Building Scalable, Hierarchical ROOFs using Distributed Hash Tables

Rohit Sardesai and Neeraj Kumar
Huawei Technologies India Pvt. Ltd.
Bangalore

Why Edge Computing ?

Increasing costs of shipping large volumes of data
to the cloud for processing and storage.

Reduce Cost

Trust & Security

Data security is critical and hence orgs prefer local data storage.

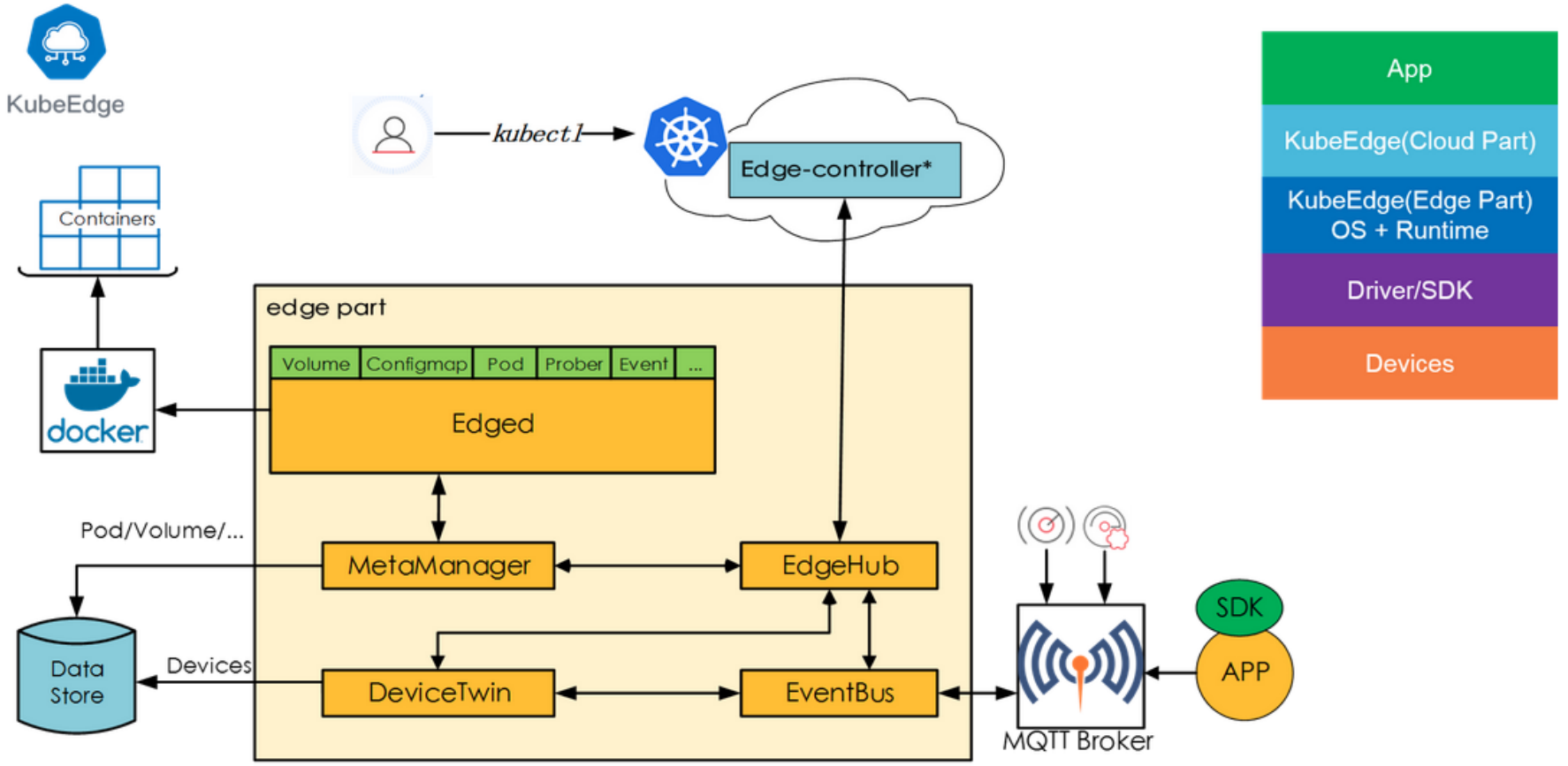
Real time decision making is the key !
Data transmission to cloud increases latency and prohibits acting in real-time**.**

Real time ,
Ultra Low Latency

Offline,
Independent

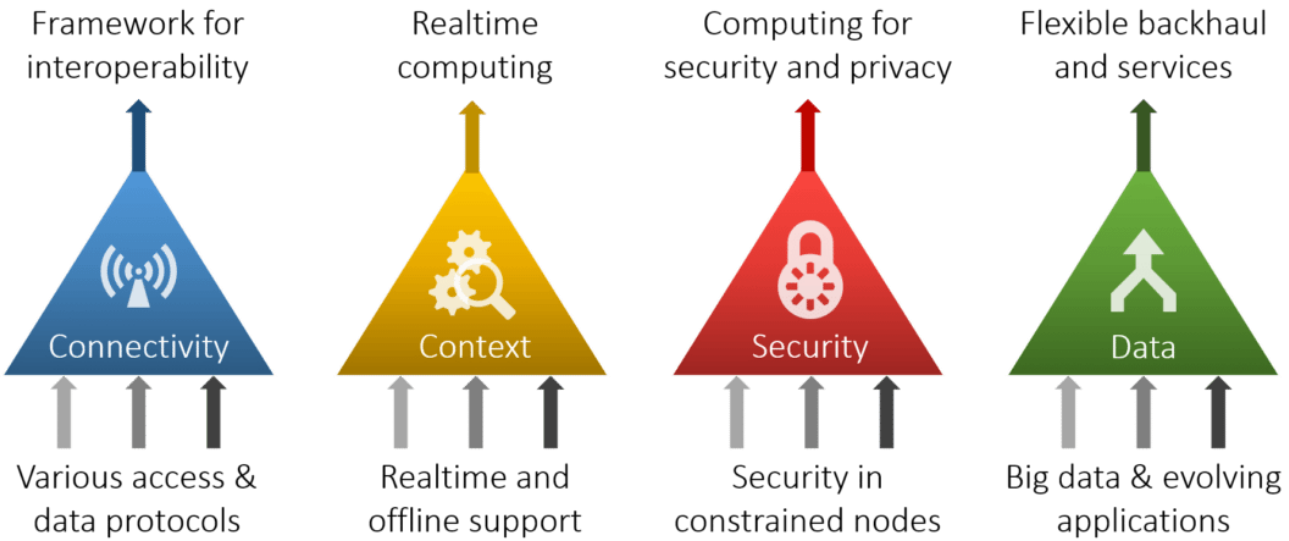
No cloud connectivity is common in IoT environments , but mission-critical IoT applications like connected vehicles need to function without this !

KubeEdge – A Kubernetes Native Edge Cloud Computing Framework



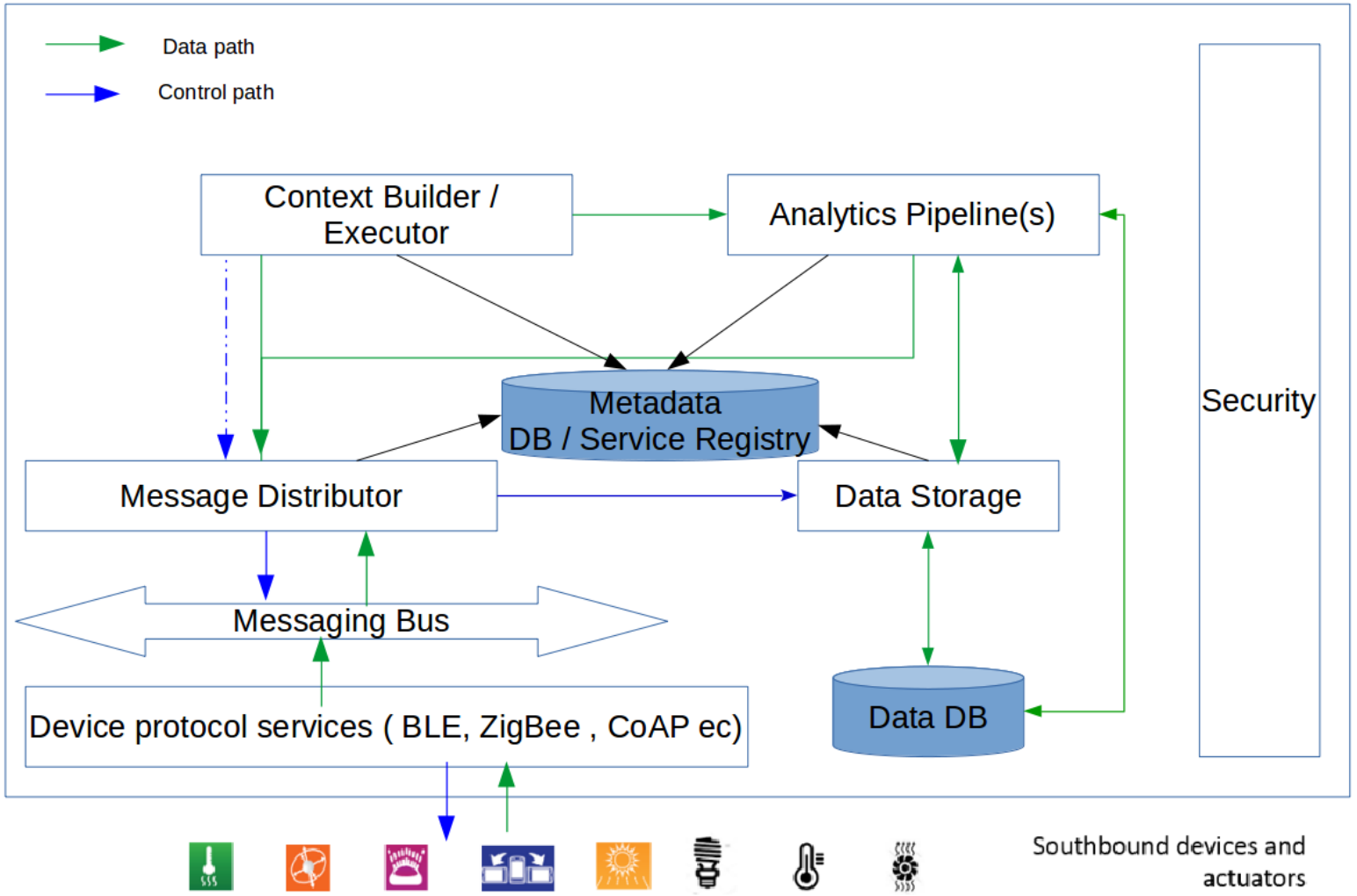
https://github.com/kubeedge/kubeedge/

IEEE P1931.1: ROOF COMPUTING



"A Standard for an Architectural Framework for Real-time Onsite Operations Facilication (ROOF) for the Internet of Things"

ROOF Microservices Platform



Blockchain in ROOF

It seems natural to use blockchain in ROOF as a validation mechanism for secure device provisioning and management, however there are a few challenges in adapting blockchain to an IoT infrastructure.

- Limited compute power and memory on the ‘things’ preclude replicating and validating against a universal ledger necessary for non-repudiation.
- An action/transaction performed by an agent (device) under certain context may be constrained by SLA/QoS and should not wait for consensus related delays.

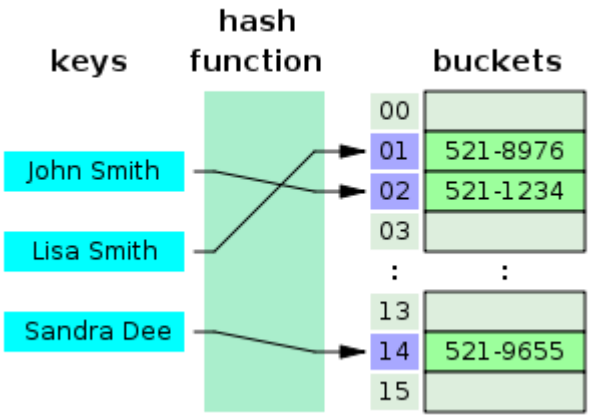
Beyond Blockchain – Distributed Hash Tables

The idea of hashing is to distribute the entries (key/value pairs) across an array of *buckets*. Given a key, the algorithm computes an *index* that suggests where the entry can be found:

index = f(key, array_size)

Often this is done in two steps:

hash = hashfunc(key)
index = hash % array_size



A distributed hash table does this in a distributed setting (nodes are buckets) and has following performance concerns:

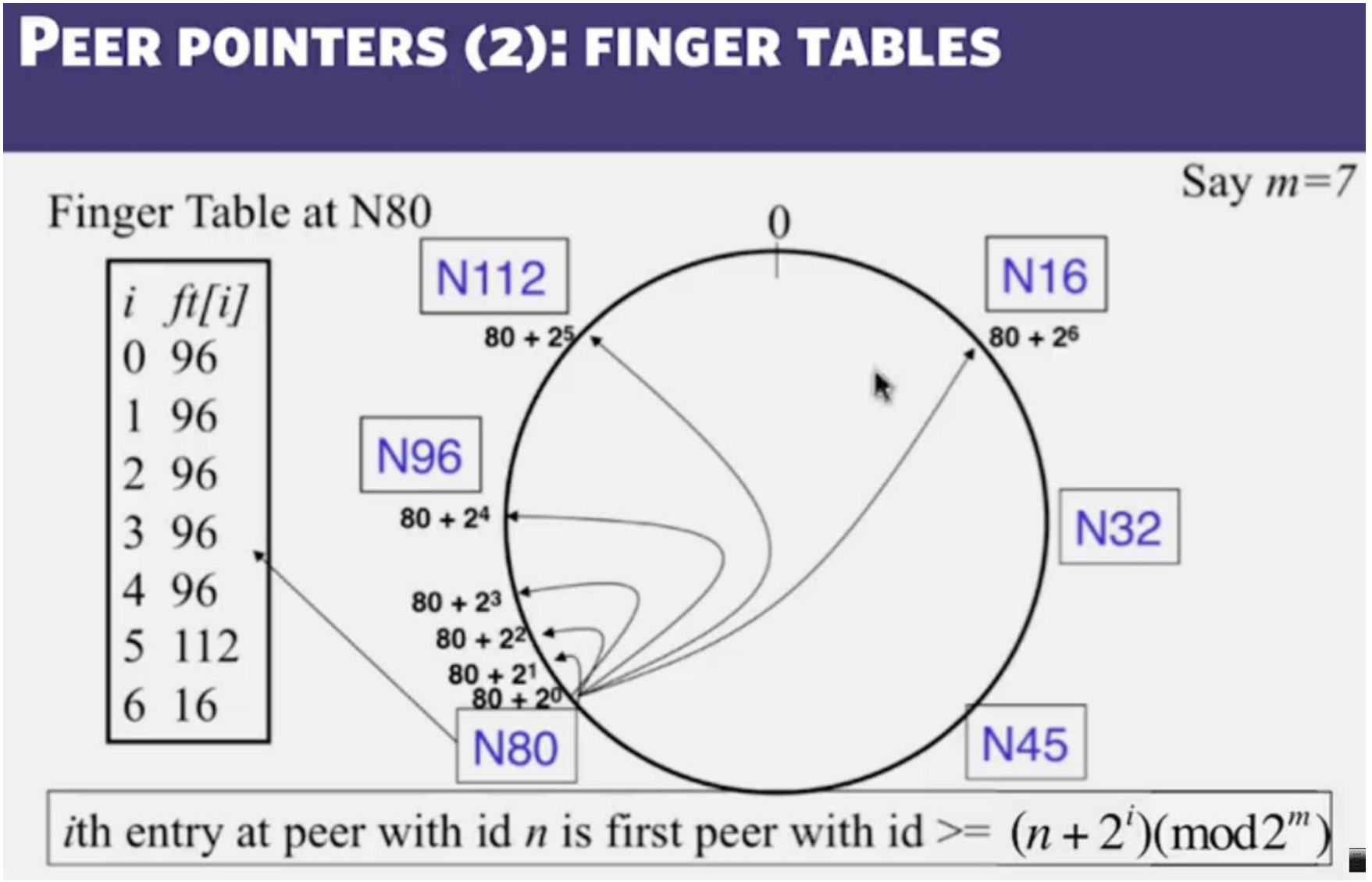
- Load balancing (nodes are uniformly loaded)
- Fault-tolerance (nodes might fail or leave the s/m, data should not be lost)
- Efficiency of lookups and inserts – O(log(N))
- Locality (communicating nodes should preferably closer to one another in the underlying n/w topology)

Beyond Blockchain – Distributed Hash Tables

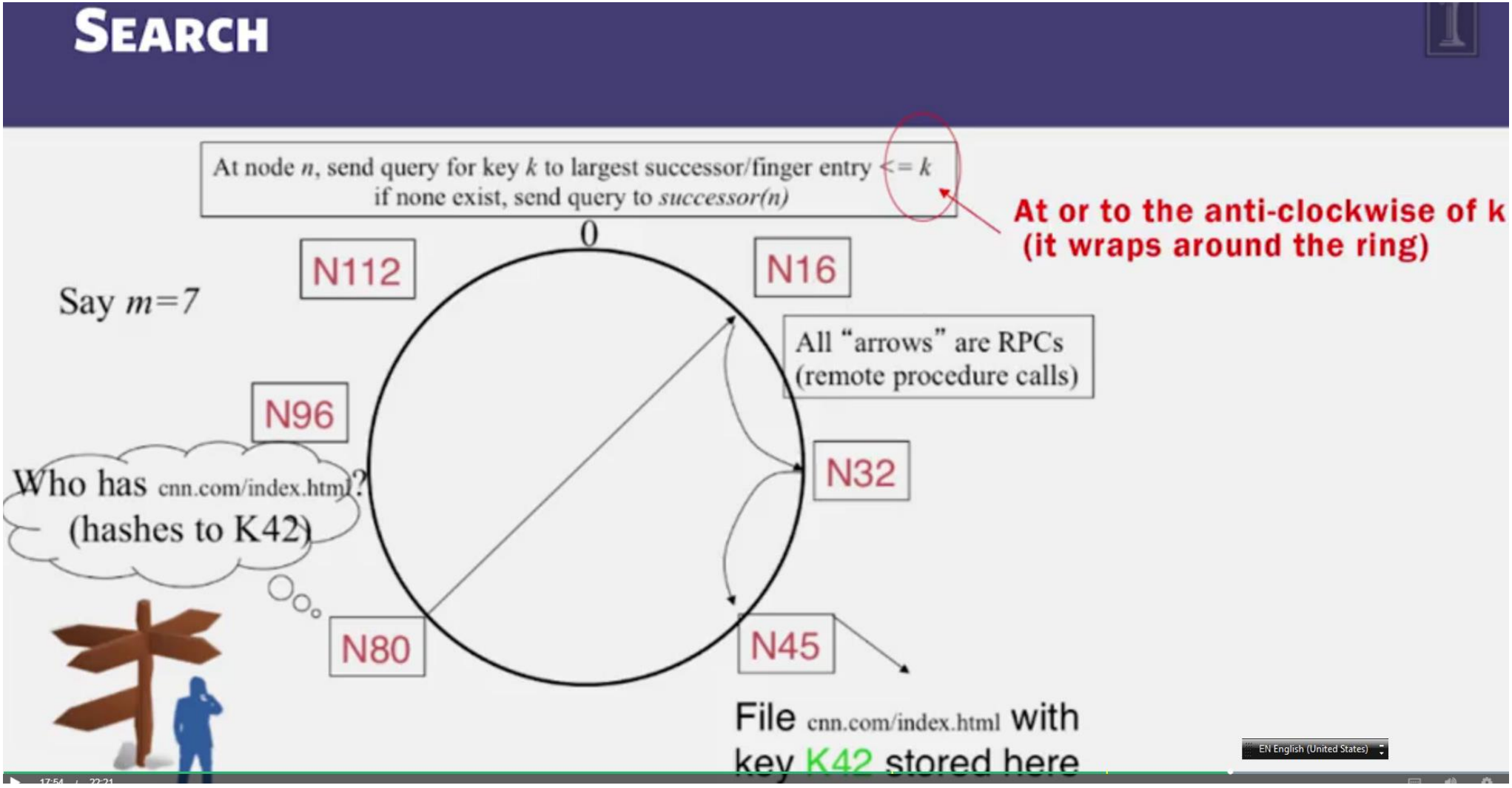
CHORD

- Developers: I. Stoica, D. Karger, F. Kaashoek, H. Balakrishnan, R. Morris, Berkeley and MIT
- Intelligent choice of neighbors to reduce latency and message cost of routing (lookups/inserts)
- Uses *Consistent Hashing* on node's (peer's) address
  - SHA-1(ip_address,port) → 160 bit string
  - Truncated to m bits
  - Called peer *id* (number between 0 and $2^m - 1$)
  - Not unique but id conflicts very unlikely
  - Can then map peers to one of 2^m logical points on a circle

Beyond Blockchain – Distributed Hash Tables



Beyond Blockchain – Distributed Hash Tables



Beyond Blockchain – Distributed Hash Tables

LDHT: Locality-aware Distributed Hash Tables*

Weiyu Wu[#1], Yang Chen[#], Xinyi Zhang[*], Xiaohui Shi[#], Lin Cong[#], Beixing Deng[#], Xing Li[#]

#Department of Electronic Engineering, Tsinghua University, China

**Department of Electrical Engineering, University of California, Los Angeles, USA*

[1]wuwy02@mails.tsinghua.edu.cn

Abstract **– As the substrate of structured peer-to-peer systems, Distributed Hash Table (DHT) plays a key role in P2P routing infrastructures. Traditional DHT does not consider the location of the nodes for the assignment of identifiers, which will result in high end-to-end latency on DHT-based overlay networks. In this paper, we propose a design of locality-aware DHT called LDHT, which exploits network locality on DHT-based systems. Instead of assigning uniform random node identifiers in traditional DHT, nodes in LDHT are assigned locality-aware identifiers according to their Autonomous System Numbers (ASNs). As a result, each node will have more nearby neighbors than faraway neighbors in the overlay. We evaluate the performance of LDHT on different kinds of typical DHT protocols and on various topologies. The results show that LDHT improves the traditional DHT protocols a lot in terms of end-to-end latency, without introducing additional overhead. It is indicated that LDHT is fit for different kinds of DHT protocols and can work effectively on all structured P2P systems including Chord, Symphony and Kademlia.**

locality in DHT-based systems. We assign the node identifiers in a geographic layout manner to ensure nodes close in the network topology to be close in the identifier space. We use a node's ASN to generate the prefix of the identifier in order to make nodes in a same AS have close identifiers. As a result, nodes in LDHT-based systems will have more close neighbors than faraway neighbors in the network topology. The end-to-end latency for the query on the overlay network will thus be reduced. We use three typical DHT-based systems, Chord [1], Symphony [2] and Kademlia [3] as the basic DHT protocols to evaluate our design. According to the simulation results on different topologies, it is indicated that LDHT can improve the performance of DHT-based systems on both path length and Relative Delay Penalty (RDP) significantly, without adding overlay hops.

The rest of this paper is organized as follows. First we review related work in Section II. Then we present the design

W. Wu *et al.*, "LDHT: Locality-aware Distributed Hash Tables," *2008 International Conference on Information Networking*, Busan, 2008, pp. 1-5.
doi: 10.1109/ICOIN.2008.4472811

Beyond Blockchain – Distributed Hash Tables

Heterogeneity and Load Balance in Distributed Hash Tables

P. Brighten Godfrey and Ion Stoica
Computer Science Division, University of California, Berkeley
{pbg,istoica}@cs.berkeley.edu

***Abstract*— Existing solutions to balance load in DHTs incur a high overhead either in terms of routing state or in terms of load movement generated by nodes arriving or departing the system. In this paper, we propose a set of general techniques and use them to develop a protocol based on Chord, called Y_0, that achieves load balancing with minimal overhead under the typical assumption that the load is uniformly distributed in the identifier space. In particular, we prove that Y_0 can achieve near-optimal load balancing, while moving little load to maintain the balance and increasing the size of the routing tables by at most a constant factor.**

Using extensive simulations based on real-world and synthetic capacity distributions, we show that Y_0 reduces the load imbalance of Chord from $O(\log n)$ to a less than 3.6 without increasing the number of links that a node needs to maintain. In addition, we study the effect of heterogeneity on both DHTs, demonstrating significantly reduced average route length as node capacities become increasingly heterogeneous. For a real-word distribution of node capacities, the route length in Y_0 is asymptotically less than half the route length in the case of a homogeneous system.

imbalance to a constant factor. To handle heterogeneity, each node picks a number of virtual servers proportional to its capacity. Unfortunately, virtual servers incur a significant cost: a node with k virtual servers must maintain k sets of overlay links. Typically $k = \Theta(\log n)$, which leads to an asymptotic increase in overhead.

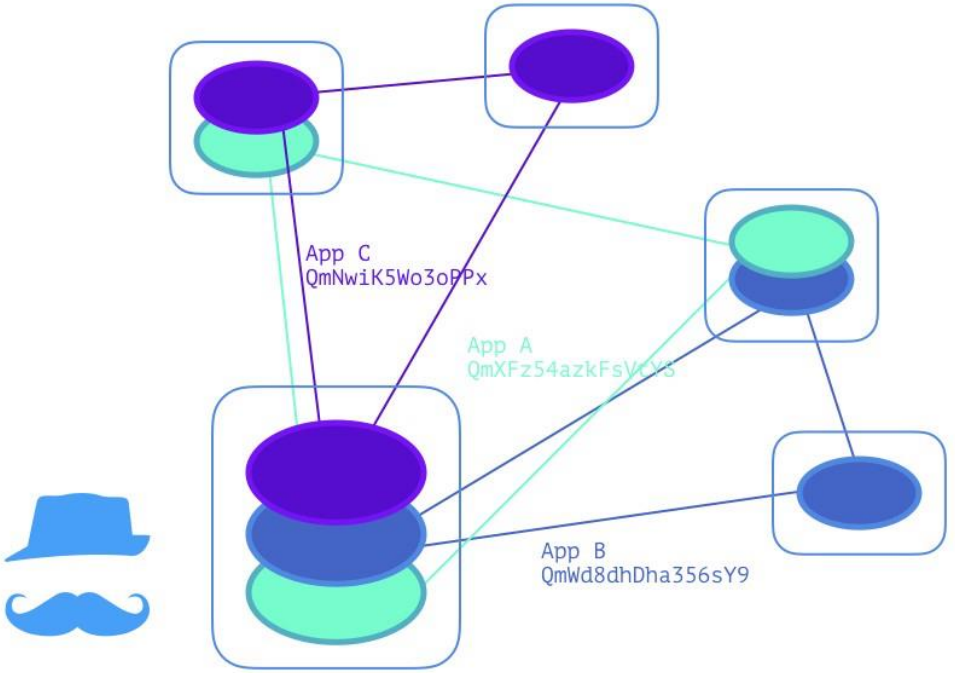
The second class of solutions uses just a single ID per node [26], [22], [20]. However, all such solutions must reassign IDs to maintain the load balance as nodes arrive and depart the system [26]. This can result in a high overhead because it involves transferring objects and updating overlay links. In addition, none of these solutions handles heterogeneity directly, although they could be combined with the virtual server technique.

In this paper, we present a simple DHT protocol, called Y_0, that addresses the above drawbacks. Y_0 is based on the concept of virtual servers, but with a twist: instead of picking

P. B. Godfrey and I. Stoica, "Heterogeneity and load balance in distributed hash tables," *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, Miami, FL, 2005, pp. 596-606 vol. 1.
doi: 10.1109/INFCOM.2005.1497926

Beyond Blockchain – Distributed Hash Tables

HOLOCHAIN



Each App can provide a context under which a device (agent) communicates with other devices using the same App. All under the same scalable (sharded), fault-tolerant (parameterized replication), and private (Hash) DHT overlay network.

This is much faster as there is no need for consensus. Holochain not only provides mechanisms to prevent malicious attacks, but also blocks that agent by broadcasting his credentials via gossip about the wrongdoing.

Thank you!