$. Another advantage over previous constructions is its relative simplicity. A major new contribution of this construction is a dynamic caching technique that maintains low load and storage even under the occurrence of hot spots. Our second construction builds a network that is guaranteed to be an expander. The resulting topologies are simple to maintain and implement. Their simplicity makes it easy to modify and add protocols. A small variation yields a DHT which is robust against random Byzantine faults. Finally we show that, using our approach, it is possible to construct *any* family of constant degree graphs in a dynamic environment, though with worse parameters. Therefore we expect that more distributed data structures could be designed and implemented in a dynamic environment.

End of lecture