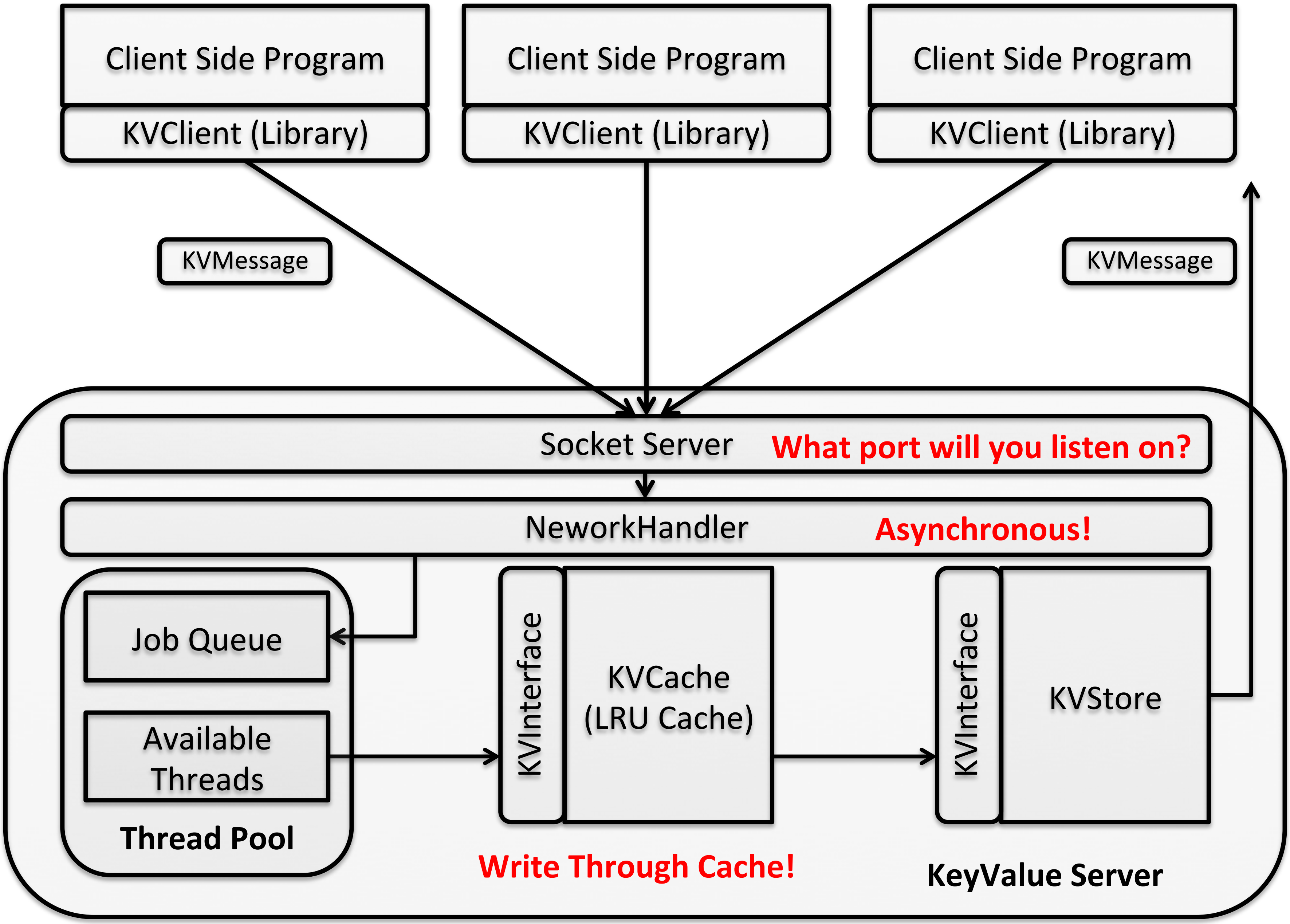
CS162 Section

Lecture 9

Project 3



Client Side Program

KVClient (Library)

Client Side Program

KVClient (Library)

Client Side Program

KVClient (Library)

KVMessage

KVMessage

Socket Server

What port will you listen on?

NeworkHandler

Asynchronous!

Job Queue

Available Threads

Thread Pool

KVInterface

KVCache (LRU Cache)

Write Through Cache!

KVInterface

KVStore

KeyValue Server

“A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable”

-- Leslie Lamport

What is Bandwidth?
What is Latency?

- RTT of 100ms
- Packet size of 1 KB
- Initial handshake of 2 RTTs
- Bandwidth is 1.5 Mbps

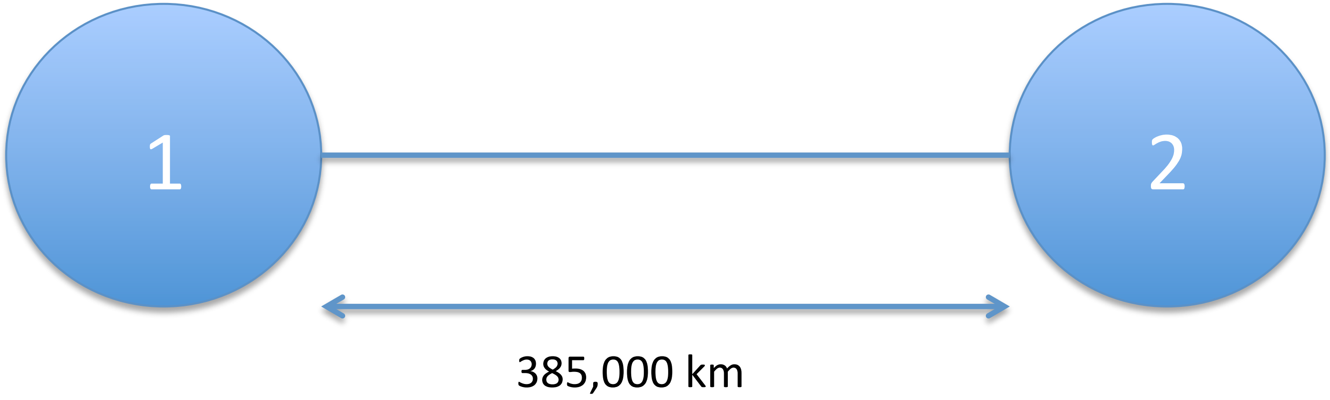
How long does it take to transfer a 1000 KB file.

Packet Transfer is continuous

- RTT of 100ms
- Packet size of 1 KB
- Initial handshake of 2 RTTs
- Bandwidth is 1.5 Mbps

How long does it take to transfer a 1000 KB file.

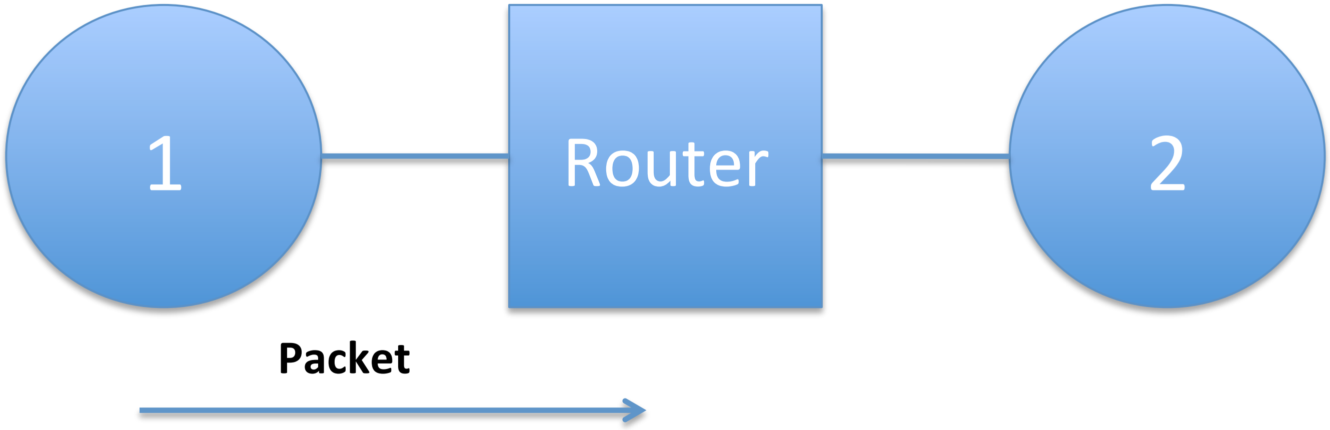
Packet Transfer is one at a time

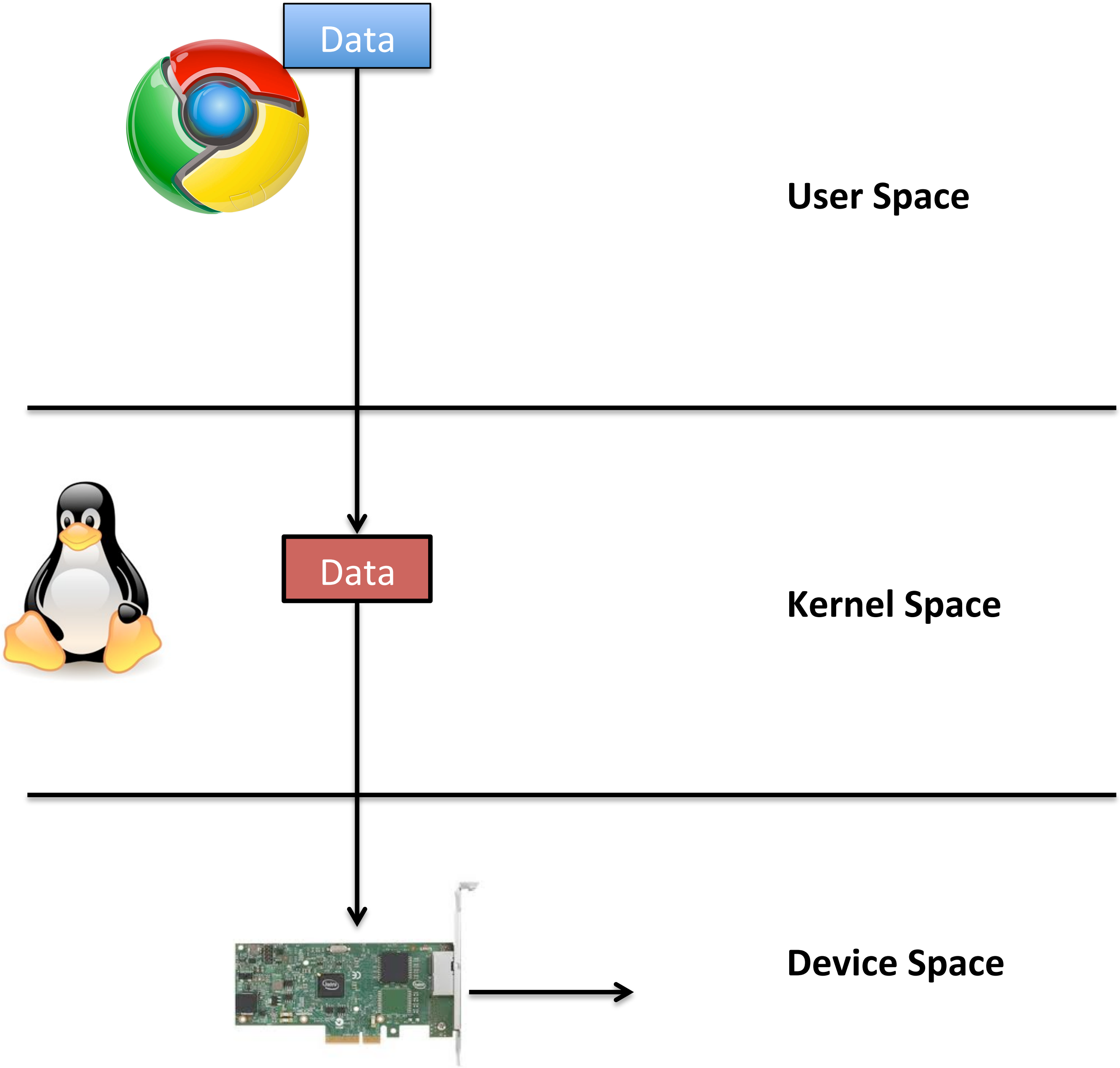


Transfer data at speed of light (3 x 10^8 m/s)

What is the RTT of the link?

How is a packet generated inside the Operating System?





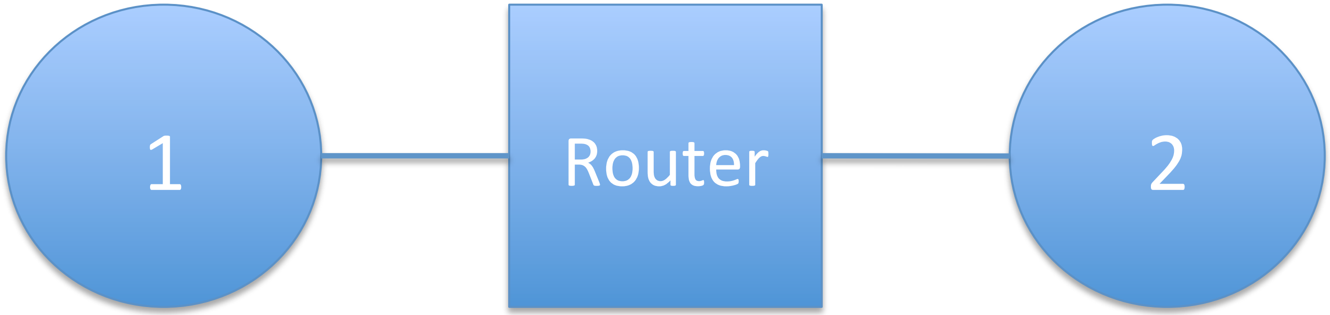
Data

User Space

Data

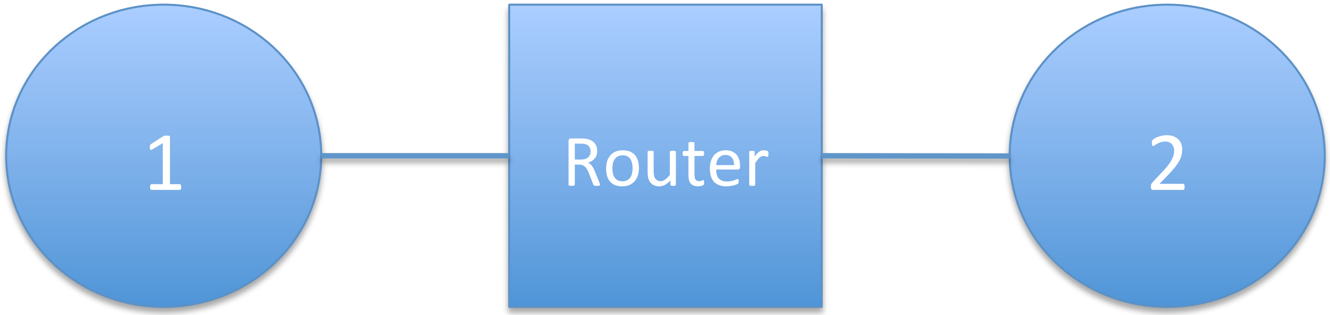
Kernel Space

Device Space



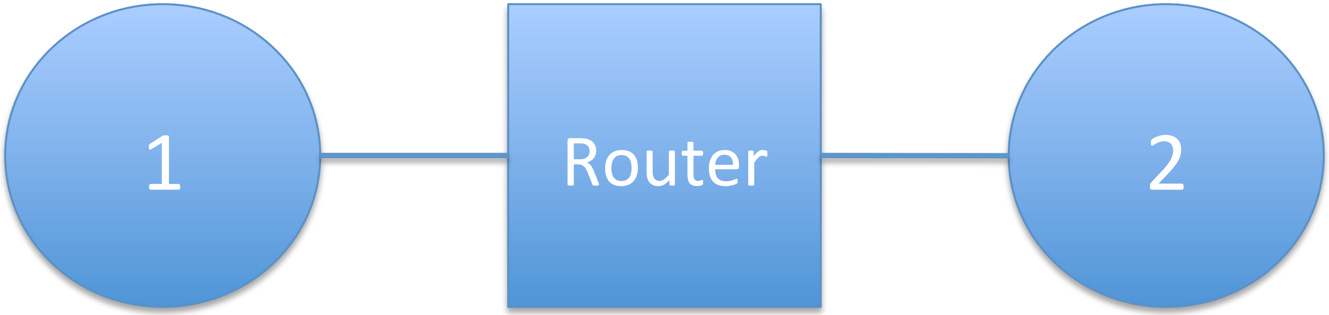
- Time to process an interrupt, T_{int} = 1.5 ms
- CPU cycle time, T_{CPU} = 10 ns
- Packet size, N_{pkt} = 1000B (ignore headers)
- Time to copy or DMA one byte, T_{copy} = 1 µs
- Latency of a link, T_{link} = 3 ms
- Bandwidth of links, B = 8Mbps

How many buffer copies are there to transmit a single packet?



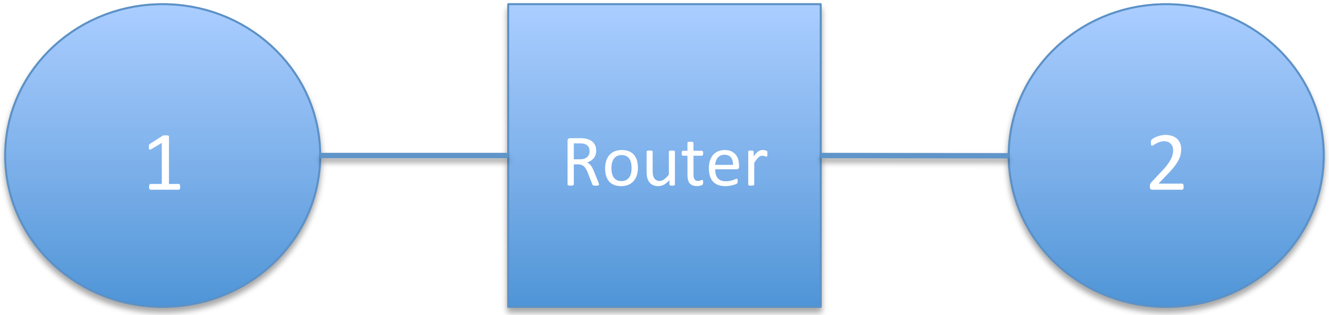
- Time to process an interrupt, T_{int} = 1.5 ms
- CPU cycle time, T_{CPU} = 10 ns
- Packet size, N_{pkt} = 1000B (ignore headers)
- Time to copy or DMA one byte, T_{copy} = 1 µs
- Latency of a link, T_{link} = 3 ms
- Bandwidth of links, B = 8Mbps

How many interrupts are generated to transmit a single packet?



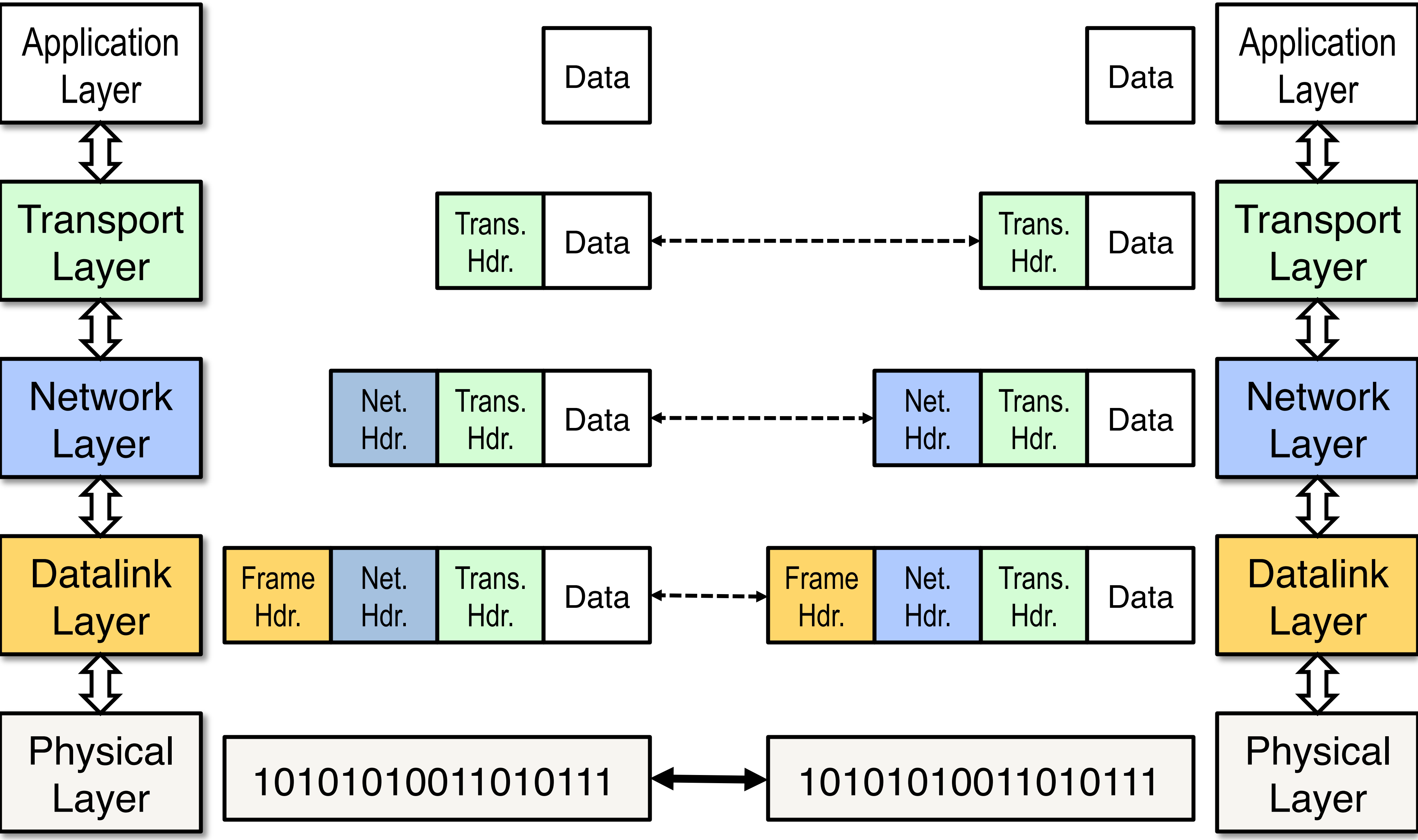
- Time to process an interrupt, T_{int} = 1.5 ms
- CPU cycle time, T_{CPU} = 10 ns
- Packet size, N_{pkt} = 1000B (ignore headers)
- Time to copy or DMA one byte, T_{copy} = 1 µs
- Latency of a link, T_{link} = 3 ms
- Bandwidth of links, B = 8Mbps

What is the time to reliably send a single packet from Node 1 to Node 2?



- Time to process an interrupt, T_{int} = 1.5 ms
- CPU cycle time, T_{CPU} = 10 ns
- Packet size, N_{pkt} = 1000B (ignore headers)
- Time to copy or DMA one byte, T_{copy} = 1 µs
- Latency of a link, T_{link} = 3 ms
- Bandwidth of links, B = 8Mbps

What is the maximum rate possible between Node 1 to Node 2?



Application Layer

Data

Data

Application Layer

Transport Layer

Trans. Hdr. | Data

Trans. Hdr. | Data

Transport Layer

Network Layer

Net. Hdr. | Trans. Hdr. | Data

Net. Hdr. | Trans. Hdr. | Data

Network Layer

Datalink Layer

Frame Hdr. | Net. Hdr. | Trans. Hdr. | Data

Frame Hdr. | Net. Hdr. | Trans. Hdr. | Data

Datalink Layer

Physical Layer

10101010011010111

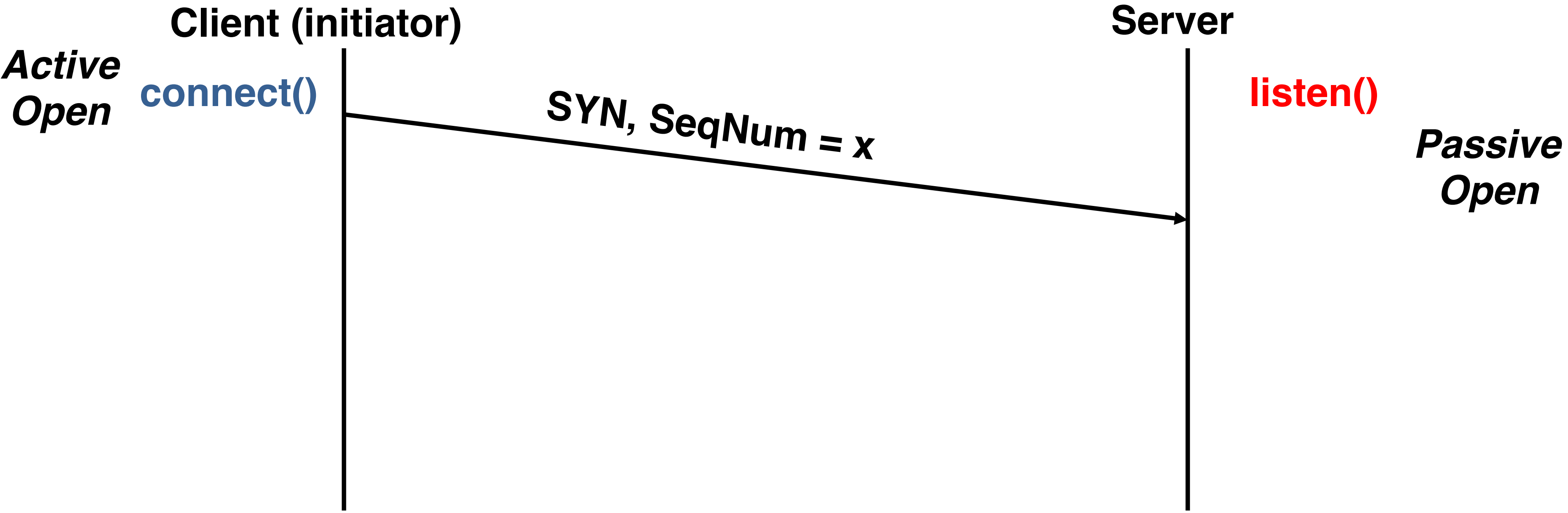
10101010011010111

Physical Layer

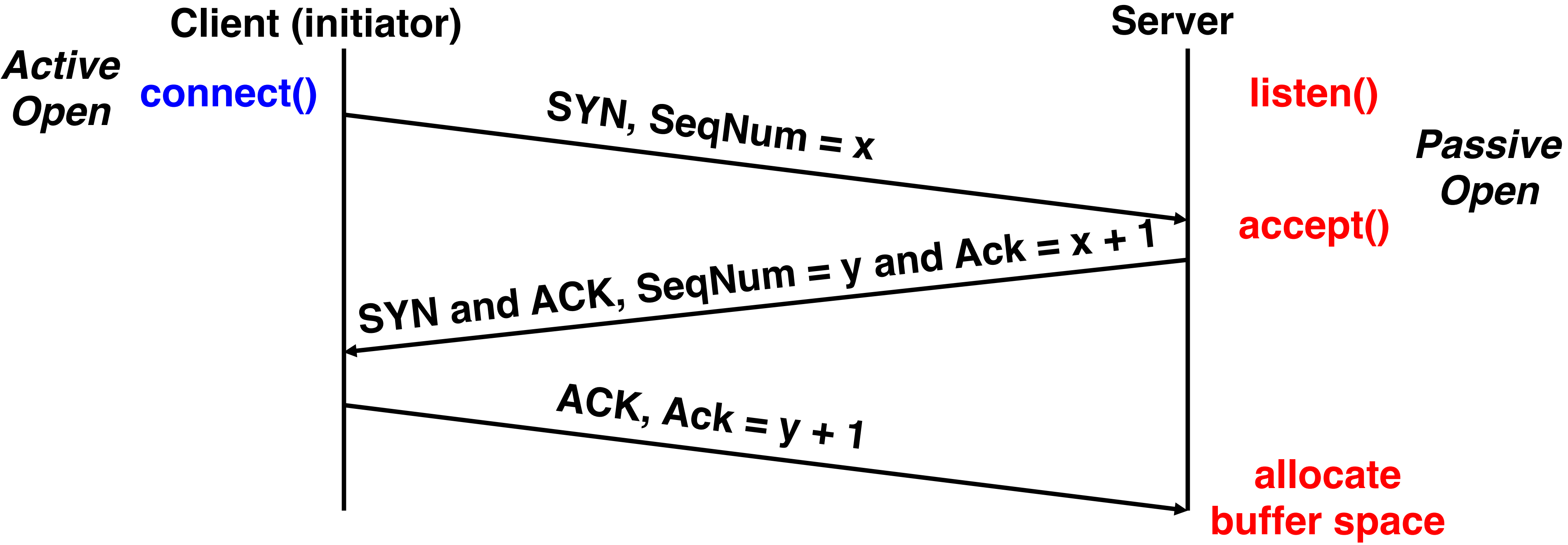
TCP

- Connection Setup – 3 way handshake
- Byte Stream Oriented (**NOTE!**)
- Reliable data transfer
- Flow control
- Congestion control
- **Connection tear down (4 way)**

TCP Connection Setup

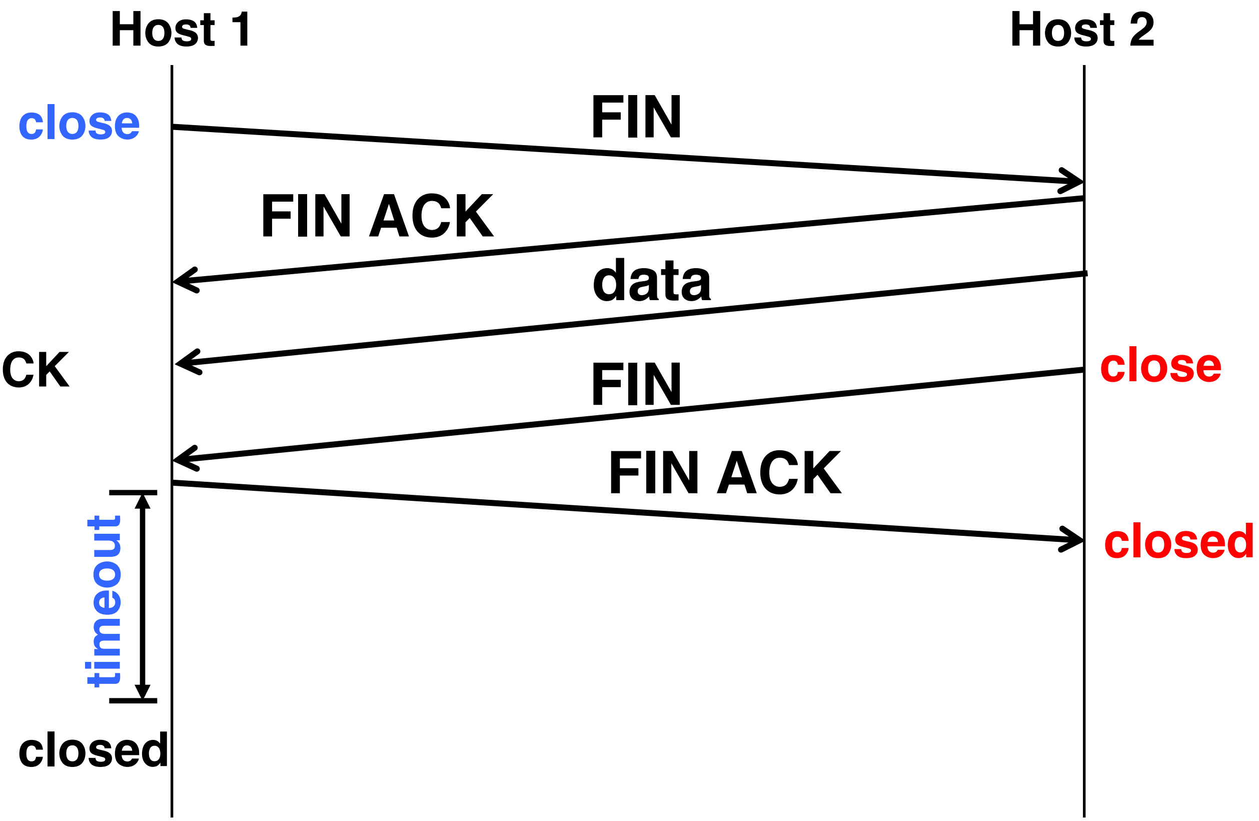


TCP Connection Setup



TCP Connection Teardown

- 4-ways tear down connection



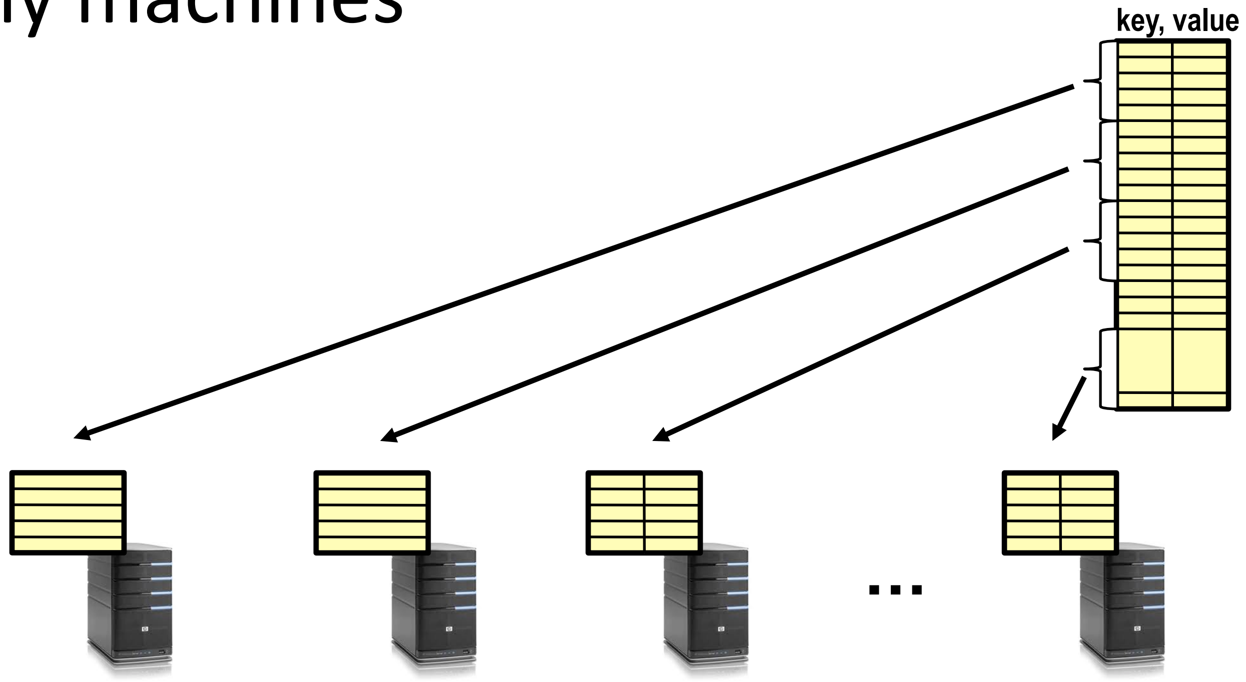
- Avoid reincarnation
- Can retransmit FIN ACK if it is lost

Key Value Stores

- Very large scale storage systems
- Two operations
  - put(key, value)
  - value = get(key)
- Challenges
  - Fault Tolerance → replication
  - Scalability → serve get()’s in parallel; replicate/cache hot tuples
  - Consistency → quorum consensus to improve put() performance

Key Value Stores

- Also called a Distributed Hash Table (DHT)
- Main idea: partition set of key-values across many machines

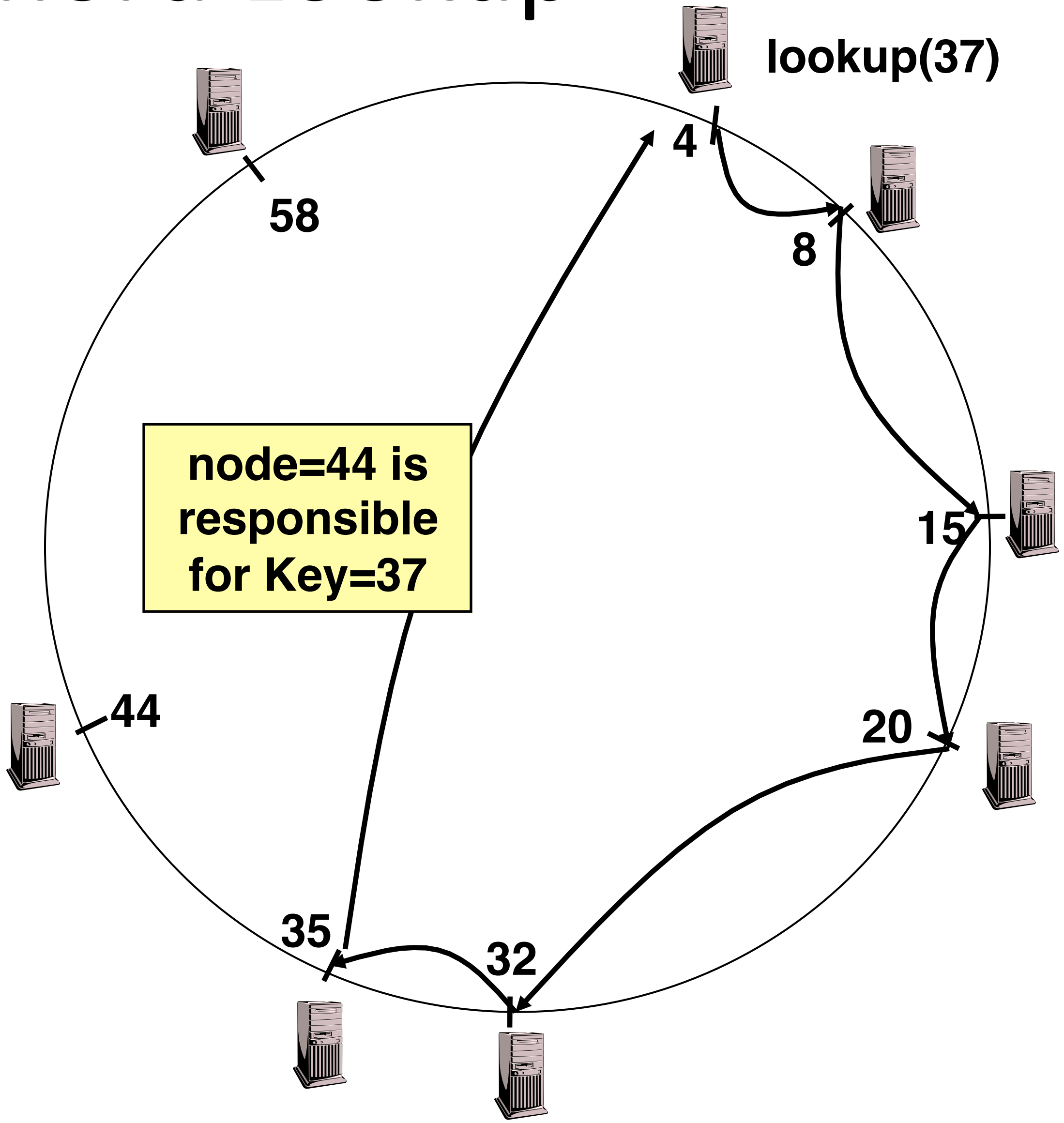


Chord Lookup

- Each node maintains pointer to its successor

- Route packet (Key, Value) to the node responsible for ID using successor pointers

- E.g., node=4 lookups for node responsible for Key=37



Chord

- Highly scalable distributed lookup protocol
- Each node needs to know about O(log(*M*)), where *M* is the total number of nodes
- Guarantees that a tuple is found in O(log(*M*)) steps
- Highly resilient: works with high probability even if half of nodes fail