Overlay Networks

Reading: 9.4

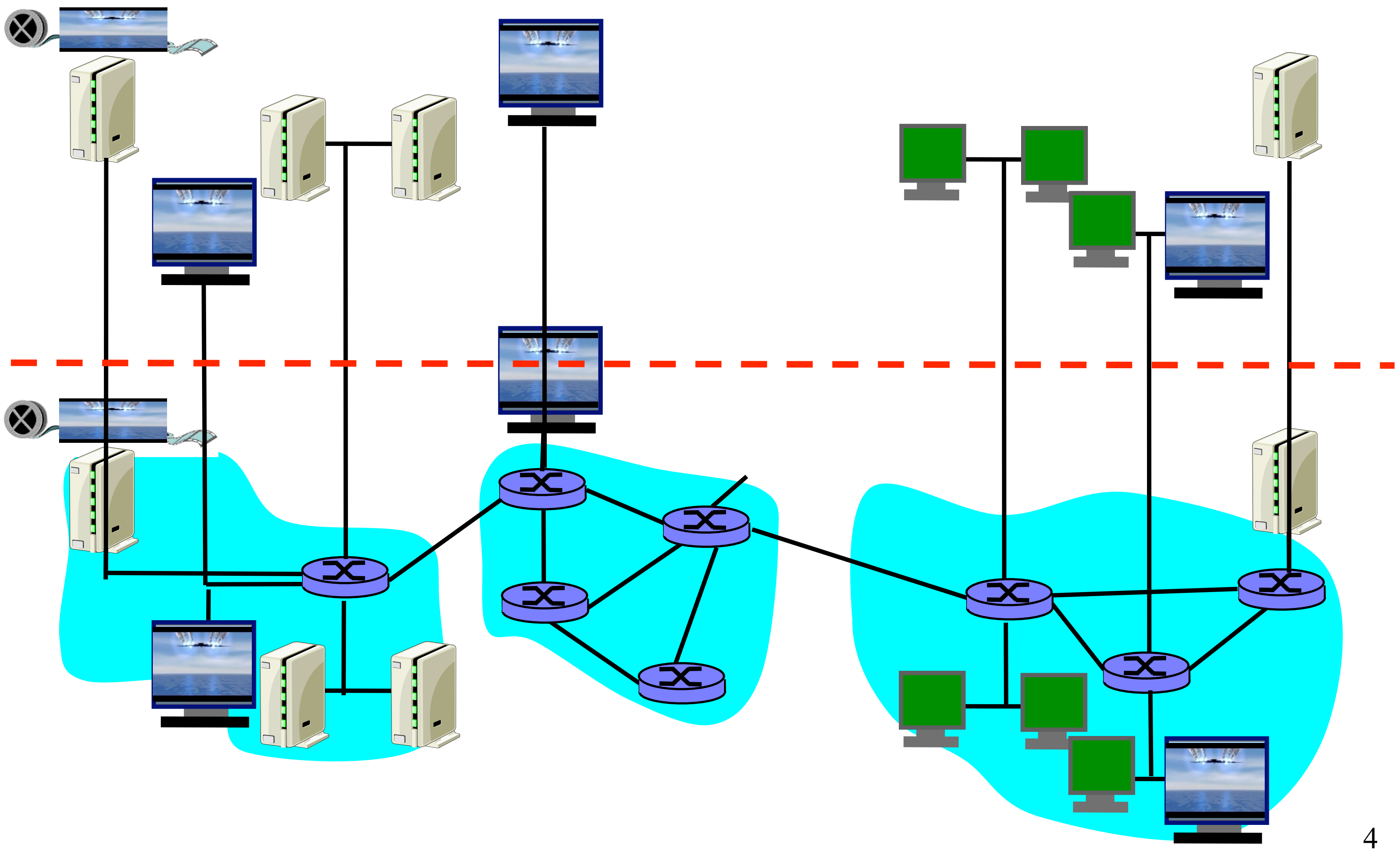
Acknowledgments: Lecture slides are from Computer networks course thought by Jennifer Rexford at Princeton University. When slides are obtained from other sources, a reference will be noted on the bottom of that slide and full reference details on the last slide.

Goals of Today's Lecture

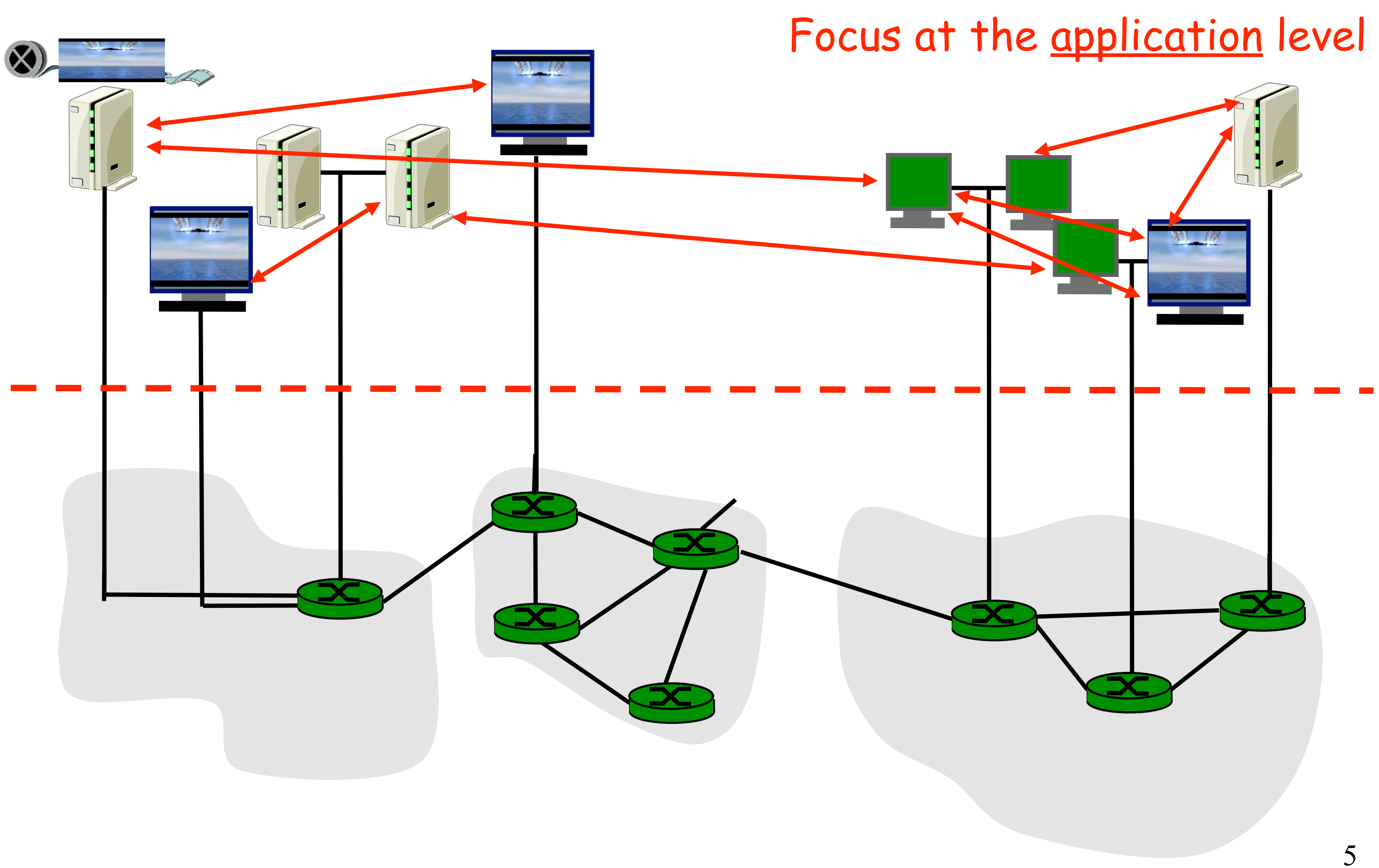
- Motivations for overlay networks
  - Incremental deployment of new protocols
  - Customized routing and forwarding solutions
- Overlays for partial deployments
  - 6Bone, Mbone, security, …
- Resilient Overlay Network (RON)
  - Adaptive routing through intermediate node
- Distributed Hash Table (DHT)
  - Overlay for look-up of <key, value> pairs

Overlay Networks

Overlay Networks

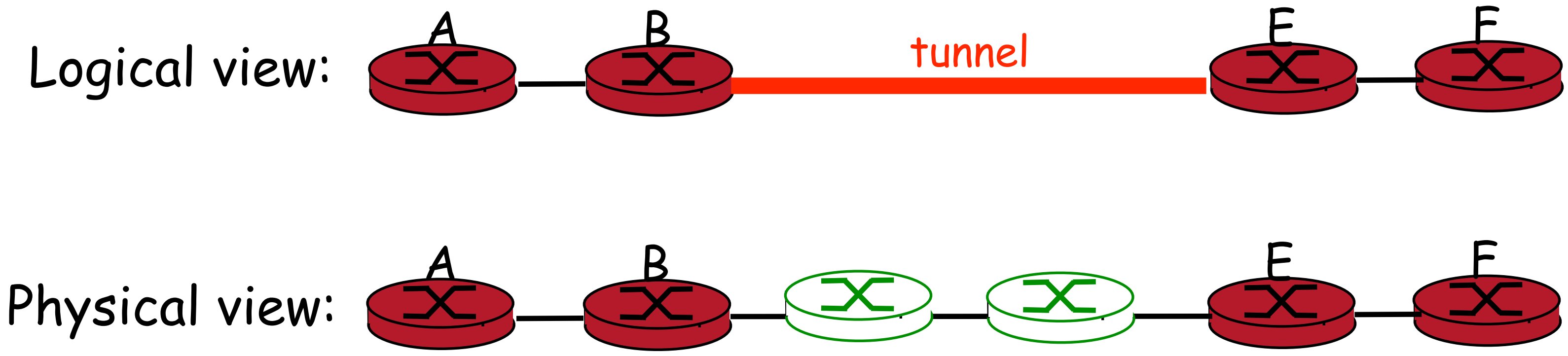


Overlay Networks



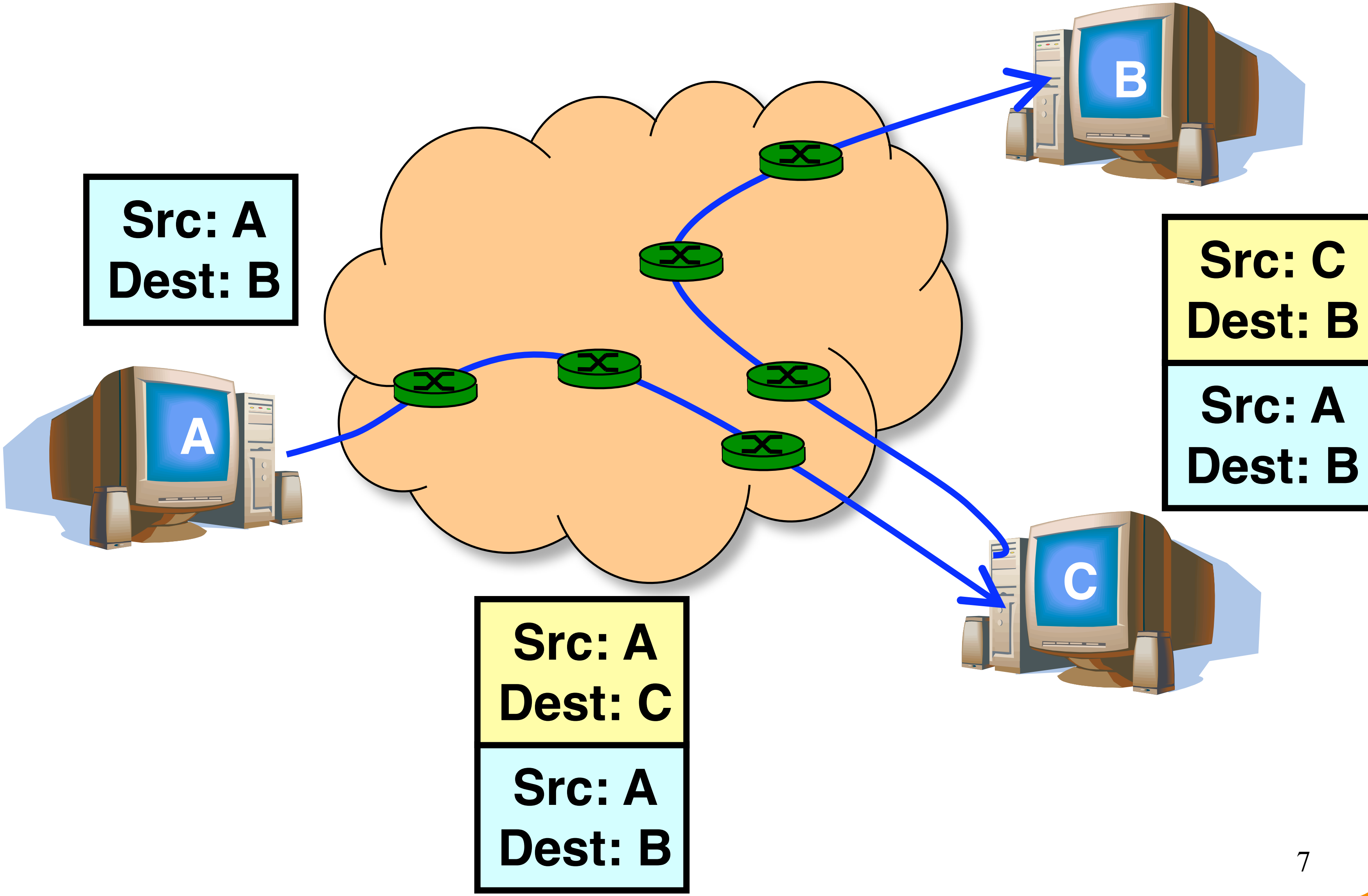
IP Tunneling to Build Overlay Links

- IP tunnel is a virtual point-to-point link
  - Illusion of a direct link between two separated nodes



- Encapsulation of the packet inside an IP datagram
  - Node B sends a packet to node E
  - … containing another packet as the payload

Tunnels Between End Hosts



Overlay Networks

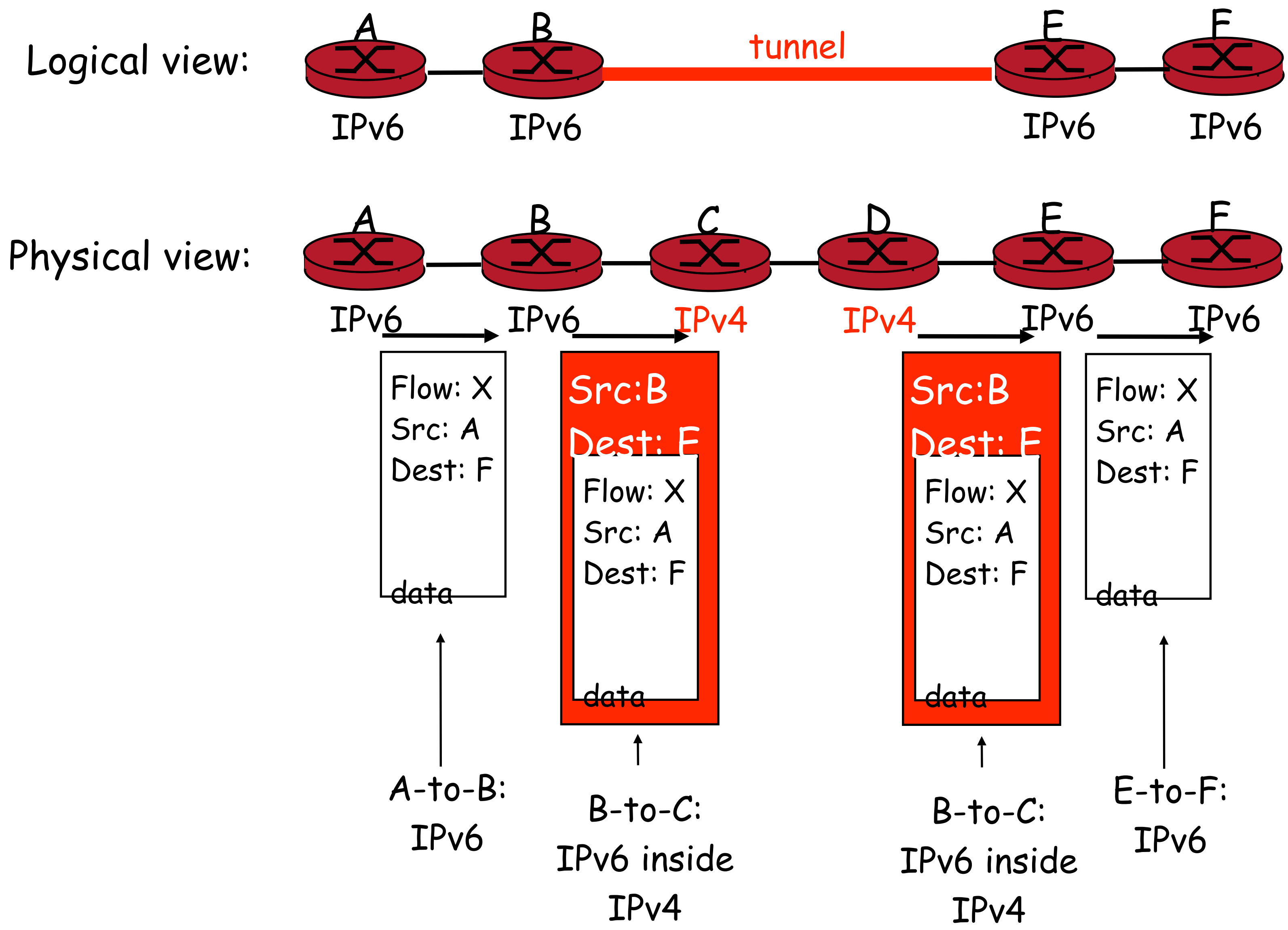
- A logical network built on top of a physical network
  - Overlay links are tunnels through the underlying network
- Many logical networks may coexist at once
  - Over the same underlying network
  - And providing its own particular service
- Nodes are often end hosts
  - Acting as intermediate nodes that forward traffic
  - Providing a service, such as access to files
- Who controls the nodes providing service?
  - The party providing the service
  - Distributed collection of end users

Overlays for Incremental Deployment

Using Overlays to Evolve the Internet

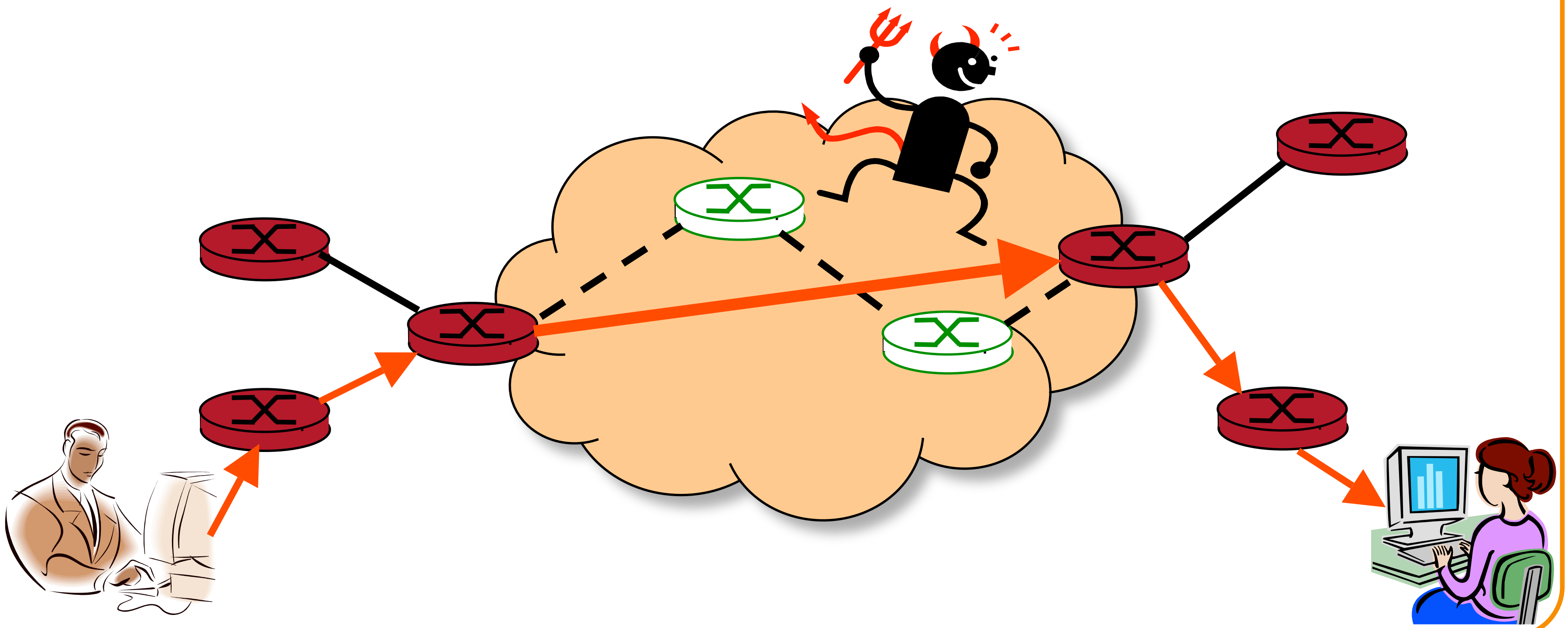
- Internet needs to evolve
  - IPv6
  - Security
  - Multicast
- But, global change is hard
  - Coordination with many ASes
  - “Flag day” to deploy and enable the technology
- Instead, better to incrementally deploy
  - And find ways to bridge deployment gaps

6Bone: Deploying IPv6 over IP4



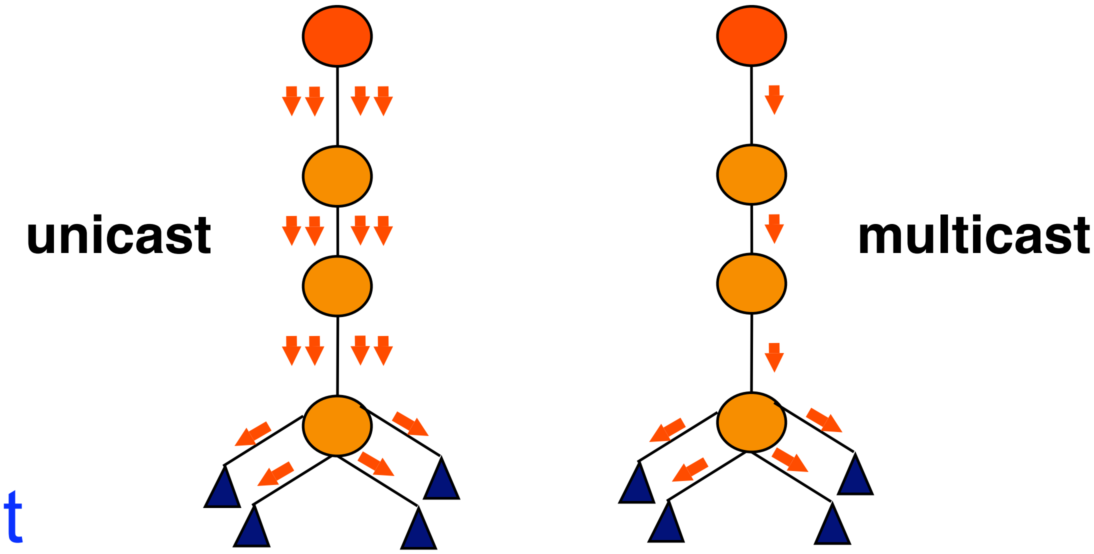
Secure Communication Over Insecure Links

- Encrypt packets at entry and decrypt at exit
- Eavesdropper cannot snoop the data
- … or determine the real source and destination



IP Multicast

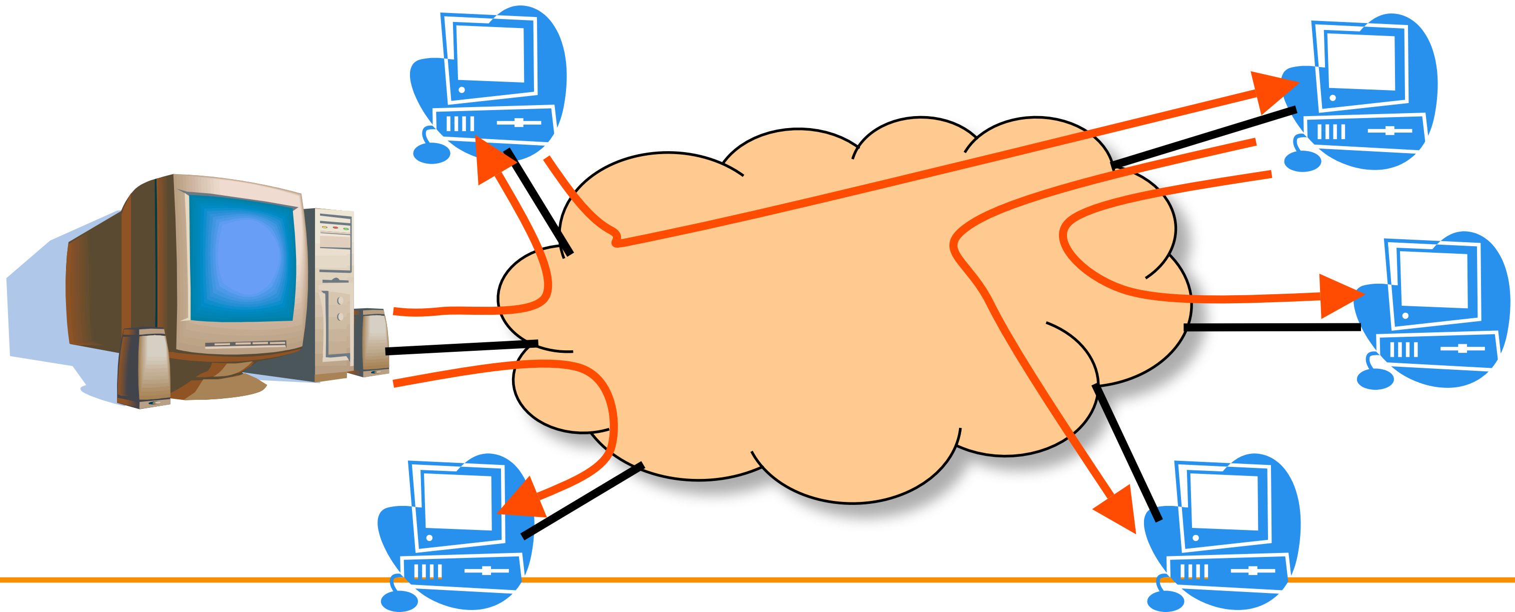
- Multicast
  - Delivering the same data to many receivers
  - Avoiding sending the same data many times



- IP multicast
  - Special addressing, forwarding, and routing schemes
  - Pretty complicated stuff (see Section 4.4)

MBone: Multicast Backbone

- Deploying multicast
  - Router vendors wouldn't support IP multicast
  - … since they weren't sure anyone would use it
  - And, since it didn't exist, nobody was using it
- Idea: software implementing multicast protocols
  - And unicast tunnels to traverse non-participants



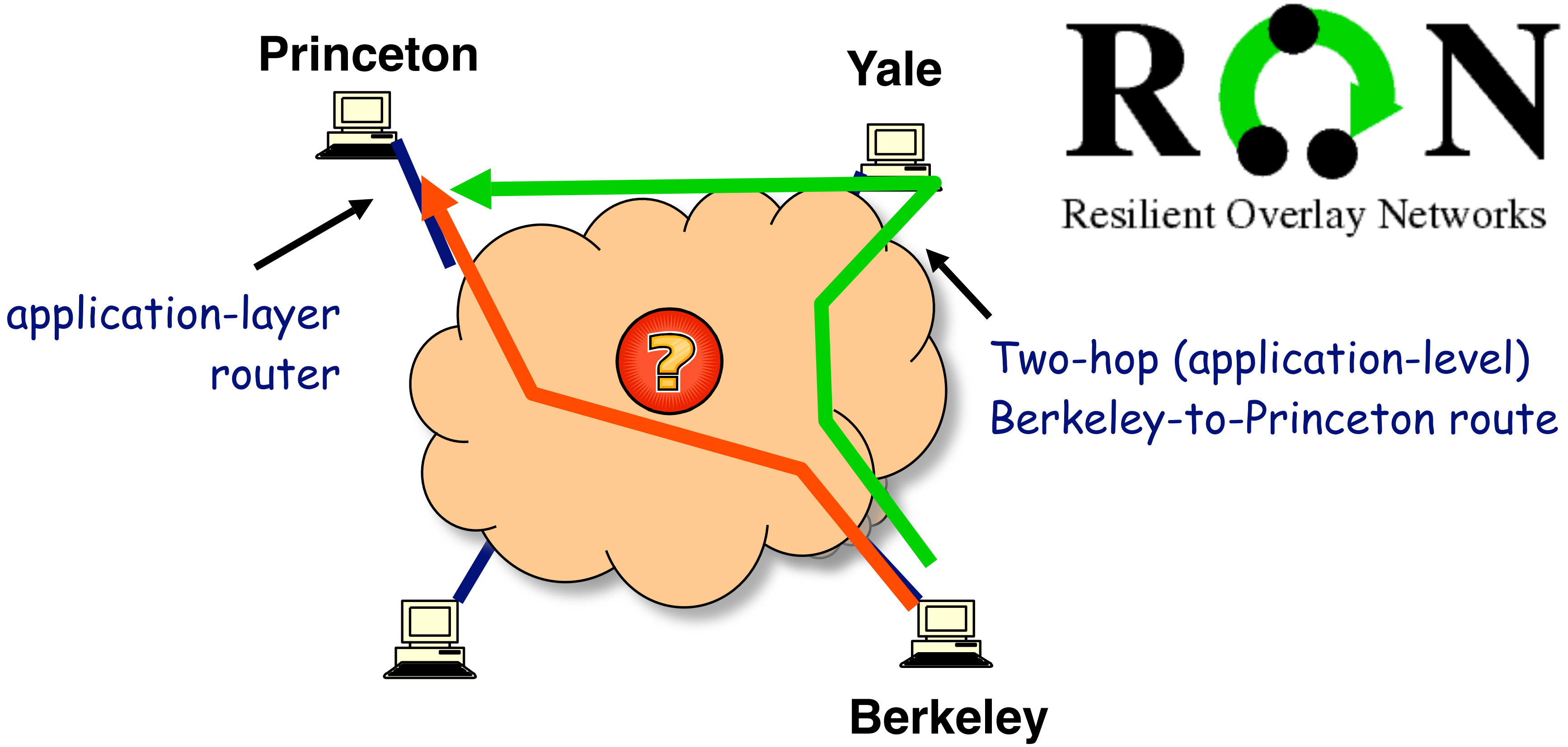
Multicast Today

- Mbone applications starting in early 1990s
  - Primarily video conferencing, but no longer operational
- Still many challenges to deploying IP multicast
  - Security vulnerabilities, business models, …
- Application-layer multicast is more prevalent
  - Tree of servers delivering the content
  - Collection of end hosts cooperating to delivery video
- Some multicast within individual ASes
  - Financial sector: stock tickers
  - Within campuses or broadband networks: TV shows
  - Backbone networks: IPTV

Case Study: Resilient Overlay Networks

RON: Resilient Overlay Networks

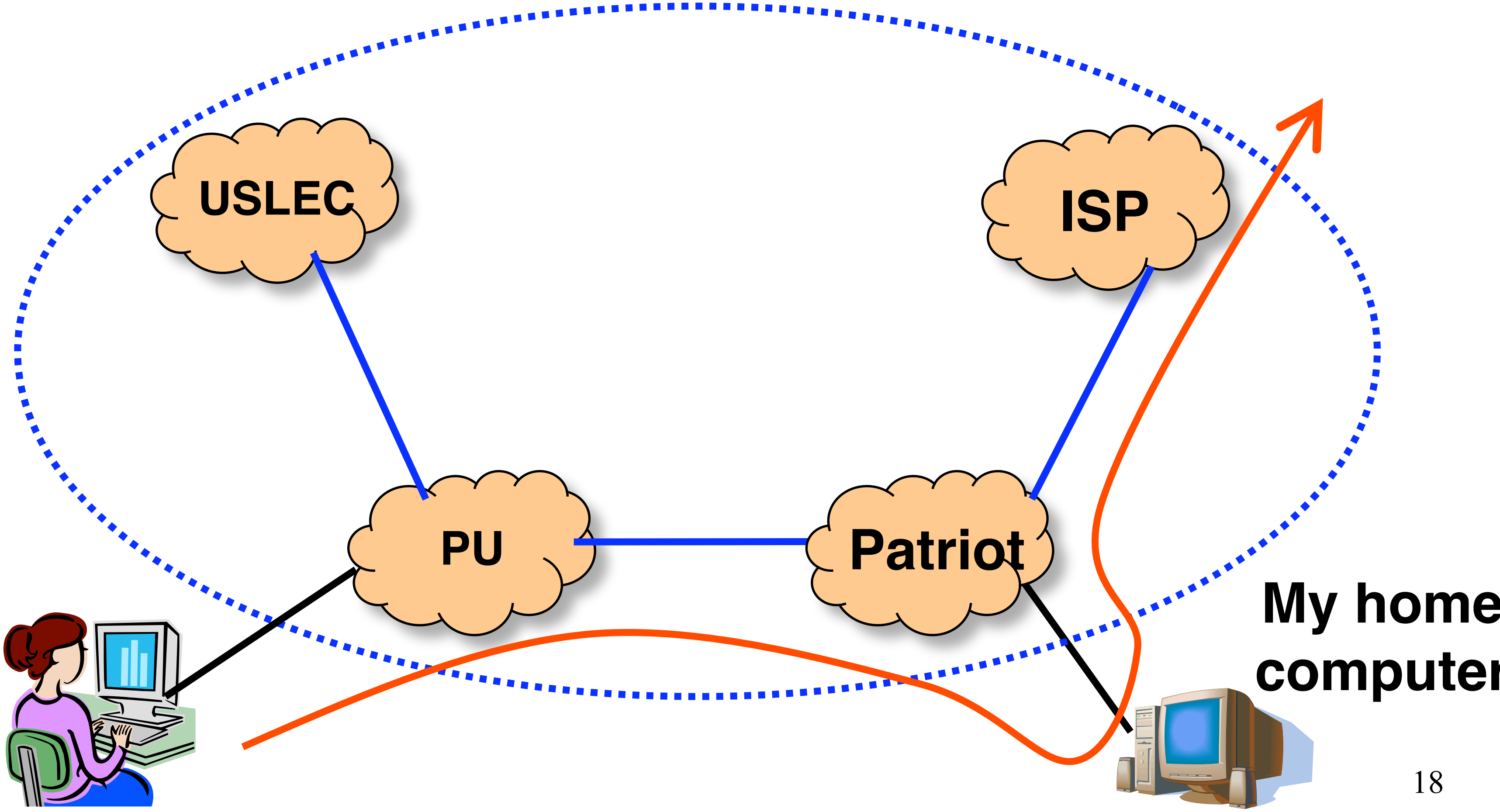
Premise: by building application overlay network, can increase performance and reliability of routing



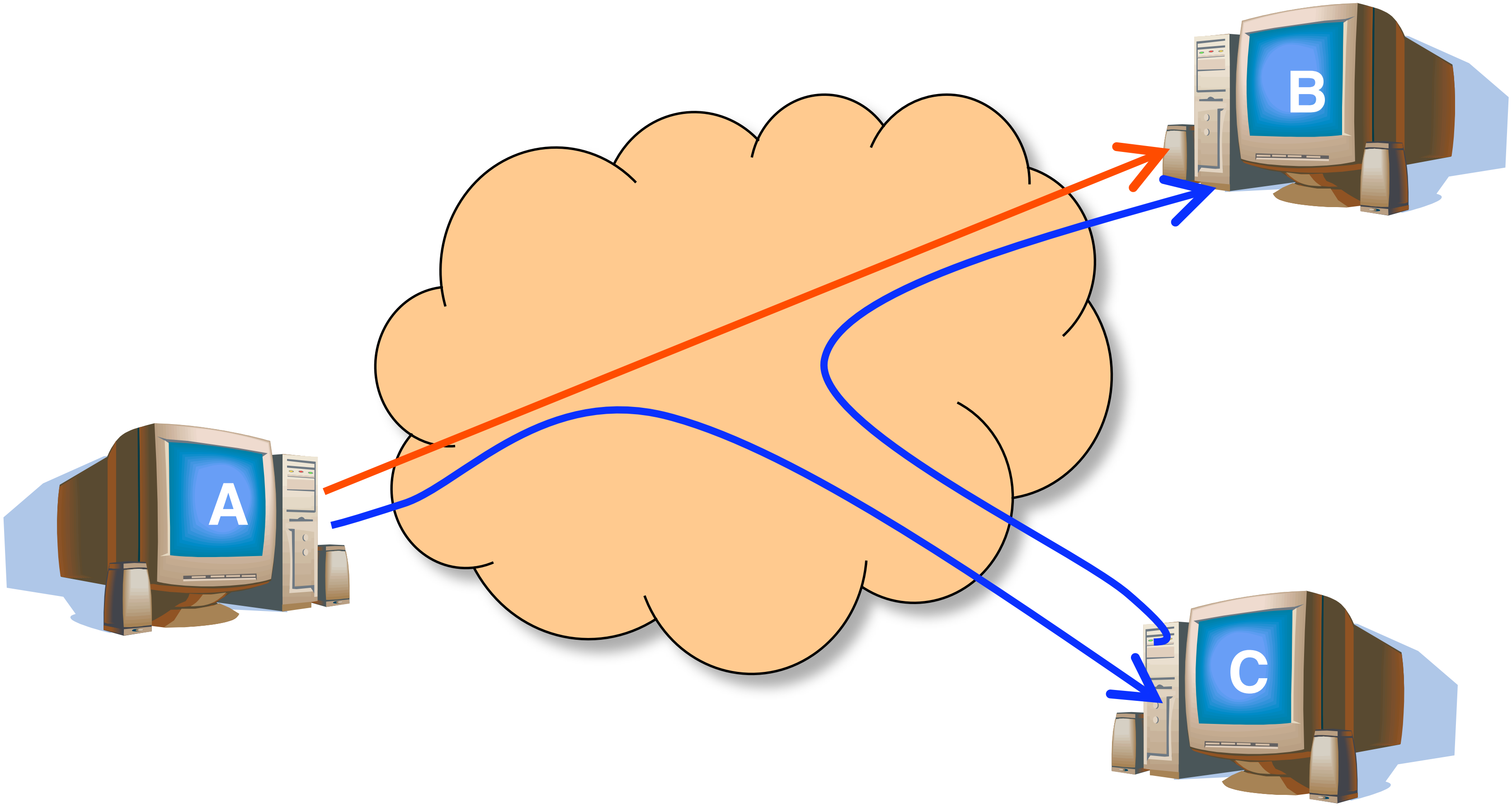
http://nms.csail.mit.edu/ron/

RON Circumvents Policy Restrictions

- IP routing depends on AS routing policies
  - But hosts may pick paths that circumvent policies

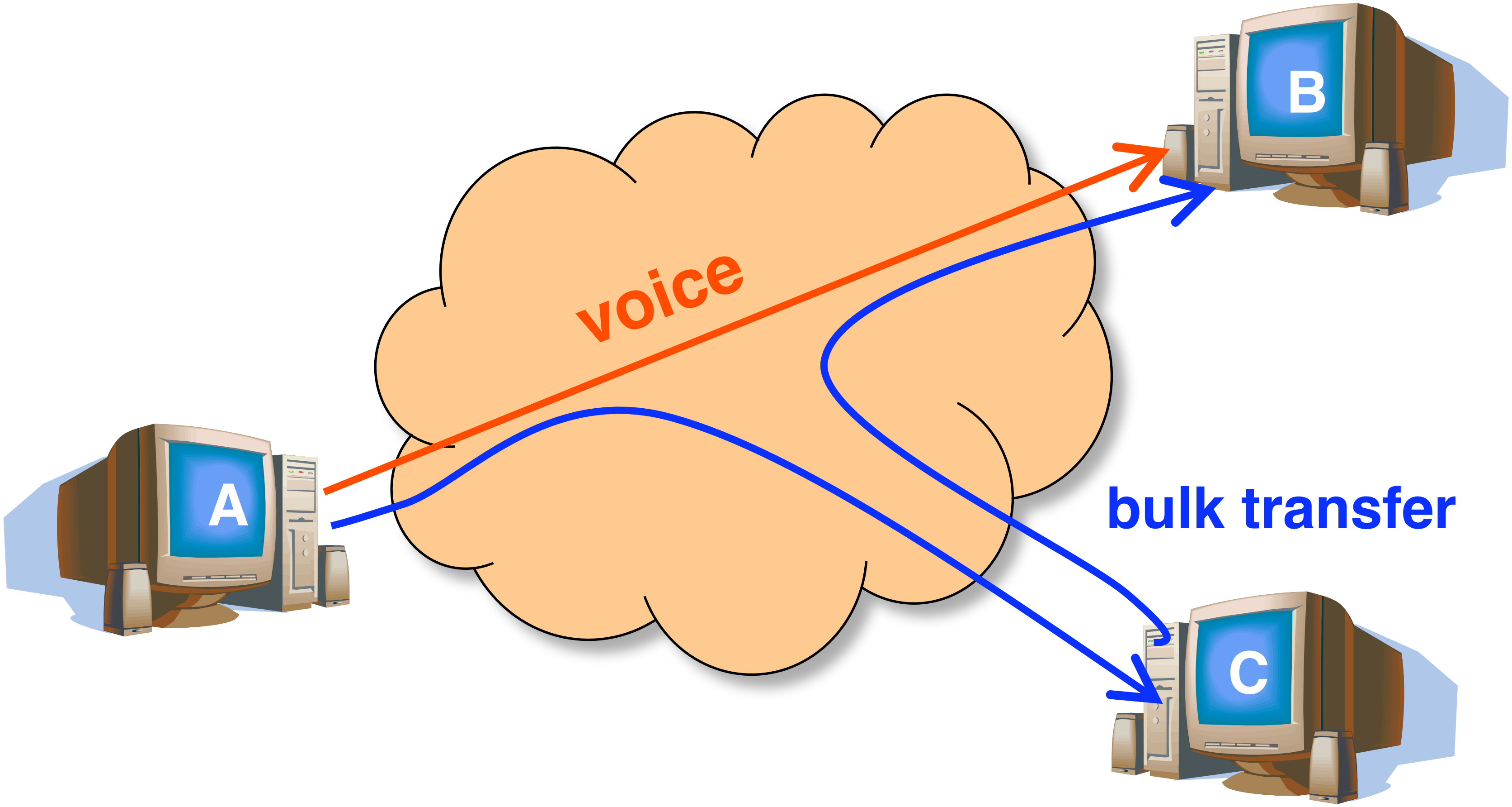


RON Adapts to Network Conditions



- Start experiencing bad performance
  - Then, start forwarding through intermediate host

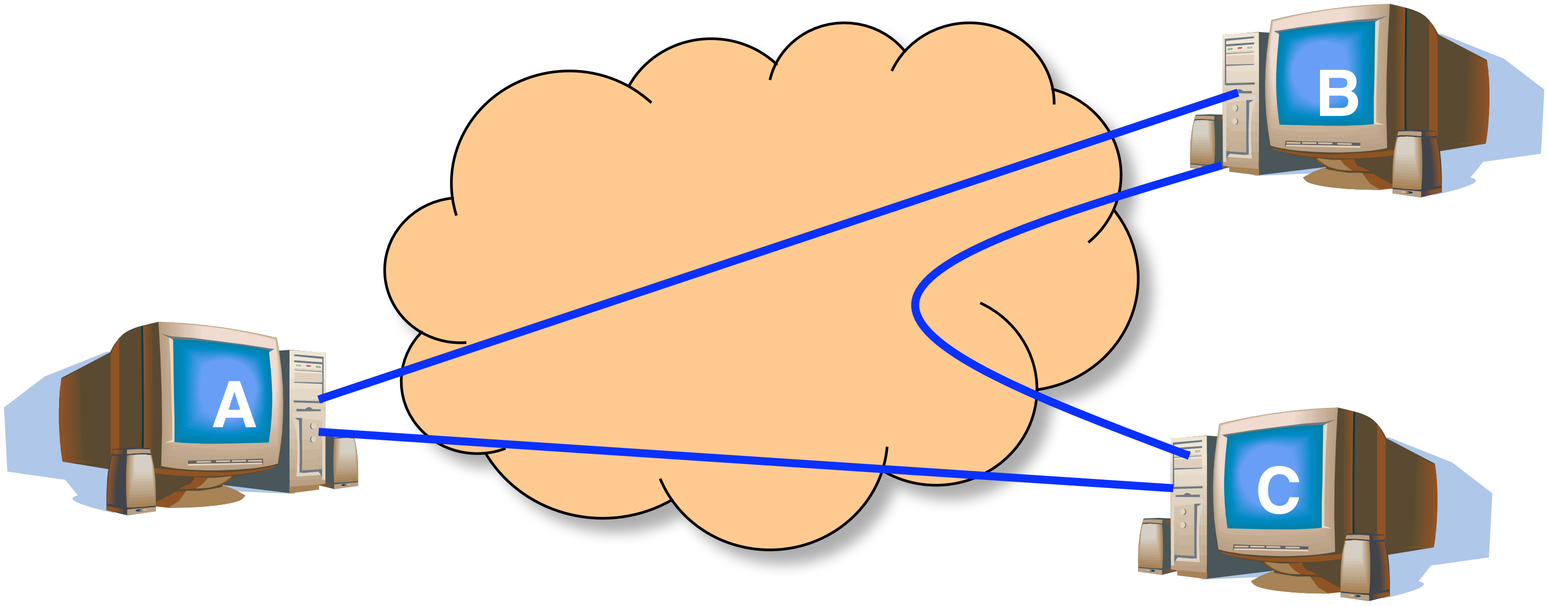
RON Customizes to Applications



- VoIP traffic: low-latency path
- Bulk transfer: high-bandwidth path

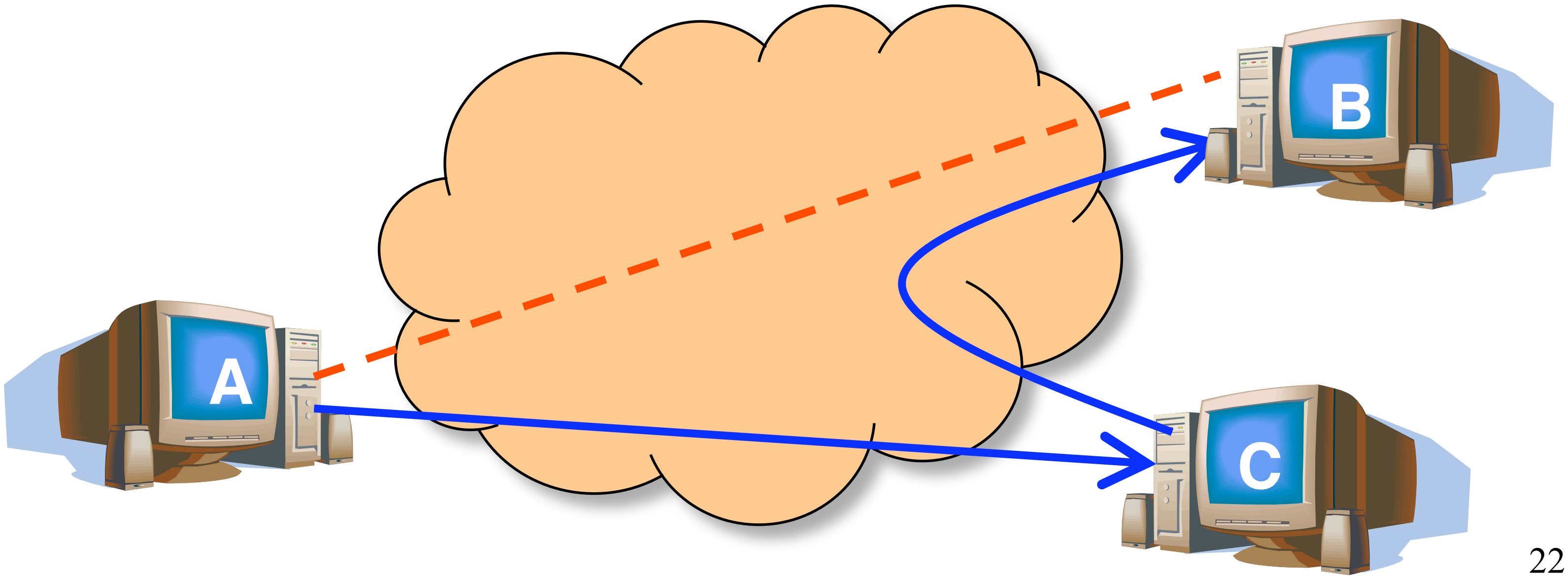
How Does RON Work?

- Keeping it small to avoid scaling problems
  - A few friends who want better service
  - Just for their communication with each other
  - E.g., VoIP, gaming, collaborative work, etc.
- Send probes between each pair of hosts



How Does RON Work?

- Exchange the results of the probes
  - Each host shares results with every other host
  - Essentially running a link-state protocol!
  - So, every host knows the performance properties
- Forward through intermediate host when needed



RON Works in Practice

- Faster reaction to failure
  - RON reacts in a few seconds
  - BGP sometimes takes a few minutes
- Single-hop indirect routing
  - No need to go through many intermediate hosts
  - One extra hop circumvents the problems
- Better end-to-end paths
  - Circumventing routing policy restrictions
  - Sometimes the RON paths are actually shorter

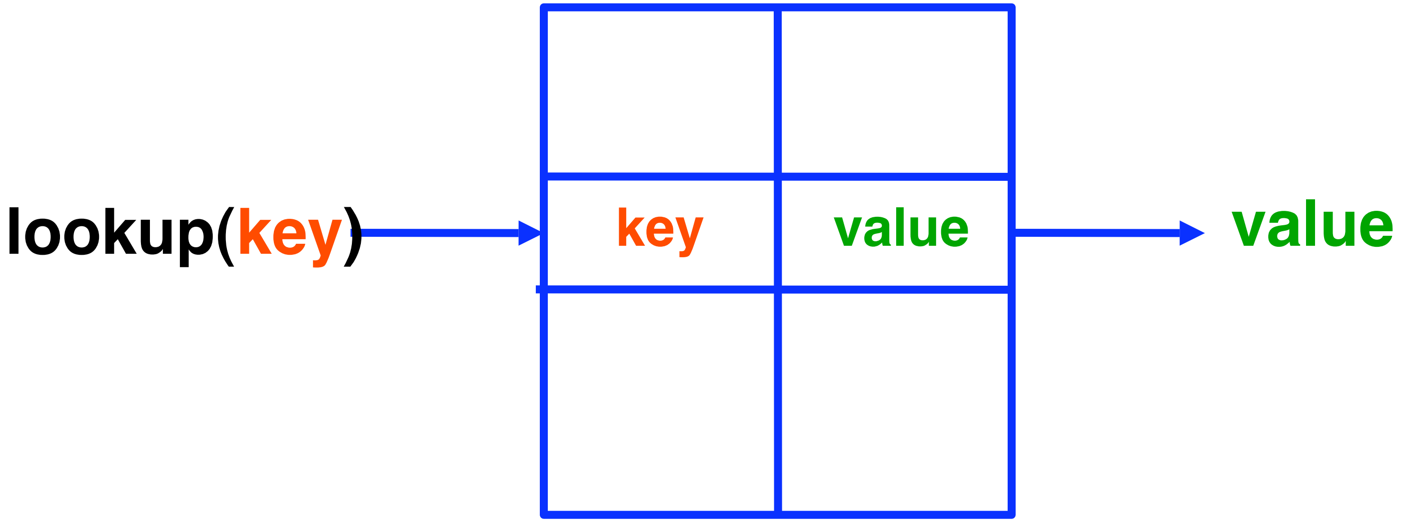
RON Limited to Small Deployments

- Extra latency through intermediate hops
  - Software delays for packet forwarding
  - Propagation delay across the access link
- Overhead on the intermediate node
  - Imposing CPU and I/O load on the host
  - Consuming bandwidth on the access link
- Overhead for probing the virtual links
  - Bandwidth consumed by frequent probes
  - Trade-off between probe overhead and detection speed
- Possibility of causing instability
  - Moving traffic in response to poor performance
  - May lead to congestion on the new paths

Case Study: Distributed Hash Tables

Hash Table

- Name-value pairs (or key-value pairs)
  - E.g,. “Website Admin” and admin@ce.sharif.edu
  - E.g., “http://www.sharif.edu/foo.html” and the Web page
  - E.g., “Betoven.mp3” and “213.233.168.67”
- Hash table
  - Data structure that associates keys with values



Distributed Hash Table

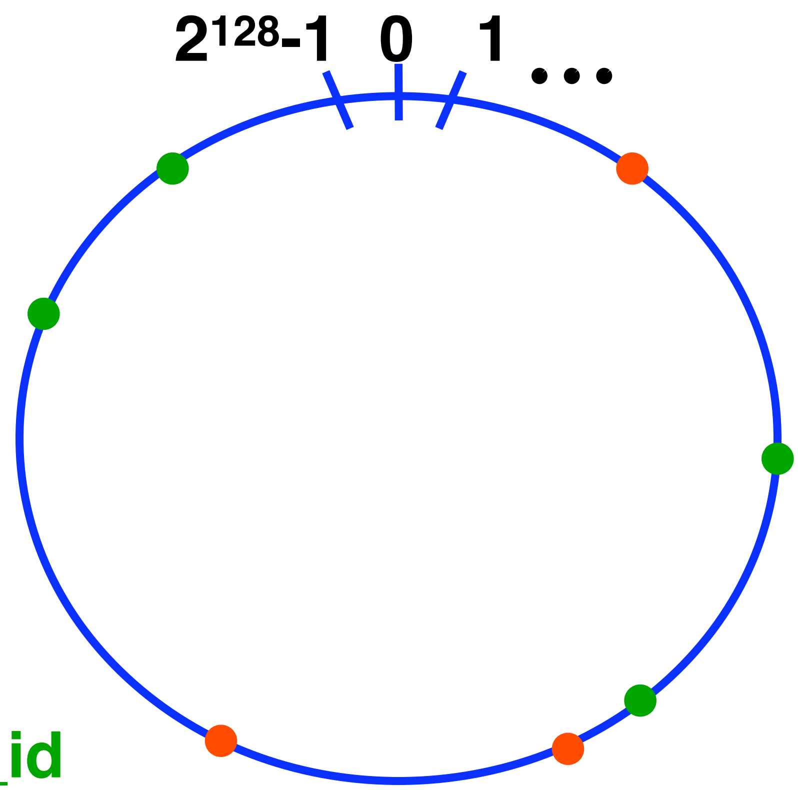
- Hash table spread over many nodes
  - Distributed over a wide area
- Main design goals
  - Decentralization: no central coordinator
  - Scalability: efficient even with large # of nodes
  - Fault tolerance: tolerate nodes joining/leaving
- Two key design decisions
  - How do we map names on to nodes?
  - How do we route a request to that node?

Hash Functions

- Hashing
  - Transform the key into a number
  - And use the number to index an array
- Example hash function
  - Hash(x) = x mod 101, mapping to 0, 1, …, 100
- Challenges
  - What if there are more than 101 nodes? Fewer?
  - Which nodes correspond to each hash value?
  - What if nodes come and go over time?

Consistent Hashing

- Large, sparse identifier space (e.g., 128 bits)
  - Hash a set of keys x uniformly to large id space
  - Hash nodes to the id space as well



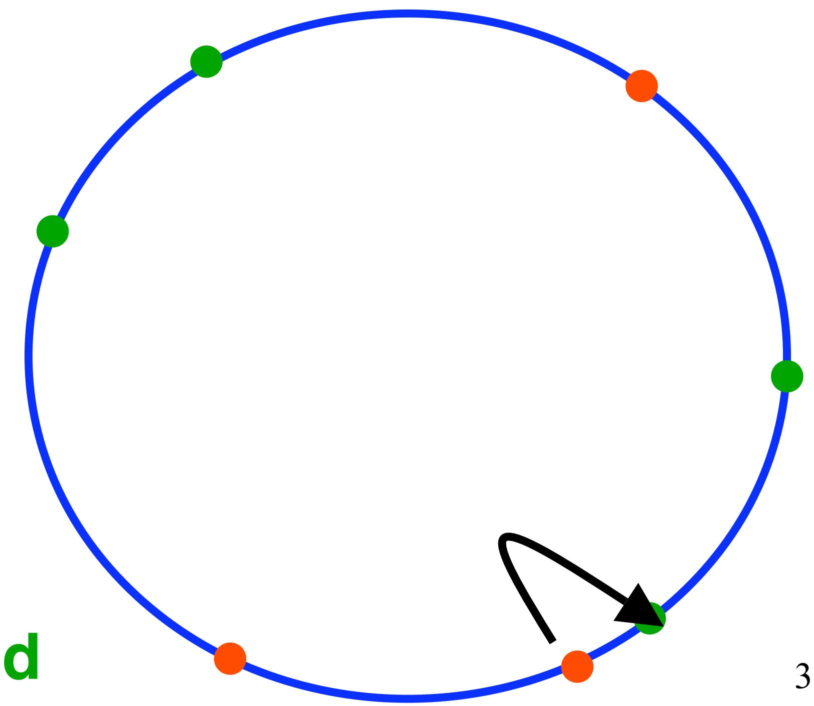
Id space represented as a ring.

Hash(name) → object_id

Hash(IP_address) → node_id

Where to Store (Key, Value) Pair?

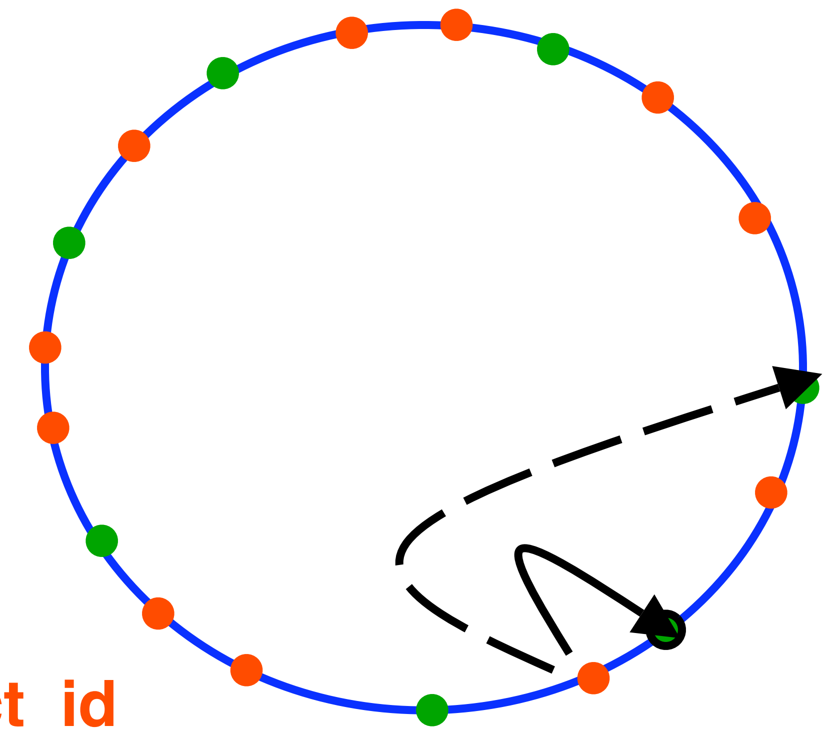
- Mapping keys in a load-balanced way
  - Store the key at one or more nodes
  - Nodes with identifiers “close” to the key
  - Where distance is measured in the id space
- Advantages
  - Even distribution
  - Few changes as nodes come and go…



Hash(name) → object_id
Hash(IP_address) → node_id

Nodes Coming and Going

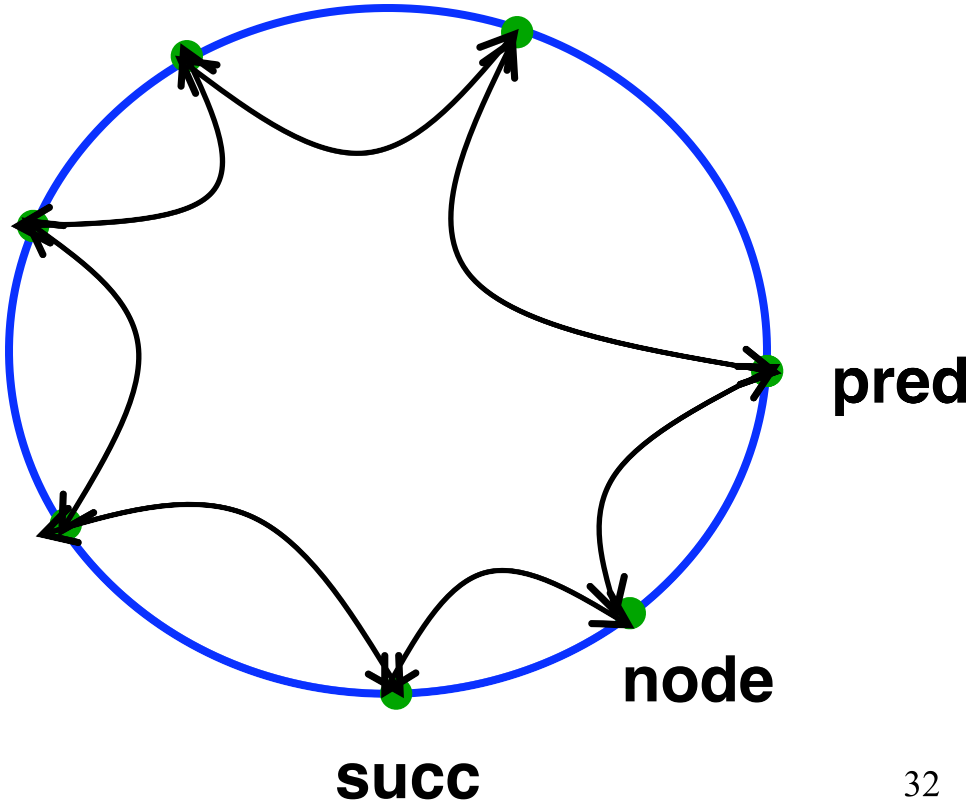
- Small changes when nodes come and go
  - Only affects mapping of keys mapped to the node that comes or goes



Hash(name) → object_id
Hash(IP_address) → node_id

Joins and Leaves of Nodes

- Maintain a circularly linked list around the ring
  - Every node has a predecessor and successor



Joins and Leaves of Nodes

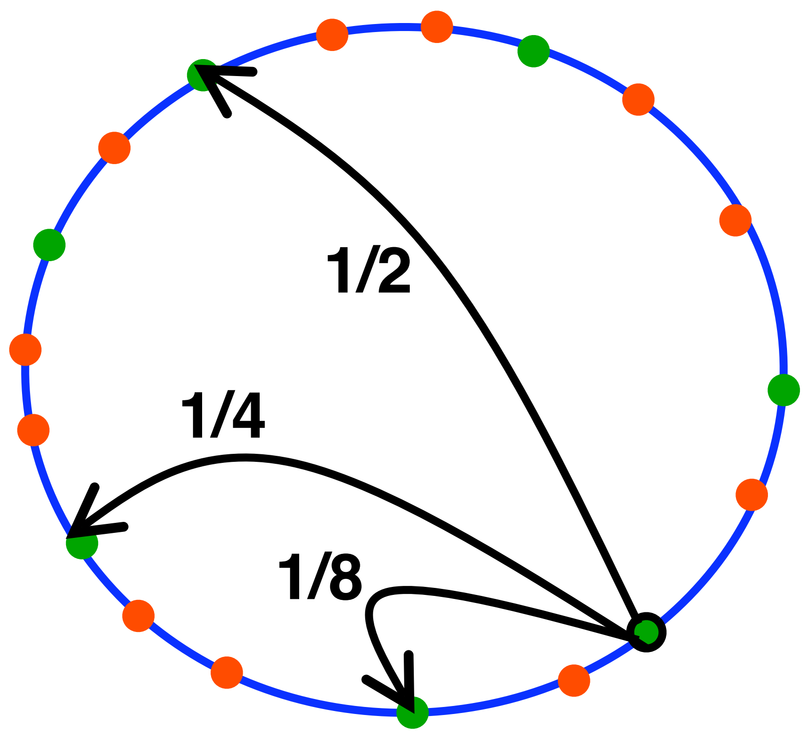
- When an existing node leaves
  - Node copies its <key, value> pairs to its predecessor
  - Predecessor points to node's successor in the ring
- When a node joins
  - Node does a lookup on its own id
  - And learns the node responsible for that id
  - This node becomes the new node's successor
  - And the node can learn that node's predecessor (which will become the new node's predecessor)

How to Find the Nearest Node?

- Need to find the closest node
  - To determine who should store (key, value) pair
  - To direct a future lookup(key) query to the node
- Strawman solution: walk through linked list
  - Circular linked list of nodes in the ring
  - O(n) lookup time when n nodes in the ring
- Alternative solution:
  - Jump further around ring
  - "Finger" table of additional overlay links

Links in the Overlay Topology

- Trade-off between # of hops vs. # of neighbors
  - E.g., log(n) for both, where n is the number of nodes
  - E.g., such as overlay links 1/2, 1/4 1/8, … around the ring
  - Each hop traverses at least half of the remaining distance



Conclusions

- Overlay networks
  - Tunnels between host computers
  - Build networks “on top” of the Internet
  - Deploy new protocols and services
- Benefits of overlay networks
  - Customization to the applications and users
  - Incremental deployment of new technologies
  - May perform better than the underlying network
- Next time
  - Peer-to-peer applications