ETH zürich Systems@ETH zürich Distributed Computing

Slides last updated: 17.11.2024

Computer Systems
Distributed Systems

Exercise Session 12 · HS 2024

Agenda

- Assignment Review
- Lecture Recap
  - Clock Synchronization
  - Distributed Storage
- Quiz
- Assignment Preview

ETH zürich Systems@ETH zürich Distributed Computing

Assignment Review

1.2 From Approximate Agreement to Byzantine Agreement

Core idea: Use Approximate Agreement and round the final value according to $x \geq 0.5$.

Does all-same validity hold?

- Yes, Approximate Agreement achieves correct-range validity and therefore also all-same validity.

Does agreement still hold?

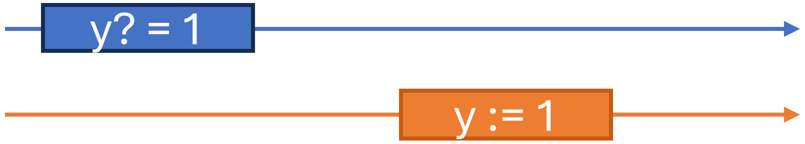
- No after the rounds, our values could be distributed around 0.5 which could lead to differing rounding decisions.

2.1 Different Consistencies

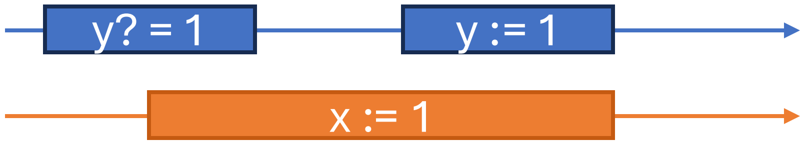
Prove or disprove:

Neither sequential consistency nor quiescent consistency imply linearizability.

- Sequential consistency: No, it does not care about real time.

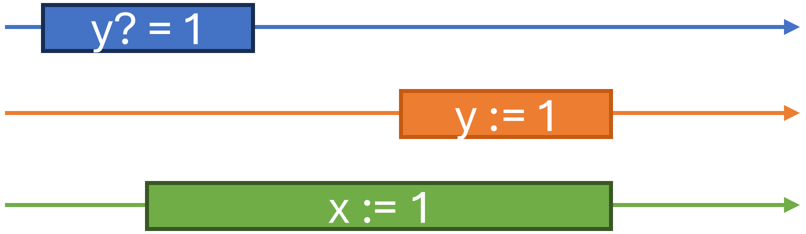


- Quiescent consistency: No, it does not care for in thread consistency.



If a system is both **sequentially** consistent **and quiescent** consistent, it is **linearizable**.

- No, we can combine the two counter examples.



Clock Synchronization

Time & Clocks

- **Wall Clock:** “true time” clock. An unrealistic baseline.
- **Clock Errors:**
  - **Drift δ:** The predictable difference in clock speed compared to the **wall clock**.
  - **Jitter ξ:** The unpredictable short-term errors in clock speed compared to the **wall clock**.

  - **Skew:** Error between two clocks: Modeled as $t = (1 + \delta)t^* + \xi^*(t^*)$

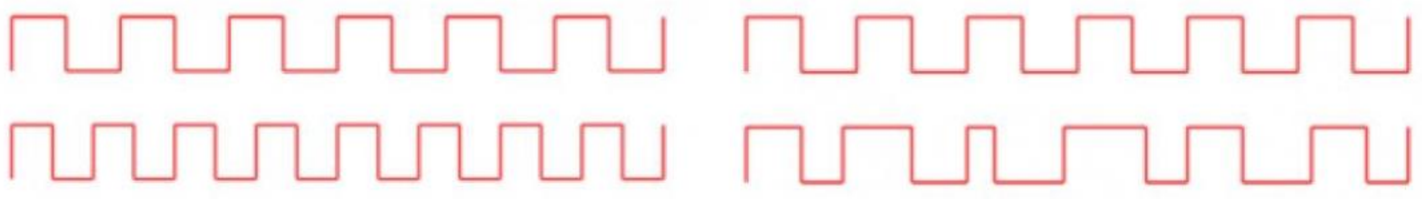


Figure 20.8: Drift (left) and Jitter (right). On top is a square wave, the wall-clock time t^*.

ETH zürich Systems@ETH zürich Distributed Computing

Clock Synchronization Algorithm

Algorithm 23.9 Clock synchronization algorithm (code for node v)

Increase clock C_v at local clock rate
upon clock value C_v reaches next integer value:
  Send C_v to all neighboring nodes
end upon
upon receiving clock value C_w from node w:
  if $C_w > C_v$ **then**
    $C_v := C_w$
    Send C_v to all neighboring nodes
  end if
end upon

Core idea:

Periodically inform neighbors about current time, whenever you are lagging behind increase clock to match others time.

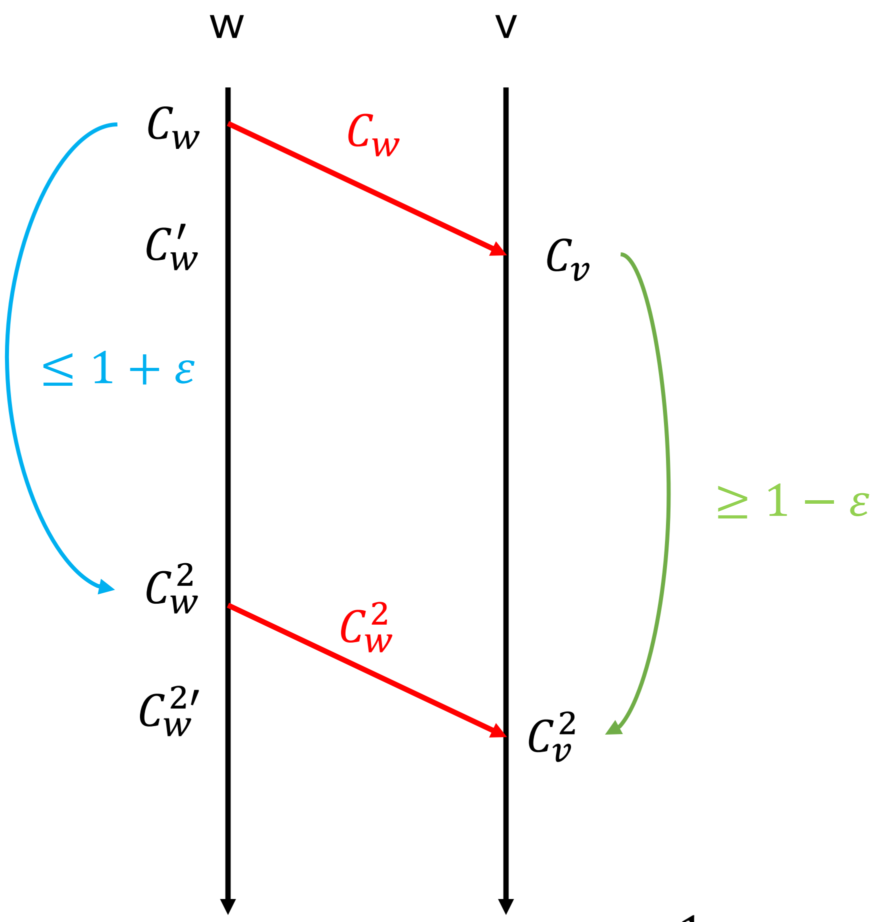
Do we expect to ever truly catch up?

- No, we expect our neighbor's time to be further already.

How good is our approximation?

- Let's analyze.

Clock Synchronization Algorithm Analysis



$$(1+\varepsilon)T + (1+\varepsilon-(1-\varepsilon))\frac{1}{1-\varepsilon}$$

1. At C_w w sends out its time
2. At C_v in v or C'_w in w, v receives C_w. v observes $C_w > C_v$ and sets its local time to match C_w.

We know that:

- The real time between (1.) and (2.) is limited by some T.
- Each clock has a drift in $[1-\varepsilon, 1+\varepsilon]$.

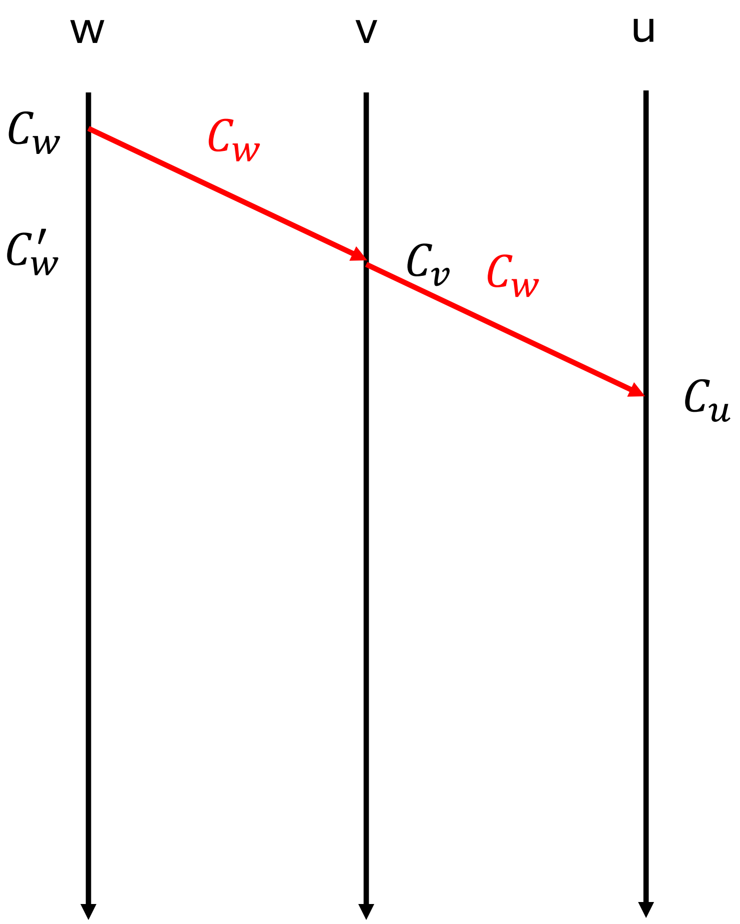
At time (2.) what is the maximal difference the clock values after the update to C_w in v?

- $(1+\varepsilon)T$ at most T passed and the clock of w runs at most $(1+\varepsilon)$ faster than real time.

What about just before the update?

- $(1+\varepsilon)T + \frac{2\varepsilon}{1-\varepsilon}$, for this we look at the next update and the different clock value differences since (2.): $\frac{1}{1-\varepsilon}$ factor is a safety margin for difference in real time broadcast.

Clock Synchronization Algorithm Analysis



With a bigger network with diameter D?

Result: Clock skew between any 2 nodes at most:

$$(1+\varepsilon)DT + \frac{2\varepsilon}{1-\varepsilon}$$

Only the $(1+\varepsilon)T$ error scales with the network.

Can we do better?

Global and Local Skew

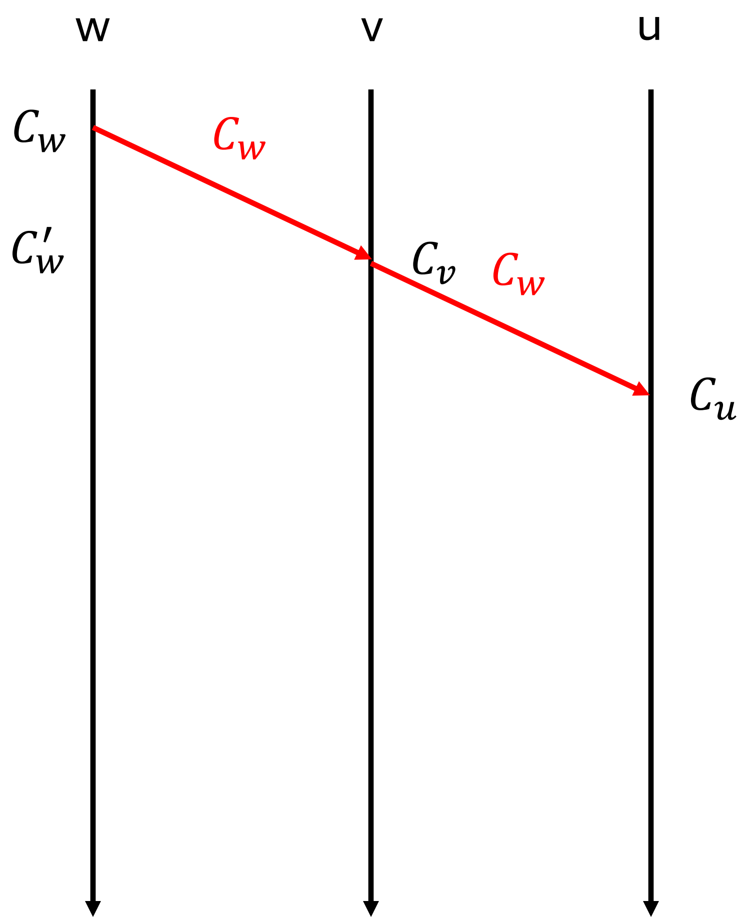
Global Skew: The maximal difference between clock values in the network.

Trivial lower bound: $DT/2$ which seems reasonable as nodes need to communicate over DT sized time differences.

Local Skew: Maximal difference between two neighboring clocks in the network.

Lower bound: $\Theta(T \log D)$ is this intuitive?

- Not really, they can communicate with time difference T, why does the diameter influence this?
- This is not as bad as it seems, because most values involved in these calculations behave very nicely according to predictable distributions.



ETH zürich | Systems@ETH zürich | Distributed Computing

Distributed Storage

Consistent Hashing
Hypercubic networks
Distributed hash table

Distributed Storage

Idea: Distribute resource like movies over many machines to have them quickly and always available.

Goals:

- High Availability for all movies.
- Good load balancing.
- Global Distribution.

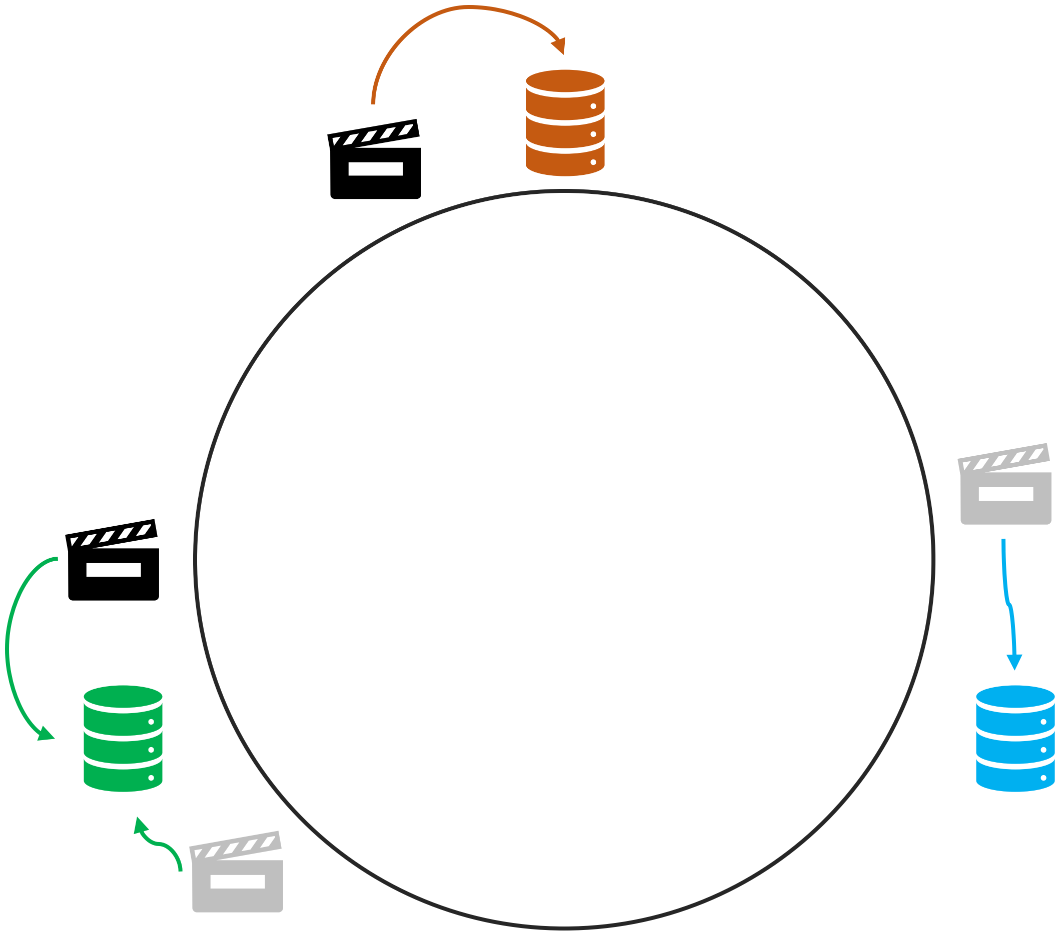
How do we decide where to store the movies and how do we give users this information?

ETH zürich Systems@ETH zürich Distributed Computing

Consistent Hashing

Idea:

- 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.



With 2 hash functions and [0, 1) mapped to a ring

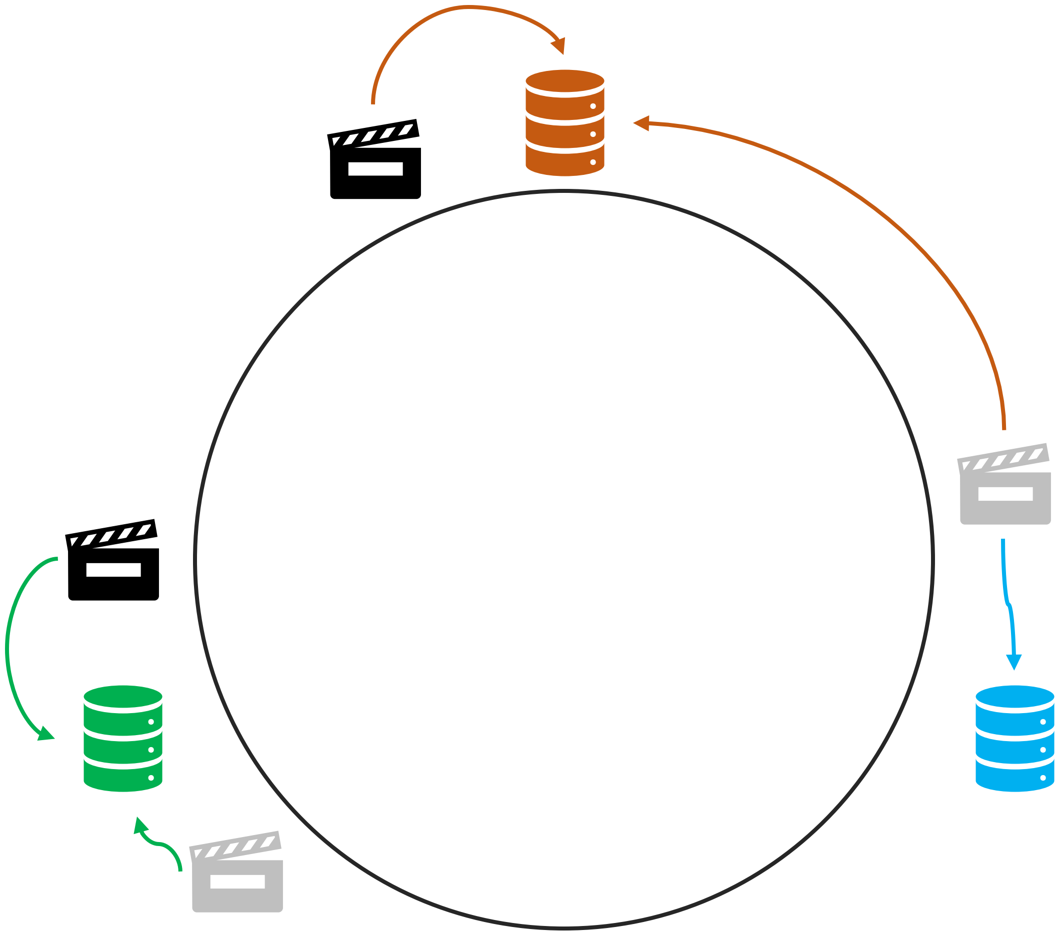
Consistent Hashing

Properties:

- In **expectation** each node stores the same amount of data.
- Duplication rate is well controlled.
- Supports nodes leaving / entering.

What would happen if the blue server leaves the system?

- Job of storing second copy of grey movie gets transferred to orange.



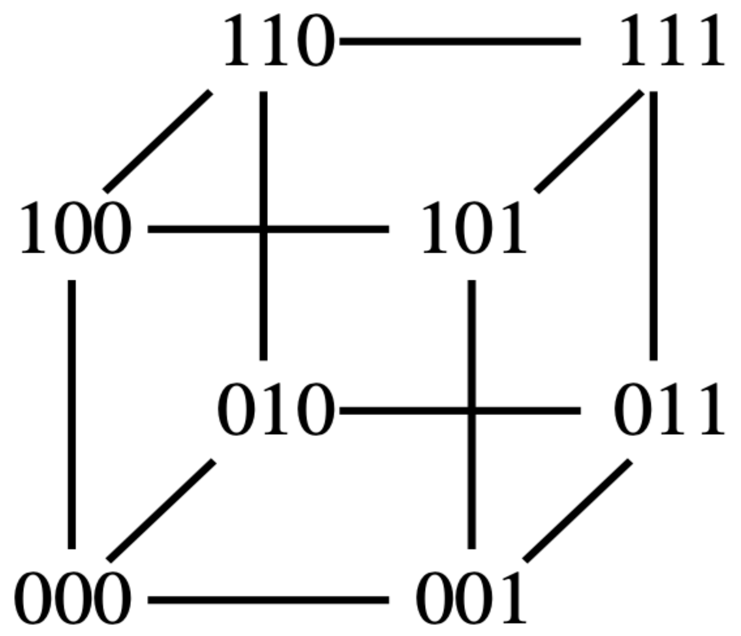
With 2 hash functions and [0, 1) mapped to a ring

Hypercubic Networks

Idea: Get good topological features from our network.

Topological Features:

- **Homogeneous:** Nodes should be similar / equal.
- **IDs:** Each node should have a unique id.
- **Degree:** Number of neighbours of each node
- **Diameter:** Furthest distance between nodes.



How are the degree and diameter optimal and why?

- Both should be small.
- Small degree to keep work of a single node low.
- Small diameter to enable fast communication.

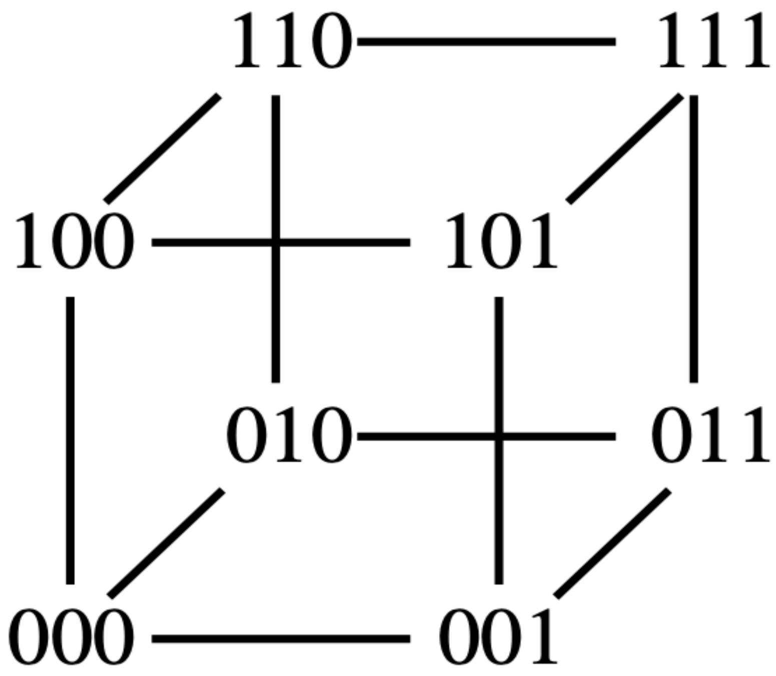
Analysis of the Hypercube Family

Idea: In each dimension increase: Double the nodes and connect those with only a single difference in ID.

Topological Features:

- **Homogeneous:**
  - **Yes**
- **Degree:**
  - $\log n$
- **Diameter:**
  - $\log n$

Can we decrease degree and keep diameter in $O(\log n)$?

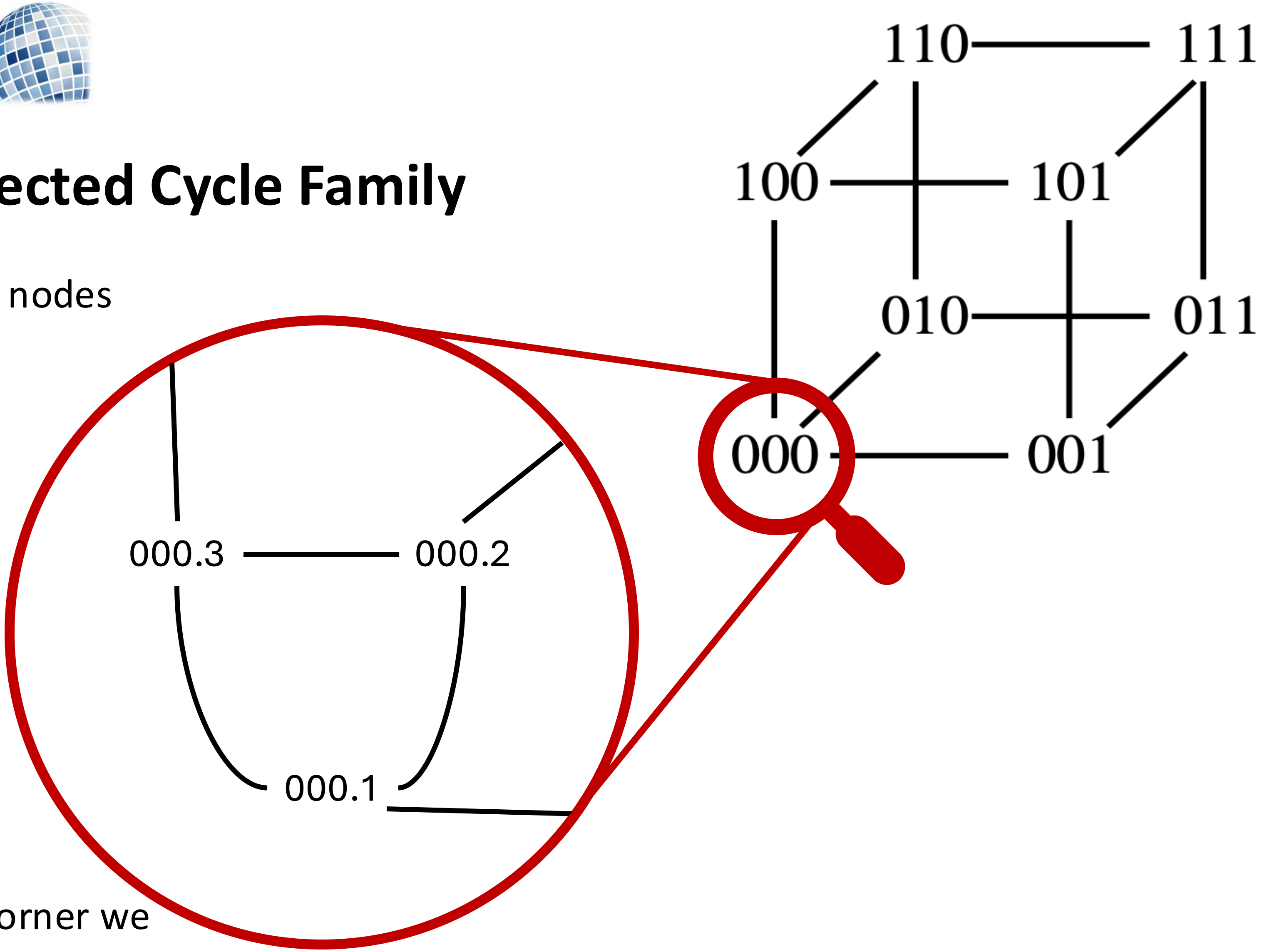


Analysis of the Cube Connected Cycle Family

Idea: Hypercube, but each corner has d nodes connected in a cycle.

Topological Features:

- **Homogeneous:**
  - Yes
- **Degree:**
  - 3
- **Diameter:**
  - $2 \log n$



Why is the diameter not $log^2 n$ as we might need to walk each cycle in each corner we pass?

- In total we just pass the cycle once if we reduce the dimensions in the right order.

ETH zürich | Systems@ETH zürich | Distributed Computing

Analysis of other Families

Butterfly

Idea: Use efficient Reduce All communication structure to design network.

Analysis: How does this differ from the CCC?

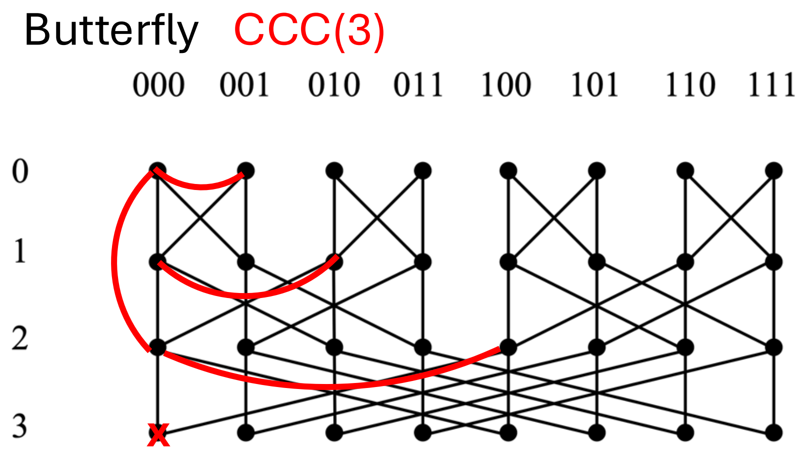
- **No loop** for each ID.
- Connections between rounds instead of in same round.
- One to many nodes per id.

Binary Tree (Fat)

Idea: Always split network in half through a capable root node.

Analysis: Which property isn't achieved?

- **Homogeneous**



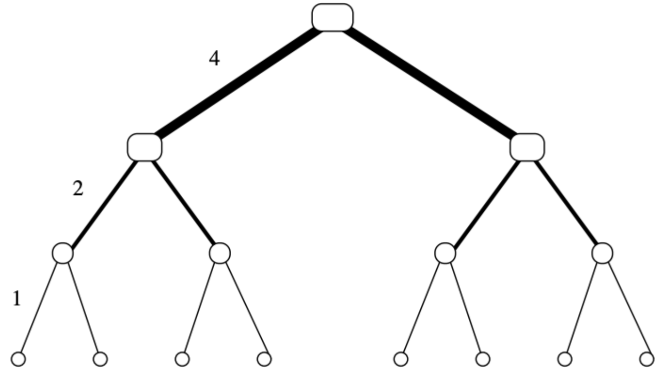
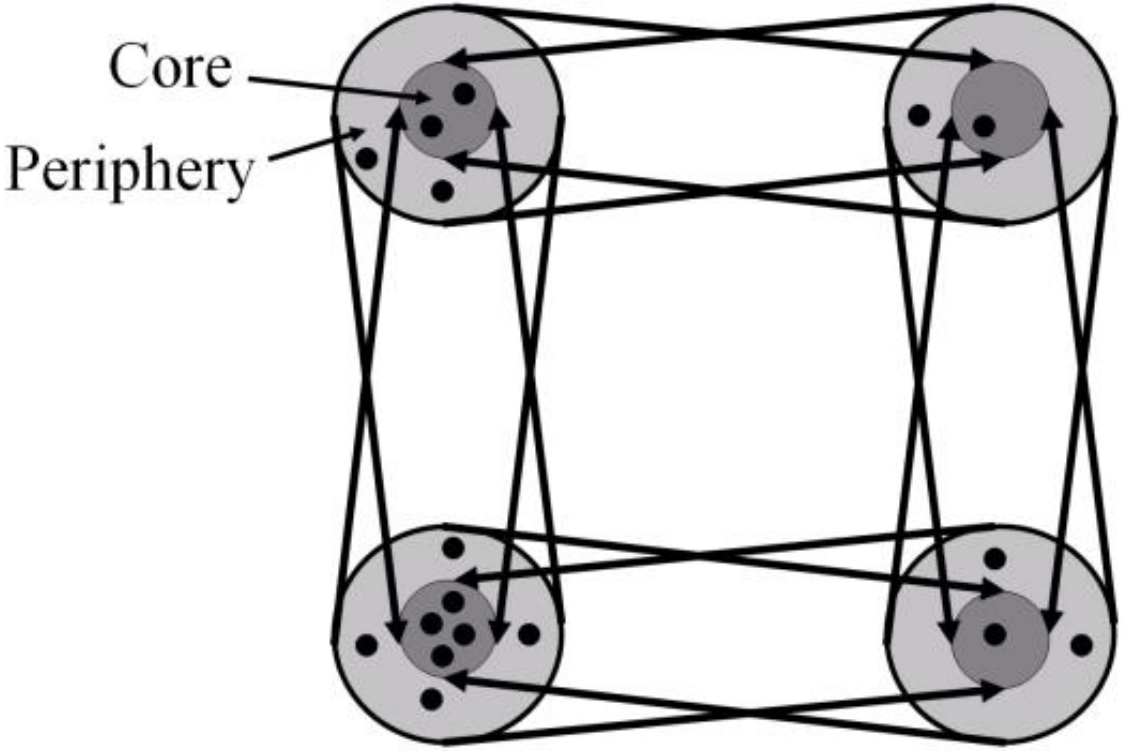


Figure 24.5: The structure of a fat tree.

Distributed Hash Table (DHT) & Churn

Idea:

- Combine Consistent Hashing with overlay networks.
- Support Inserting and Searching data.
- Use Hypercubic networks with **Hyper nodes**.



Hyper Node:

- Collection of fully connected nodes, that are split into a core and periphery group.
- Act in the network like a single dynamic node, which has a single state.
- Core nodes store data and are interconnected.
- Periphery nodes don't store data and are ready to hop to load balance.

Robustness against Churn

Churn:

- High turn over of participating nodes.
- Nodes often join, leave or crash.

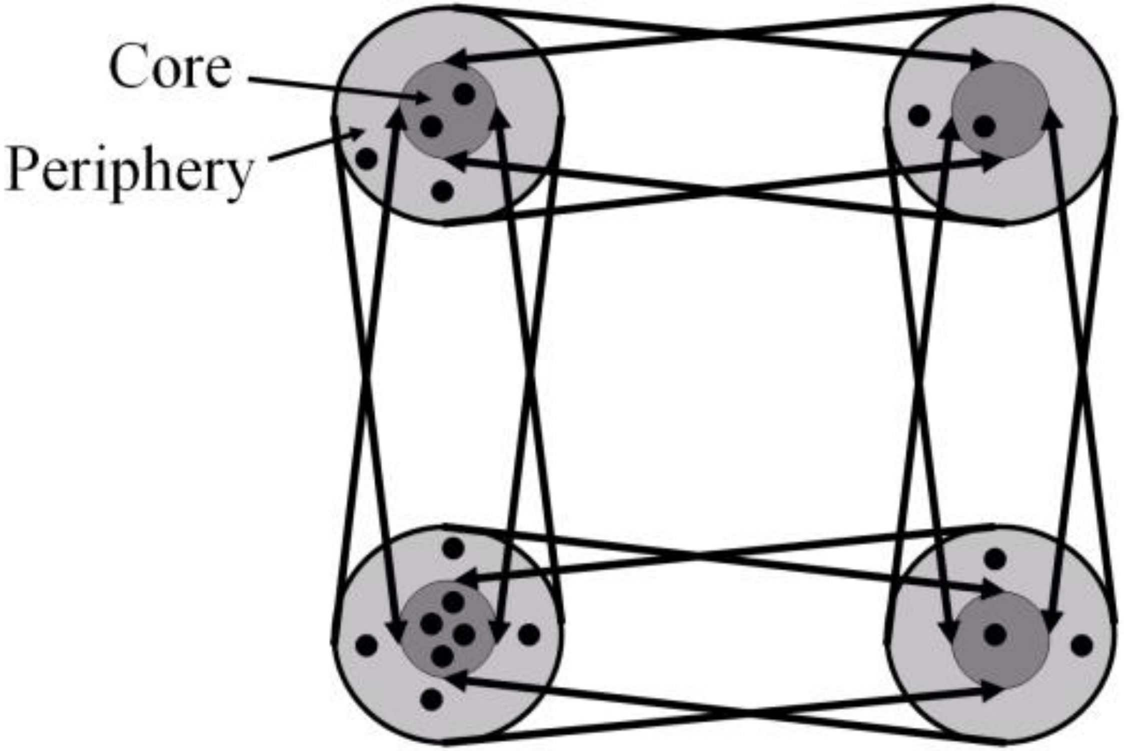
How does the DHT react?

Attacker targets some hypernode and starts crashing those nodes:

- **Hyper node** weakness is picked up by neighbours which send over its periphery nodes.

Why are periphery nodes even associated with hyper nodes to begin with?

- **Load balancing** and smaller more consistent state for ready, but none core nodes.



ETH zürich

Systems@ETH zürich

Distributed Computing

Quiz

ETH zürich Systems@ETH zürich Distributed Computing

Quiz questions

1. Local skew can be independent of the network diameter.

   False

2. Global skew is always bigger than local skew.

   True

3. The fully connected graph is a good hyper cubic network.

   False

4. In consistent hashing node positions are determined randomly.

   False

5. The butterfly network has similar properties to the CCC.

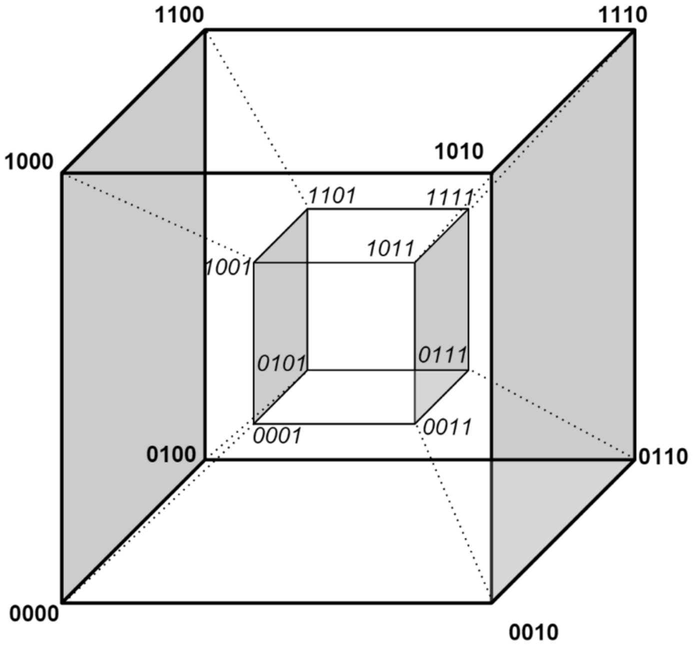
   True

6. In DHT hyper nodes move to different positions to load balance.

   False

Questions and Tasks

Draw a 4d Hypercube



When load on the electric grid is high, the frequency of the grid is lowered. If this is sustained over a while, what clock error do we expect on oven clocks?

- **Drift** they will slowly start falling behind, because they expect net frequency to be exactly 50 Hz.

ETH zürich

Systems@ETH zürich

Distributed Computing

Assignment Preview

Assignment Preview

- Clocks and Time Algorithms
- Drawing Hypercubic Networks
- Hash functions

Final Words