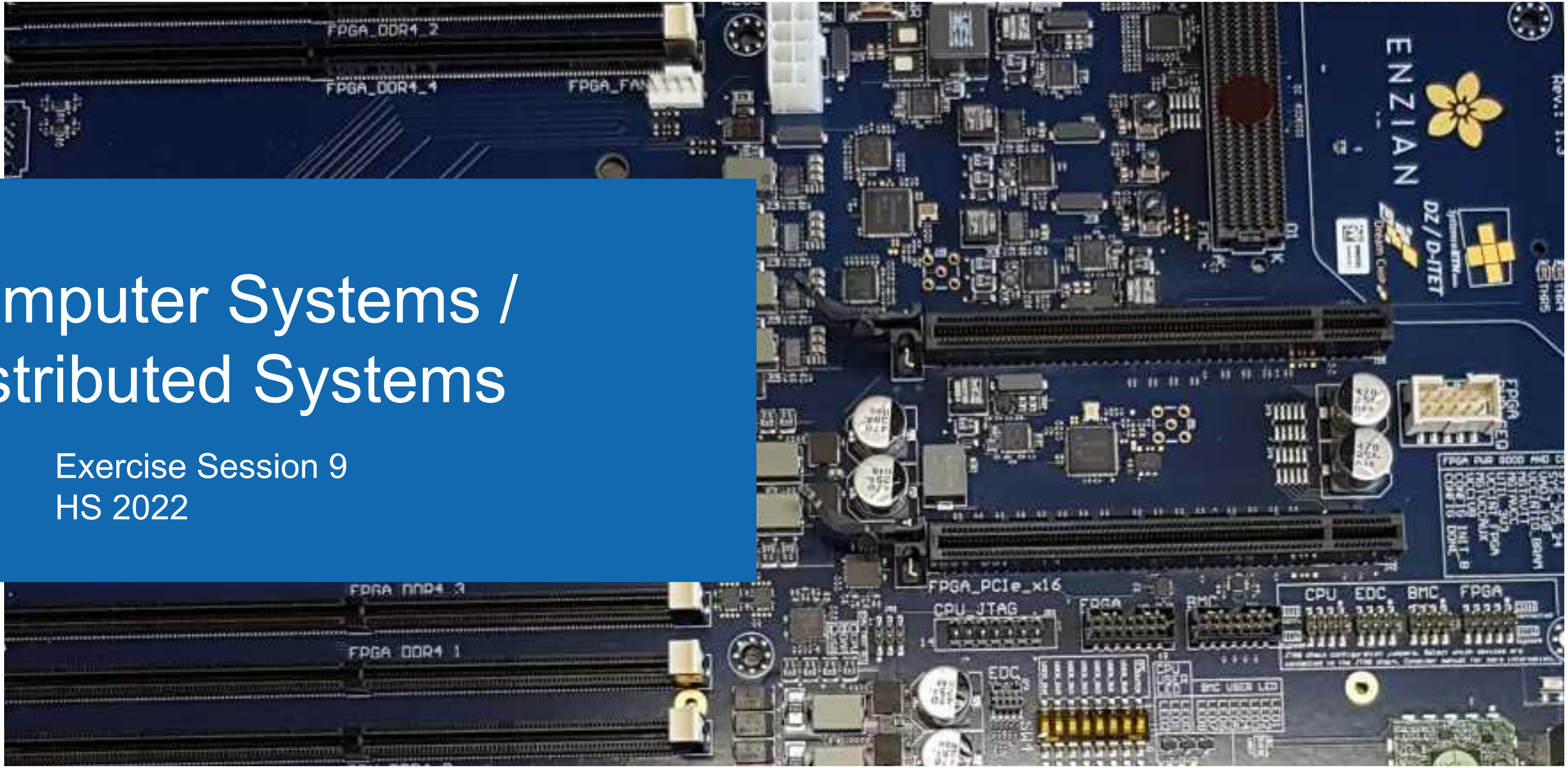
ETH zürich Systems@ETH zürich Distributed Computing

Slides last updated: 08.12.2022



Computer Systems / Distributed Systems

Exercise Session 9
HS 2022

Distributed Storage

Consistent Hashing

How to store many items on many nodes in a “**consistent**” manner?

Use **hash functions** to transform item and node IDs into values in **[0,1)**

For each hash function, item is stored on machine with the closest hash.

Consistent Hashing

Some properties of **consistent hashing**:

- Each node stores the same number of items in expectation
- Number of hash functions determines degree of duplication
- Any single node’s memory consumption is bounded (by Chernoff bound)
- Supports nodes leaving/joining

Hypercubic Networks

- How should our distributed storage system look like?
- How should the nodes be connected?
- How do we find a particular item?
- ...

Hypercubic Networks

In a classic distributed system one node can have a view of the entire system because nodes rarely leave/join

However, we are considering **very large networks** with **high churn** in which it becomes impossible for nodes to have an accurate and updated picture of large parts of the network topology

Thus, we want a system that only relies on every node knowing its **small neighborhood**

What kind of network topology should we use?

Hypercubic Networks

Consistent hashing reminder: Where to store items

Hypercubic networks: Arrange nodes such that they form a virtual network, also called an **overlay network**

In general, the overlay network gives us the possibility to “navigate” our distributed storage system, i.e., do **routing**. This is necessary since each node only has a local view, but we still want to find any item, even if it is not in the neighborhood of the node we are currently querying

Hypercubic Networks

A good overlay topology should fulfill the following properties (more or less):

- **Homogeneity**: No single point of failure, all nodes are "equal"
- **Node IDs in [0,1)** for consistent hashing
- Nodes have **small degree**, i.e., only relatively few neighbours
- **Small diameter** and easy routing: Any node should be reachable within reasonable time

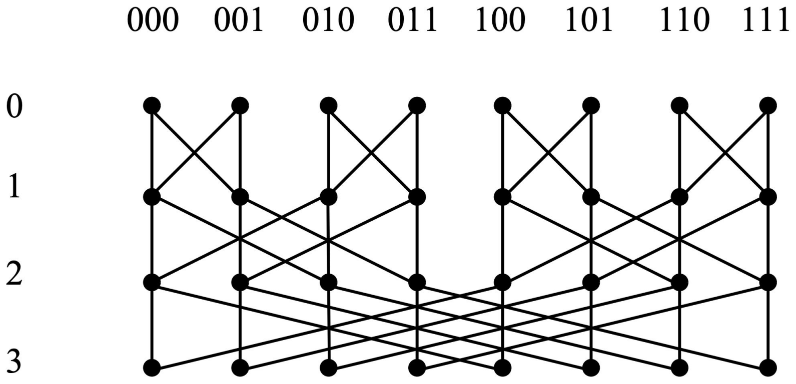
ETH zürich Systems@ETH zürich Distributed Computing

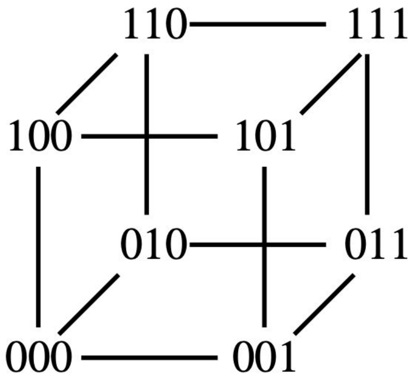
Hypercubic Networks

Different overlay topologies make different trade-offs, for example:

Butterflies: Constant small node degree

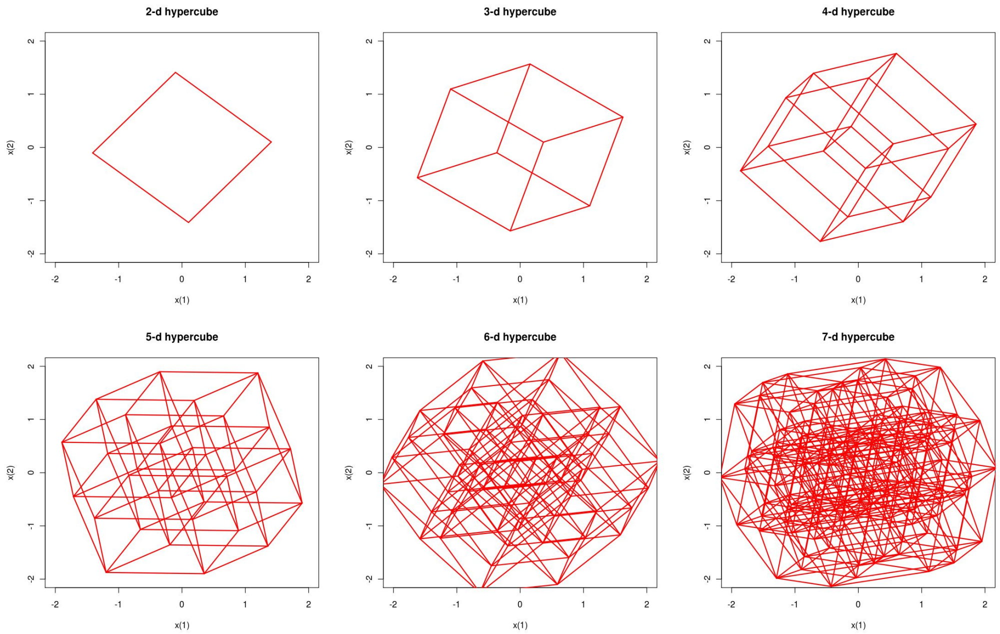
Hypercube: More fault tolerant routing; i.e. more short routes between nodes (k! routes of length k)





Hypercubic Networks

You will draw some simple hypercubic graphs in the quiz

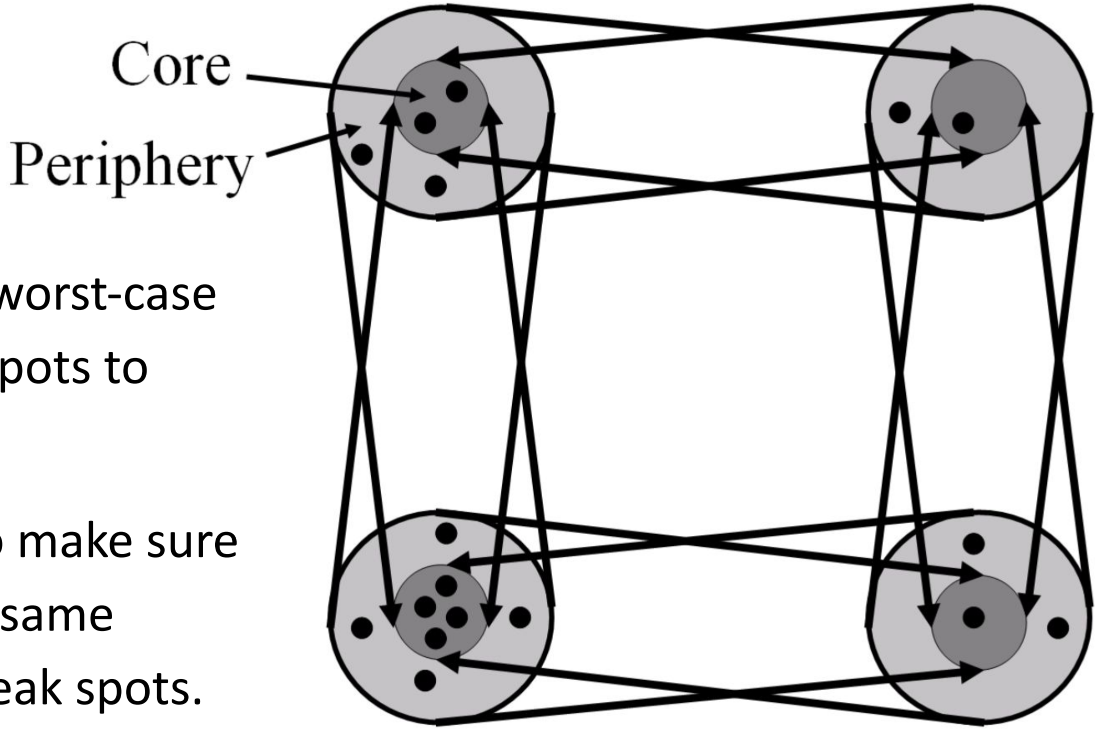


DHT & Churn

DHT: Distributed Hash Table

- Combines consistent hashing with overlay networks
- Supports searching, insertion and (maybe) deletion
- For example: Use hypercube with hyper nodes. “Core” nodes store data, “periphery” nodes can move around.

DHT & Churn



Robustness against Churn

- Attacker crashes nodes in worst-case manner. Can target weak spots to partition the DHT.
- DHT redistributes nodes to make sure each hypernode has ≈ the same number of nodes → No weak spots.

Quiz

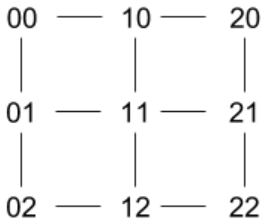
Draw the following hypercubic graphs:

- M(3,1)
- M(3,2)
- SE(2)
- M(2,4)

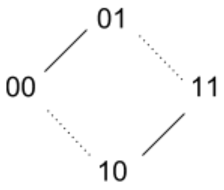
Quiz Solutions

M(3,1)

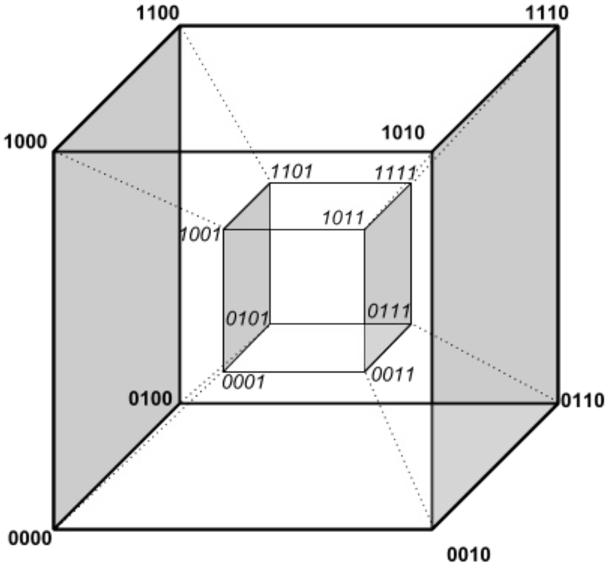
M(3,2)



SE(2)



M(2,4)



Assignment Outlook

Basic

2.2 Iterative vs. Recursive Lookup

There are two fundamental ways to perform a lookup in an overlay network: recursive and iterative lookup.

Assume node n_0 is attempting to look up an object in a DHT. In the recursive lookup n_0 selects a node n_1 which is closest according to the DHT metric and sends a request to it. Upon receiving the request n_1 selects its closest known neighbor n_2 and forwards the request to it and so on. The request either ends up at the node storing the object, returning the object along the same path, or it ends at a node that does not store the object and does not have a closer neighbor.

In the iterative case n_0 looks up the closest neighbor n_1 and sends it the request. Upon receiving the request n_1 is either the node storing the object and it returns the object, or it knows a closer node n_2 and returns n_2 to the n_0. If n_0 receives a node n_2 it will add it to its neighbor set and sends a new request to n_2 which is now its closest neighbor. The lookup terminates either when n_0 sends a request to the node storing the object, or no closer node can be found.

a) What are the advantages of recursive lookups over the iterative lookups?

b) Most systems that are in use today use the iterative lookup, and not the recursive lookup, why?

Assignment Outlook

2.3 Building a set of Hash functions

Consistent hashing relies on having k hashing functions $\{h_0, \ldots, h_{k-1}\}$ that map object ids to hashes. There are several constructions for these hash functions, the most common being iterative hashing and salted hashing. In iterative hashing we use a hash function h and apply it iteratively so that the hashes of an object id o are defined as

$$h_i(o) = \begin{cases} h(o) & \text{if} \quad i = 0 \\ h(h_{i-1}(o)) & \text{otherwise.} \end{cases}$$

With salted hashing the object id is concatenated with the hash function index i resulting in the following definition

$$h_i(o) = h(o|i).$$

Which hashing function derivation is better and why?

Assignment Outlook

Advanced

2.4 Multiple Skiplists

In the lecture we have seen the simple skip list in which at each level nodes have probability 1/2 of being promoted to the next level. We have also discussed a variation known as a skip graph. For yet another option, we once again redefine the promotion so that a node is promoted to a list s if s is a suffix of the binary representation of the node's id. At each level l we now have 2^l lists (some empty), each defined by a string of bits s of length l. In particular, the root level $l = 0$ is constructed with s being the empty string. The second level has one list for each $s \in \{0, 1\}$, the third level one list for each $s \in \{00, 01, 10, 11\}$, and so on. We call the resulting network a multi-skiplist. For the purposes of this question, assume that all lists are circular.

a) Assuming we have an 8 node network, with ids $\{000, \ldots, 111\}$, draw the multi-skiplist graph.

b) What is the minimum degree of a node in the multi-skiplist if we have d levels?

c) What is the maximum number of hops a lookup has to perform?

Q & A Session