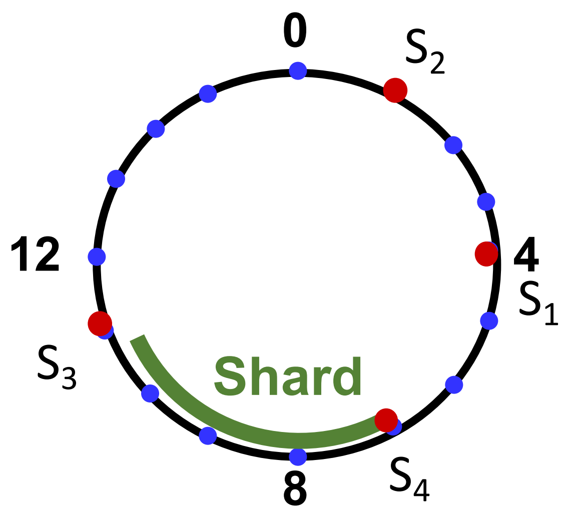
Discussion 07

Spring 2019 – CS 188

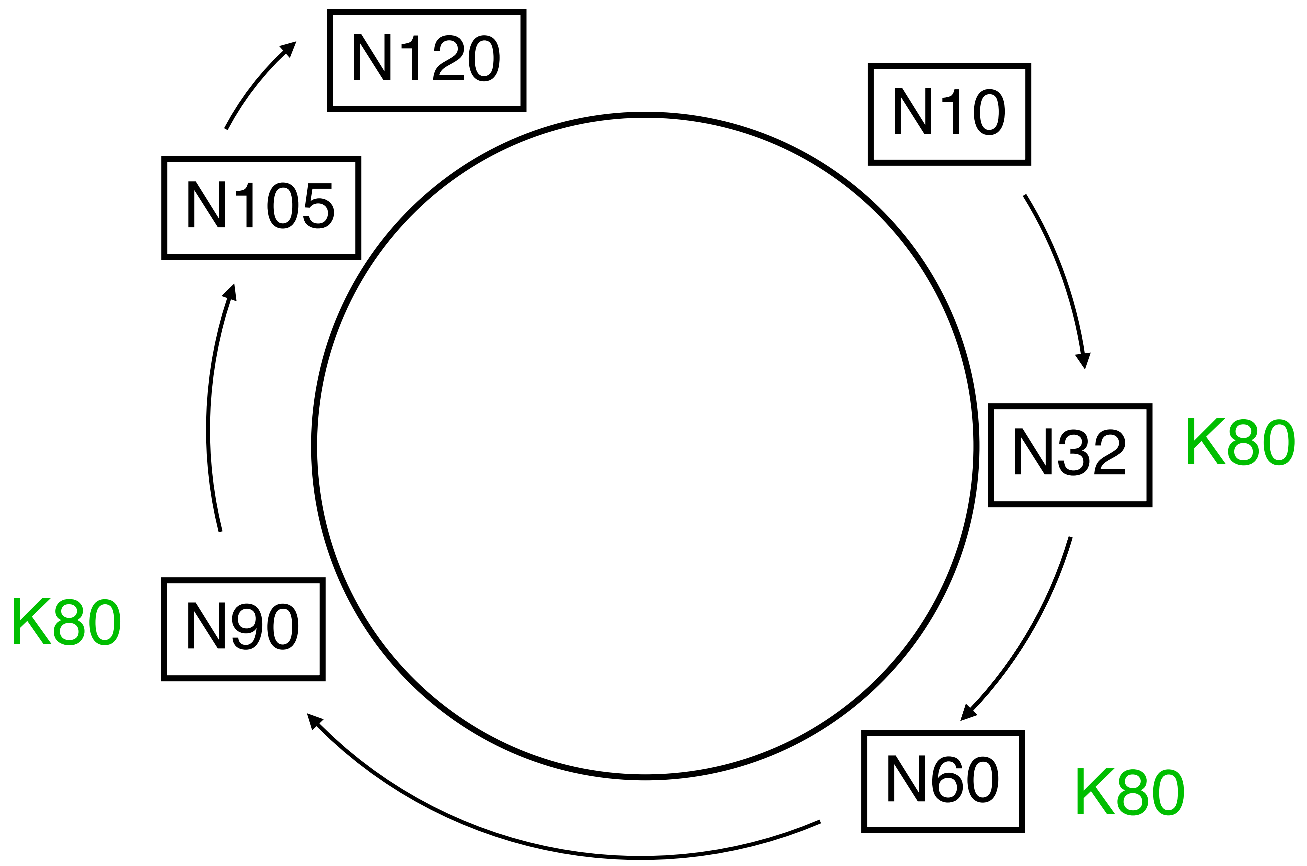
Section 2B

Consistent Hashing

- Represent hash space as a circle
- Partition keys across servers
  - Assign every server a random ID
  - Hash server ID
  - Server responsible for keys between predecessor and itself
- How to map a key to a server?
  - Hash key and execute read/write at successor



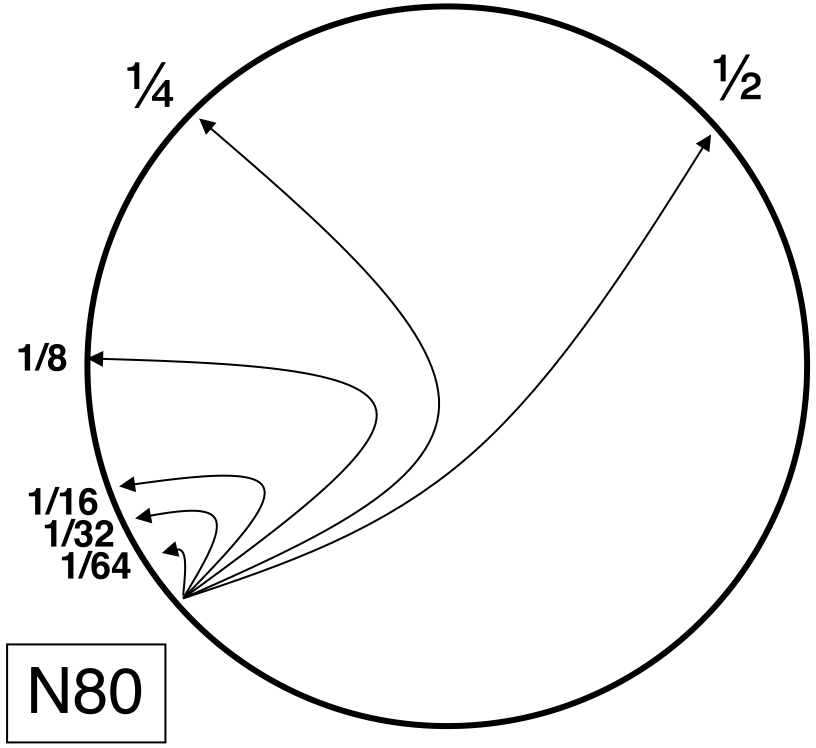
Successor Pointers



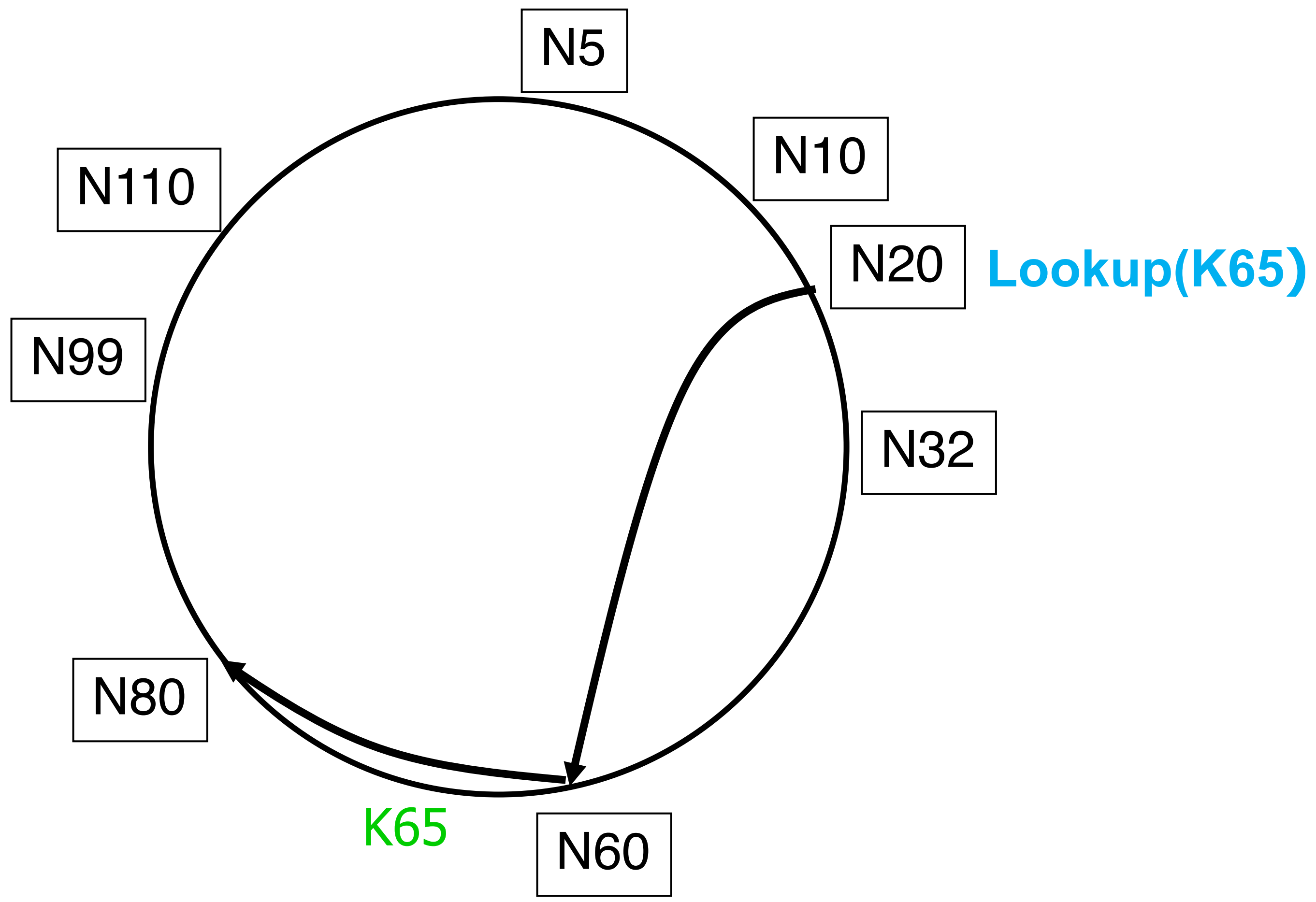
- If you don't have value for key, forward to successor

Finger Tables

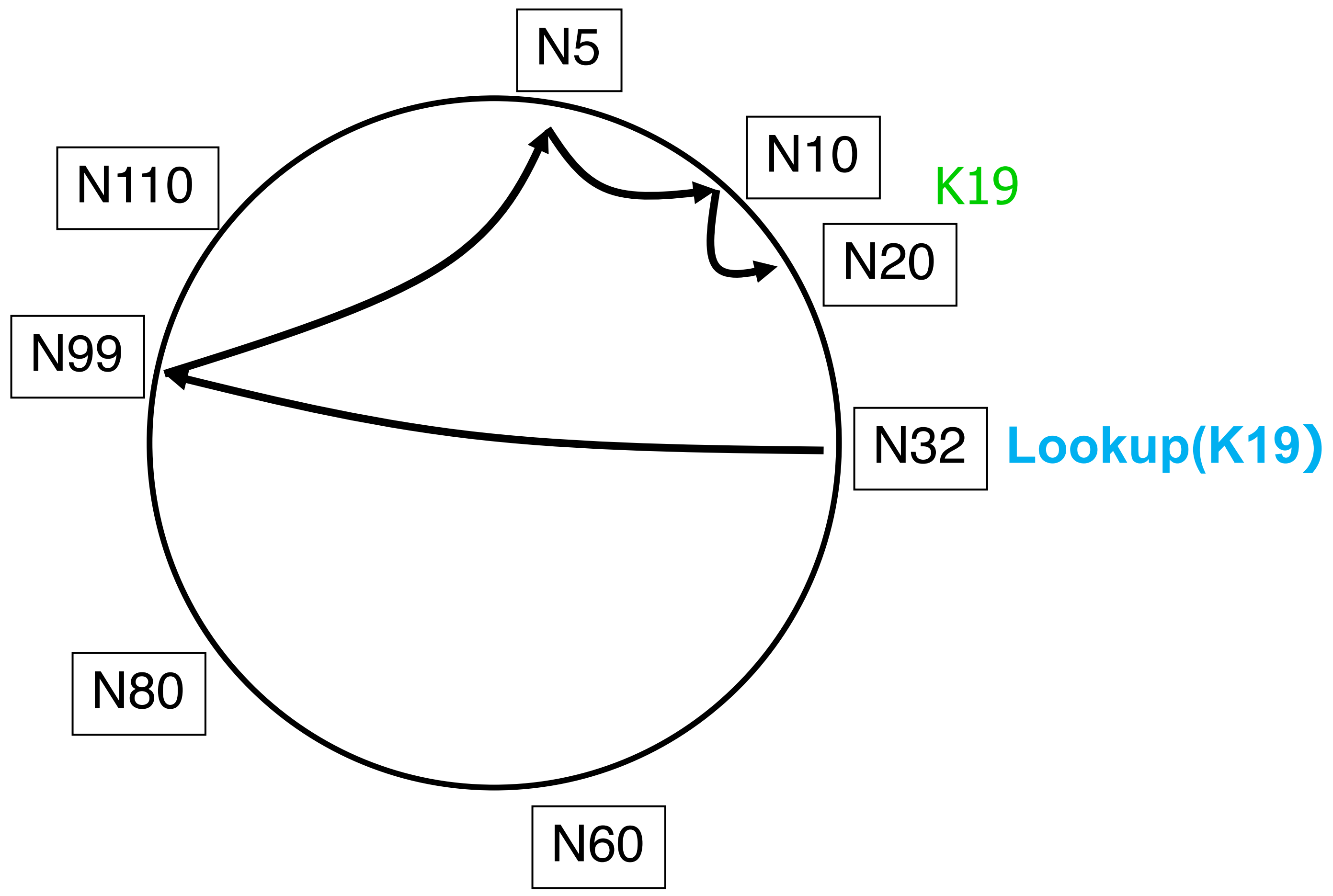
- i’th entry at node n points to successor of hash(n)+2^i
  - # of entries = # of bits in hash value
- Binary lookup tree rooted at every node
  - Threaded through others’ finger tables



Example1: Lookup with Finger Table



Lookups take O(log N) hops



Example2: Serializability

- Concurrent execution of transactions:
  - T1: transfer $10 from Alice to Bob
  - T2: Read balance in Alice's and Bob's accounts
  - T3: Read balance in Charlie's account.
  - Initial balance is $100 of 3 accounts

- Permissible outputs for T2?
  - (Alice: $100, Bob: $100) or (Alice: $90, Bob: $110)

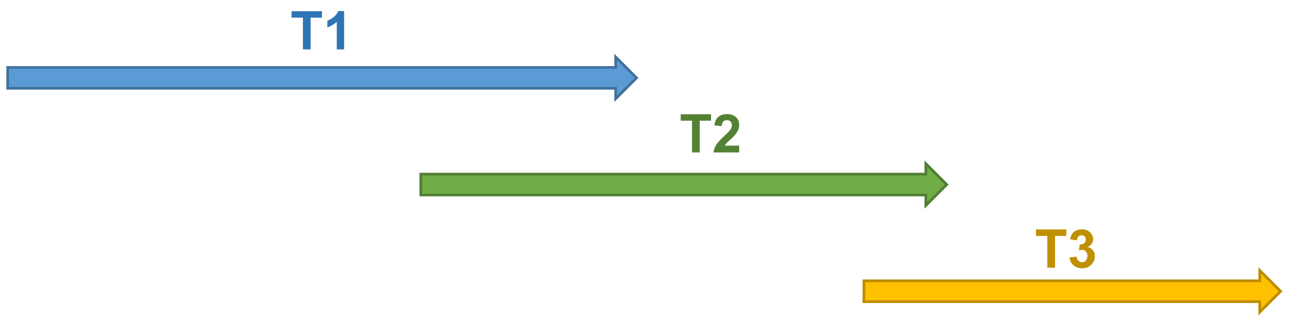
- Invalid outputs for T2?
  - (Alice: $90, Bob: $100) or (Alice: $100, Bob: $110)

Transactions and Serializability

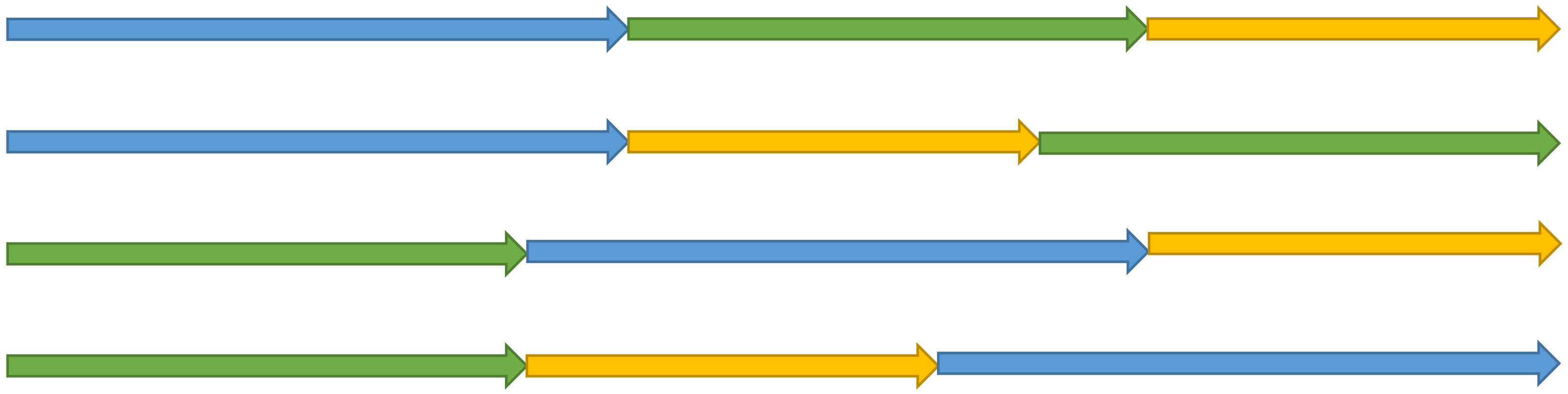
- Concurrent execution of transactions:
  - T1: transfer $10 from Alice to Bob
  - T2: Read balance in Alice’s and Bob’s accounts
  - T3: Read balance in Charlie’s account.
  - Initial balance is $100 of 3 accounts

- From user’s perspective: these transactions output must be the same as some serial execution.

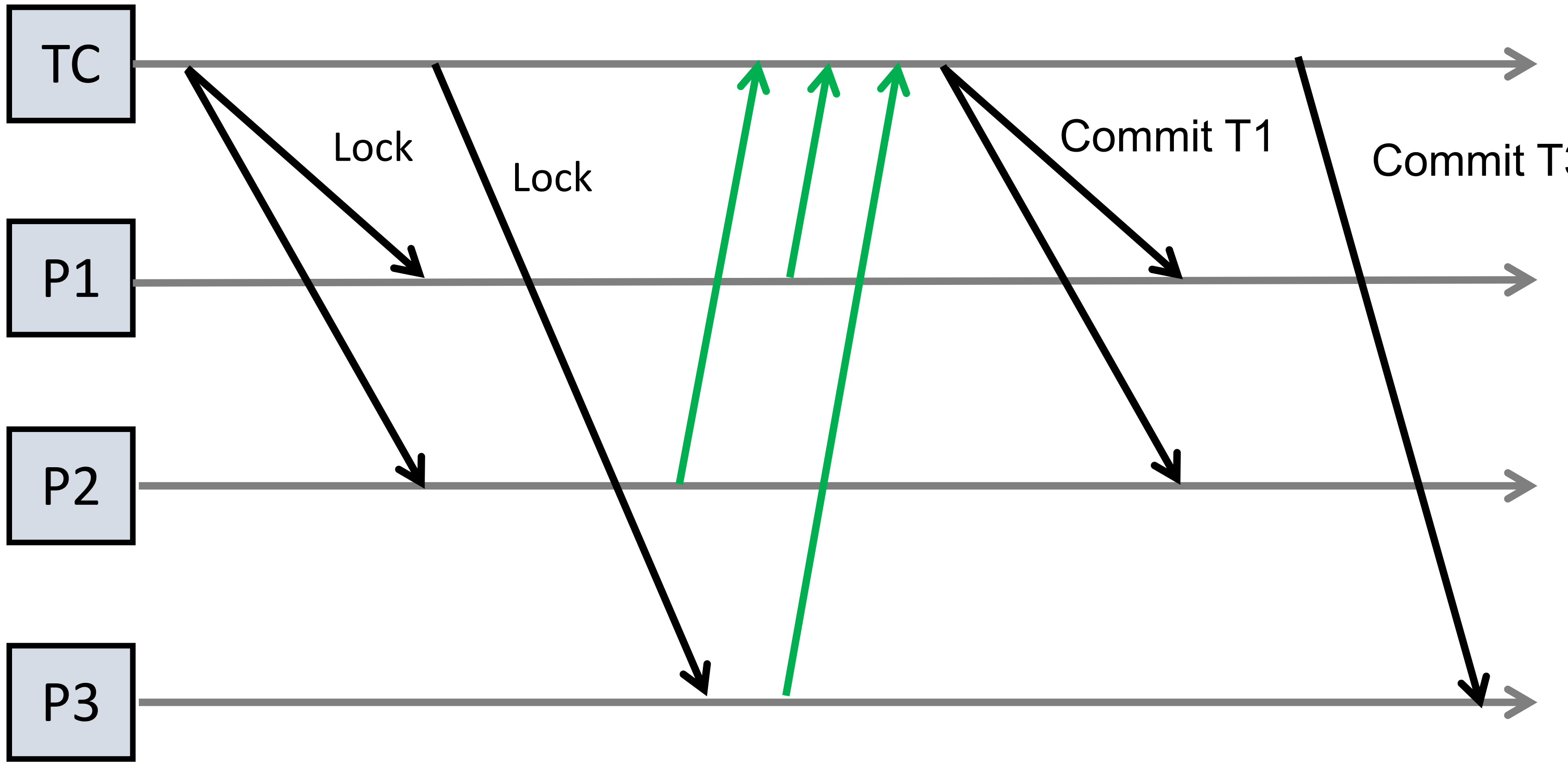
Example2: Serializability



Some of the possible Orderings:



Two Phase Locking


TC
P1
P2
P3
Lock
Lock
Commit T1
Commit T3