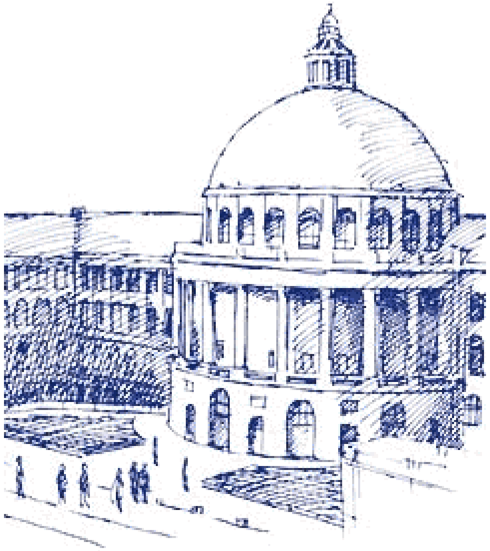
Dynamic Hypercube Topology

Stefan Schmid

URAW 2005

Upper Rhine Algorithms Workshop

University of Tübingen, Germany

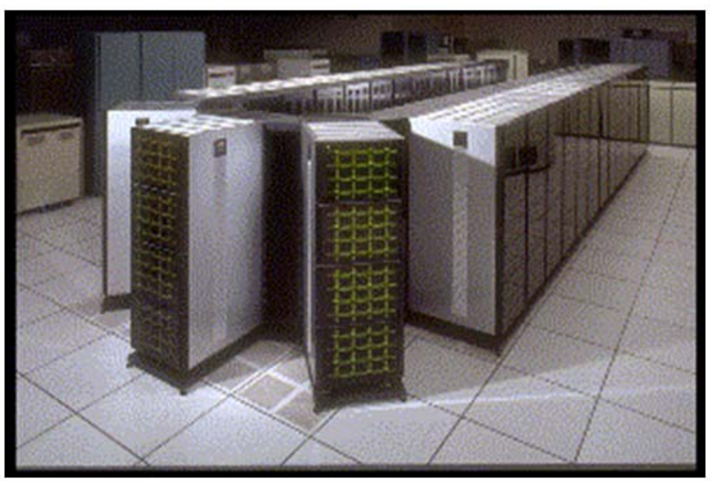


ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

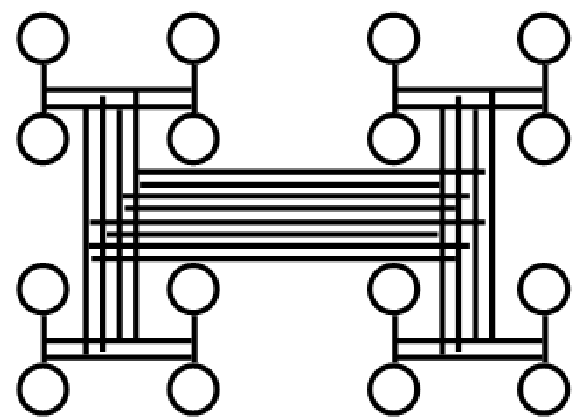
Static vs. Dynamic Networks (1)

- Network graph *G=(V,E)*
  - *V* = set of vertices ("nodes", machines, peers, …)
  - *E* = set of edges ("connections", wires, links, pointers, …)

- "Traditional", static networks
  - Fixed set of vertices, fixed set of edges
  - E.g., interconnection network of parallel computers



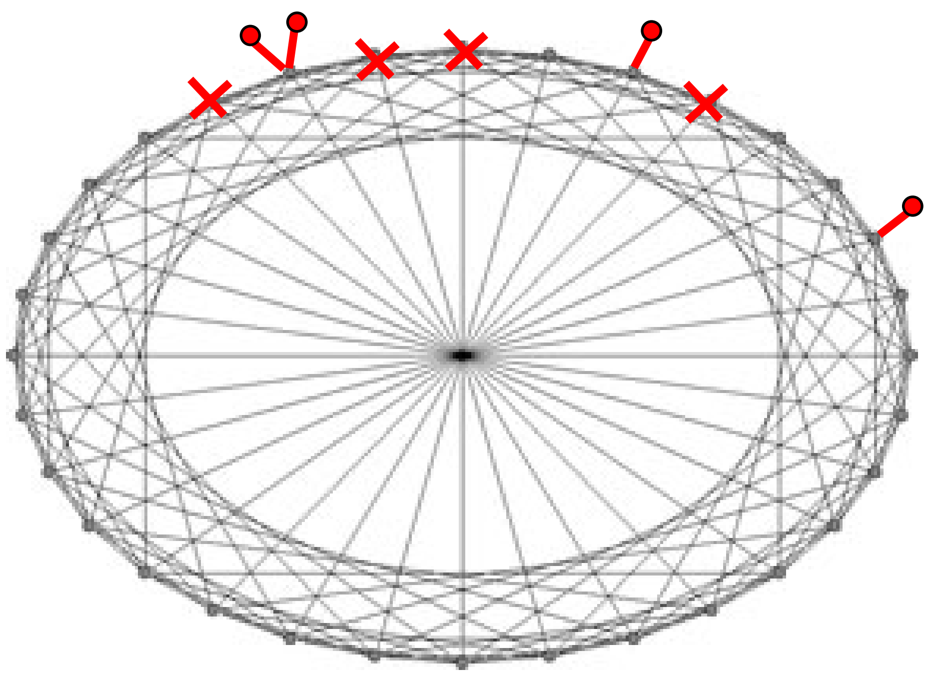
Parallel Computer



Fat Tree Topology

Static vs. Dynamic Networks (2)

- Dynamic networks
  - Set of nodes and/or set of edges is dynamic
  - Here: nodes may join and leave
  - E.g., peer-to-peer (P2P) systems (Napster, Gnutella, …)

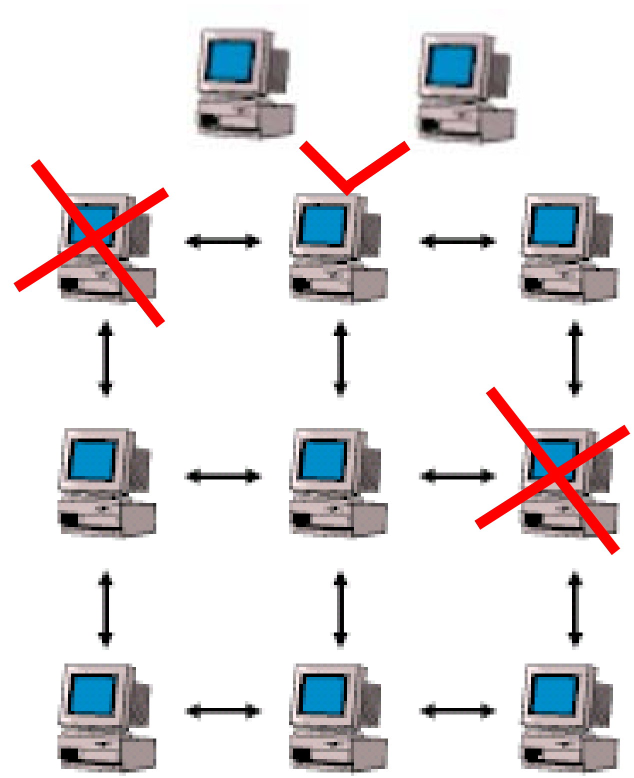


Dynamic Chord Topology

Dynamic Peer-to-Peer Systems

Peer-to-Peer Systems

- cooperation of many machines (to share files, CPU cycles, etc.)
- usually desktop computers under control of individual users
- user may turn machine on and off at any time
- => Churn



How to maintain desirable properties such as connectivity, network diameter, node degree, ...?

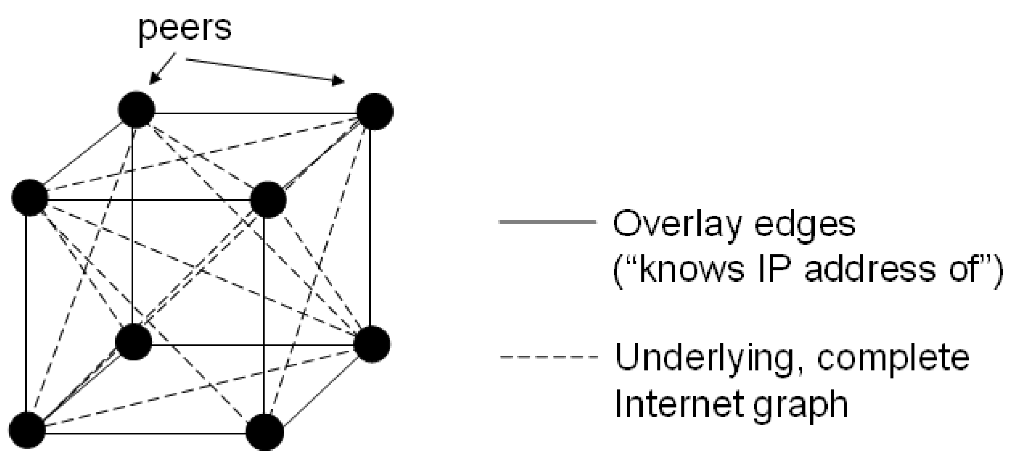
Talk Overview

- Model
- Ingredients: basic algorithms on hypercube graph
- Assembling the components
- Results for the hypercube
- Conclusion, generalization and open problems
- Discussion

Model (1): Network Model

- Typical P2P overlay network
  - Vertices $v \in V$: peers (dynamic: may join and leave)
  - Directed edges $(u,v) \in E$: u knows IP address of v (static)

- Assumption: Overlay network builds upon *complete* Internet graph
  - Sending a message over an overlay edge => routing in the underlying Internet graph

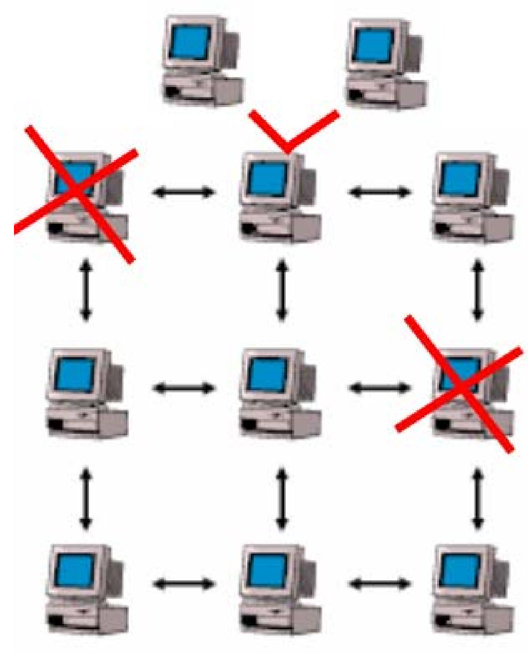


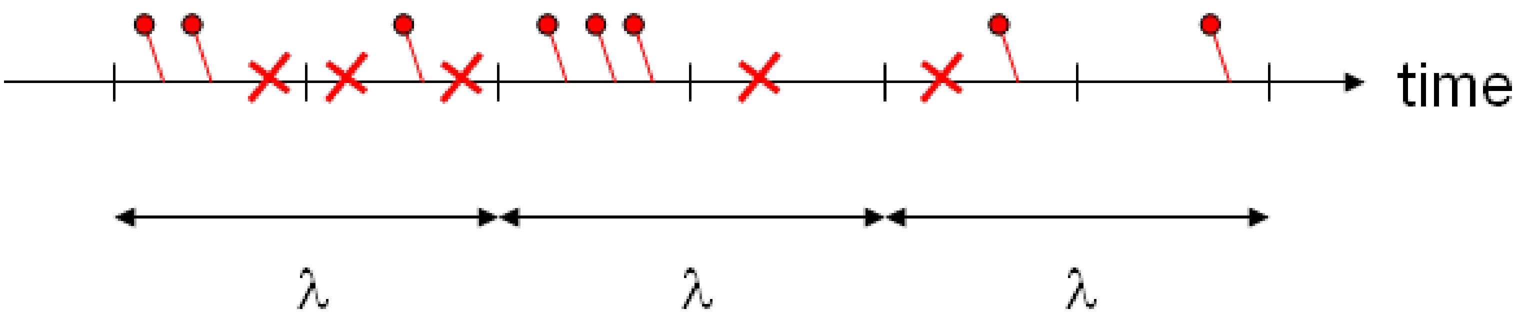
Model (2): Worst-Case (Adversarial) Dynamics

- 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



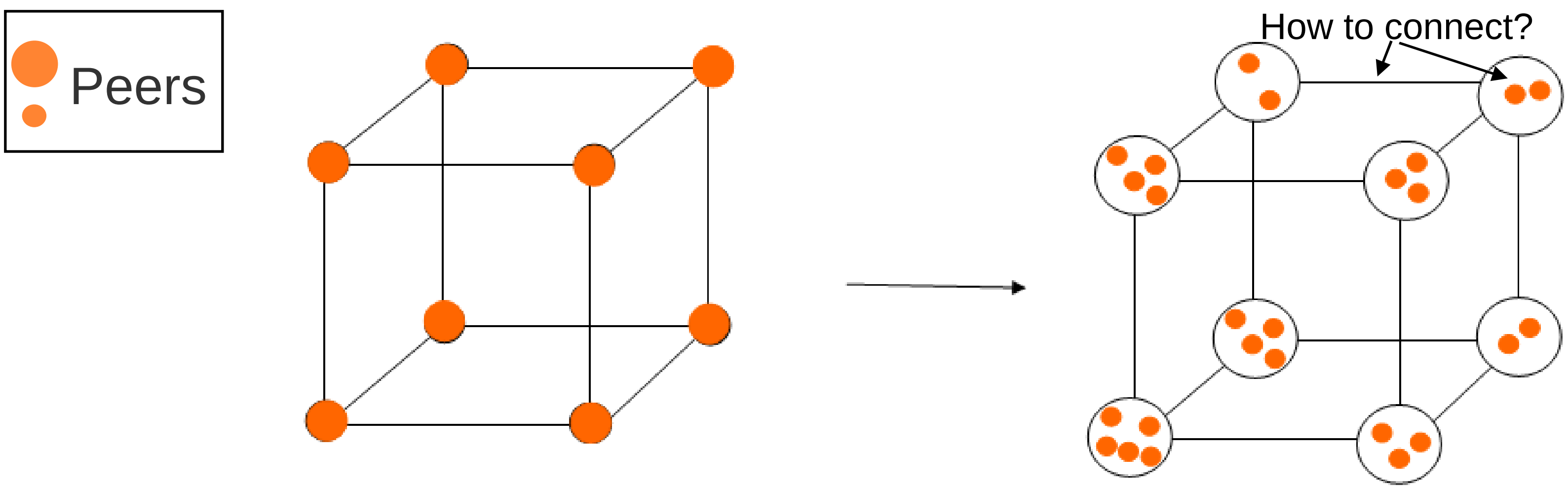


Model (3): Communication Rounds

- 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

Overview of Dynamic Hypercube System

- Idea: Arrange peers into a *simulated* hypercube where each node consists of *several* (logarithmically many) peers!
  - Gives a certain redundancy and thus time to react to changes.
  - But still guarantees diameter *D = O(log n)* and degree *Δ = O(log n)*, as in the normal hypercube (*n* = total number of peers)!



Normal Hypercube Topology

Simulated Hypercube Topology

Ingredients for Fault-Tolerant Hypercube System

Simulation: Node consists of several peers!

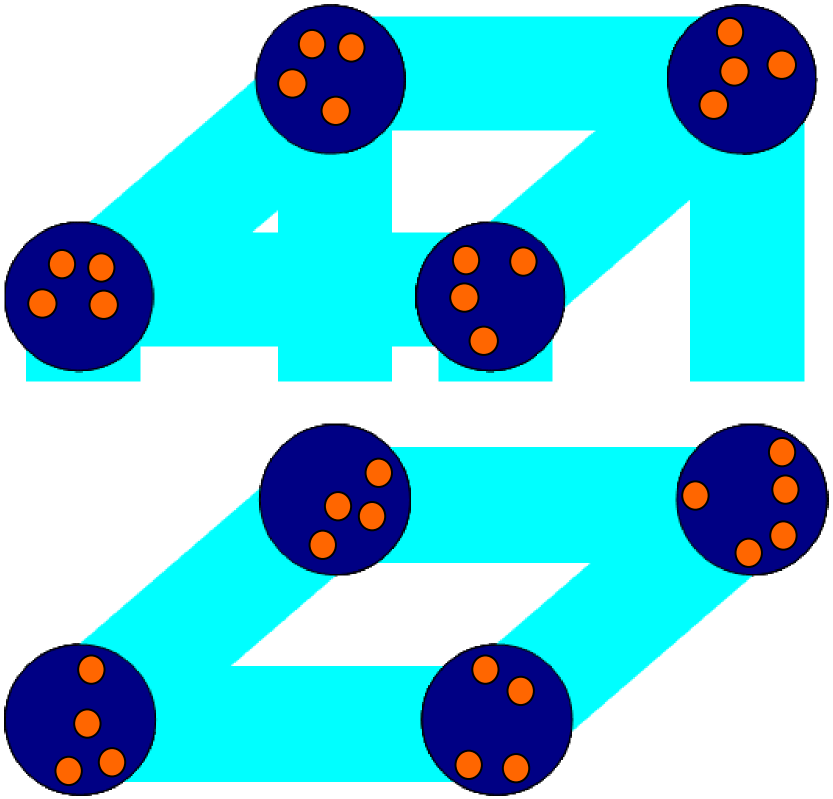
Basic components:

- Route peers to sparse areas

Token distribution algorithm!

- Adapt dimension

Information aggregation algorithm!



Components: Peer Distribution and Information Aggregation

Peer Distribution

Tackled next!

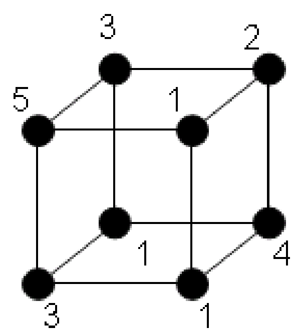
- Goal: Dis[illegible] peers evenly among all hypercube nodes in orde[illegible]ance biased adversarial churn
- [illegible]ally a token distribution problem

Counting the total number of peers (information aggregation)

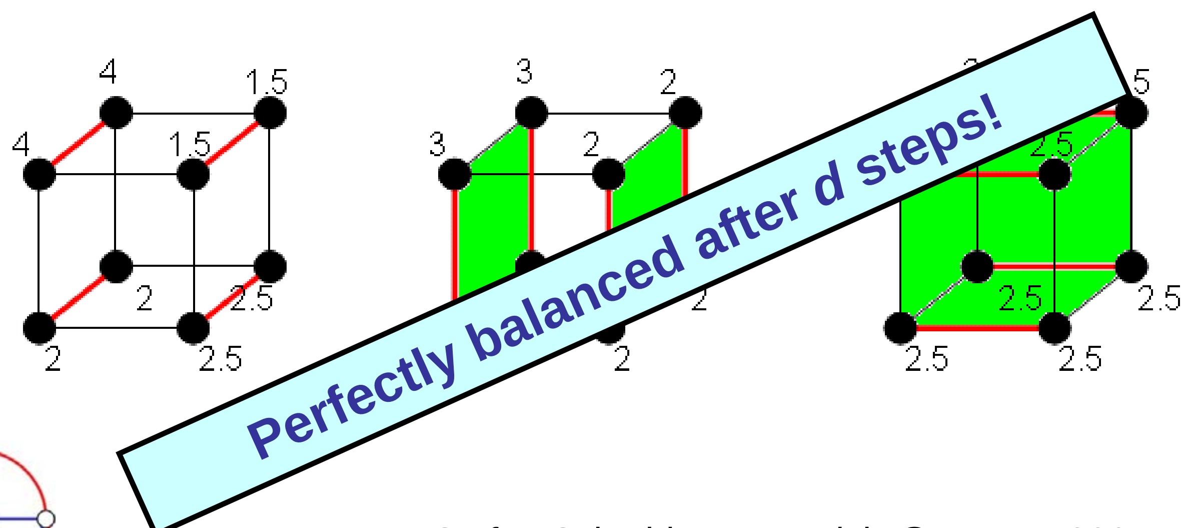
- Goal: Estimate the total number of peers in the system and adapt the dimension accordingly

Dynamic Token Distribution Algorithm (1)

Algorithm: Cycle over dimensions and balance!

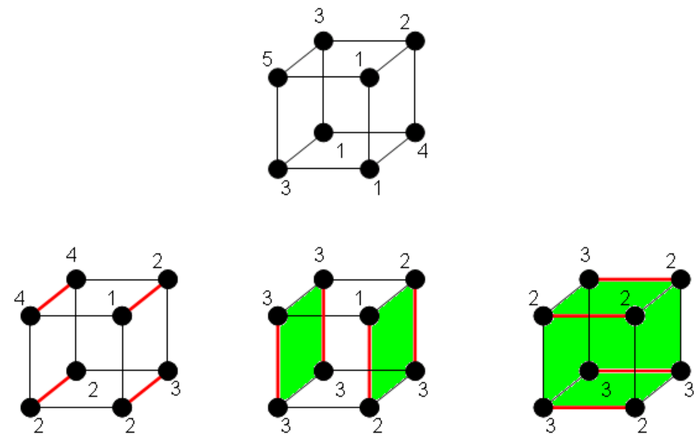


(* algorithm running on node $b_0 \ldots b_{d-1}$ *)
$my\_id := b_0 \ldots b_{d-1}$;
$\mathcal{T}_{my\_id}$:=tokens at this node;
for $i := 0$ to $d-1$ **do**
    $buddy\_id := b_0 \ldots \overline{b_i} \ldots b_{d-1}$;
    SEND $|\mathcal{T}_{my\_id}|/2$ tokens to node $buddy\_id$;
    update $\mathcal{T}_{my\_id}$ accordingly;
    $\mathcal{T}_{buddy\_id}$:=REVC tokens from node $buddy\_id$;
    $\mathcal{T}_{my\_id} := \mathcal{T}_{my\_id} \cup \mathcal{T}_{buddy\_id}$;
end for



Dynamic Token Distribution Algorithm (2)

- Problem 1: Peers are not fractional!



- However, by induction, the integer discrepancy is at most *d* larger than the fractional discrepancy.

$$
\begin{aligned}
|v|_{t+1}^{int} &\leq \left\lceil \frac{|v|_t^{int} + |u|_t^{int}}{2} \right\rceil \leq \left\lceil \frac{\left\lfloor |v|_t^{frac} + \frac{i}{2} \right\rfloor + \left\lfloor |u|_t^{frac} + \frac{i}{2} \right\rfloor}{2} \right\rceil \\
&\leq \frac{\left\lfloor |v|_t^{frac} + \frac{i}{2} \right\rfloor + \left\lfloor |u|_t^{frac} + \frac{i}{2} \right\rfloor}{2} + \frac{1}{2} \\
&\leq \frac{|v|_t^{frac} + |u|_t^{frac} + i + 1}{2} = |v|_{t+1}^{frac} + \frac{i+1}{2}.
\end{aligned}
$$

Dynamic Token Distribution Algorithm (3)

- Problem 2: An adversary inserts at most *J* and removes at most *L* peers *per step*!

- Fortunately, these dynamic changes are balanced quite fast (geometric series).

$$\underbrace{J_t + \frac{J_{t-1}}{2} + \frac{J_{t-2}}{4} + ... + \frac{J_{t-(d-1)}}{2^{d-1}}}_{< 2J} + \underbrace{\frac{J_{t-d}}{2^d} + \frac{J_{t-(d+1)}}{2^d} + \frac{J_{t-(d+2)}}{2^d} + ...}_{\text{shared by all nodes}}$$

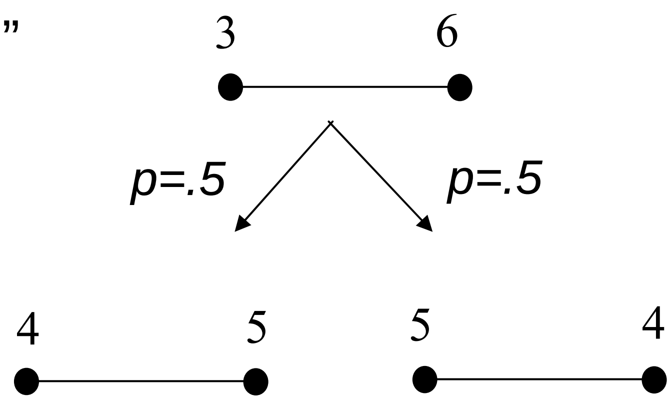
- Thus

Theorem 1: Given adversary *ADV(J,L,1)*, discrepancy never exceeds *2J+2L+d*!

Excursion: Randomized Token Distribution

- Again the static case, but this time assign "dangling" token to one of the edge's vertices *at random*

- "Randomized rounding"



- Dangling tokens are binomially distributed => Chernoff lower tail

Theorem 2: The expected discrepancy is *constant* (~ 3)!

Components: Peer Distribution and Information Aggregation

Peer Distribution

- Goal: Distribute peers evenly among all hypercube nodes in order to balance biased adversarial churn
- Basically a token distribution problem

Counting the total number of [illegible] (information aggregation)

- Goal: Estimate the [illegible]mber of peers in the system and adapt the dimension [illegible]ngly

Tackled next!

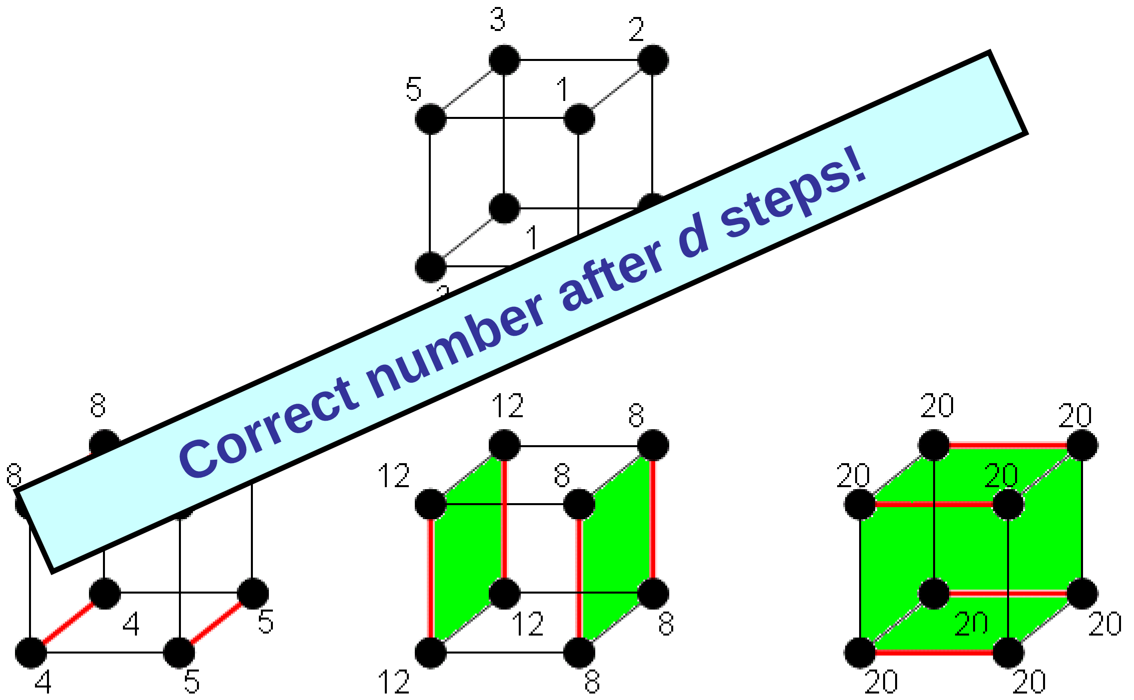
Information Aggregation Algorithm (1)

- Goal: Provide the same (and good!) estimation of the total number of peers presently in the system to all nodes
  - Thresholds for expansion and reduction

- Means: Exploit again the recursive structure of the hypercube!

Information Aggregation Algorithm (2)

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



Correct number after *d* steps!

Information Aggregation Algorithm (3)

- But again, we have a concurrent adversary!
- Solution: Pipelined execution!

Theorem 3: The information aggregation algorithm yields the same estimation to all nodes. Moreover, this number represents the correct state of the system *d* steps ago!

Composing the Components

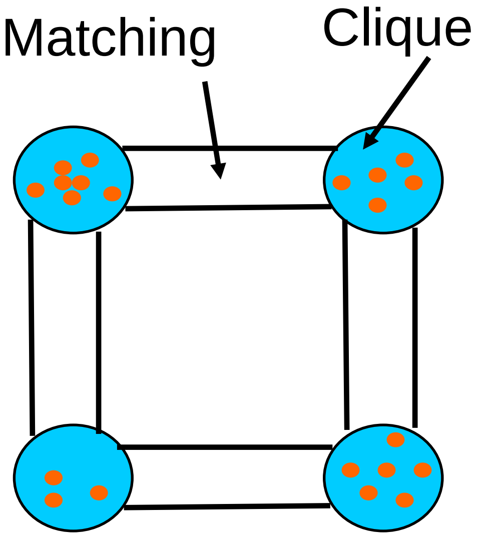
- Our system permanently runs
  - Peer distribution algorithm to balance biased churn
  - Information aggregation algorithm to estimate total number of peers and change dimension accordingly

But: How are peers connected inside a node, and how are the edges of the hypercube represented?

Intra- and Interconnections

- Peers inside the same hypercube vertex are connected *completely* (clique).

- Moreover, there is a matching between the peers of neighboring vertices.



Maintenance Algorithm

- Maintenance algorithm runs in *phases*
  - Phase = 6 rounds

- In phase *i*:
  - Snapshot of the state of the system in round 1
  - One exchange to estimate number of peers in sub-cubes (information aggregation)
  - Balances tokens in dimension *i mod d*
  - Dimension change if necessary

All based on the snapshot made in round 1, ignoring the changes that have happened in-between!

Results for Hypercube Topology

- Given an adversary *ADV(d+1,d+1,6)...*
  => Peer discrepancy at most *5d+4* (Theorem 1)
  => Total number of peers with delay *d* (Theorem 3)

- ... we have, in spite of *ADV(O(log n), O(log n), 1)*:

  – always at least one peer per node,

  – peer degree bounded by *O(log n)* (asymptotically opitmal!),

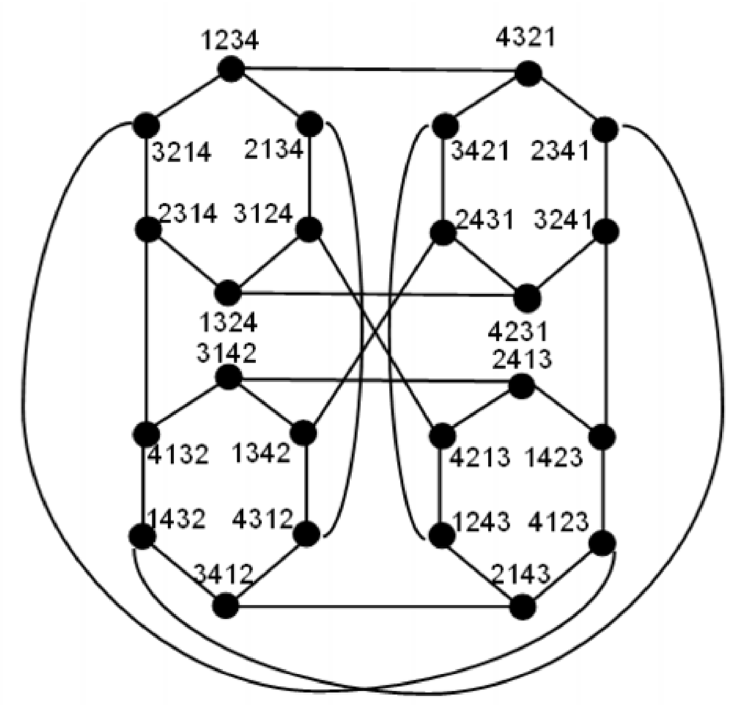
  – network diameter *O(log n).*

A Blueprint for Many Graphs?

- Conclusion: We have achieved an asymptotically optimal fault-tolerance for a *O(log n)* degree and *O(log n)* diameter topology.

- Generalization? We could apply the same tricks for general graphs *G=(V,E)*, given the ingredients (on *G*):
  - –token distribution algorithm
  - –information aggregation algorithm

- For instance: Easy for skip graphs (Δ = *D* = O(*log n*)), possible for pancake graphs (Δ = *D* = *O(log n / loglog n)*).



Open Problems

- Experiences with other graphs?
- Other models for graph dynamics?
- Less messages?

Thank you for your attention!

Discussion

- Questions?
- Inputs?
- Feedback?