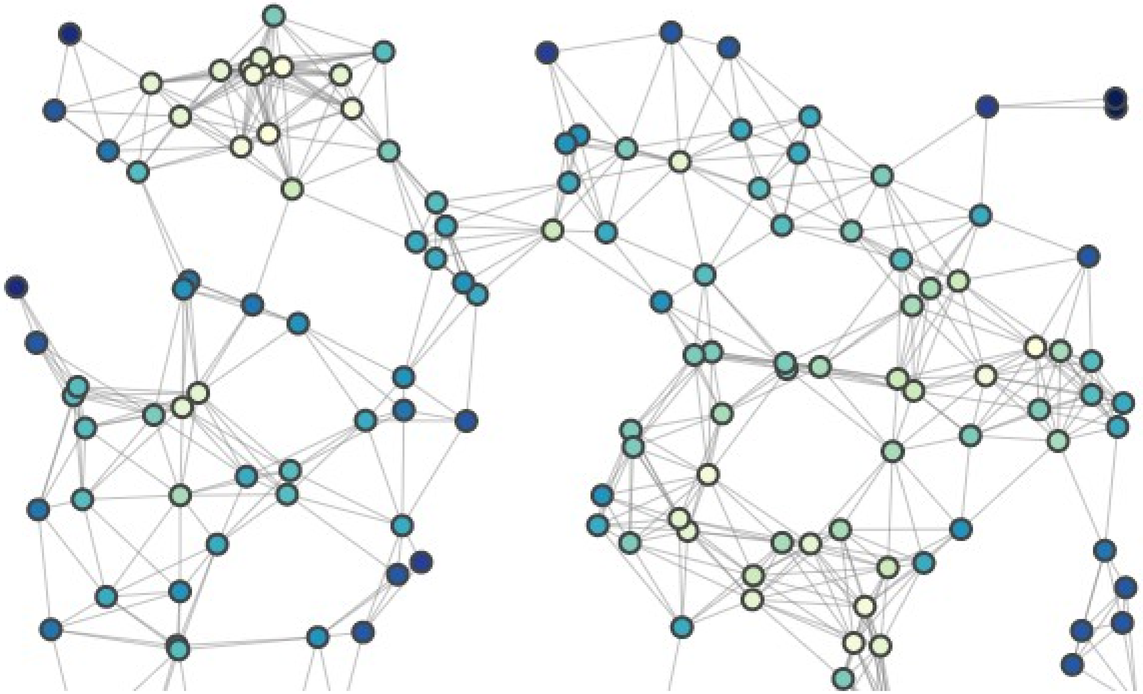
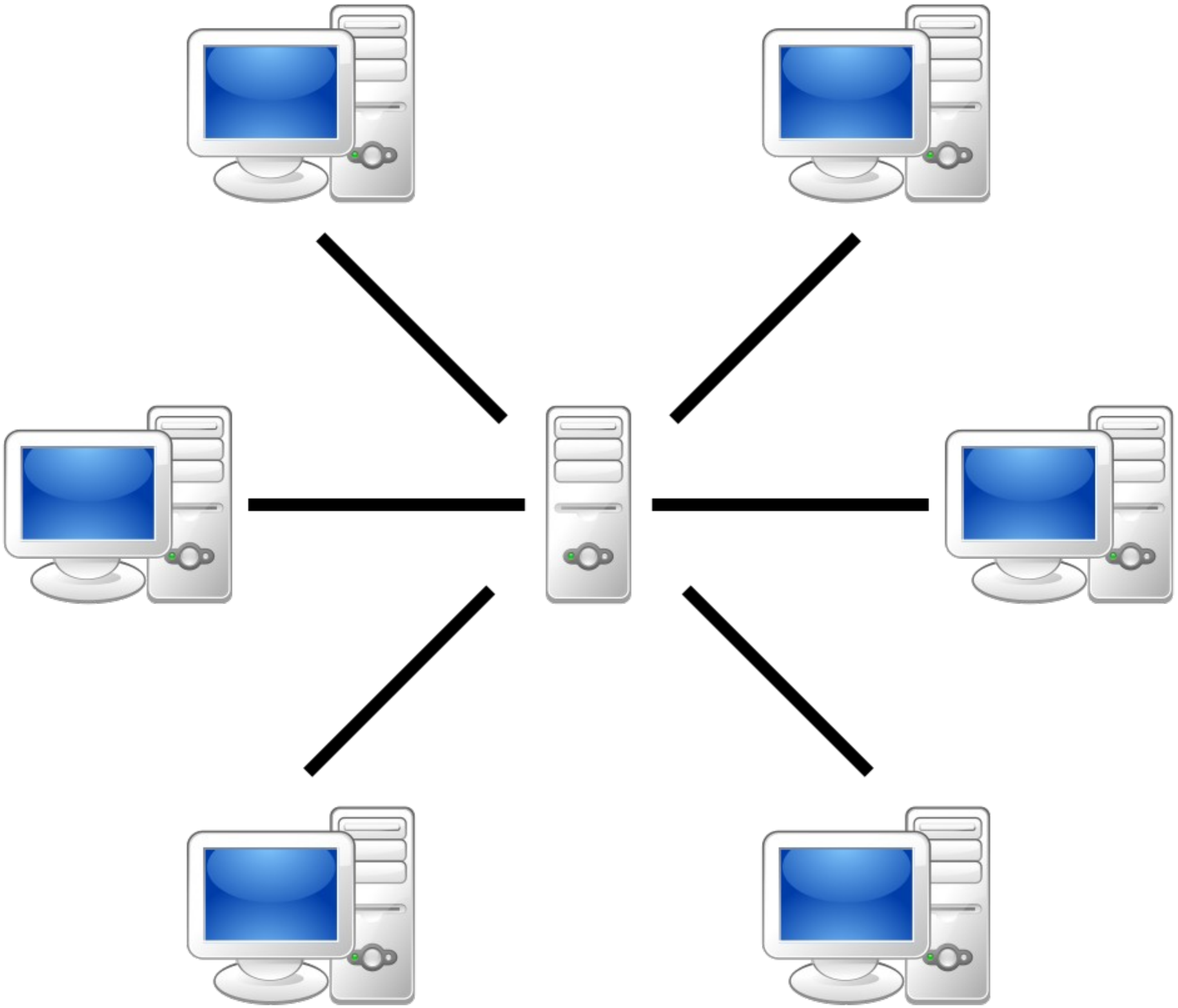


DHT

Distributed Hash Table

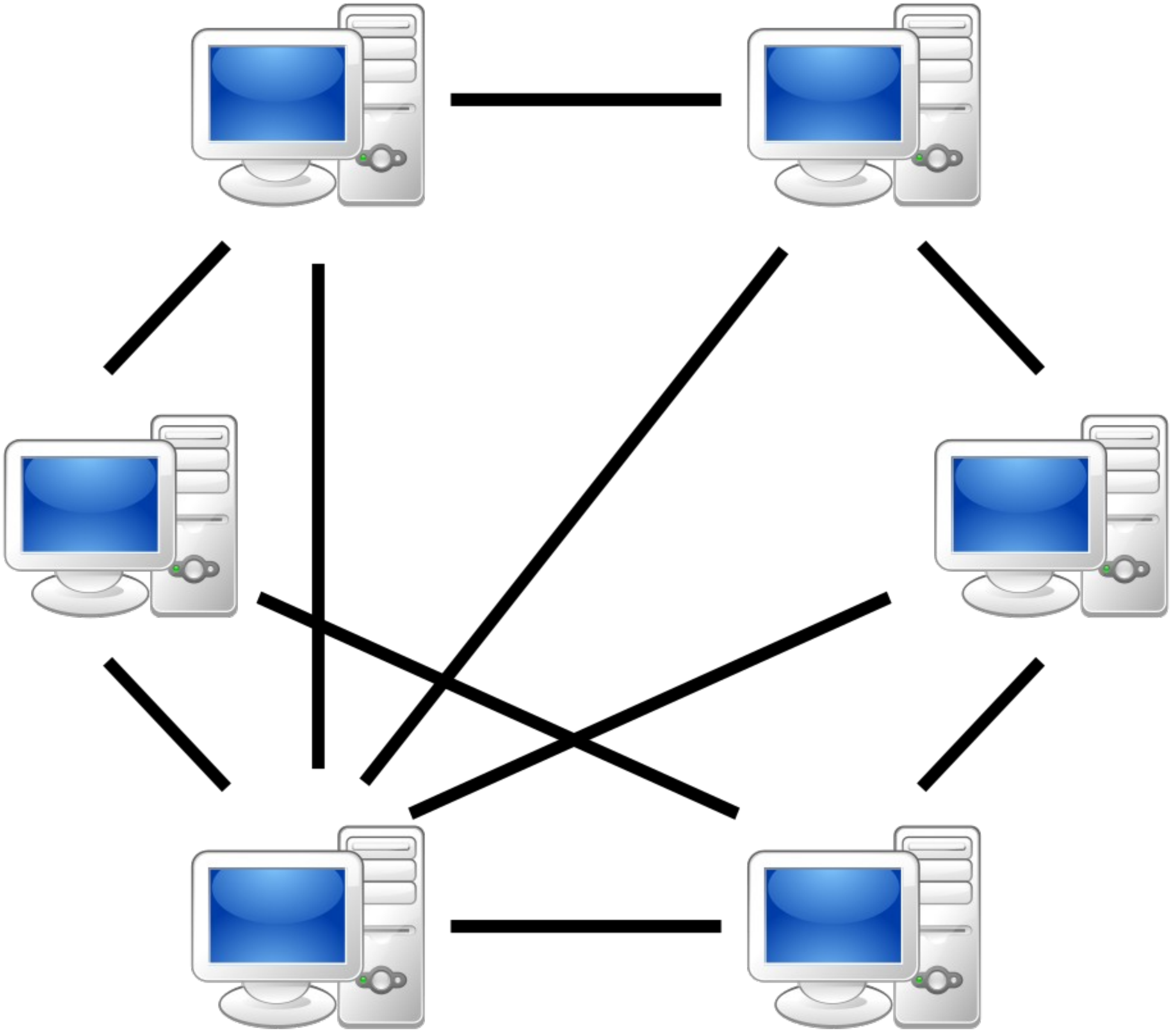
/

client-server



/

Peer-to-peer?



BitTorrent

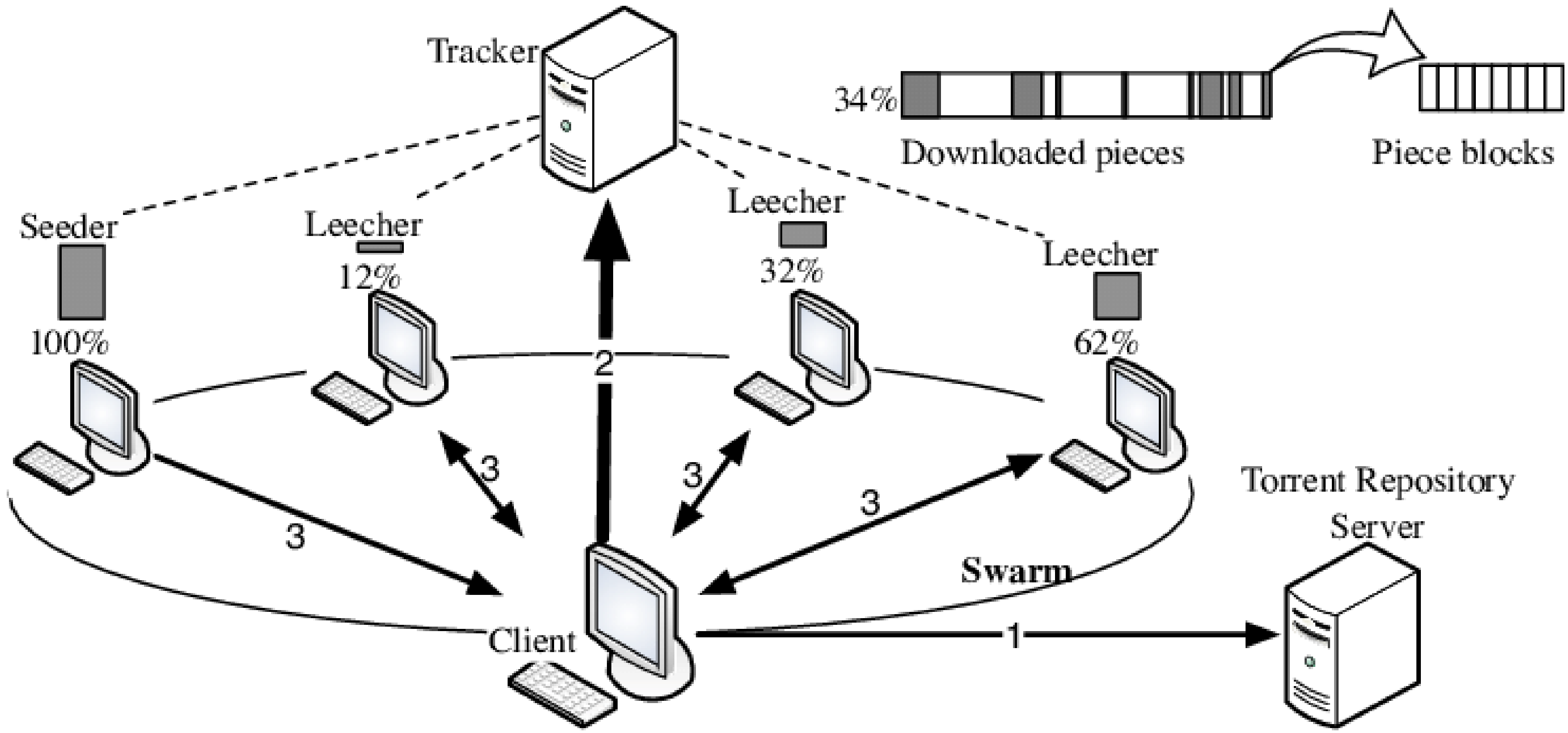
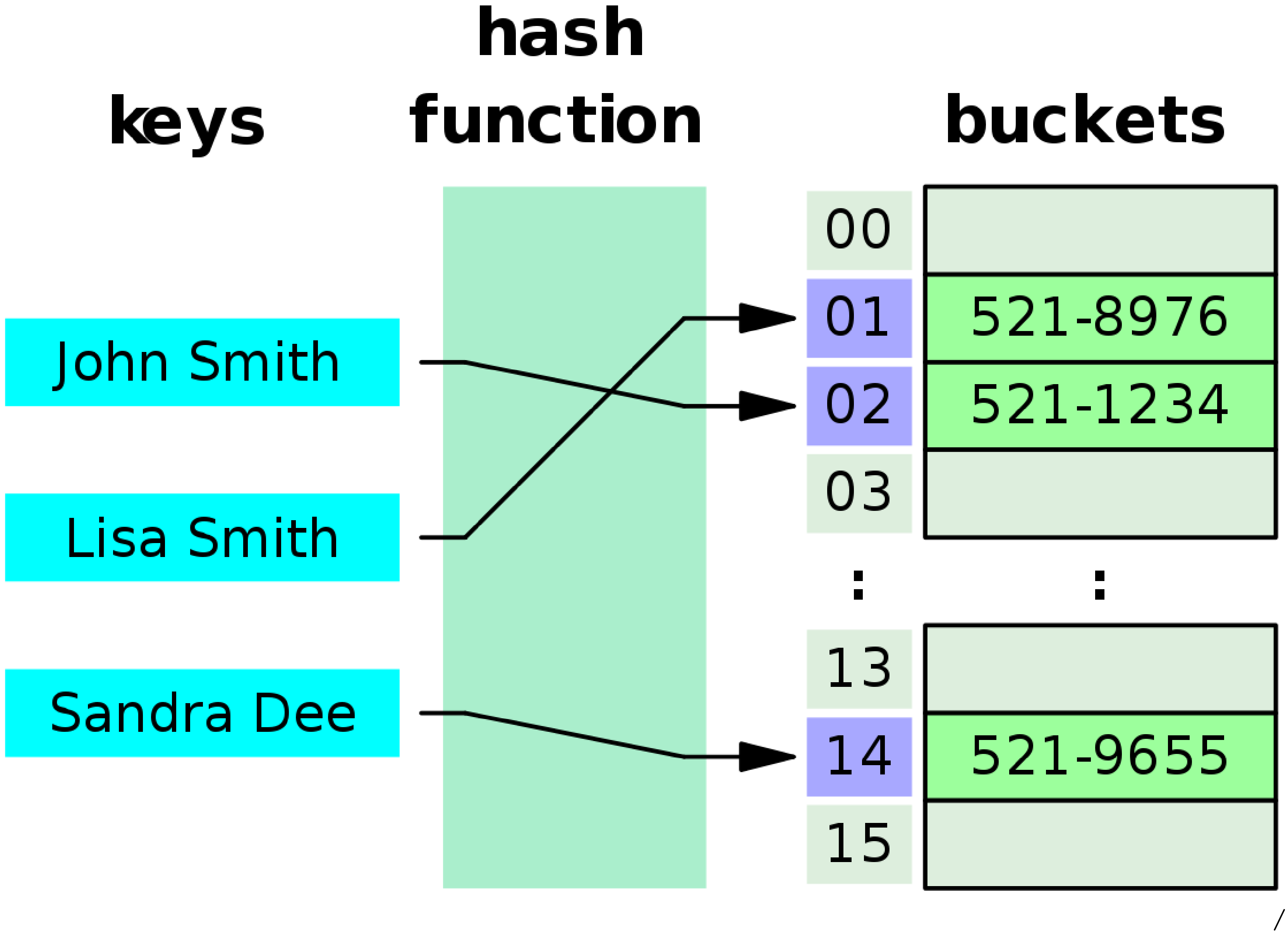


Fig: Evangelista, Pedro & Amaral, Marcelo & Miers, Charles & Goya, Walter & Simplicio, Marcos & Carvalho, Tereza & Souza, Victor. (2011). EbitSim: An Enhanced BitTorrent Simulation Using OMNeT++ 4. 437-440. 10.1109/MASCOTS.2011.46.

Solution? DistributedHashTable

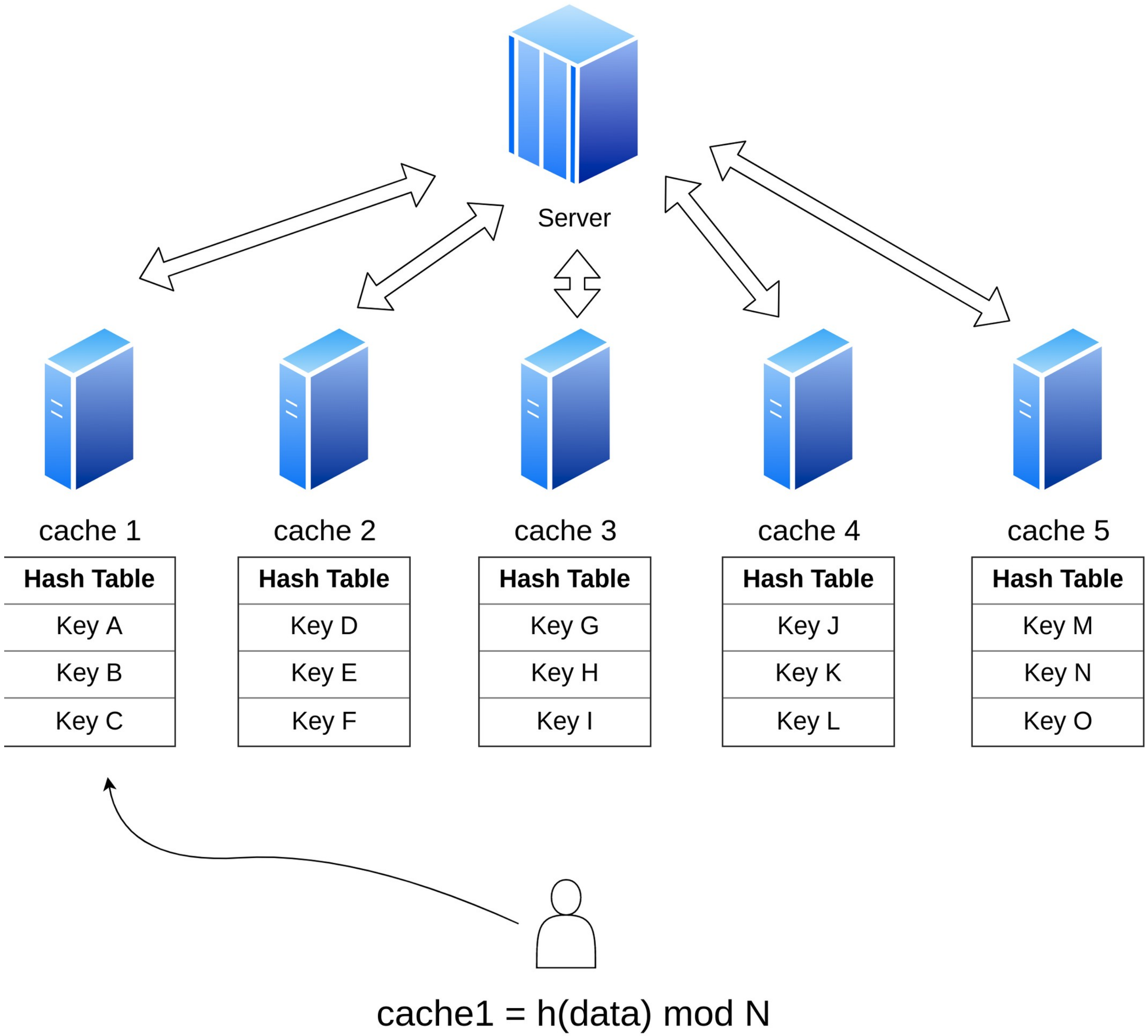
- High efficiency
- Decentralization
- Scalability

Hash Table

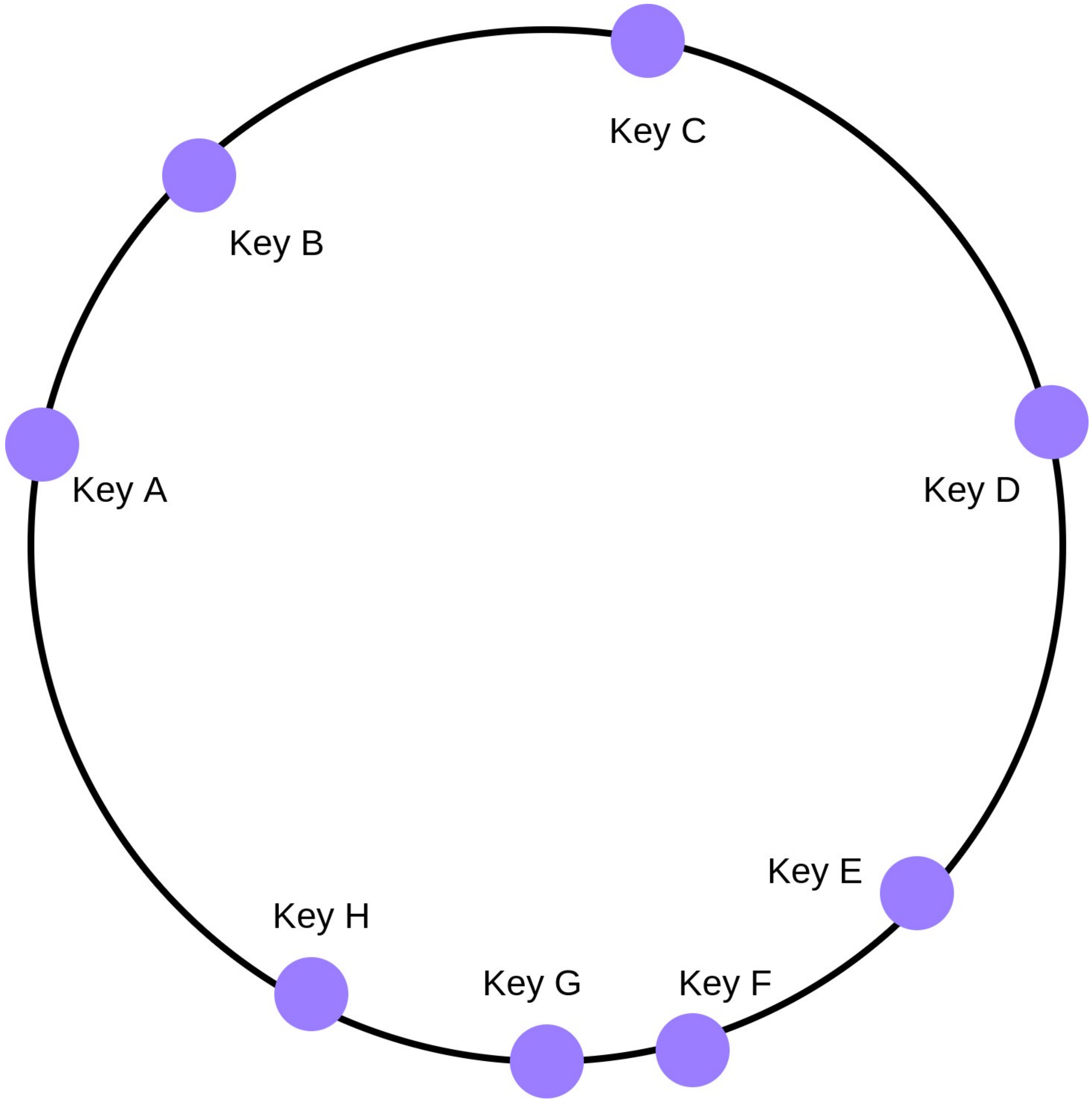

hash function
keys
buckets
John Smith
Lisa Smith
Sandra Dee
00
01
02
03
13
14
15
521-8976
521-1234
521-9655

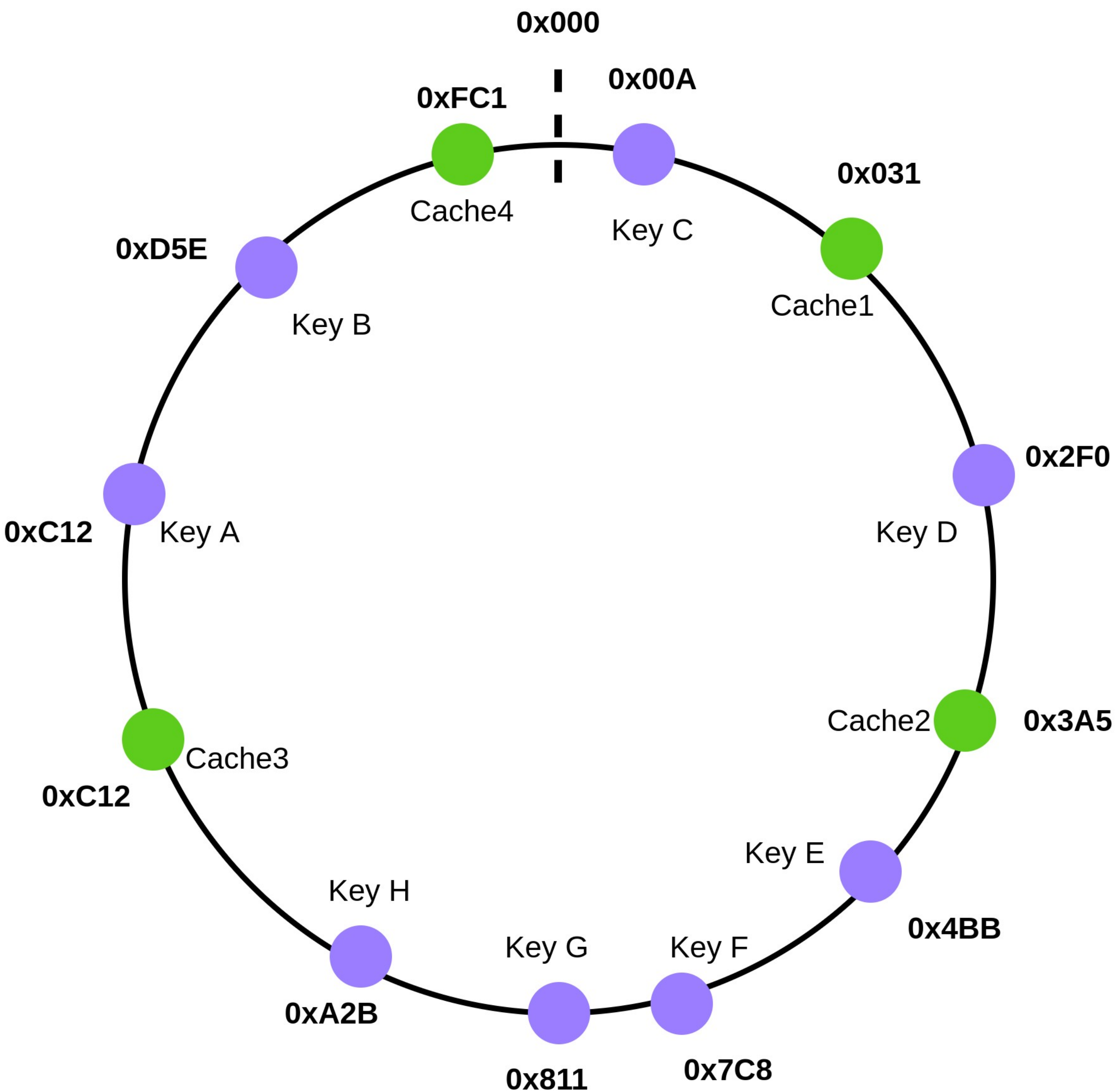
Distributed Hashing

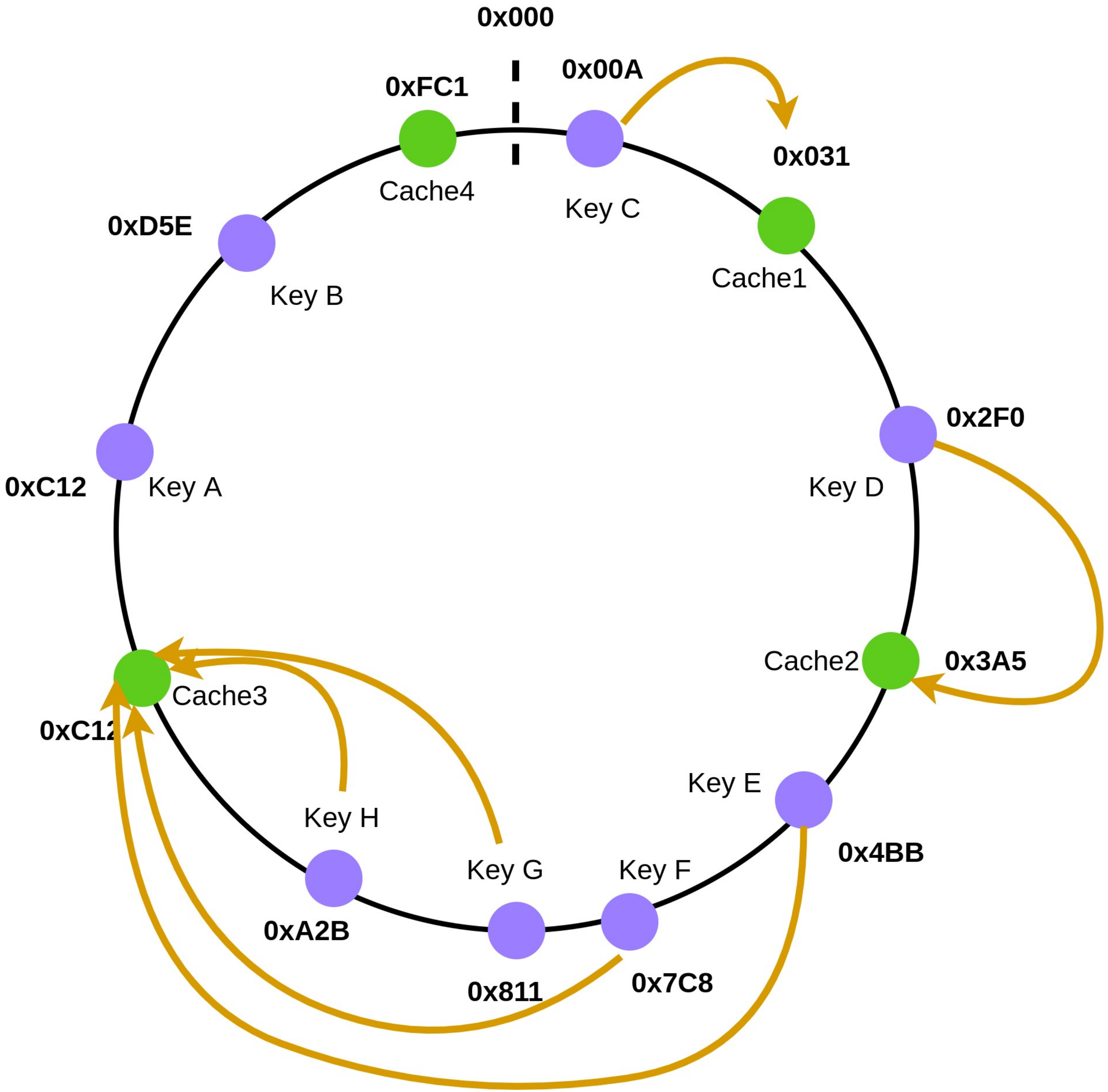
- caching



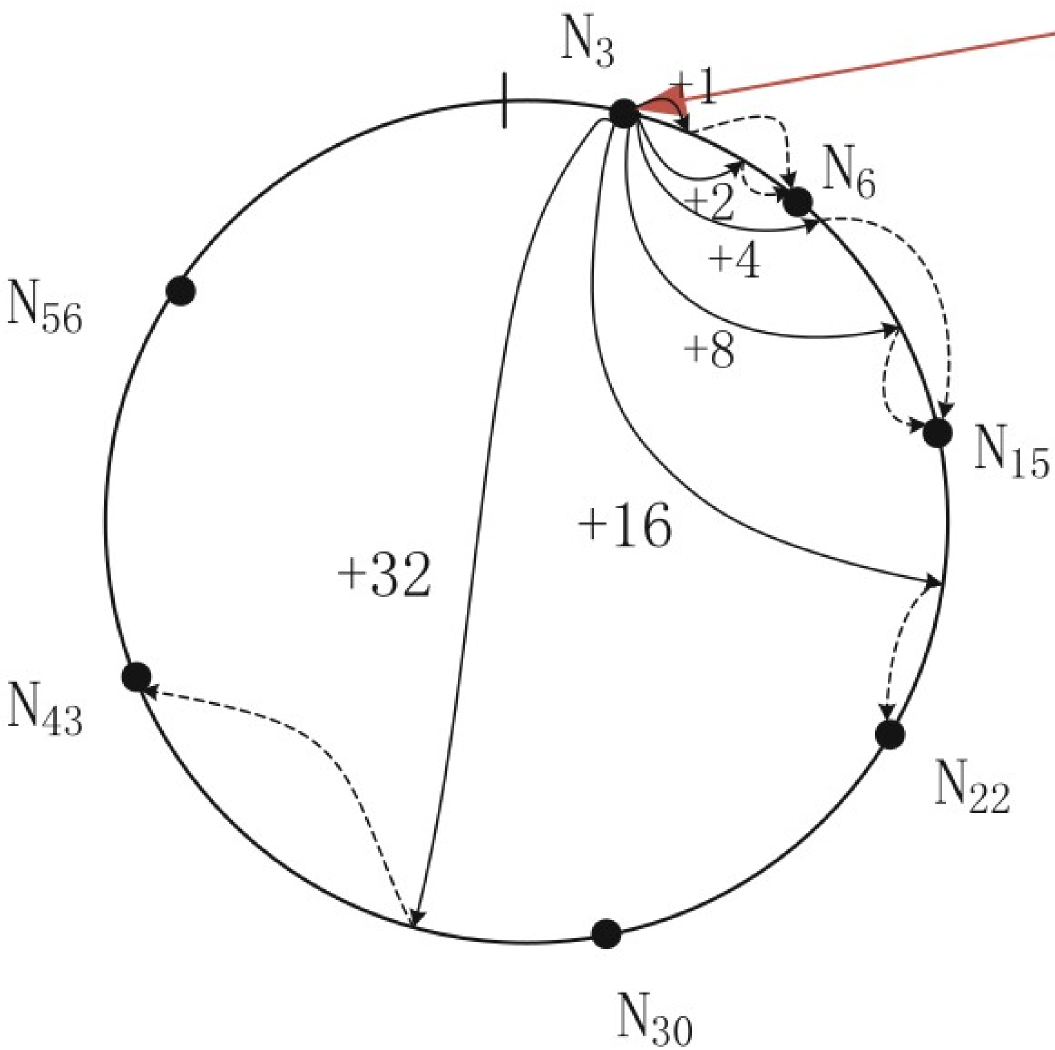
Consistent Hashing




0x000
0x00A
Key C
0x031
Cache1
0x2F0
Key D
Cache2
0x3A5
Key E
0x4BB
Key F
0x7C8
Key G
0x811
Key H
0xA2B
Cache3
0xC12
Key A
0xC12
0xD5E
Key B
0xFC1
Cache4

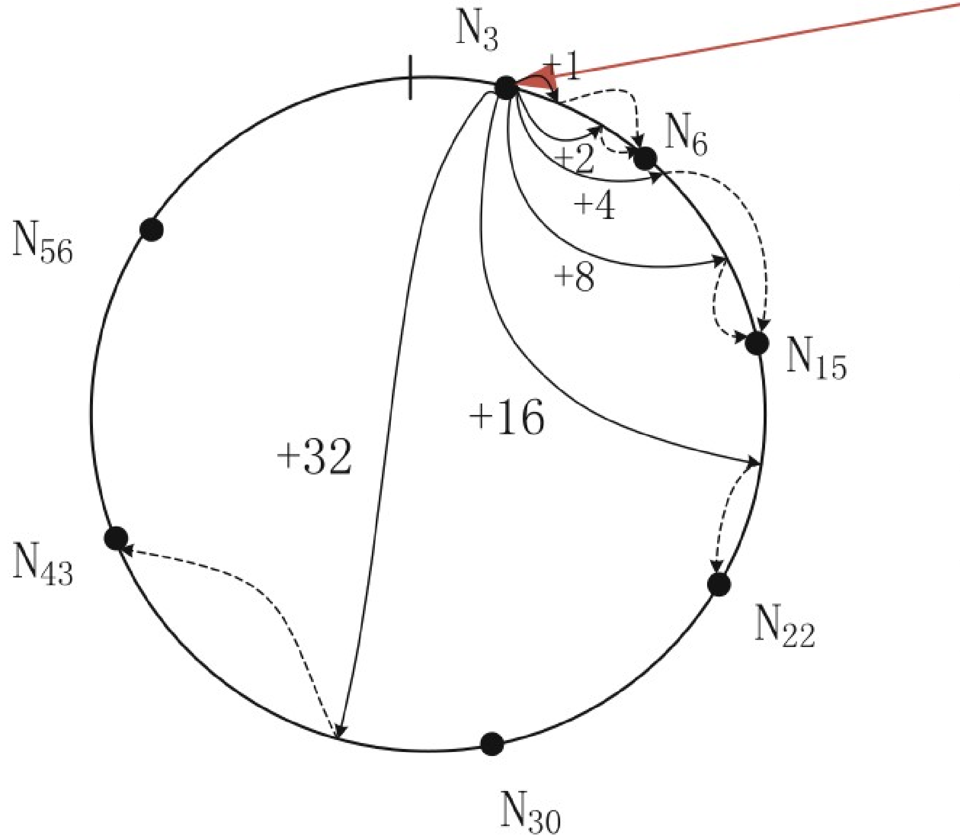

0x000
0x00A
0xFC1
0x031
Cache4
Key C
0xD5E
Cache1
Key B
0x2F0
0xC12
Key A
Key D
Cache2
0x3A5
Cache3
0xC12
Key E
Key H
0x4BB
Key G
Key F
0xA2B
0x811
0x7C8

Chord(DHT)

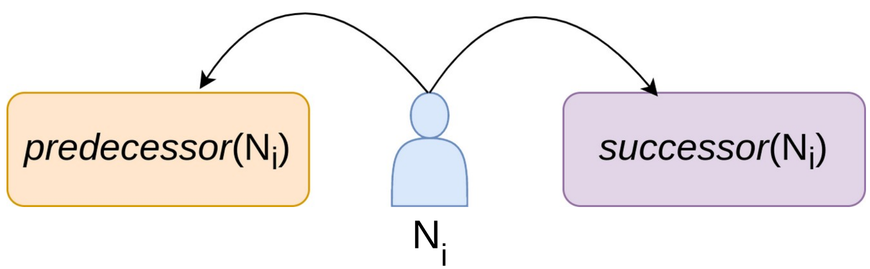


finger table	
N_3+1	N_6
N_3+2	N_6
N_3+4	N_{15}
N_3+8	N_{15}
N_3+16	N_{22}
N_3+32	N_{43}

Fig: H. Zhang et al., Distributed Hash Table; Theory, Platforms and Applications, SpringerBriefs in Computer Science, DOI 10.1007/978-1-4614-9008-1__2,



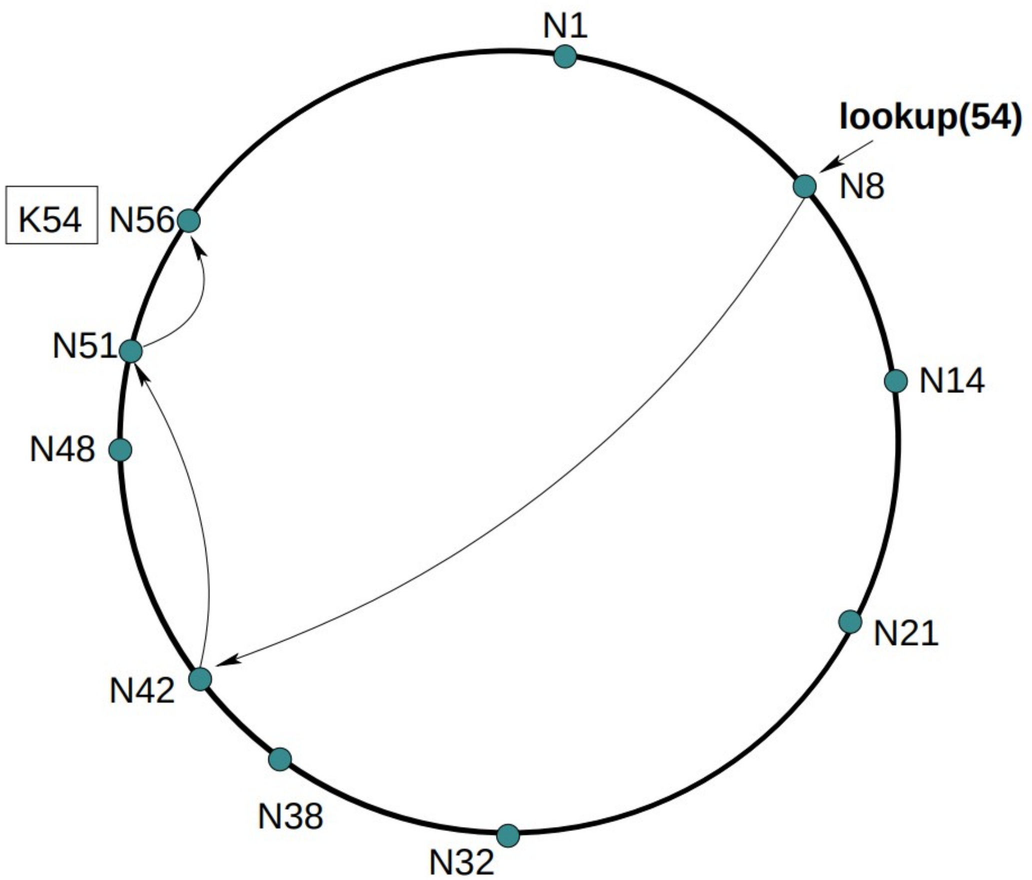
finger table	
N_3+1	N_6
N_3+2	N_6
N_3+4	N_{15}
N_3+8	N_{15}
N_3+16	N_{22}
N_3+32	N_{43}



ID = ID mod m

m=number of bits in the key

$$N.finger[i] = successor(N + 2^i)$$



k – key
v – value
k = h(*v*)

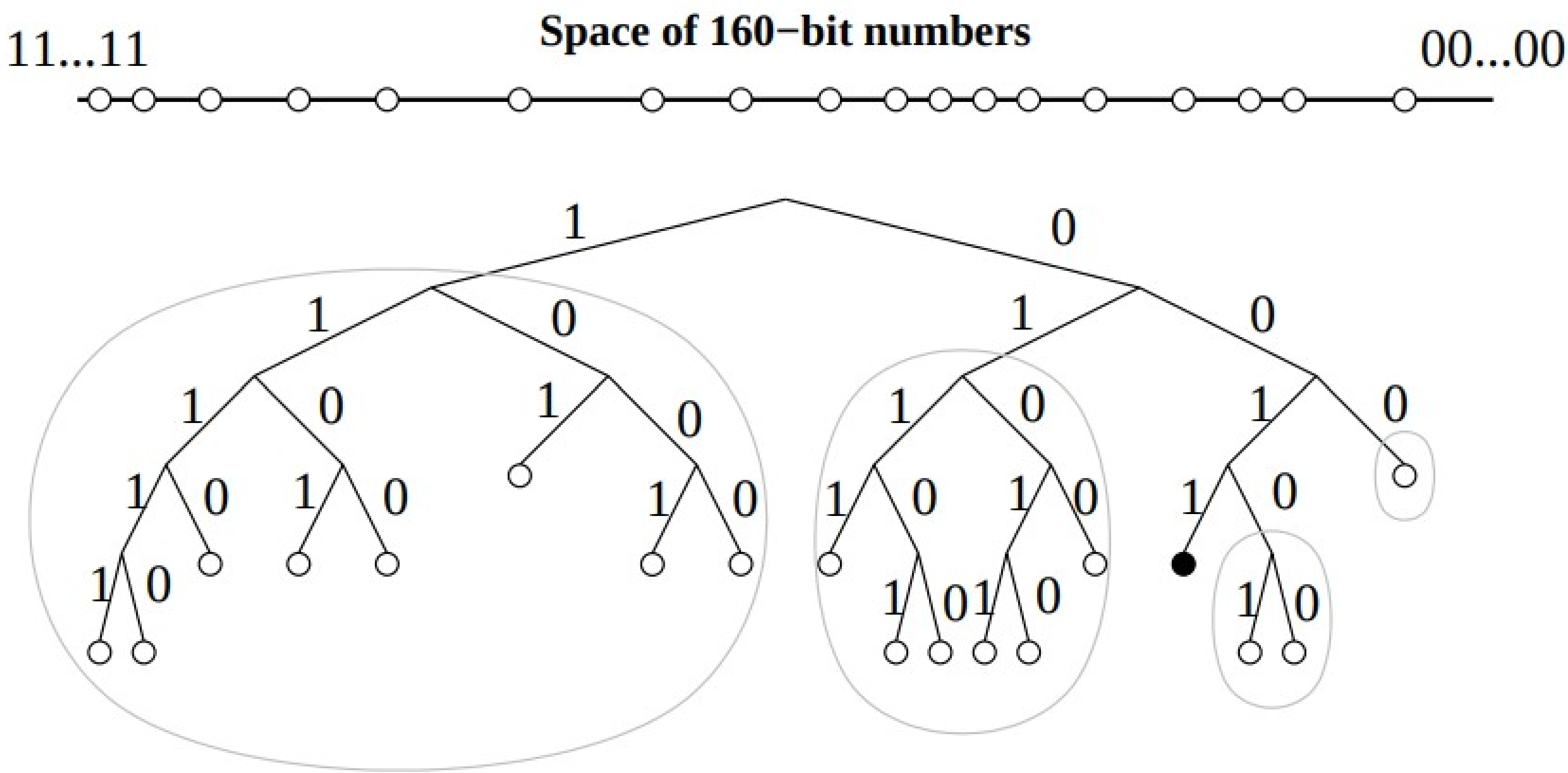
RPC:
put(*k*,*v*)
get(*k*)

Which node should store data with key *k?*

$$N' = \begin{cases} \mathtt{N.finger}[0] & \mathrm{d}(k,j) \le \mathrm{d}(\mathtt{N.finger}[0], j) \\ \mathtt{N.finger}[i] & \mathrm{d}(k,j) \le \mathrm{d}(\mathtt{N.finger}[i+1], j) \text{ and } \mathrm{d}(k,j) > \mathrm{d}(\mathtt{N.finger}[i], j) \\ \mathtt{N.finger}[m-1] & \text{otherwise} \end{cases}$$

/

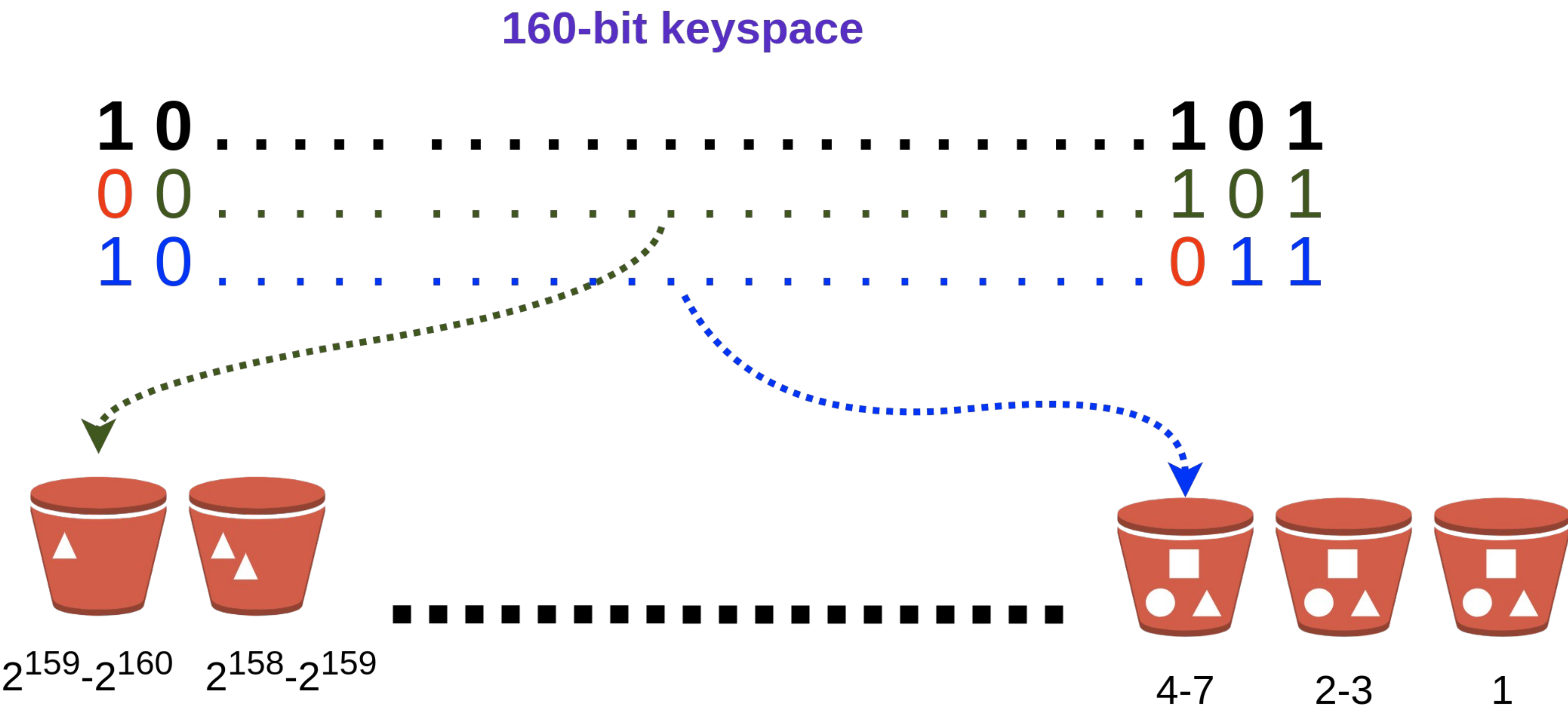
Kademlia (DHT)

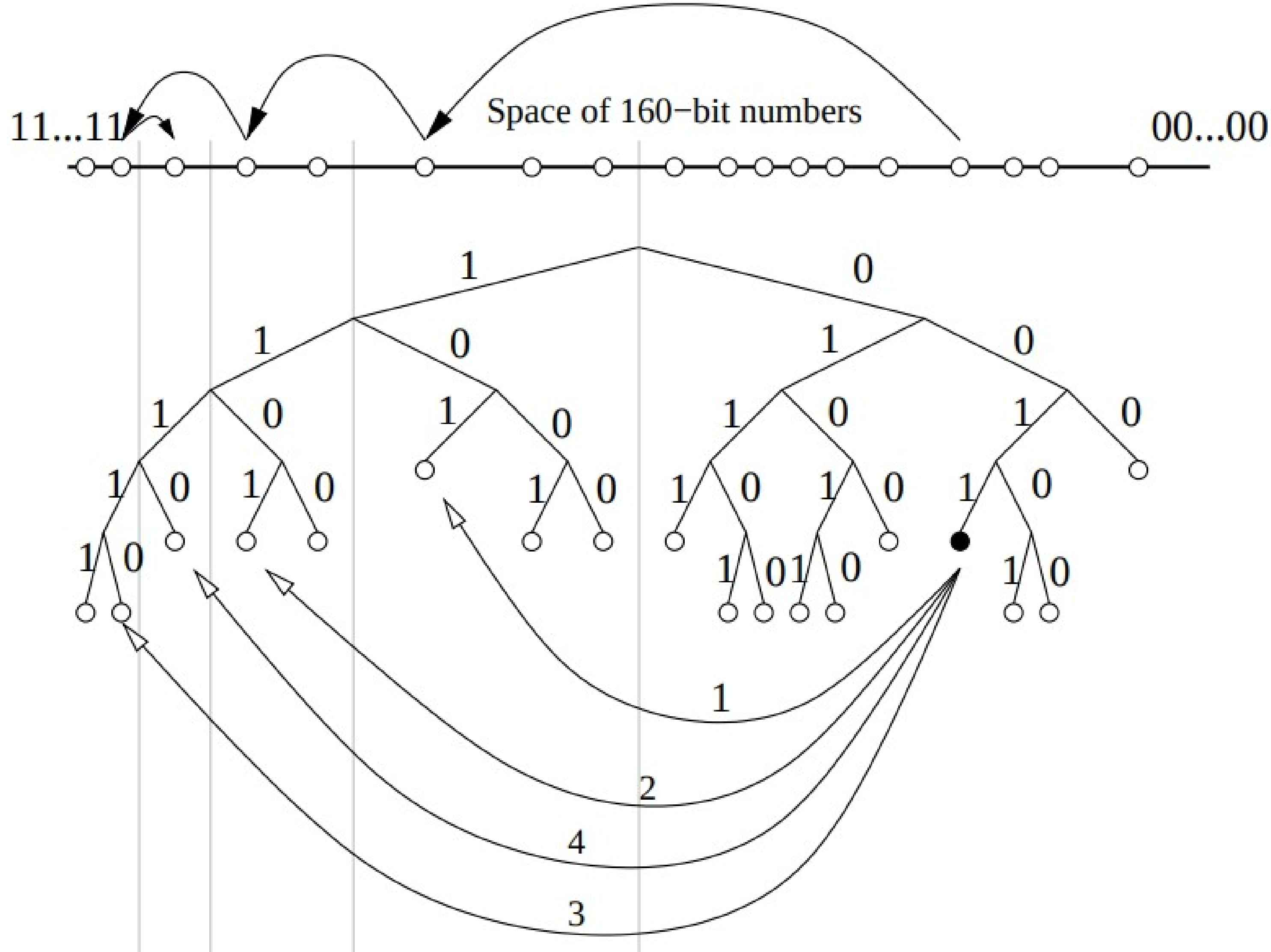


d(x,y) = x **XOR** y

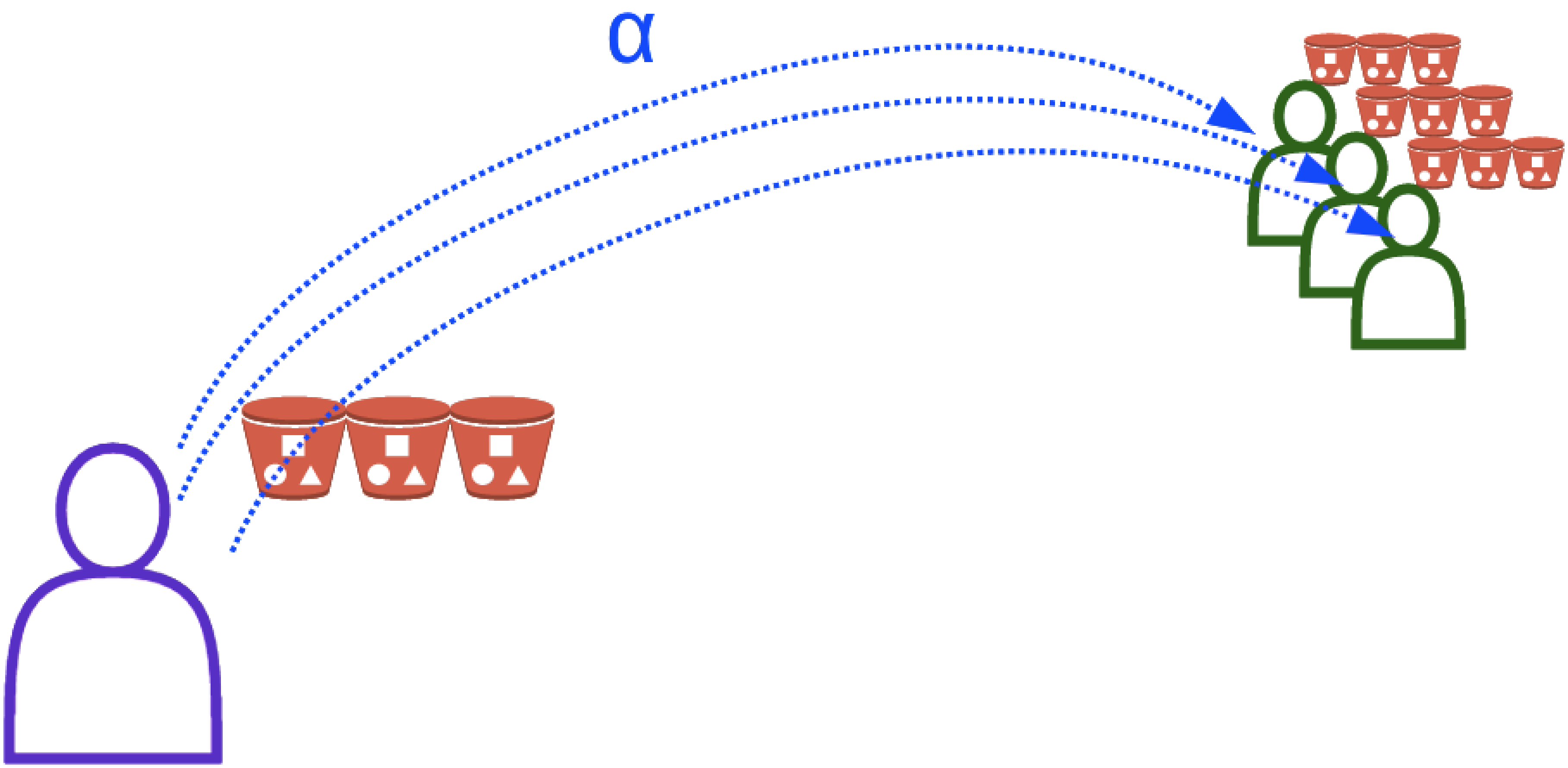
α – parametr zrównoleglelnia
k – wielkość bucket’ów

k-buckets

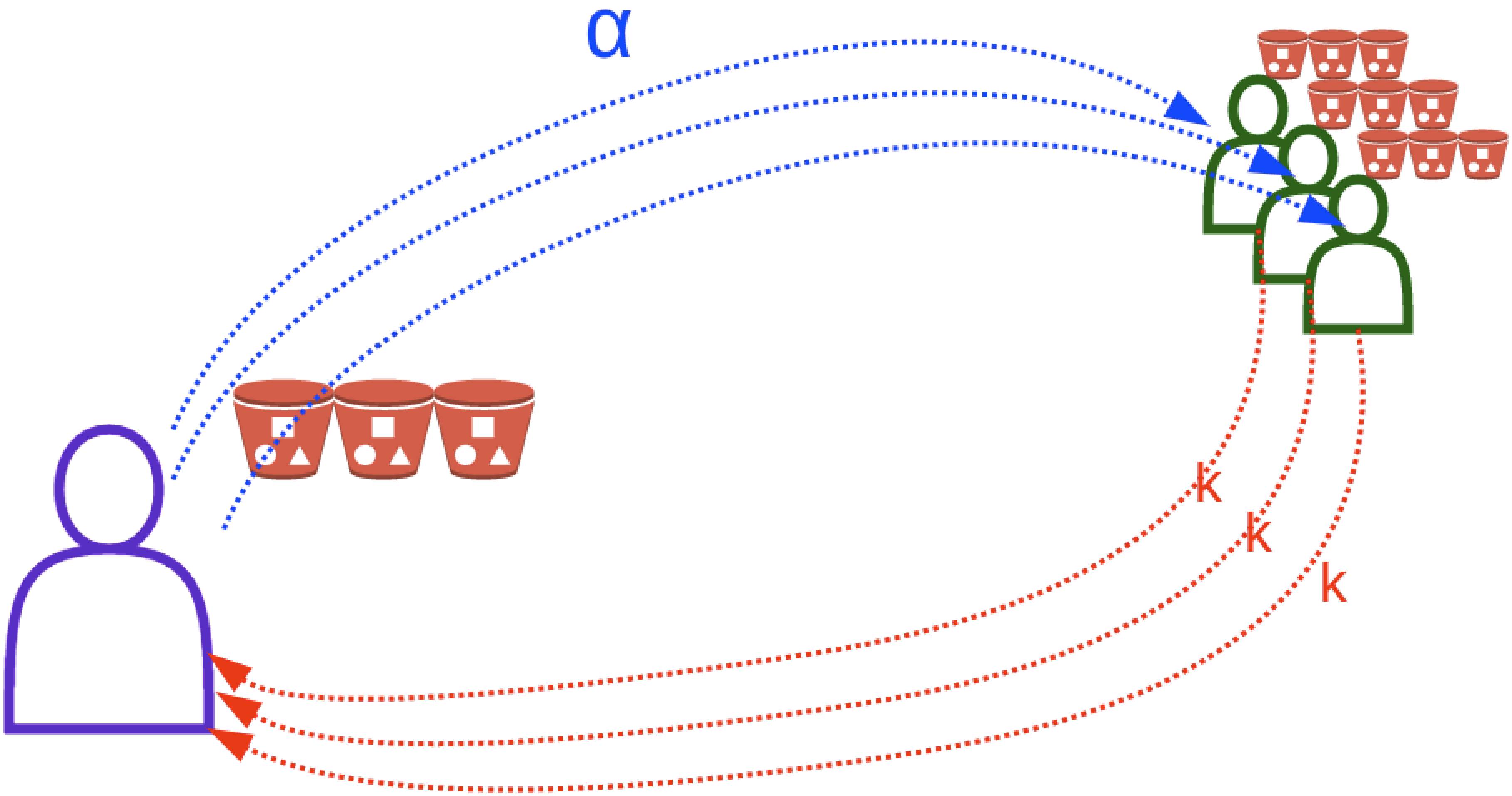

160-bit keyspace
1 0 . . . . 1 0 1
0 0 . . . . 1 0 1
1 0 . . . . 0 1 1
2^{159}-2^{160}
2^{158}-2^{159}
4-7
2-3
1


11...11
Space of 160−bit numbers
00...00
1
0
1
0
1
0
1
2
4
3

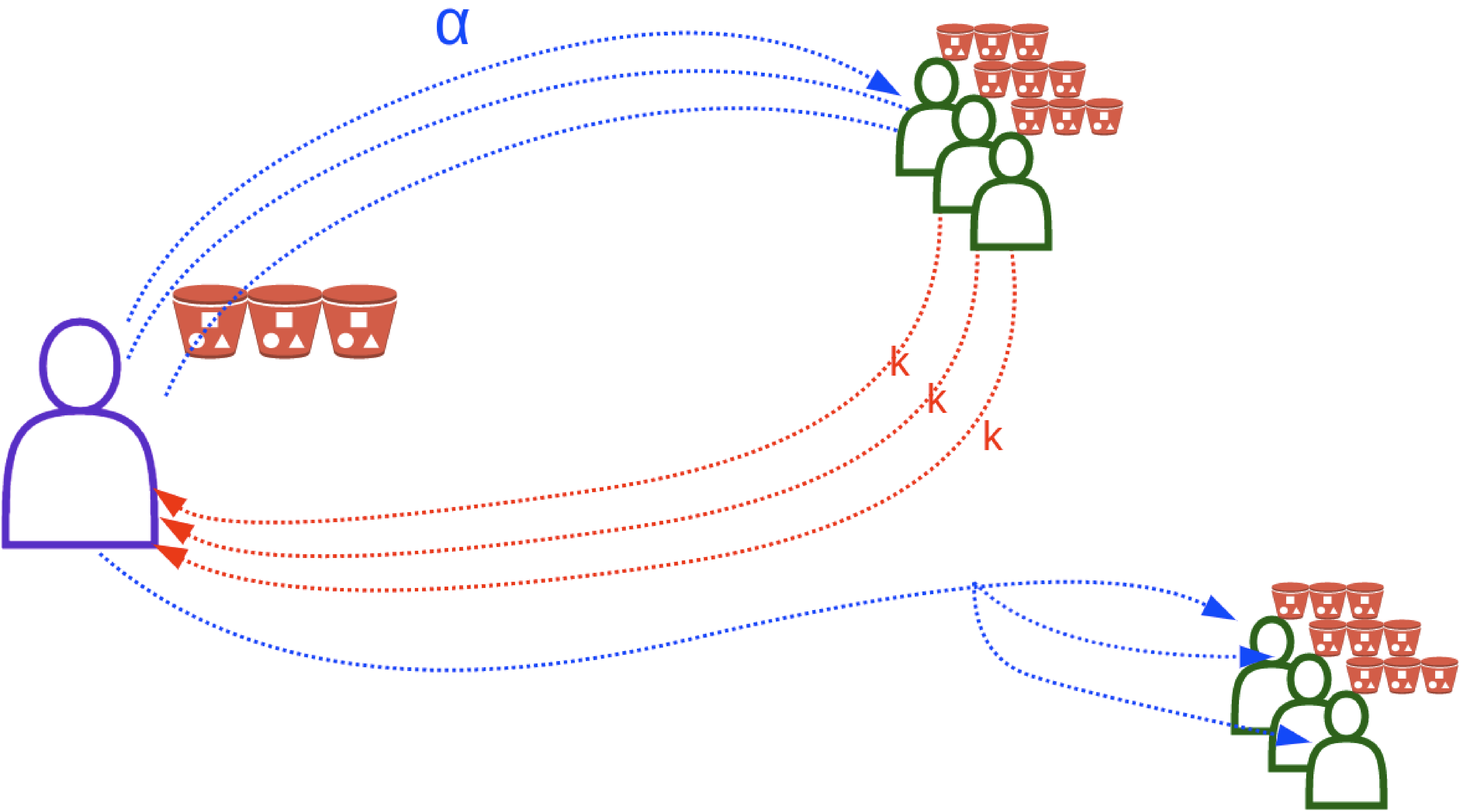
Node lookup



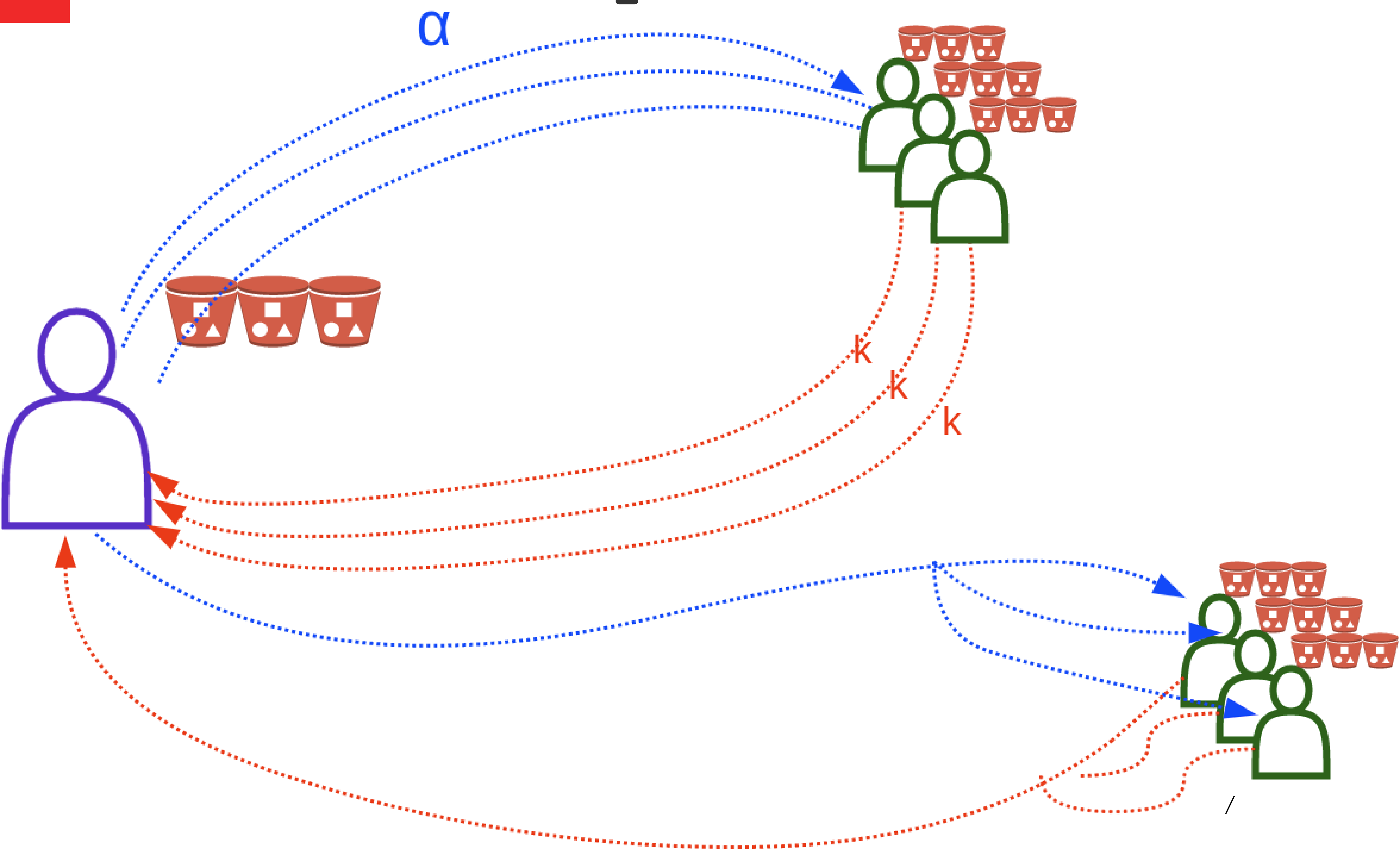
Node lookup



Node lookup



Node lookup



Przykładowe Zastosowanie