Constant-Degree KBRs

With m=O(log(n)) state and L=O(log(n)) DHTs do not achieve the performance of random graphs/small-world graphs. Lets recap the

$$L_{random} \sim \frac{\log n}{\log(m/n)} \underset{\frac{m}{n}=const}{\sim} \log n$$

- ➔ Random graphs achieve L=O(log(n)) with constant degree. This is an average and the O(logn) we give for the KBRs is a maximum with high probability.
- ➔ Can we build structured networks with constant degree and O(logn) hops?

We can, even with degree 2, e.g. binary trees, Viceroy (KBR based on butterfly graph), de Bruijn graphs, Kautz graphs, Distance-Halving.

- ❑ However, short distances are not for free, constant-degree graphs have longer average paths because they have significantly less links!

De Bruijn graphs

Operations

- Let Σ be a set of symbols, say $\Sigma = \{0,1\}$.
- Shuffle S $(s_1, s_2, s_3, ..., s_k) \rightarrow (s_2, ..., s_k, s_1)$
- Shuffle-Exchange SE $(s_1, s_2, s_3, ..., s_k) \rightarrow (s_2, ..., s_k, \Sigma \setminus s_1)$

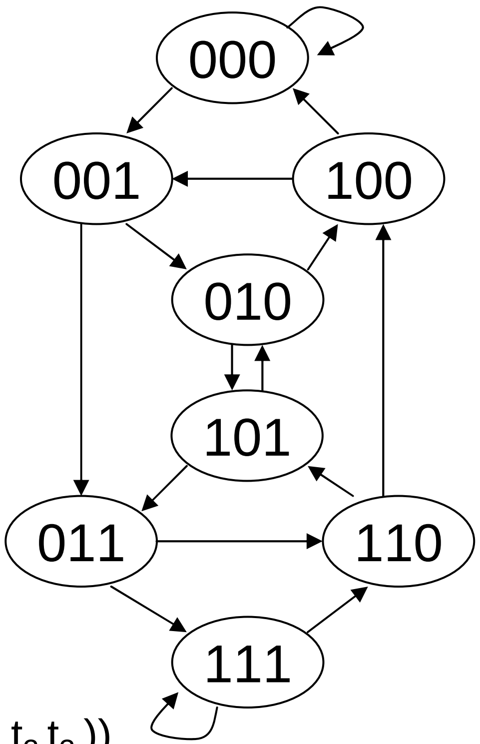
De Buijn graphs

- A node identifyer is then a fixed-length string of these symbols.
- Each node with node_ID has links to nodes that are either S(node_ID) or SE(node_ID).
- Formally:

$$V = \{(s_1 s_2 ... s_k) \mid s_i \in \Sigma\}$$

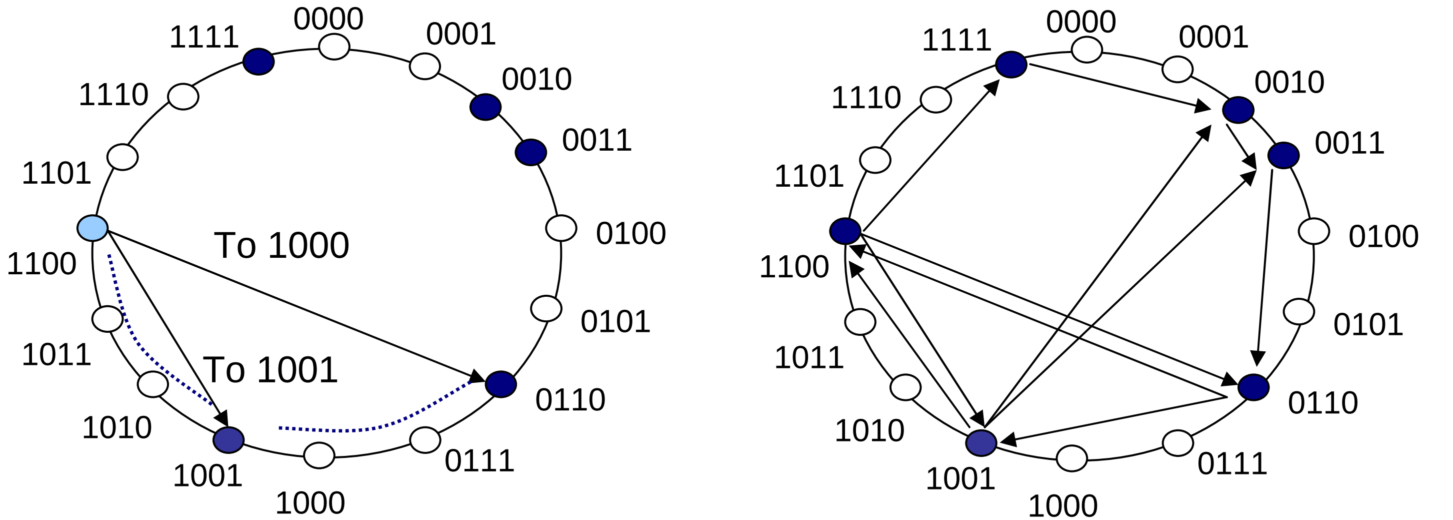
$$E = \{((s_1 s_2 ... s_k), (t_1 t_2 ... t_k)) \mid t_1 = s_2, t_2 = s_3, ..., t_{k-1} = s_k\}$$

- Routing
  - From s=(s_1 s_2 s_3) to t=(t_1 t_2 t_3) use the links ((s_1 s_2 s_3), (s_2 s_3 t_1)) then ((s_2 s_3 t_1), (s_3 t_1 t_2)) then ((s_3 t_1 t_2), (t_1 t_2 t_3))



Koorde

- Ring-based DHT with De Bruijn Graph as embedding.
- Nodes virtually represent all de Bruijn nodes between themselves and their successor.
- Long distance links
  - Outgoing: Link to the nodes according to the de Bruijn neighbors of the node_ID
  - Incoming: Accept incoming links for all your virtual nodes.



Koorde / De Bruijn Graphs – Results

Results

- The diameter of Koorde is with high probability O(logn).
- The outdegree is per design constant O(2+2k).
- The indegree is w.h.p. O(logn).
- *There is yet no stabilization for Koorde.*
- There are other embeddings of de Bruijn graphs, e.g. D2B (in CAN), Broose (in Kademlia).

Extentions

- De-Bruijn graphs are not necessarily binary, but can be defined for arbitrary character size (like Pastry).

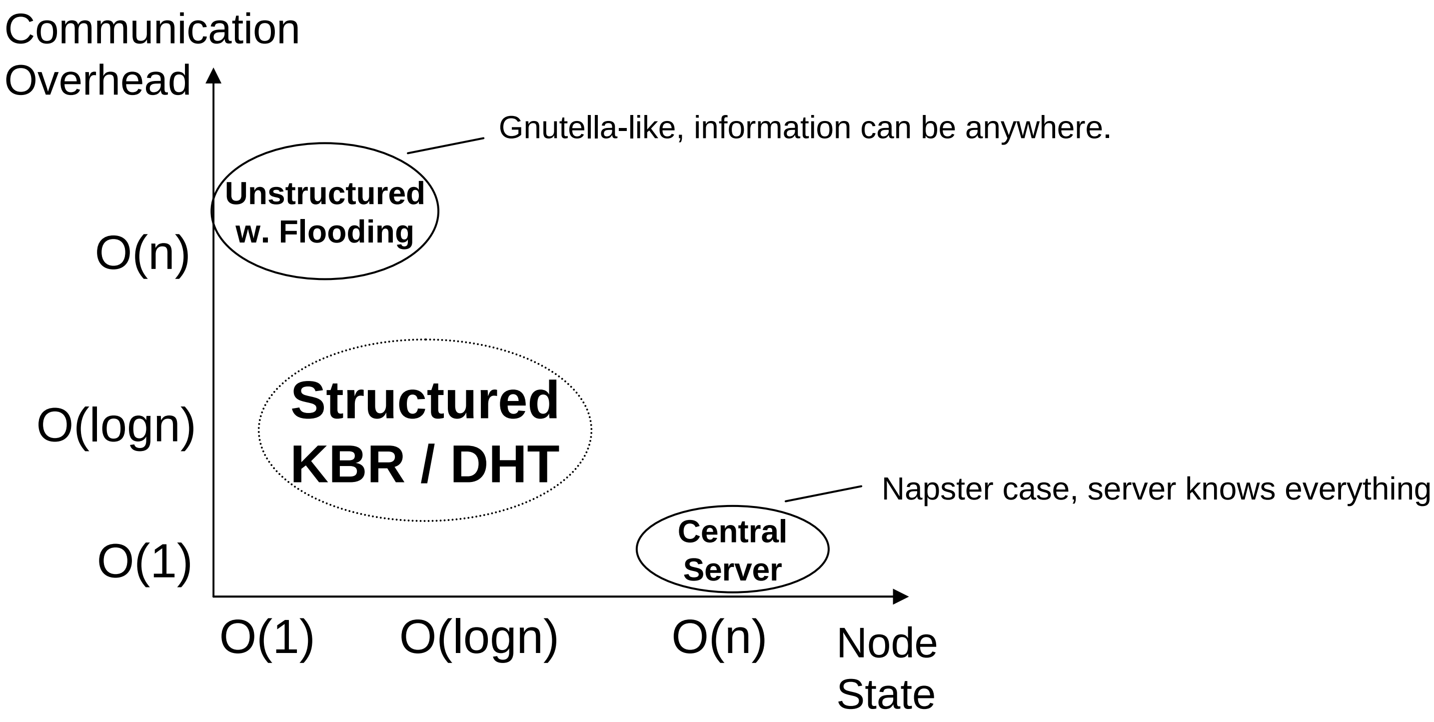
Constant-Distance KBRs

One-Hop-DHT

- Structure: Full Mesh / Clique
  - All nodes know each other.
- Limited scalability due to O(n) state per node and O(n) operations per change
  - Hard to maintain for large networks.
  - Authors claim that routing tables with millions of nodes are no problem with current RAM.
- Trade-off: If one allows more than one hop to all destinations, one can reduce the size of the routing table.

Structured vs Unstructured vs Server

- Comparing DHTs with unstructured networks and central servers

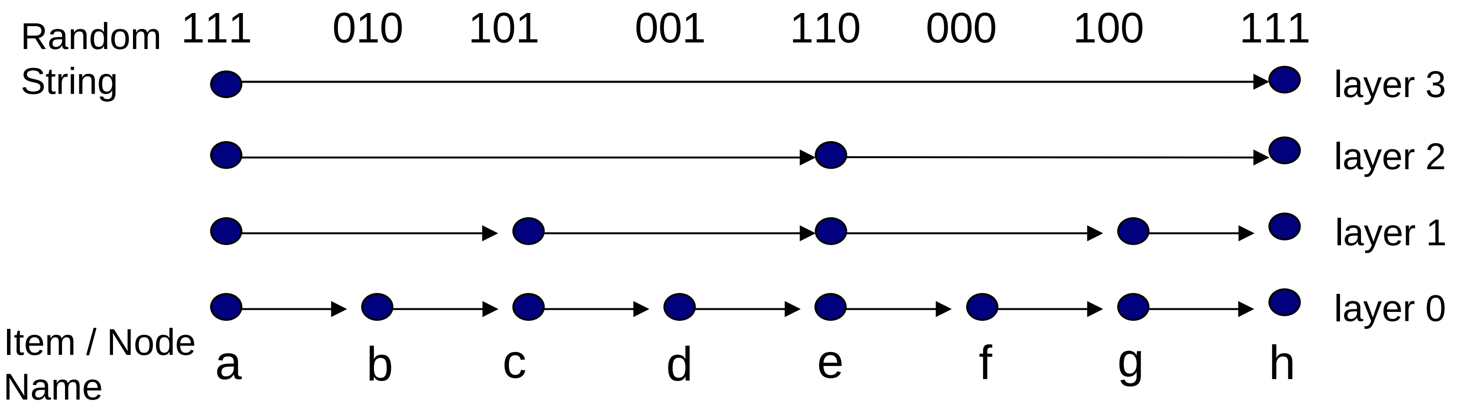


Ordered Indexing

DHT vs Ordered Indexing

- Common DHT use case
  - ID = hash item name
  - Why the hash function?
    - Fixed bitlength
    - Balance the items over ID space
  - Problem: Find all words that start with „Peer“ is not effient in DHTs
  - Other options instead of hash function
    - Use DHT without hashing, but with load balancing (→ next chapter)
    - Ordered Indexing
- Ordered Indexing
  - Build an efficient structure without hashing
- Trie
  - reTRIEval tree

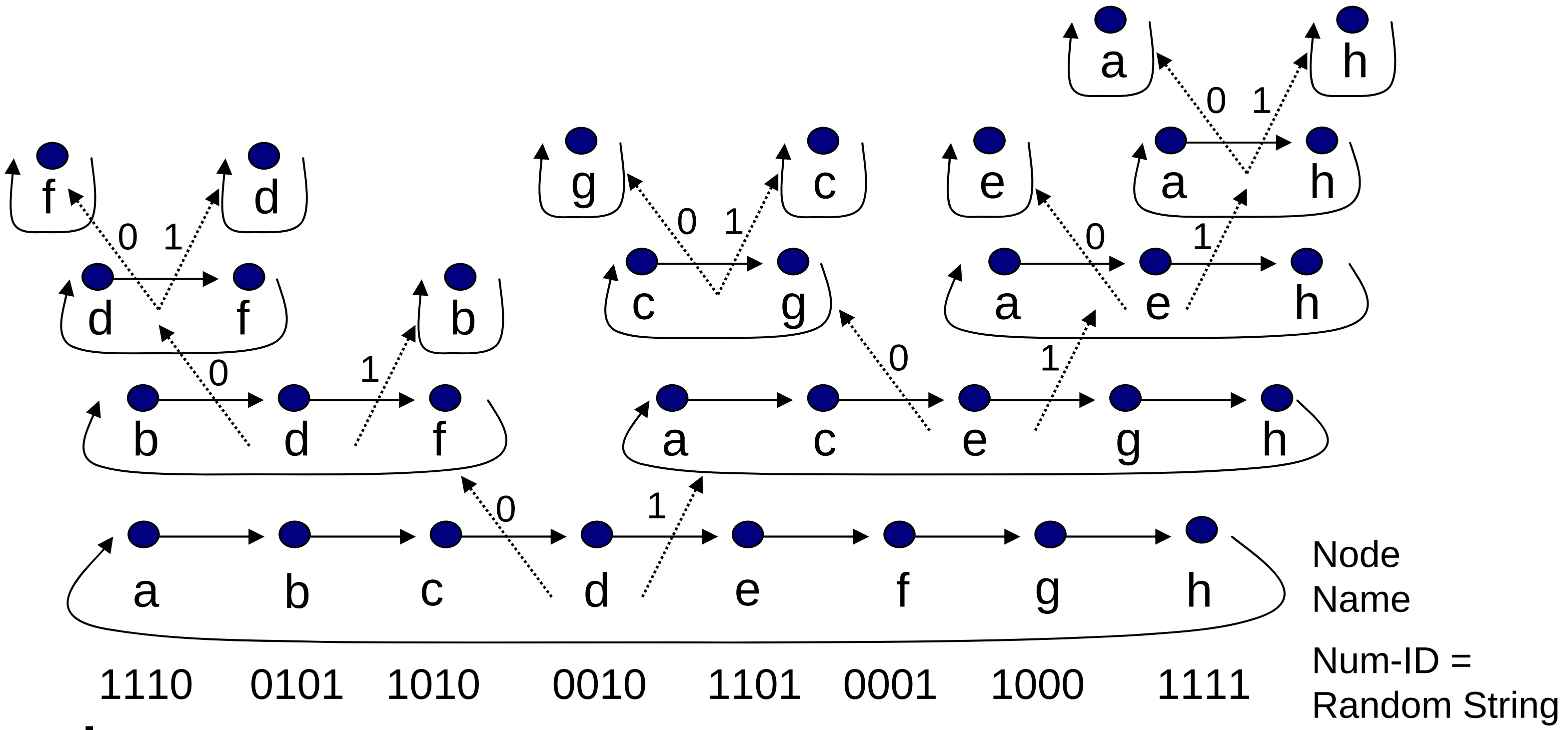
Skip List



Skip List

- ❑ A linear list is inefficient → add „long-distance“ links
- ❑ Idea
  - Add layers as long as there are more than 2 nodes per layer.
  - A n-layer consists of a randomly selected subset of the (n-1)-layer.
    - The random string for a peer corresponds to this random selction process.
      - 1 means „part of this layer“
      - 0 means „not part of any further layer“
- ❑ Achieves O(log n) hops with O(log n) in- and out-degree.

Skip Graph



Skip Graph

- Adapts the skip list idea to Peer-to-Peer networks.
- Idea
  - Layer 0 is a circle with all nodes.
  - Recursion: Split (n-1)-layer nodes into two random sets according to the bit of random string at position n. Each set forms again a circle.

Skip Graph

Node Identities

- Name: arbitrary name of node (item), e.g. tum.i8.heiko
- Num-ID: random number for each node

Search for Name

- Next hop selection
  - Start with the highest layer.
  - IF the next hop is closer to the name and still before the name in the order of the names (e.g. alphabetical)
  - ELSE Check lower layer for next hop. ENDIF

Search for Num-ID

- Start search on lowest layer for node with correct next bit, then go to next higher layer.

Results

- Both search operations take O(log n).
- Skip Graphs support range queries (e.g. all names from c to e).

Example (graph on last slide):
A looks for F
→ next hop: E
E looks for F
→ next hop: F

Example:
A looks for 0000
→ next hop: B (0101)
B looks for 0000
→ next hop: D (0010)
D looks for 0000
→ next hop: F (0001)
(closest match)