Foundations of Distributed Systems:

Topology

(with an excursion to P2P)
Three administrative comments...

1. There will be a „Skript“ for this part of the lecture. (More formal details compared to slides... 😊 )
Will be updated weekly.

2. Course follows the cool book by Peleg (but only first, simple chapters are covered).
Further reading, e.g., «Networks» by Newman.

3. Exam: How about on Monday, August 6, 2012?
Please register for the exam until July 1, 2012!
(Will work via QISPOS.)
Communication over networks!

- no shared memory
- focus on message or communication (bit-) complexity
- goal: compute global task in a decentralized and local manner: only few nodes (neighbors) are involved
- hope: fast reaction and convergence, robust, ...

- Problem / design space
  - sometimes the topology is given (e.g., social network or Gnutella) sometimes it can be designed (e.g., smart grid network, overlay p2p network),
  - sometimes nodes are more heterogenous (e.g., in open peer-to-peer systems) sometimes less (e.g., in datacenters, parallel architectures),
  - sometimes communication occurs along wires sometimes it is wireless (broadcast),
  - ...

Stefan Schmid @ T-Labs Berlin, 2012
Networks

Social Networks.

Internet.

Nervous system.

Google+ users.

Smart grid.
Shared Memory vs Message Passing?

Similarities to first part of lecture: models (and results) can sometimes be transformed!

SHARED-MEMORY COMPUTER with 4 PUs

Send...

... receive...

... compute.
What you will learn!

Topology: Which (communication) networks are good?

The basics, e.g., leader election

Classical TCS reloaded: Maximal spanning tree, maximal independent sets, graph colorings computed distributedly?

Distributed lower bounds: what is impossible?

maybe: social networks
Good Topologies?

Topology („network graph“)

- sometimes given (e.g., social networks)
- sometimes semi-structured (e.g., unstructured peer-to-peer networks with heterogeneous clients and join protocols)
- sometimes subject to design and optimization (e.g., parallel computer architectures, structured peer-to-peer networks, etc.)

Which topologies do you know?
What is a „good topology“?!
What is a „good topology“? It depends...

- How to interconnect the cities of a country with an efficient railroad or telecommunication infrastructure? (Expensive?)
- How to interconnect components of a parallel computer? (Space?)
- How to interconnect peers of a peer-to-peer system? (Latency?)
- Or even: how to control the „topology“ of a wireless network?! (How to control a wireless topology? Transmission power, choose subset of links for routing, ...)

Possible 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
bisection bandwidth / cutwidth, 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?

It depends... Generally, the 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 packets. (Which ones such that connectivity is preserved but as short links as possible?)

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)

Note: In wireless networks, routing over many short hops may be more efficient than routing over few long ones, as the required energy grows at least quadratically with distance.
**Napster:**
centralized, „no topology“

**Gnutella:**
fully decentralized, „arbitrary topology“

**DHT:**
„structured“, often hypercubic topology (why?)
Napster: Centralized index
Napster
Napster

Aphex Twin: Ptolemy @ 212.17.11.69

Beach Boys: Pet Sounds @ 170.13.01.02
Napster
Napster

"Aphex Twin: Ptolemy"?

<Beach Boys: Pet Sounds @ 170.13.01.02>

<Aphex Twin: Ptolemy @ 212.17.11.69>
Napster

@ 212.17.11.69!

@ 170.13.01.02

<Beach Boys: Pet Sounds @ 170.13.01.02>

<Aphex Twin: Ptolemy @ 212.17.11.69>

Stefan Schmid @ T-Labs Berlin, 2012
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...
Lookup: flooding.
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* (simple 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
DHTs: decentralized peer-to-peer systems with routing wrt to keys

Basic idea: virtual ring

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!)
The Kad system: DHT accessed by eMule client
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

files:

h(f1): <k1, k3>
h(f2): <k1, k2, k3>
h(f3): <k1, k2', k3>

Requester

Closest peer

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)
Peer provides requester with a list of peers storing a copy of the file.
Eventually, the requester can download the data from these peers.
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. \( D(G)=? \)

- \( \Gamma(U) \): neighbor set of nodes \( U \) (not including nodes in \( U \))
  - \( \alpha(U) = |\Gamma(U)| / |U| \) (size of neighbor set compared to size of \( U \))
  - \( \alpha(G) = \min_{U, |U| \leq v/2} \alpha(U) \): *expansion* of \( G \) (meaning?)

Expansion captures „bottlenecks“!
**Explanation:** $\Gamma(U), \alpha(U)$?

Neighborhood is just C, so...

... $\alpha = 1/3$. 

Stefan Schmid @ T-Labs Berlin, 2012
Graph Theory

**Explanation:** $\Gamma(U), \alpha(U)$?

\[ \alpha(U) = 1/3 \text{ (bottleneck!)} \]
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?, ...)

Stefan Schmid @ T-Labs Berlin, 2012
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: \( U (|V|/2 \text{ nodes}) \quad \Gamma(U) (= 1 \text{ node}) \)

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:

$U \sim |V|/2$ nodes

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

Stefan Schmid @ T-Labs Berlin, 2012
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:

U (~n/2 nodes)

Γ(U) (= sqrt(n) nodes)
Good Topologies?

**d-dim Hypercube**: Formalization?

Nodes \( V = \{(b_1, \ldots, b_d), b_i \vdash \text{binary}\} \) (nodes are bitstrings!)

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

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

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

\( 2^d = n \) nodes \( \Rightarrow d = \log(n): \) degree

Diameter: fix one bit after another \( \Rightarrow \log(n) \) too
**Good Topologies?**

**d-dim Hypercube:**
Nodes $V = \{(b_d, \ldots, b_1), b \in \{0, 1\}\}$
Edges $E = \text{for all } i: (b_d, \ldots, b_i, \ldots, b_1)$
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 $i \times 1$ are connected to which nodes?
To nodes with $(i-1) \times 1$ and $(i+1) \times 1$...:
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!

Many computer networks are variants or generalizations of hypercubes!

E.g., *peer-to-peer systems* (Chord, Pastry, Kademlia, ...)

E.g., *datacenter* topologies (container-based datacenters, BCube, MDCCube, ...)

E.g., *parallel architectures* (butterfly variants, etc.)
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 $E = \text{for all } i: (k-1, b_0...b_k...b_d)$
connected to $(k, b_1...b_k...b_d)$ and $(k, b_1...1-b_k...b_d)$
(i.e., to nodes on next level with same and opposite bit at only this position)

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

$d=1$: 

```
0 0 1
1
```

$d=2$: 

```
00
01
10
11
```

Degree 4, Diameter 2d (e.g., go to corresponding „bottom“, then up)
Many networks are hypercubic!

**Butterfly graph:**
Nodes $V = \{(k, b_1...b_d) \in \{0,...,d\} \times \{0,1\}^d\}$

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

Expansion:

$U (~n/2 \text{ nodes})$  
$\Gamma(U) (\text{only at low dimension}) \sim n/d$

Expansion roughly $1/d$.  

Stefan Schmid @ T-Labs Berlin, 2012
Many networks are hypercubic!

**Cube-Connected Cycles**: Hypercube with „replaced corners“

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

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

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

Example:
Many networks are hypercubic!

**De Bruijn Graph:**

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

Edges $E =$ for all $i$: $(b_1...b_k...b_d)$ connected to $(b_2...b_d0)$ and $(b_2...b_d1)$ ("shift left and add 0 and 1")

Example (undirected version):

How to route on this topology?
Fill in bits from the back...

Stefan Schmid @ T-Labs Berlin, 2012 49
Theorem

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

In two steps, at most

\[
d (d-1)
\]

additional nodes can be reached!

So in \( k \) steps at most:

\[
1 + \sum_{i=0}^{k-1} d \cdot (d - 1)^i = 1 + d \cdot \frac{(d - 1)^k - 1}{(d - 1) - 1} \leq \frac{d \cdot (d - 1)^k}{d - 2}
\]

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

\[
(d - 1)^k \geq \frac{(d - 2) \cdot n}{d} \iff k \geq \log_{d-1} \left( \frac{(d - 2) \cdot n}{d} \right) \iff k \geq \log_{d-1} n + \log_{d-1} \left( \frac{d - 2}{d} \right)
\]

Reformulating this yields the claim... 😊
Example: Pancake Graphs

Graph which minimizes \( \max(\text{degree, diameter}) \)?
Solution: Pancake graph gives \( \log n / \log \log n \)

Example: \( d \)-dim Pancake graph

Nodes = permutations of \{1,\ldots,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, ...

What 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 \textit{routing algorithms} on continuous graph etc.)
3. „Discretize graph“: nodes are responsible for the links in their neighborhood (routing adapted easily)

\begin{itemize}
  \item \textbf{Continuous graph:} e.g., node at position \(x\) connects to points \(x/2\) and \((1+x)/2\)
  \item \textbf{Discrete graph:} responsibility zones...
\end{itemize}

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:

Foundations of Distributed Systems
(Part 2: Message Passing)

Dr. Stefan Schmid
Co-lecturer (shared memory): Dr. Petr Kaznetsov
Thanks to Prof. Dr. Roger Wattenhofer for basis of manuscript!

Spring 2012

Novel Architectures for P2P Applications: the Continuous-Discrete Approach

MOMI 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 Hashing, 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 $O(d \log n)$. 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.

Towards worst-case churn resistant peer-to-peer systems

Fabian Kuhn - Stefan Schmid - Roger Wattenhofer

Received: 24 March 2009 / Accepted: 25 February 2010
© Springer-Verlag 2010

Abstract

Until now, the analysis of fault tolerance of peer-to-peer systems usually only covers random faults of some kind. Contrary to traditional algorithmic research, faults as well as joins and leaves occurring in a worst-case manner are hardly considered. In this article, we devise techniques to build dynamic peer-to-peer systems which remain fully functional in spite of an adversary who continuously adds and removes peers. We exemplify our algorithms on hypercube and pancake topologies and present a system which maintains small peer degree and network diameter.

Keywords
Churn - Dynamic networks - Fault-tolerance - Overlay network - Peer-to-peer

overlay networks, are becoming more and more wide spread. A major complication in these networks is that they can be highly dynamic, which requires fast and robust recovery mechanisms.

Today, the analysis of fault tolerance of p2p systems usually only covers random faults of some kind. Contrary to traditional algorithmic research, faults as well as joins and leaves occurring in a worst-case manner in p2p systems are hardly considered. Moreover, most fault tolerance analyses are static in the sense that only a functionally bounded number of random peers can be crashed. After removing a few peers the system is given sufficient time to recover again. The much more realistic dynamic case where faults steadily