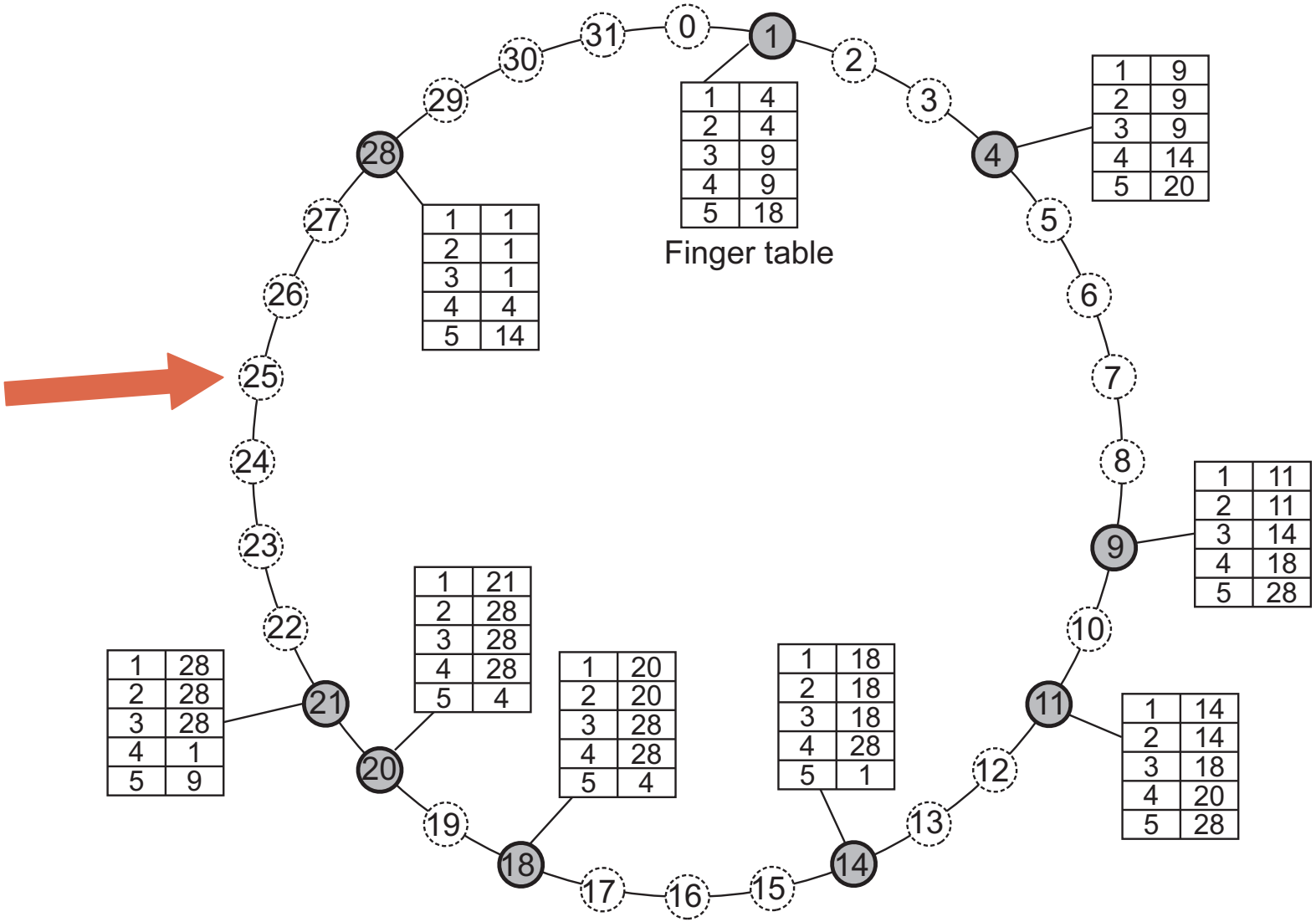
# Project 3

Yao Liu

# Nodes join and leave the DHT

- In Project 2, DHT nodes are started once and remain accessible. Fingertable is computed given the information about all nodes in the DHT and sent to each node.
- In Project 3, nodes will join the DHT one by one. Once a new node joins, it will:
  - update the fingertables of existing nodes that are affected by the join.
  - assume the responsibility for files that were held by other node.
- Nodes may also gracefully leave the DHT

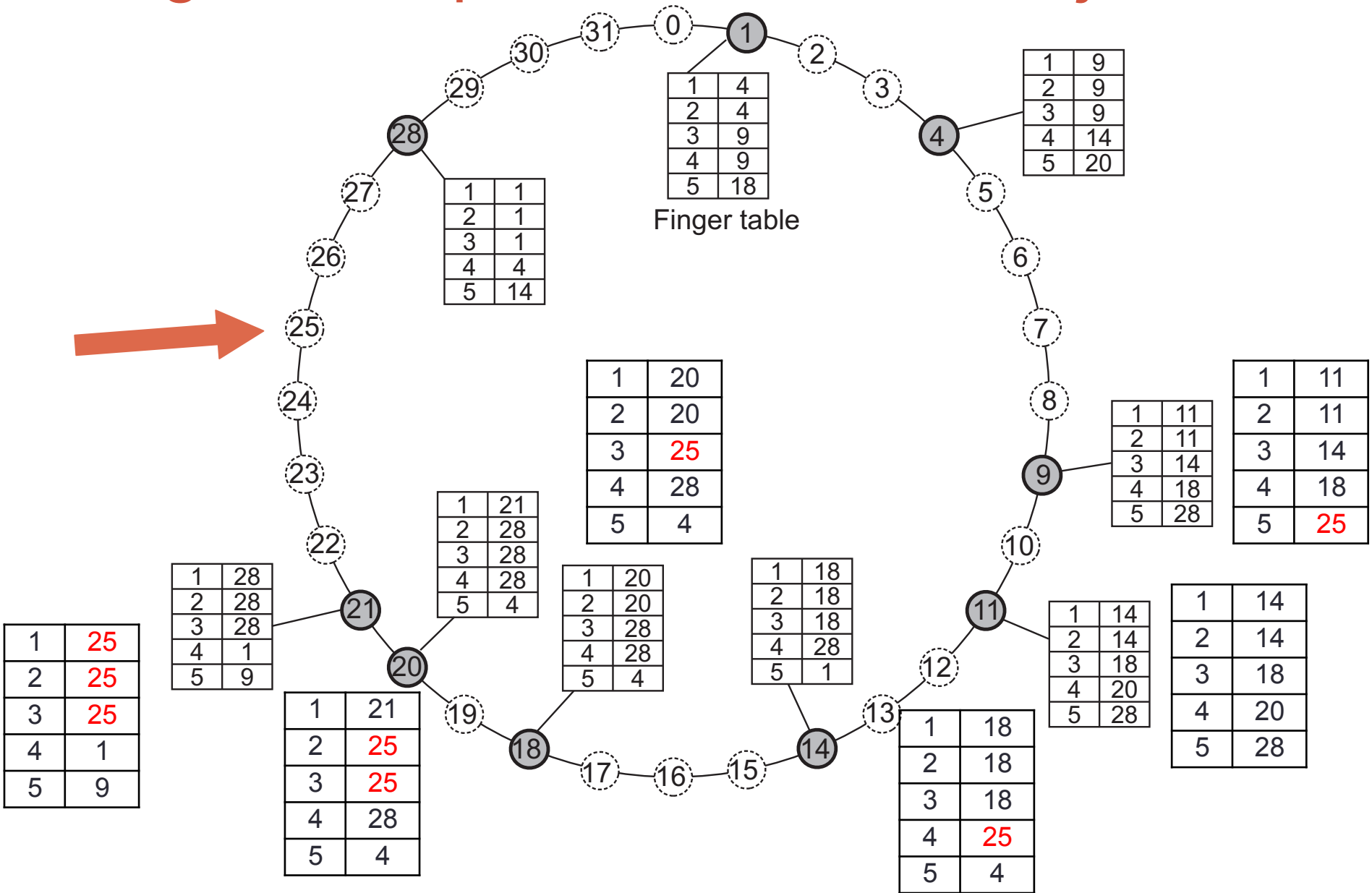
# Node 25 joins the Chord DHT



# Create a new fingertable for Node 25

- Node 25 can contact an arbitrary node that already exists in the DHT, and ask this node to compute fingertable entries for it using the *findSucc()* call.

# Fingertable updates after Node 25 joins



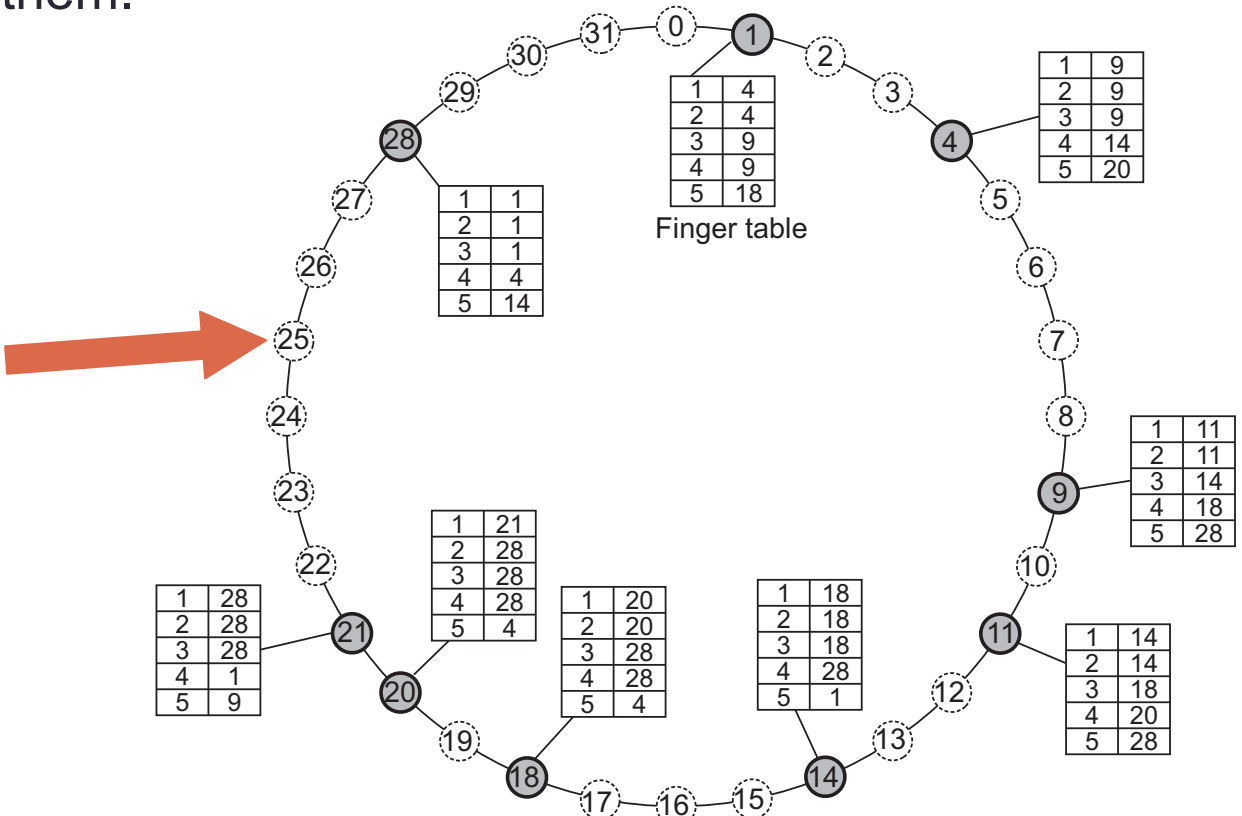
# Fingertable updates

- For each node *p* that $p+2^i$ belongs to the interval (pred($node_{new}$), $node_{new}$], the new node will update node *p*'s $i^{th}$ entry in the fingertable. (counting from 0)
  - use the *updateFinger()* call
- A new node affects *O(log(N))* other fingertable entries in the system, on average
- Number of messages per node join= *O(log(N)*log(N))*

# Pull files from Node 28 to Node 25

Node 25 also need to assume responsibility for files with key {22, 23, 24, 25}.

1. Node 25 uses *setNodePred()* to set the new predecessor for Node 28
2. Send *pullUnownedFiles*() call to Node 28
3. Node 28 retrieves a set of RFiles whose keys are among {22, 23, 24, 25}, and returns them.



# An existing node leaves the DHT

- Similar to how we deal with node joining
- For each node *p* that $p+2^i$ belongs to the interval (pred($node_{leave}$), $node_{leave}$], we need to update node *p*'s $i^{th}$ entry (counting from 0) in the fingertable from $node_{leave}$ to succ($node_{leave}$).
- Push files it was responsible to its successor. Use the *pushUnownedFiles()* call.

# For debugging

- Use the `cmp_fingertables` code supplied.
- Compare your fingertable with the output from `cmp_fingertables`.