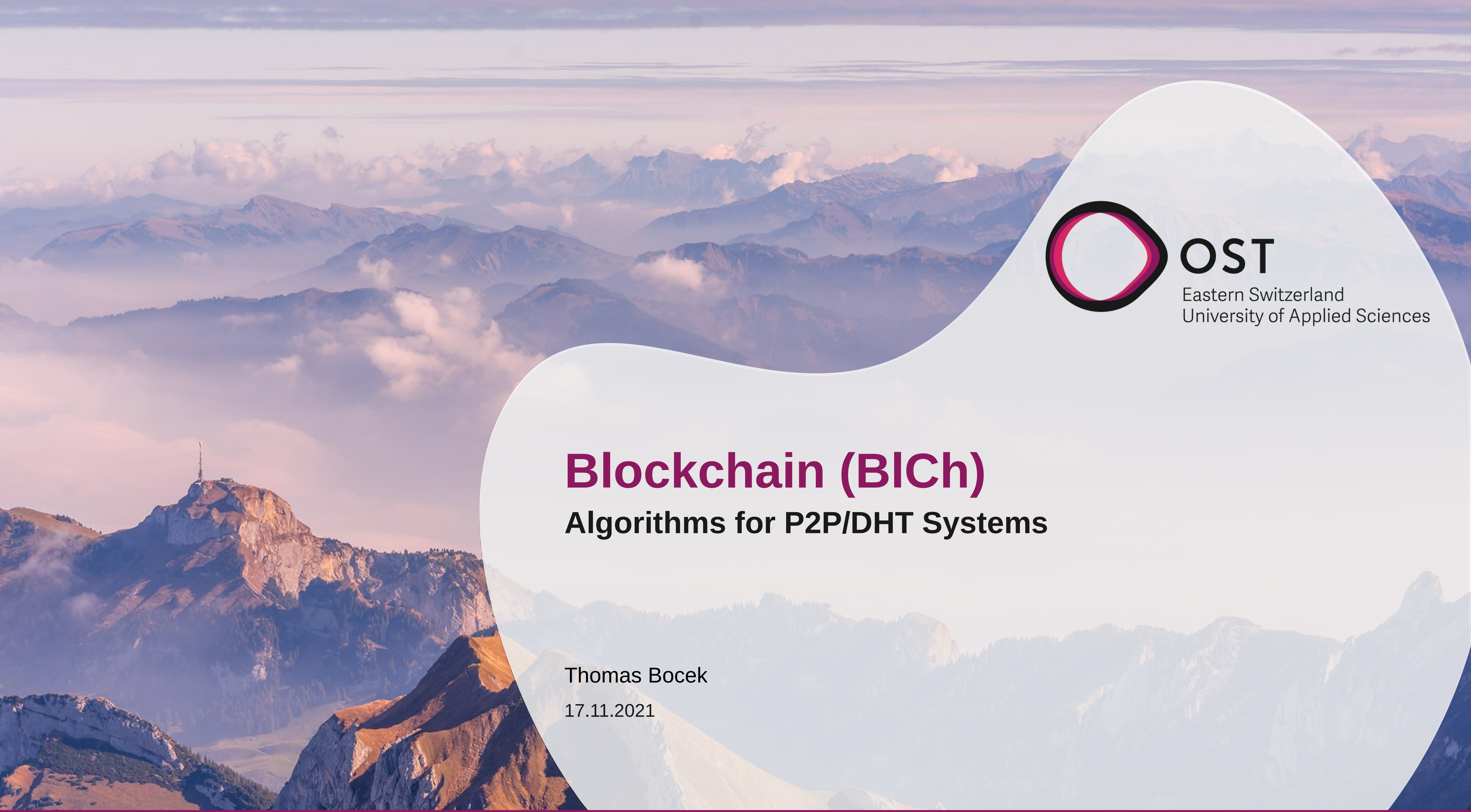


Blockchain (BlCh)

Algorithms for P2P/DHT Systems

Thomas Bocek

OST Eastern Switzerland University of Applied Sciences

17.11.2021

Mechanisms based on Hashing in KV storage

- Search in DHTs / consistent hashing
  - DHT.get(h(«Institut für Software»))
  - In order to find it: DHT.put(h(«Institut für Software»), value)
- Keywords
  - DHT.get(h(«Institut»))
  - Find it: DHT.put(h(«Institut»), value), DHT.put(h(«für»), value), DHT.put(h(«Software»), value)
  - value points to h(«Institut für Software»)
- Keywords drawbacks
  - Find good keywords → “the”, “a” are not good keywords
  - Exact matches only

OST

Mechanisms based on Hashing in KV storage

- Find “Institut” or “Software” - OR Systems
  - DHT.get(h(«Institut»)) and DHT.get(h(«Software»)), combine results
- Find “Institut” and “Software” - AND Systems

1) DHT.get(h(«Institut»)) and DHT.get(h(«Software»)), intersect results

2) DHT.get(h(«Institut») xor h(«Software»))

- In order to find it:
  - DHT.put(h(«Institut») xor h(«Software»), value),
  - DHT.put(h(«Institut») xor h(«für»), value)
  - DHT.put(h(«für») xor h(«Software»), value)
- Combination needs to be known in advance

3) Use Bloom Filters

- bf = DHT.getBF(h(«Institut»)) and DHT.get(h(«Software», bf))
- Sequential (less network, slower) vs. parallel (more network, faster)

Mechanisms based on Hashing in KV storage

- Similarity Search in DHT
  - https://fastss.csg.uzh.ch
- Project that brings similarity search to HT / DHT
  - Problem: Search for “netwrk” fails for DHTs
- Similarity: Edit distance / Levenshtein distance
  - Min operations to transform one string into another, operations: insert, delete, replace
  - Calculated in matrix size O(m x n)

$$
\begin{aligned}
&d[i,0] = i,\ d[0,j] = j,\\
&d[i,j] = min\,(d[i-1,j]+1,\ d[i,j-1]+1,\\
&\quad d[i-1,j-1] + (if\ s1[i] = s2[j]\ then\ 0\ else\ 1))
\end{aligned}
$$

OST

Mechanisms based on Hashing in KV storage

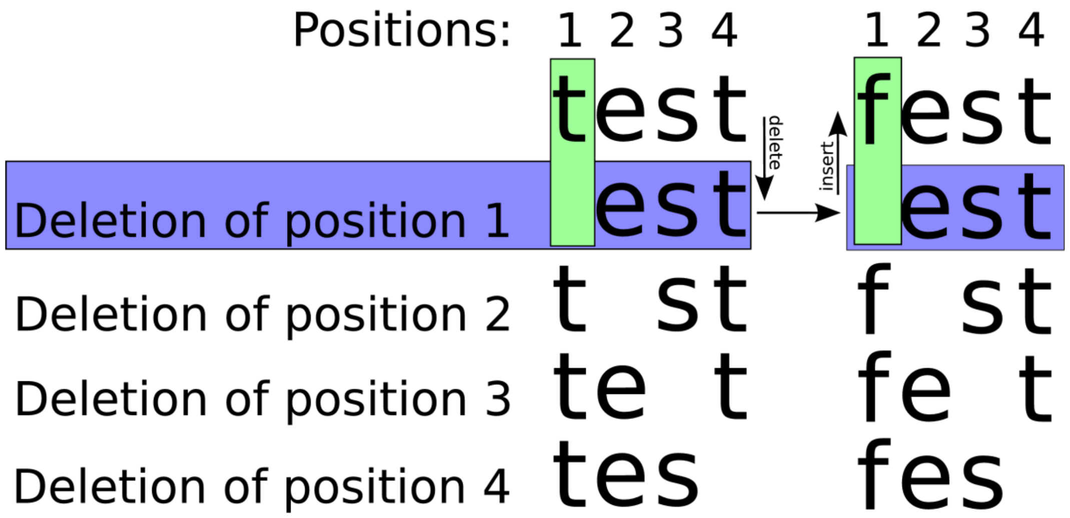
- Example d(test,east) = 2 (remove a, insert t)
- Expensive operation if all words need testing
- Main idea: pre-calculate errors
  - All possible errors? Neighbors for test with ed 2: test, testa, testaa, testab, ... , tea, teb, tec, ..., teaa, teab, ... → 23883 more of those!

		T	E	S	T
	0	1	2	3	4
E	1	1	1	2	3
A	2	2	2	2	3
S	3	3	3	2	3
T	4	3	4	3	2

$$d[i,0] = i,\ d[0,j] = j,$$
$$d[i,j] = min\ (d[i-1,j]+1,\ d[i,j-1]+1,$$
$$d[i-1,j-1] + (if\ s1[i] = s2[j]\ then\ 0\ else\ 1))$$

Mechanisms based on Hashing in KV storage

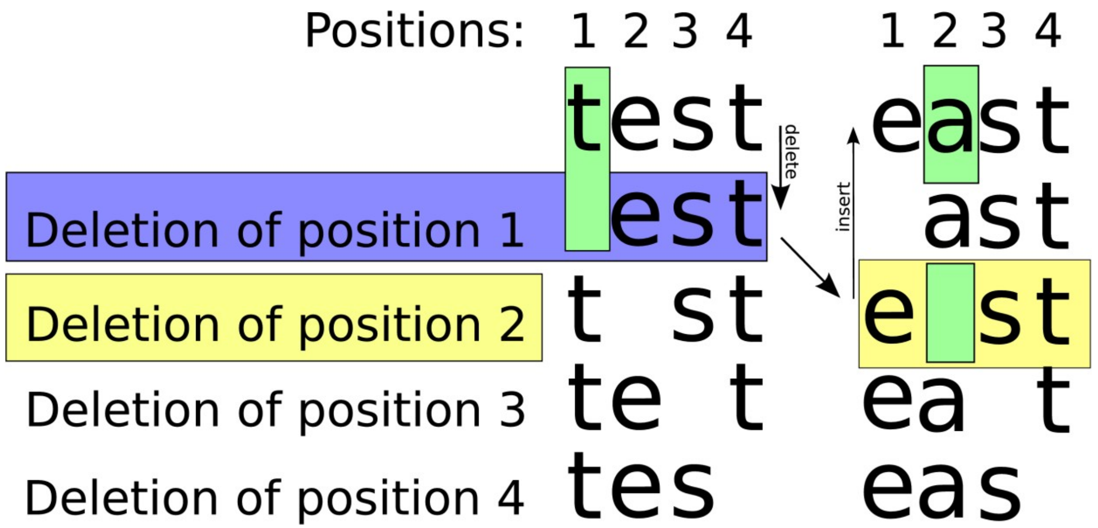
- FastSS pre-calculates with deletions only
  - Neighbors for test with ed 2: test, est, st, et, es, tst, tt, ts, tet, te, tes
  - Pre-calculation on query and index
  - 11 neighbors → 11 more queries, indexed enlarged by 11 entries
- Example d(test,fest)=1
  - test: indexed
  - fest: query



OST

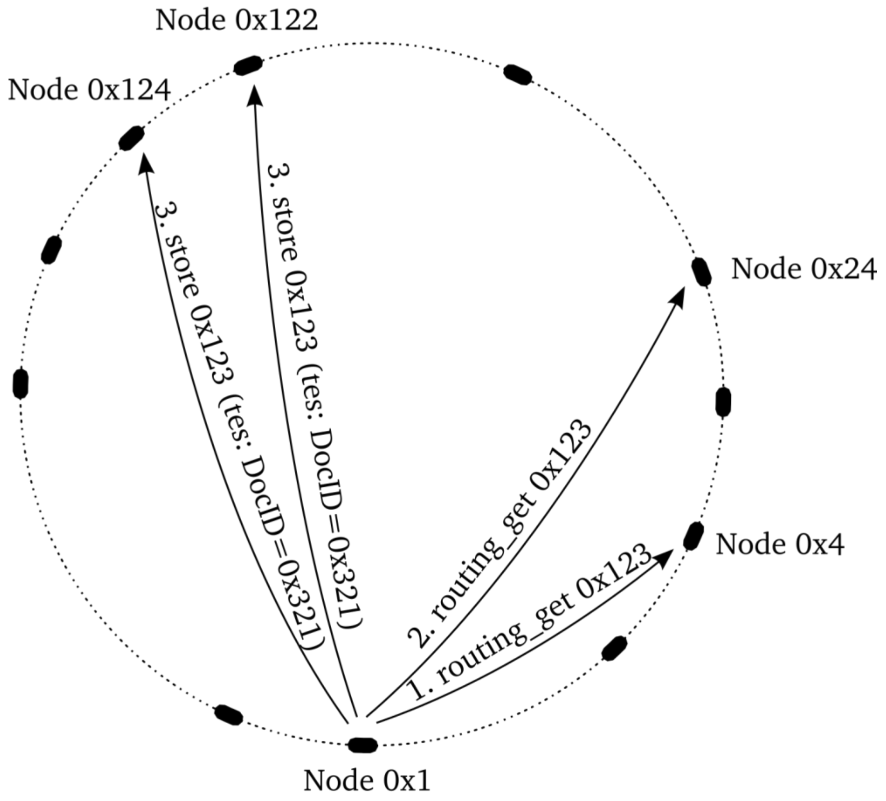
Mechanisms based on Hashing in KV storage

- Example d(test,east)=2
  - test: indexed
  - east: query
- FastSS with indexing Wikipedia documents in systems with consistent hashing



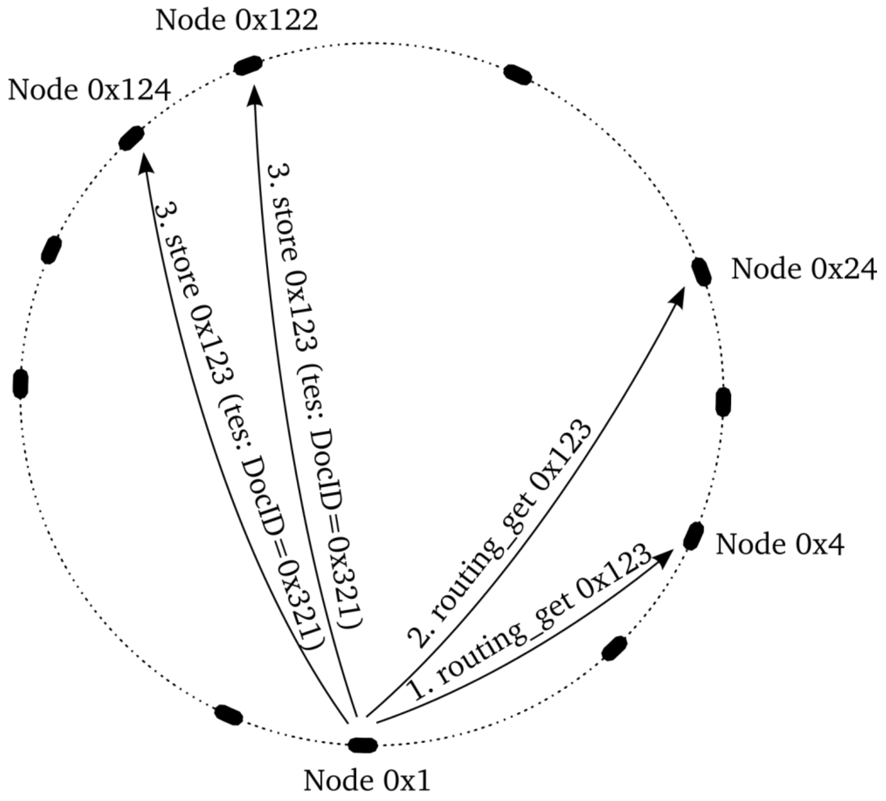
Mechanisms based on Hashing in KV storage

- Index documents using put(hash(document), document)
  - Document (0x321) contains word test
- Index all neighbors (test, tes, tst, tet, est) using put(hash(neighbor), point to document)
  - hash("tes") = 0x123



Mechanisms based on Hashing in KV storage

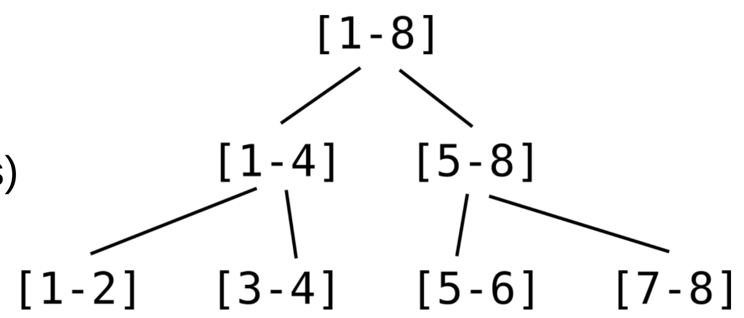
- User searches for “tesx”
- Neighbors are generated (tesx, esx, tsx, tex, tes)
  - get(hash(neighbor)) → 0x123
  - Find pointer to document (0x321)
  - document = get(0x321)
- Tests with edit distance 1, partially 2, ignoring delete pos.
  - Overhead (n choose k) for query and index
- Similarity search as series of put() and get()



OST

Mechanisms based on Hashing in KV storage

- Range Queries
  - Problem: random insert vs. sequence insert
  - Sequence → [0..n-1] [n..2n-1] [2n..3n-1] […] → peer responsible for range, hash it, store it, done.
    - Insert 10 items: N = 5 → [0, 1, 2, 3, 4], [5, 6, 7, 8, 9] – sequential, 2 peers
    - Insert 10 items: N = 5 → [0], [5], [10], [15], [20], [25], [30], [35], [40], [45] – random, 10 peers
    - But random: worst case: 1 peers has 1 data item, range query for range [0..x] contacts x/n peers.
- Over-DHT
  - PHT: trie (prefix tree); DST: segment → tree on top of DHT
  - Main idea: hash of tree-node (resp. for range) → DHT
  - PHT: Peer stores n data items, if n reached, splits data (moves data across peers)
  - DST: stores data on each level (redundancy) up to a threshold
    - No data splitting

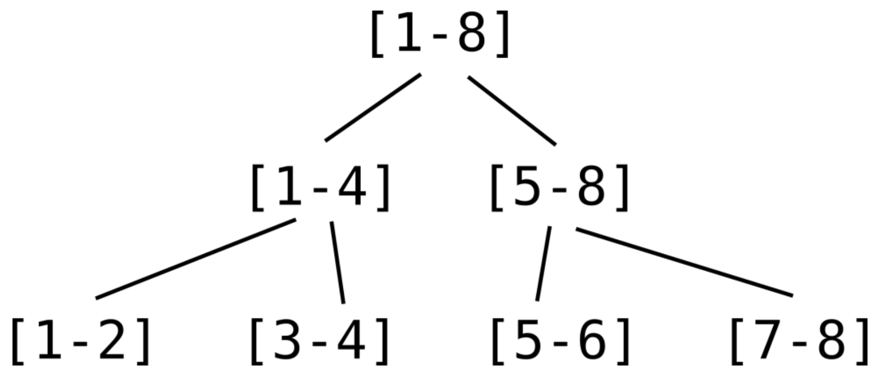


OST

Mechanisms based on Hashing in KV storage

- Example:
  - Set n = 2, m=8
  - 1, “test”; 2, “hallo”;
    3, “world”; 5, “sys”; 6, “ost”; 7, “ifs”
- Tree: store value
  - Translate putDST(1, “test”) to
    - put(hash([1-8]),”test”) → may be stored (only if threshold not reached)
    - put(hash([1-4]),”test”) → may be stored
    - put(hash([1-2]),”test”) → will be stored
    - Store put(2, “hallo”), put(3, “world”), put(5, “sys”), …

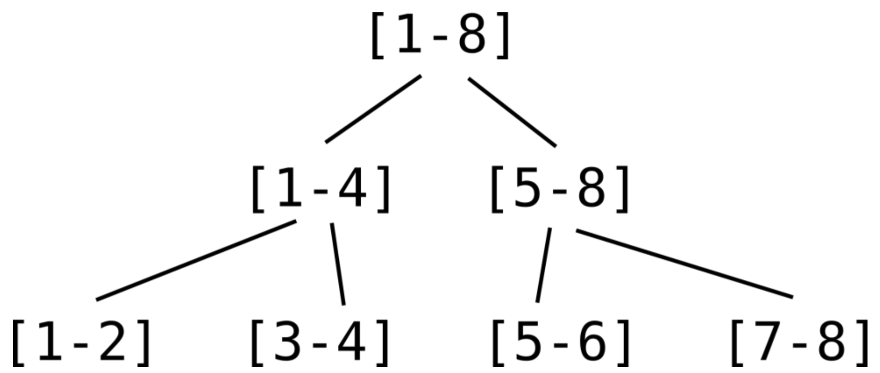
- Query `getDST(1..5)` translates to
  - `get(hash[1-8])` → returns “1,test; 2,hallo”
  - `get(hash[1-4])` → returns “1,test; 2,hallo”
  - `get(hash[1-2])` → returns “1,test; 2,hallo”
  - `get(hash[3-4])` → returns “3,world”
  - `get(hash[5-8])` → returns “5,sys; 6,ost”
  - `get(hash[5-6])` → returns “5,sys; 6,ost”



Mechanisms based on Hashing in KV storage

- Example:
  - Set n = 2, m=8
  - 1, “test”; 7, “ifs”
- Tree: store value
  - Translate putDST(1, “test”) to
    - put(hash([1-8]),”test”) → may be stored (only if threshold not reached)
    - put(hash([1-4]),”test”) → may be stored
    - put(hash([1-2]),”test”) → will be stored
    - Store put(7, “ifs”)

- Query `getDST(1..5)` translates to
  - `get(hash[1-8])` → returns “1,test; 7,ifs”
  - `get(hash[1-4])` → returns “1,test;”
  - `get(hash[5-8])` → returns “7,ifs”
- Range query as series of `put()` and `get()`



OST