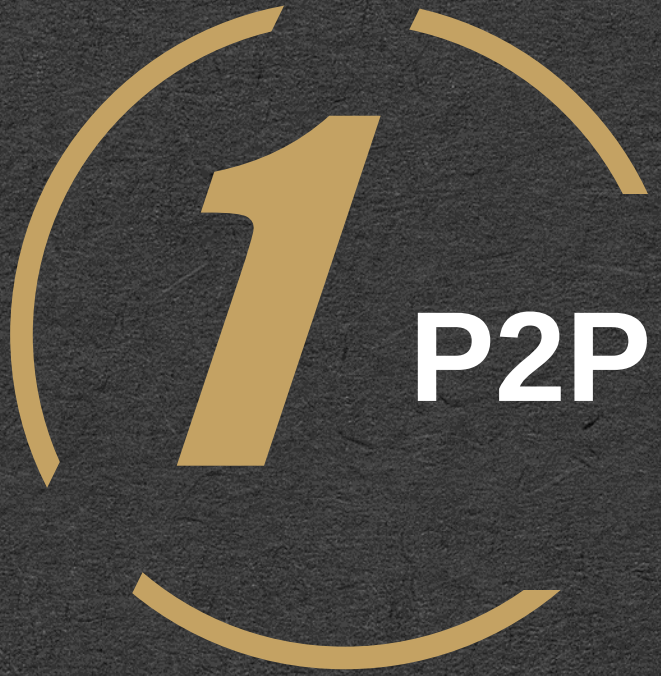
BlockChain

- *p2p network*

Horizon Wang

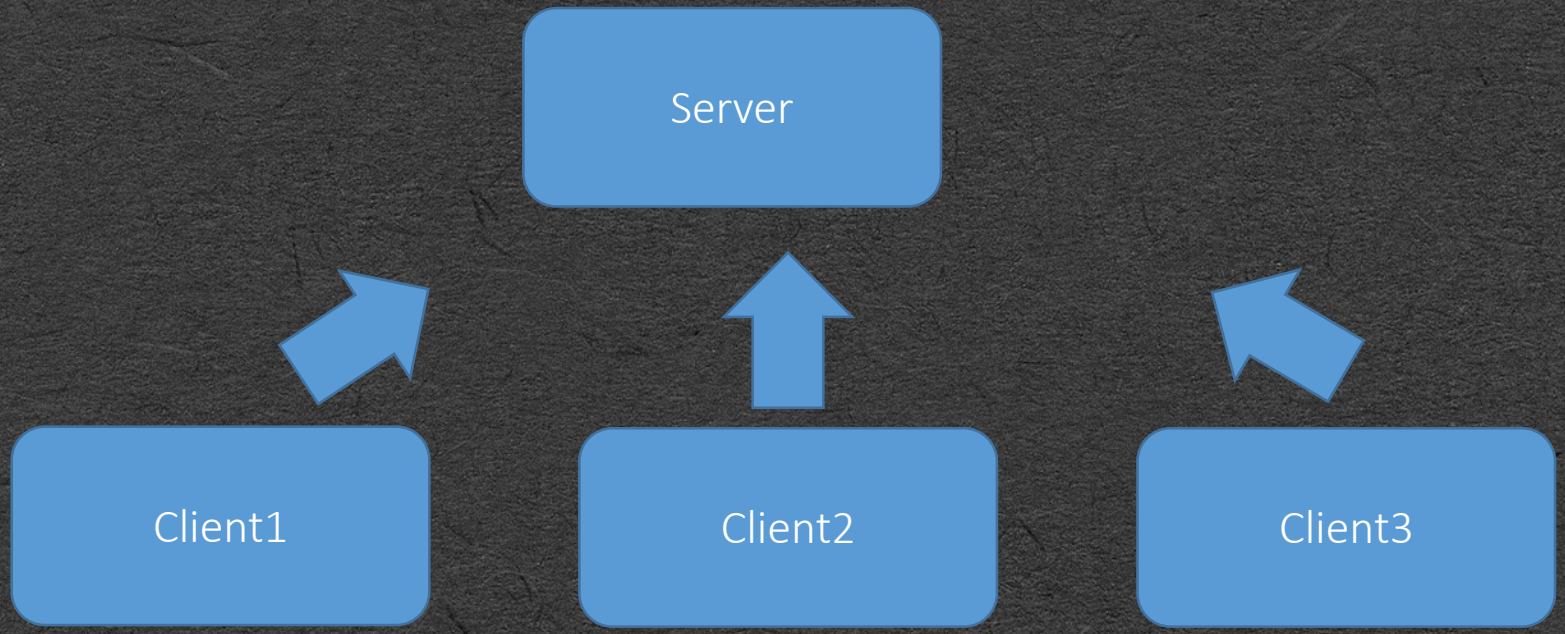
2019.03

CONTENTS

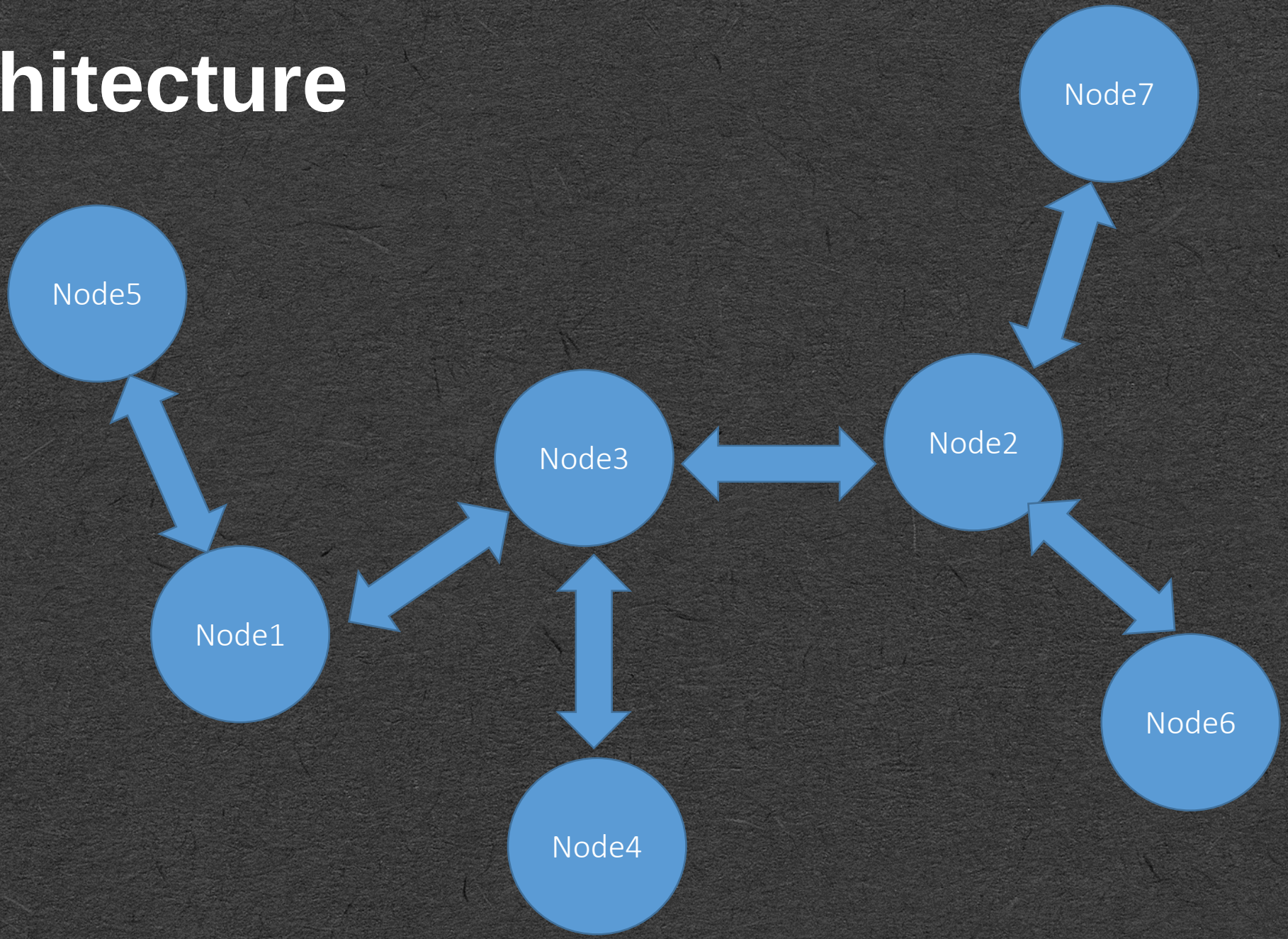


P2P Network Architecture

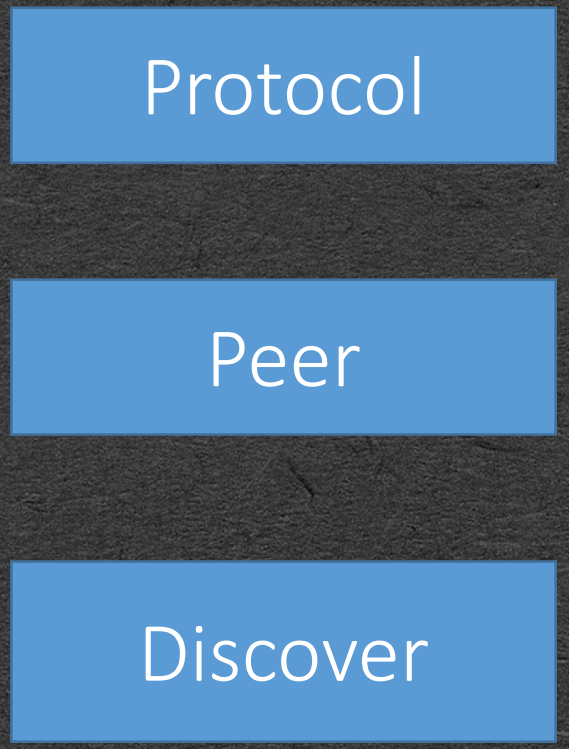
Architecture



Architecture

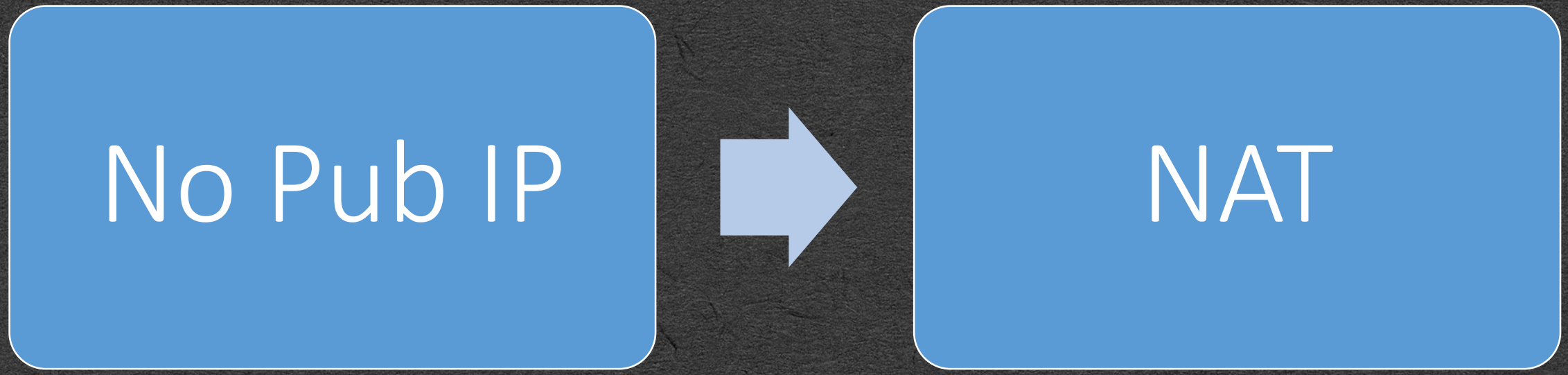


Architecture

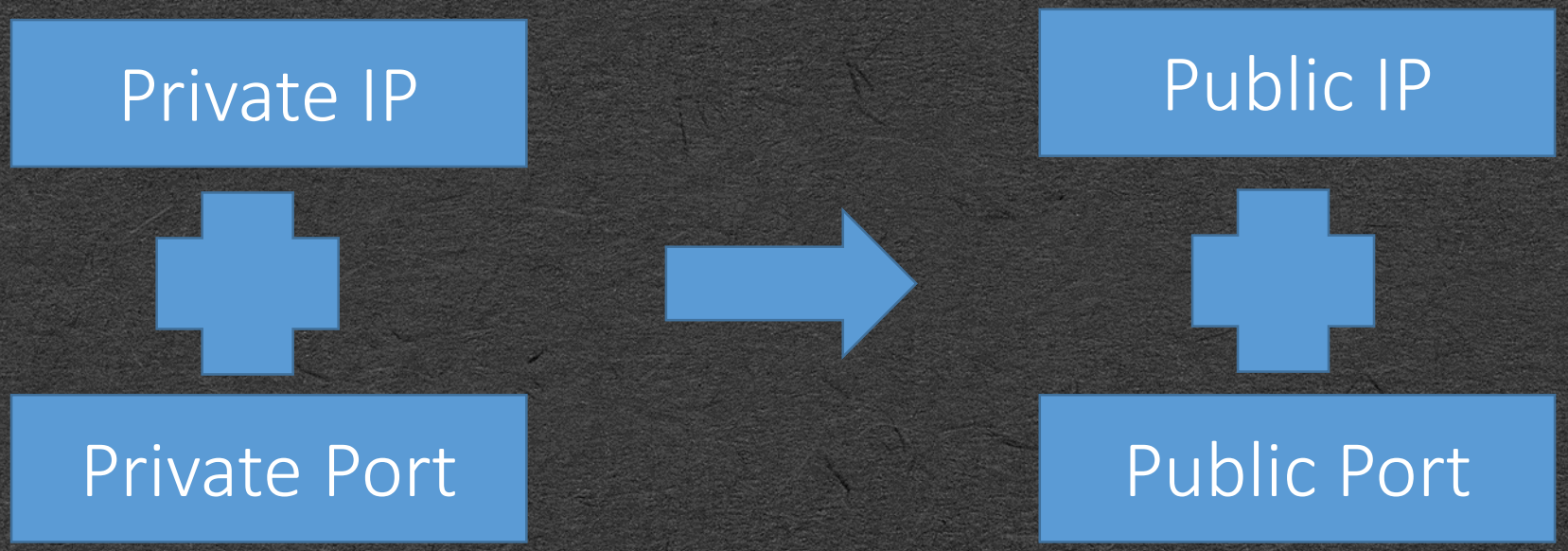


Algorithm

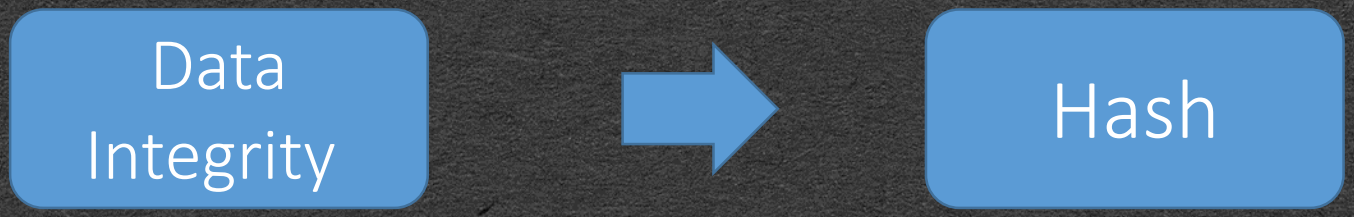
Algorithm



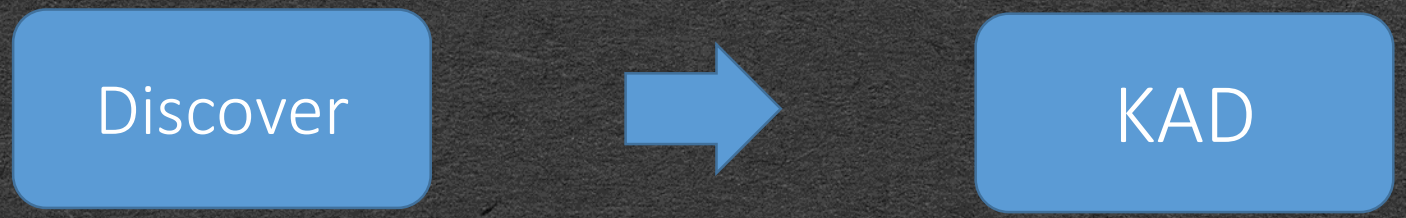
Algorithm



Algorithm



Algorithm



Algorithm

This section is simplified to use a single bit; see the section accelerated lookups for more information on real routing tables.

Kademlia routing tables consist of a *list* for each bit of the node ID. (e.g. if a node ID consists of 128 bits, a node will keep 128 such *lists*.) A list has many entries. Every entry in a *list* holds the necessary data to locate another node. The data in each *list* entry is typically the *IP address*, *port*, and *node ID* of another node. Every *list* corresponds to a specific distance from the node. Nodes that can go in the n^{th} *list* must have a differing n^{th} bit from the node's ID; the first n-1 bits of the candidate ID must match those of the node's ID. This means that it is very easy to populate the first *list* as 1/2 of the nodes in the network are far away candidates. The next *list* can use only 1/4 of the nodes in the network (one bit closer than the first), etc.

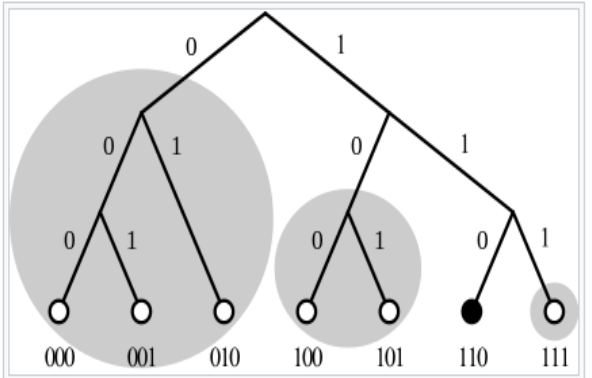
With an ID of 128 bits, every node in the network will classify other nodes in one of 128 different distances, one specific distance per bit.

As nodes are encountered on the network, they are added to the *lists*. This includes store and retrieval operations and even helping other nodes to find a key. Every node encountered will be considered for inclusion in the *lists*. Therefore, the knowledge that a node has of the network is very dynamic. This keeps the network constantly updated and adds resilience to failures or attacks.

In the Kademlia literature, the *lists* are referred to as *k-buckets*. *k* is a system wide number, like 20. Every k-bucket is a *list* having up to *k* entries inside; i.e. for a network with k=20, each node will have *lists* containing up to 20 nodes for a particular bit (a particular distance from itself).

Since the possible nodes for each *k-bucket* decreases quickly (because there will be very few nodes that are that close), the lower bit *k-buckets* will fully map all nodes in that section of the network. Since the quantity of possible IDs is much larger than any node population can ever be, some of the k-buckets corresponding to very short distances will remain empty.

Consider the simple network to the right. The network size is 2^3 or eight maximum keys and nodes. There are seven nodes participating; the small circles at the bottom. The node under consideration is node six (binary 110) in black. There are three *k-buckets* for each node in this network. Nodes zero, one and two (binary 000, 001, and 010) are candidates for the farthest *k-bucket* . Node three (binary 011, not shown) is not participating in the network. In the middle *k-bucket*, nodes four and five (binary 100 and 101) are placed. Finally, the third *k-bucket* can only contain node seven (binary 111). Each of the three *k-buckets* are enclosed in a gray circle. If the size of the *k-bucket* was two, then the farthest *2-bucket* can only contain two of the three nodes. For example, if node six has node one and two in the farthest 2-bucket, it would have to request a node ID lookup to these nodes to find the location (ip address) of node zero. Each node *knows* its neighbourhood well and has contact with a few nodes far away which can help locate other nodes far away.



Network partition for node **110**

It is known that nodes that have been connected for a long time in a network will probably remain connected for a long time in the future.[3][4] Because of this statistical distribution, Kademlia selects long connected nodes to remain stored in the k-buckets. This increases the number of known valid nodes at some time in the future and provides for a more stable network.

Algorithm

PING

PONG

FINDNODE

NEIGHBORS

Protocol

Protocol

HandShake

TX

Status

Block

State

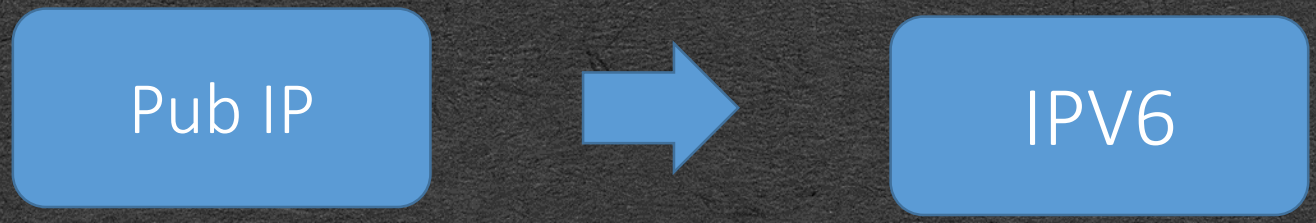
Protocol

FULL

LIGHT

Improvement

Improvement



Improvement

THANK YOU