Middleboxes and Tunneling
Reading: Sect 8.5, 9.4.1, 4.5

COS 461: Computer Networks
Spring 2011

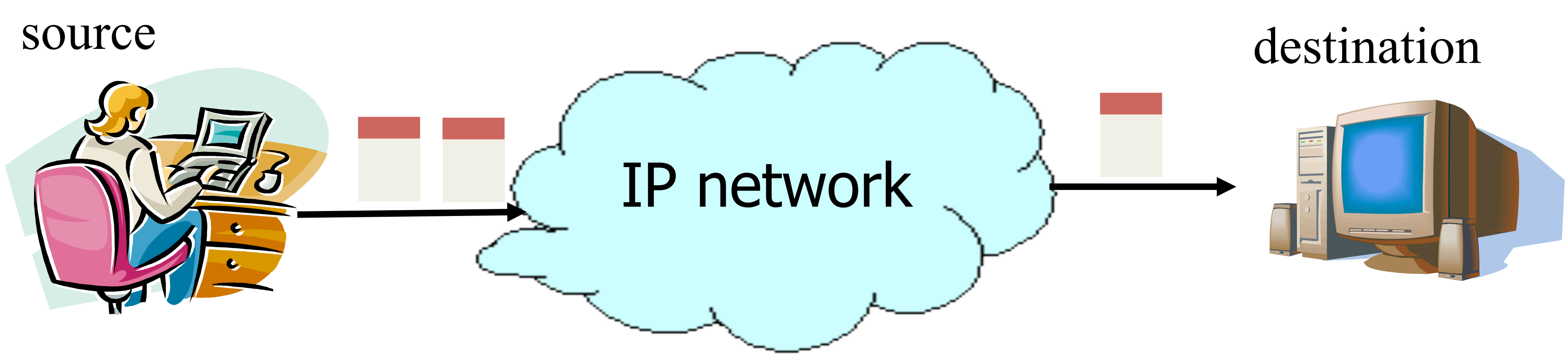
Mike Freedman
http://www.cs.princeton.edu/courses/archive/spring11/cos461/

Outline today

- Network Address Translation (NAT)
  – Multiple machines w/ private addrs behind a single public addr
- Firewalls
  – Discarding unwanted packets
- LAN appliances
  – Improve performance/security via middlebox at endpoint sites
- Overlay networks: “on top” of Internet
  – Tunnels between host computers
  – Provide better control, flexibility, QoS, isolation, …
- Underlay tunnels: “below” IP route
  – Across routers within AS
  – Provide better control, flexibility, QoS, isolation, …

Network-Layer Principles

- Globally unique identifiers
  - Each node has a unique, fixed IP address
  - … reachable from everyone and everywhere
- Simple packet forwarding
  - Network nodes simply forward packets
  - … rather than modifying or filtering them



Internet Reality

- Host mobility
  – Changes in IP addresses as hosts move
- IP address depletion
  – Dynamic assignment of IP addresses
  – Private addresses (10.0.0.0/8, 192.168.0.0/16, …)
- Security concerns
  – Discarding/detecting suspicious or unwanted packets
- Performance concerns
  – Controlling how link bandwidth is allocated
  – Caching popular content near the clients

Topic today: Middleboxes

- Middleboxes are intermediaries
  - Interposed in-between the communicating hosts
  - Often without knowledge of one or both parties
- Myriad uses
  - Network address translators
  - Firewalls
  - Tunnel endpoints
  - Traffic shapers
  - Intrusion detection systems
  - Transparent Web proxy caches
  - Application accelerators

"An abomination!"
- Violation of layering
- Hard to reason about
- Responsible for subtle bugs

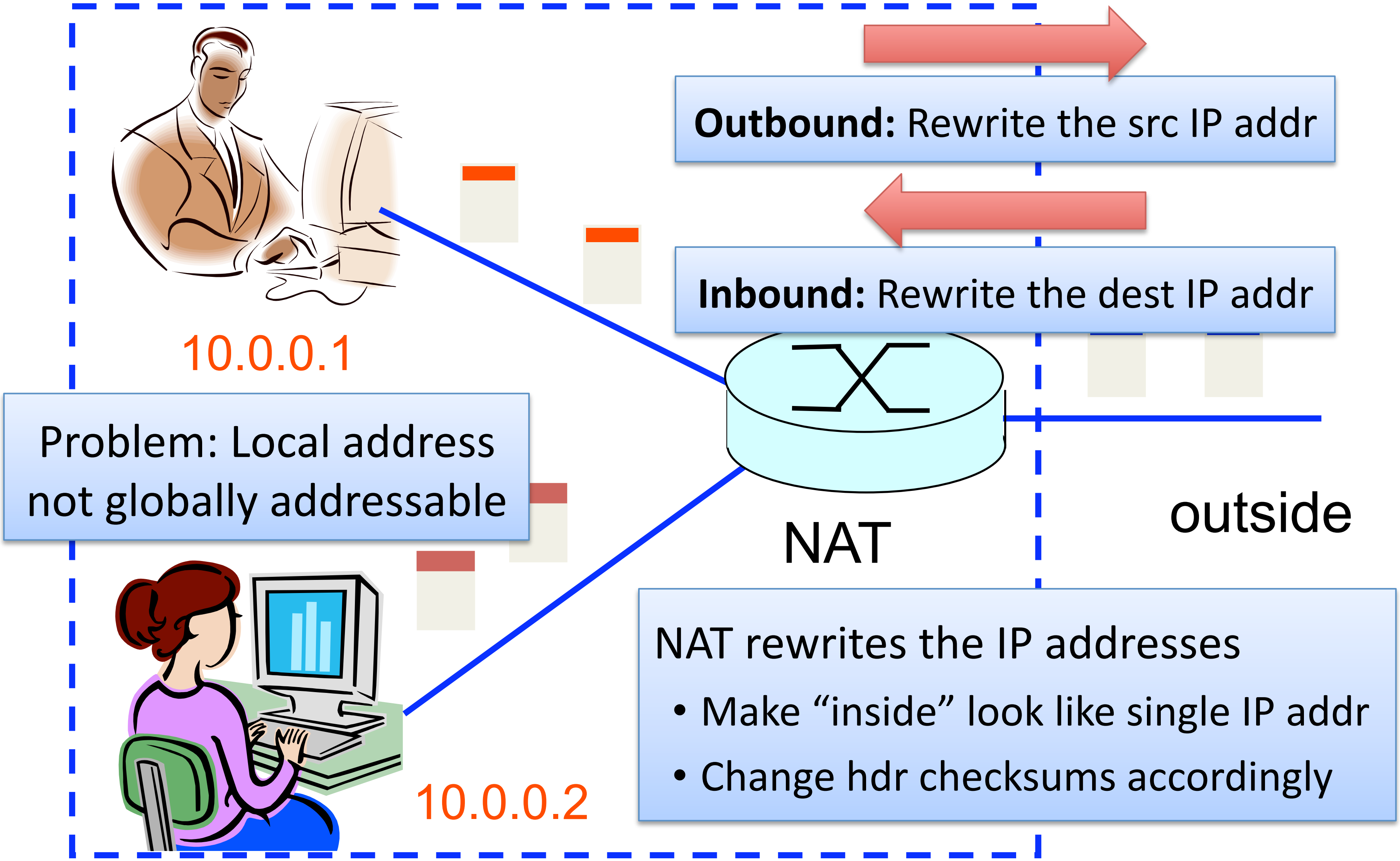
"A practical necessity!"
- Solve real/pressing problems
- Needs not likely to go away

Network Address Translation

History of NATs

- IP address space depletion
  - Clear in early 90s that 2^{32} addresses not enough
  - Work began on a successor to IPv4
- In the meantime…
  - Share addresses among numerous devices
  - … without requiring changes to existing hosts
- Meant to provide short-term remedy
  - Now: NAT is widely deployed, much more than IPv6

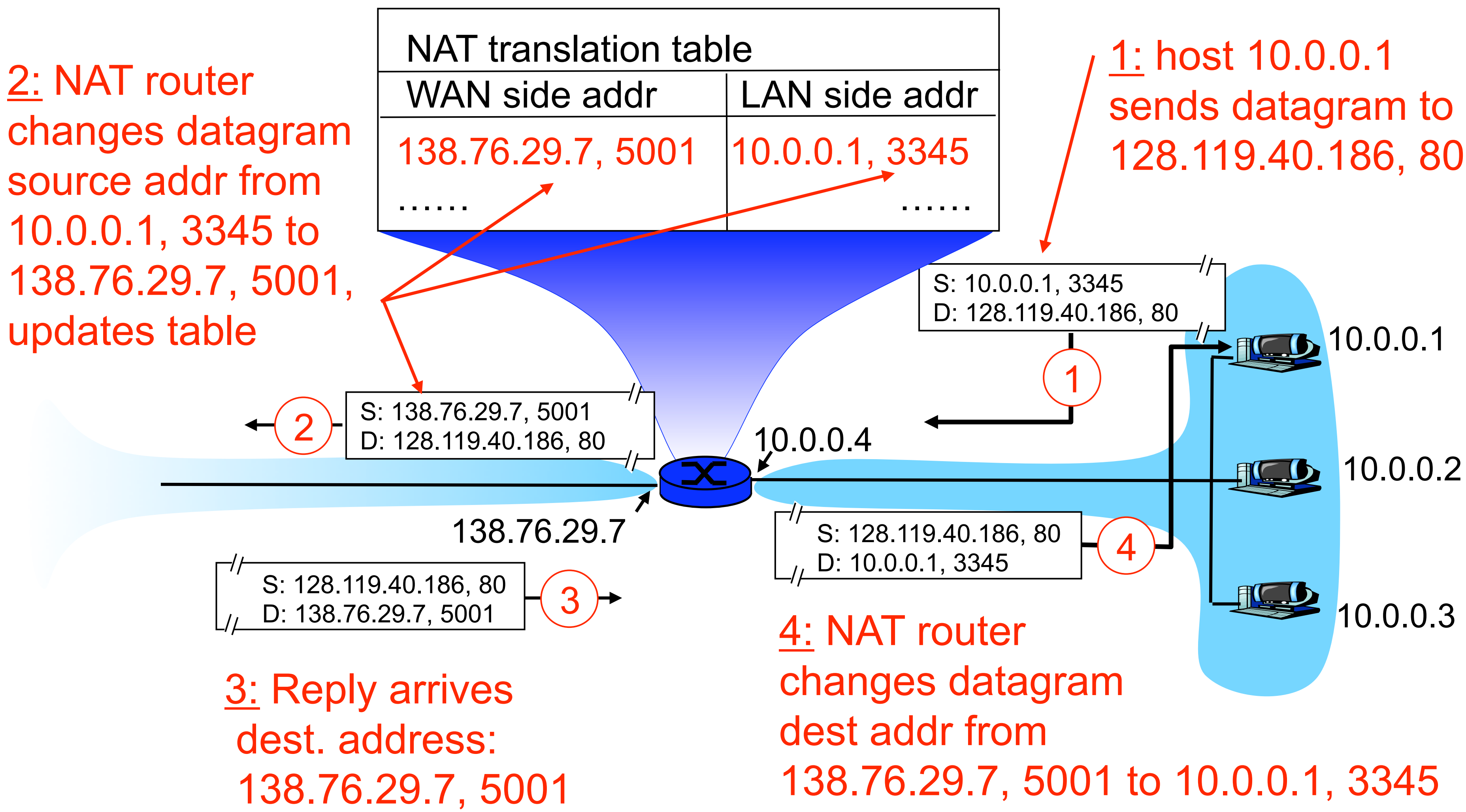
Active Component in the Data Path



Port-Translating NAT

- What if both NATted sources use same source port?
  - Can dest differentiate hosts? Can response traffic arrive?
- Map outgoing packets
  - Replace: src addr → NAT addr, source port # → new port #
  - Remote hosts respond using (NAT address, new port #)
- Maintain a translation table
  - Store map of (src addr, port #) to (NAT addr, new port #)
- Map incoming packets
  - Consult the translation table and map the dest addr/port
  - Local host receives the incoming packet

Network Address Translation Example



Maintaining the Mapping Table

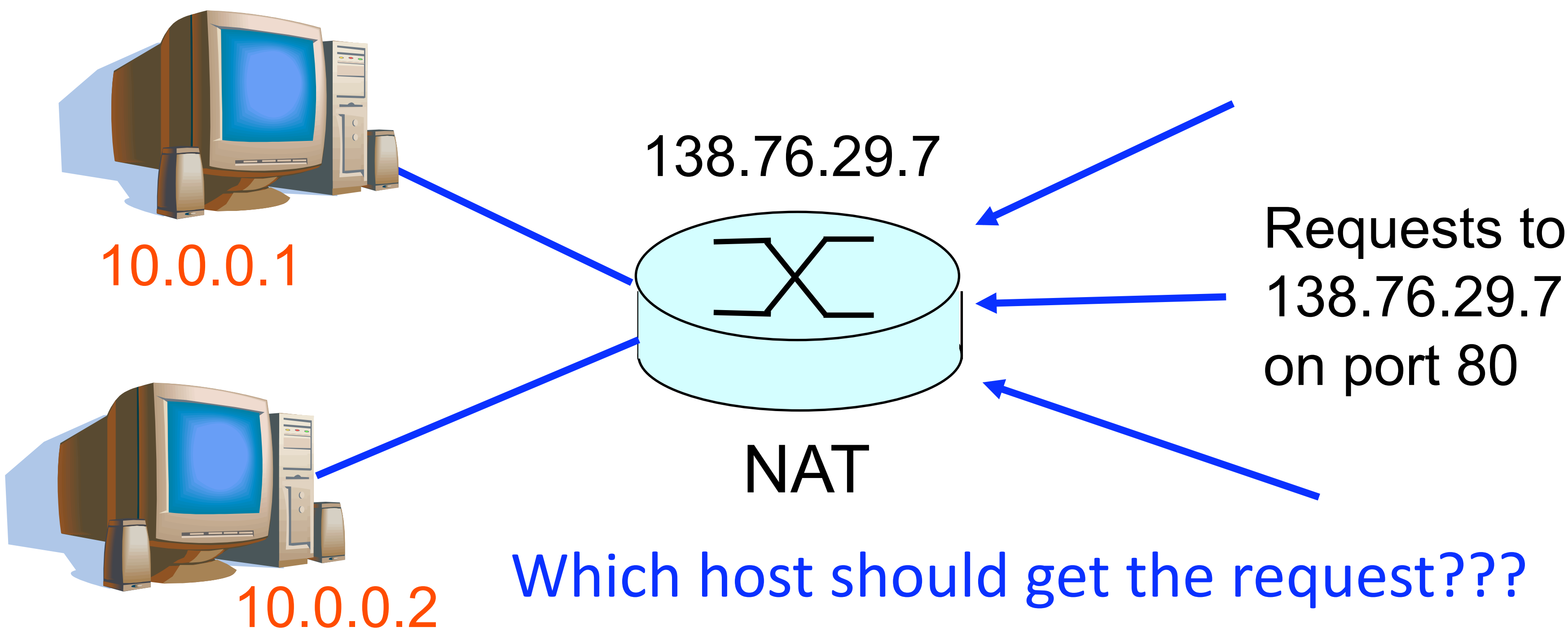
- Create an entry upon seeing an outgoing packet
  - Packet with new (source addr, source port) pair
- Eventually, need to delete entries to free up #'s
  - When? If no packets arrive before a timeout
  - (At risk of disrupting a temporarily idle connection)
- Yet another example of "soft state"
  - I.e., removing state if not refreshed for a while

Where is NAT Implemented?

- Home router (e.g., Linksys box)
  - Integrates router, DHCP server, NAT, etc.
  - Use single IP address from the service provider
- Campus or corporate network
  - NAT at the connection to the Internet
  - Share a collection of public IP addresses
  - Avoid complexity of renumbering hosts/routers when changing ISP (w/ provider-allocated IP prefix)

Practical Objections Against NAT

- Port #s are meant to identify *sockets*
  - Yet, NAT uses them to identify *end hosts*
  - Makes it hard to run a server behind a NAT



Which host should get the request???

- Explicit config at NAT for incoming conn's

Principled Objections Against NAT

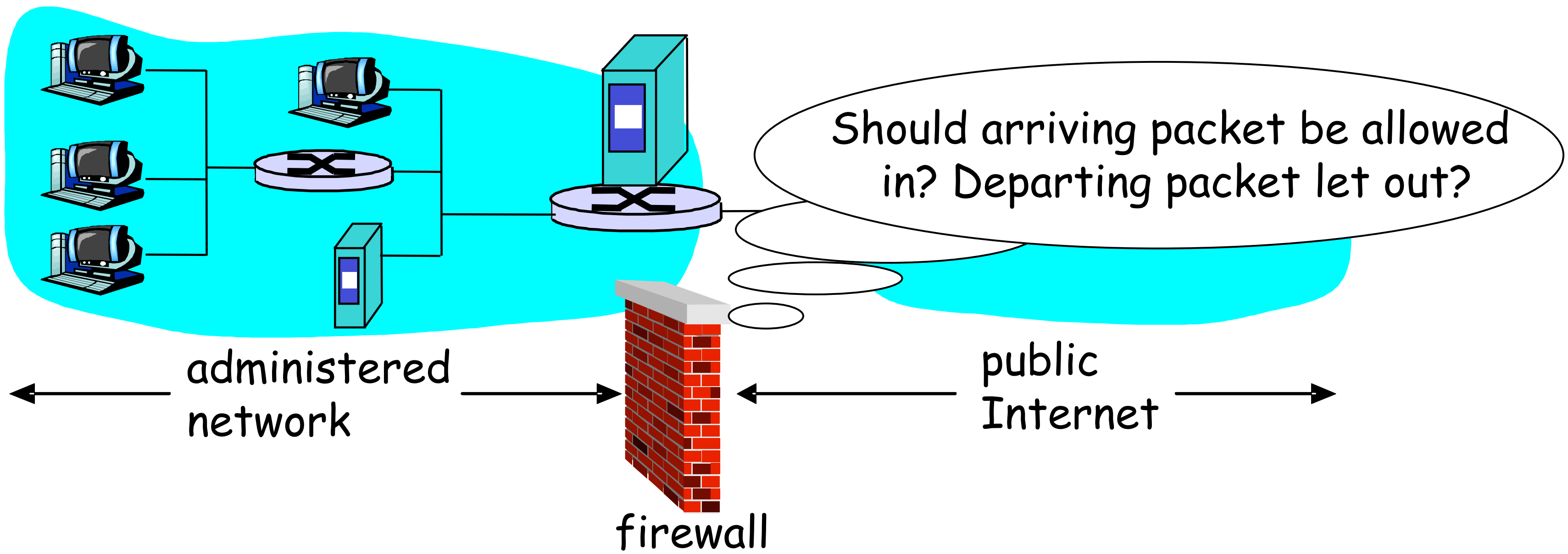
- Routers are not supposed to look at port #s
  - Network layer should care *only* about IP header
  - … and *not* be looking at the port numbers at all
- NAT violates the *end-to-end* argument
  - Network nodes should not modify the packets
- IPv6 is a cleaner solution
  - Better to migrate than to limp along with a hack

That's what happens when network puts power in hands of end users!

Firewalls

Firewalls

Isolates internal net from larger Internet, allowing some packets to pass, blocking others.



- Firewall filters packet-by-packet, based on:
  - Source/Dest IP address; Source/Dest TCP/UDP port numbers
  - TCP SYN and ACK bits; ICMP message type
  - Deep packet inspection on packet contents (DPI)

Internet Attacks: Denial of Service

- Denial-of-service attacks
  - Outsider overwhelms the host with unsolicited traffic
  - … with the goal of preventing any useful work
- Example: attacks by botnets
  - Bad guys take over a large collection of hosts
  - … and program these hosts to send traffic to your host
  - Leading to excessive traffic
- Motivations for denial-of-service attacks
  - Malice (e.g., just to be mean)
  - Revenge (e.g., for some past perceived injustice)
  - Greed (e.g., blackmailing)

Internet Attacks: Break-Ins

- Breaking in to a host
  - Outsider exploits a vulnerability in the end host
  - … with the goal of changing the behavior of the host
- Example
  - Bad guys know a Web server has a buffer-overflow bug
  - … and, say, send an HTTP request with a long URL
  - Allowing them to run their own code
- Motivations for break-ins
  - Take over the machine to launch other attacks
  - Steal information stored on the machine
  - Modify/replace the content the site normally returns

Packet Filtering Examples

- Block all packets with IP protocol field = 17 and with either source or dest port = 23.
  - All incoming and outgoing UDP flows blocked
  - All Telnet connections are blocked
- Block inbound TCP packets with SYN but no ACK
  - Prevents external clients from making TCP connections with internal clients
  - But allows internal clients to connect to outside
- Block all packets with TCP port of Quake

Firewall Configuration

- Firewall applies a set of rules to each packet
  - To decide whether to permit or deny the packet
- Each rule is a test on the packet
  - Comparing IP and TCP/UDP header fields
  - … and deciding whether to permit or deny
- Order matters
  - Once packet matches a rule, the decision is done

Firewall Configuration Example

- Alice runs a network in 222.22.0.0/16
  - Wants to let Bob’s school access certain hosts
    - Bob is on 111.11.0.0/16
    - Alice’s special hosts on 222.22.22.0/24
  - Alice doesn’t trust Trudy, inside Bob’s network
    - Trudy is on 111.11.11.0/24
  - Alice doesn’t want any other traffic from Internet
- Rules
  - #1: Don’t let Trudy’s machines in
    - Deny (src = 111.11.11.0/24, dst = 222.22.0.0/16)
  - #2: Let rest of Bob’s network in to special dsts
    - Permit (src=111.11.0.0/16, dst = 222.22.22.0/24)
  - #3: Block the rest of the world
    - Deny (src = 0.0.0.0/0, dst = 0.0.0.0/0)

A Variation: Traffic Management

- Permit vs. deny is too binary a decision
  - Maybe better to classify the traffic based on rules
  - … and then handle the classes of traffic differently
- Traffic shaping (rate limiting)
  - Limit the amount of bandwidth for certain traffic
  - E.g., rate limit on Web or P2P traffic
- Separate queues
  - Use rules to group related packets
  - And then do round-robin scheduling across groups
  - E.g., separate queue for each internal IP address

Firewall Implementation Challenges

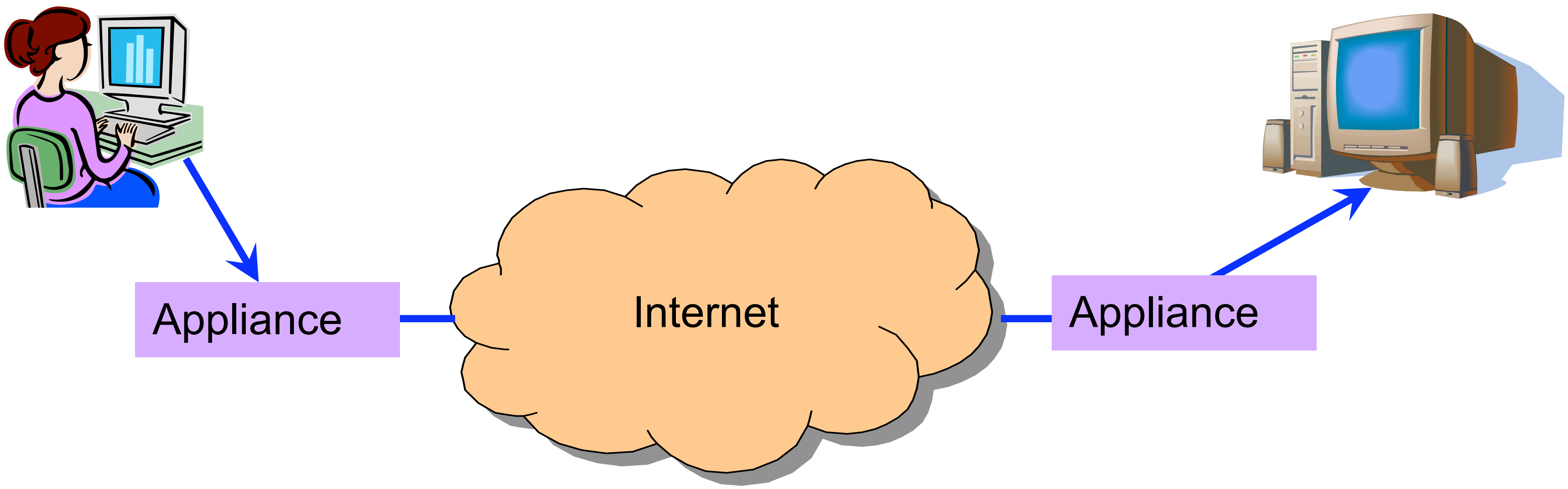
- Per-packet handling
  – Must inspect every packet
  – Challenging on very high-speed links
- Complex filtering rules
  – May have large # of rules
  – May have very complicated rules
- Location of firewalls
  – Complex firewalls near the edge, at low speed
  – Simpler firewalls in the core, at higher speed

Clever Users Subvert Firewalls

- Example: filtering dorm access to a server
  - Firewall rule based on IP addresses of dorms
  - … and the server IP address and port number
  - Problem: users may log in to another machine
    - E.g., connect from the dorms to another host
    - … and then onward to the blocked server
- Example: filtering P2P based on port #s
  - Firewall rule based on TCP/UDP port numbers
    - E.g., allow only port 80 (e.g., Web) traffic
  - Problem: software using non-traditional ports
    - E.g., write P2P client to use port 80 instead

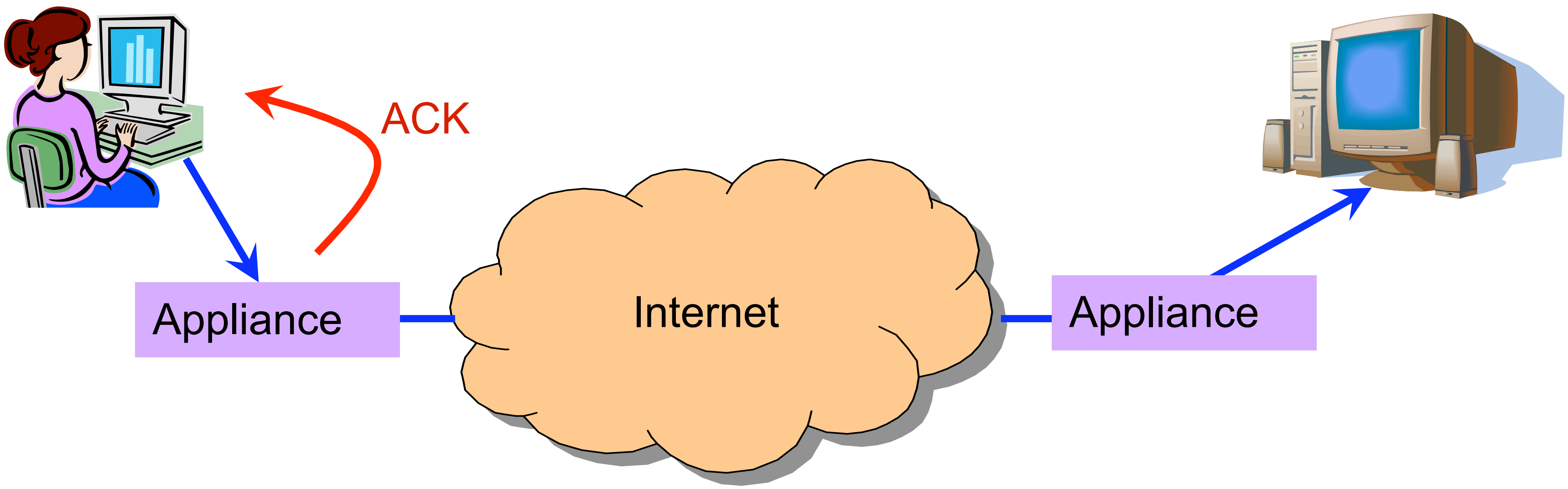
Tunneling
via on-path middleboxes

At Connection Point to the Internet



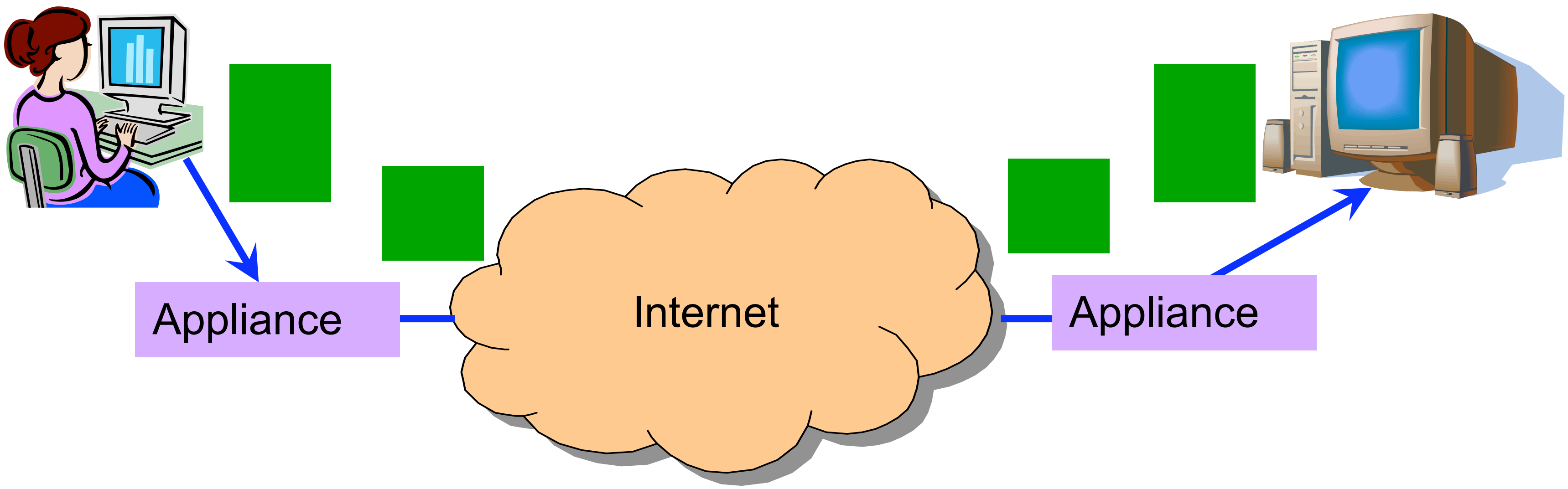
- Improve performance between edge networks
  - E.g., multiple sites of the same company
  - Through buffering, compression, caching, …
- Incrementally deployable
  - No changes to the end hosts or the rest of the Internet
  - Inspects the packets as they go by, and takes action

Example: Improve TCP Throughput



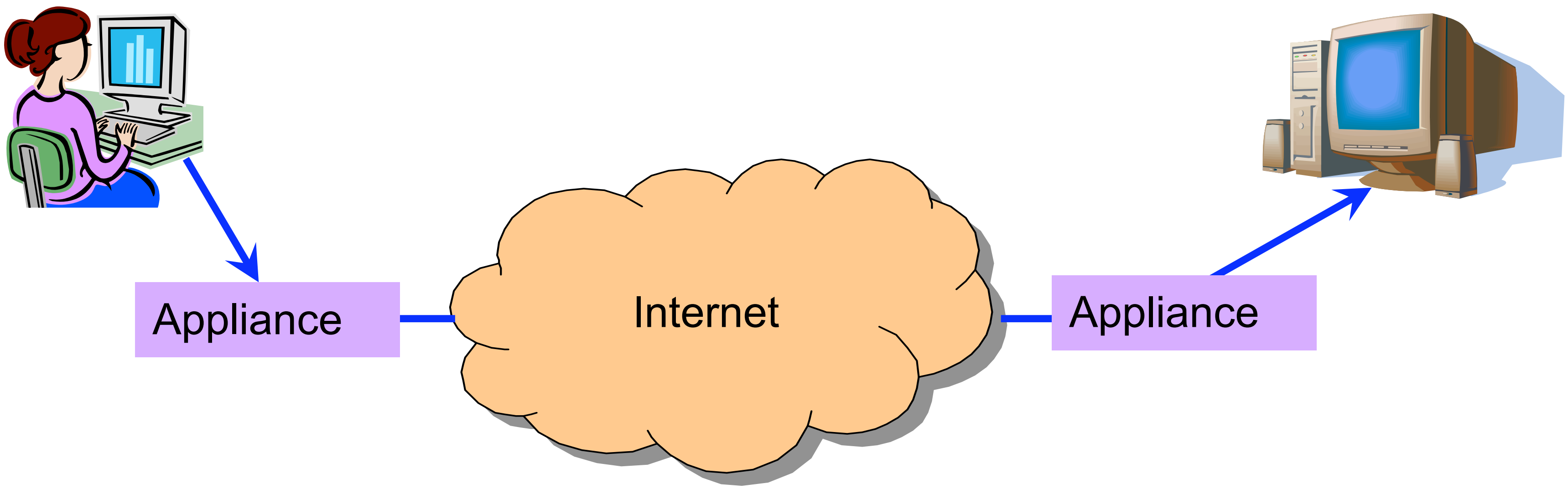
- Appliance with a lot of local memory
- Sends ACK packets quickly to the sender
- Overwrites receive window with a large value
- Or, even run a new and improved version of TCP

Example: Compression



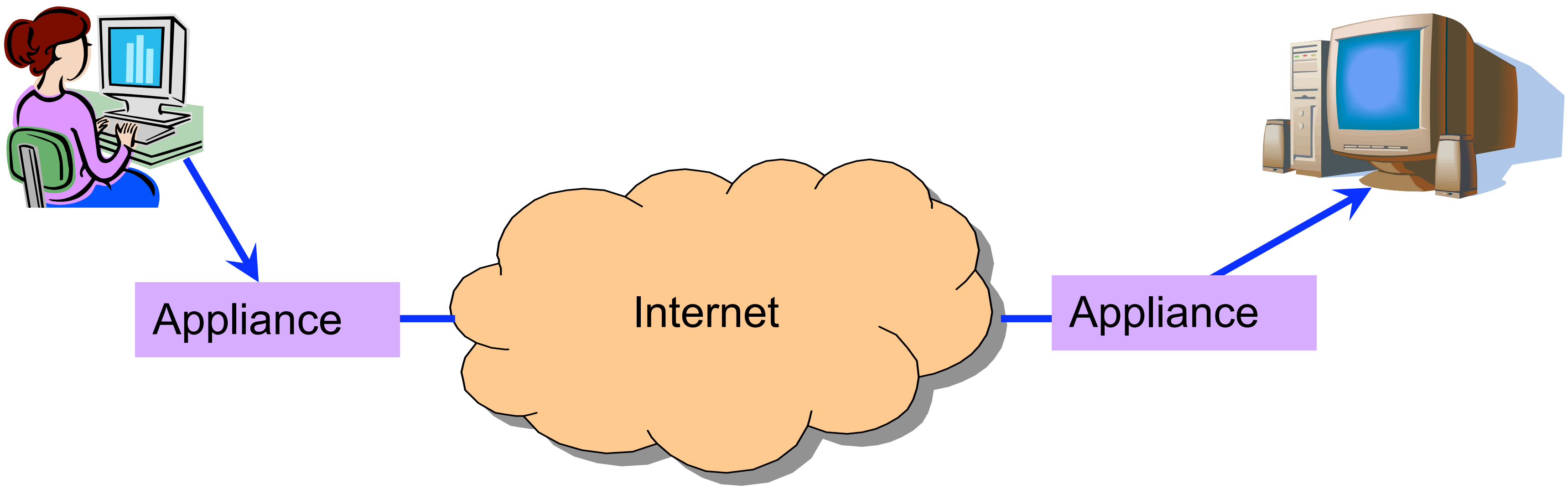
- Compress the packet
- Send the compressed packet
- Uncompress at the other end
- Maybe compress across successive packets

Example: Caching



- Cache copies of the outgoing packets
- Check for sequences of bytes that match past data
- Just send a pointer to the past data
- And have the receiving appliance reconstruct

Example: Encryption



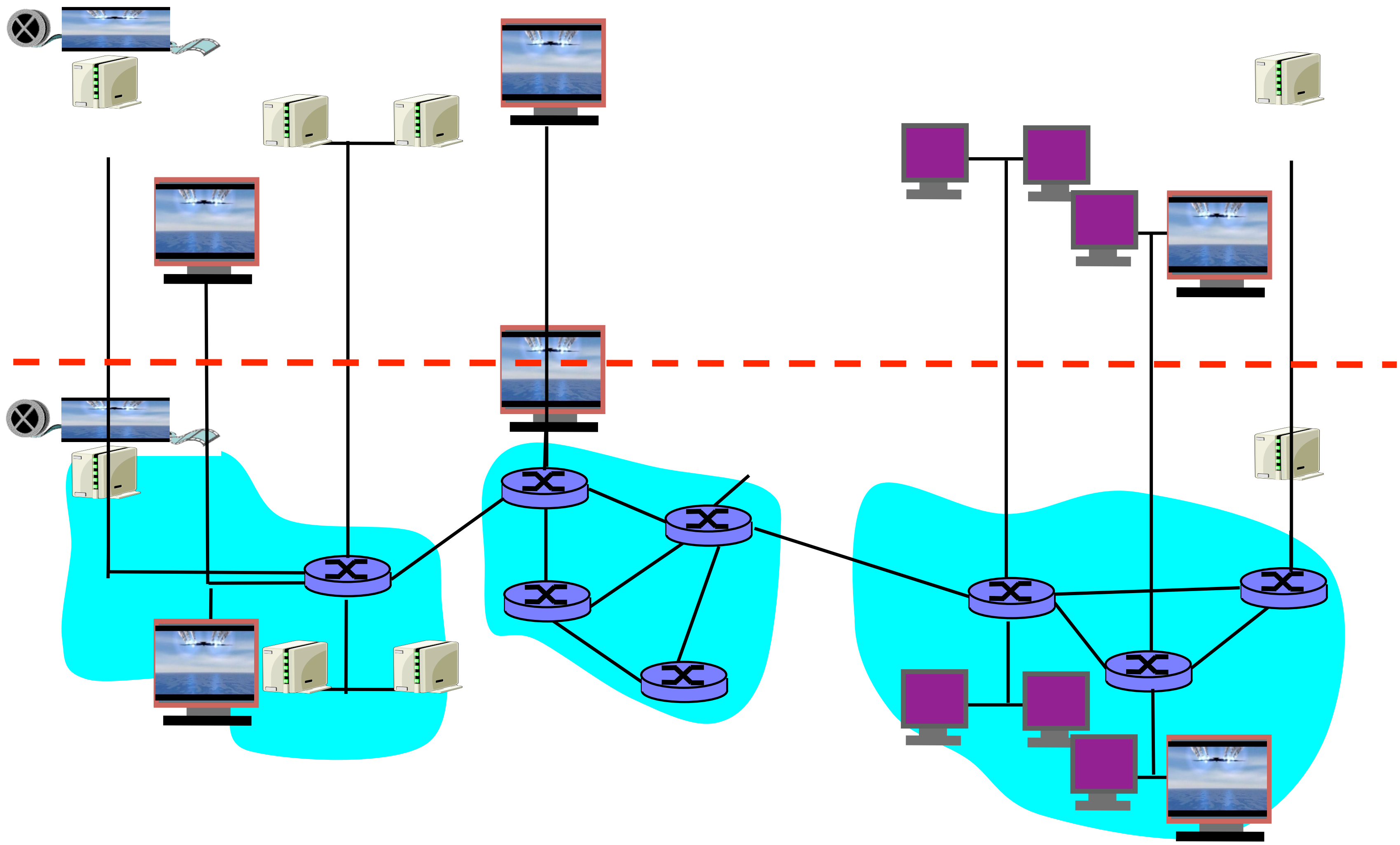
- Two sites share keys for encrypting traffic
- Sending appliance encrypts the data
- Receiving appliance decrypts the data
- Protects the sites from snoopers on the Internet

Tunneling via Overlay Networks

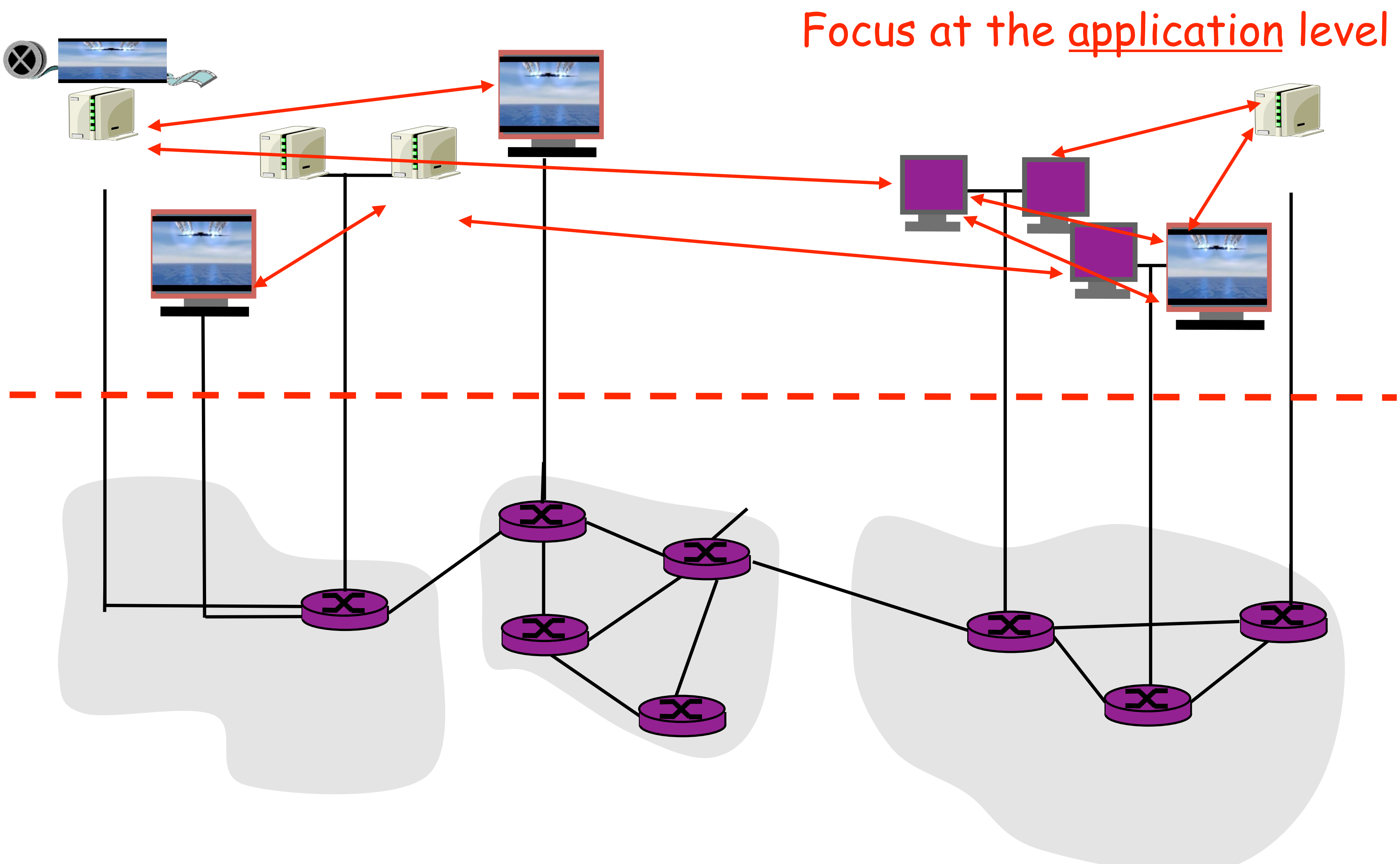
Using Overlays to Evolve the Internet

- Internet needs to evolve
  - IPv6
  - Mobility
  - Security
  - IP 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

Overlay Networks



Overlay Networks

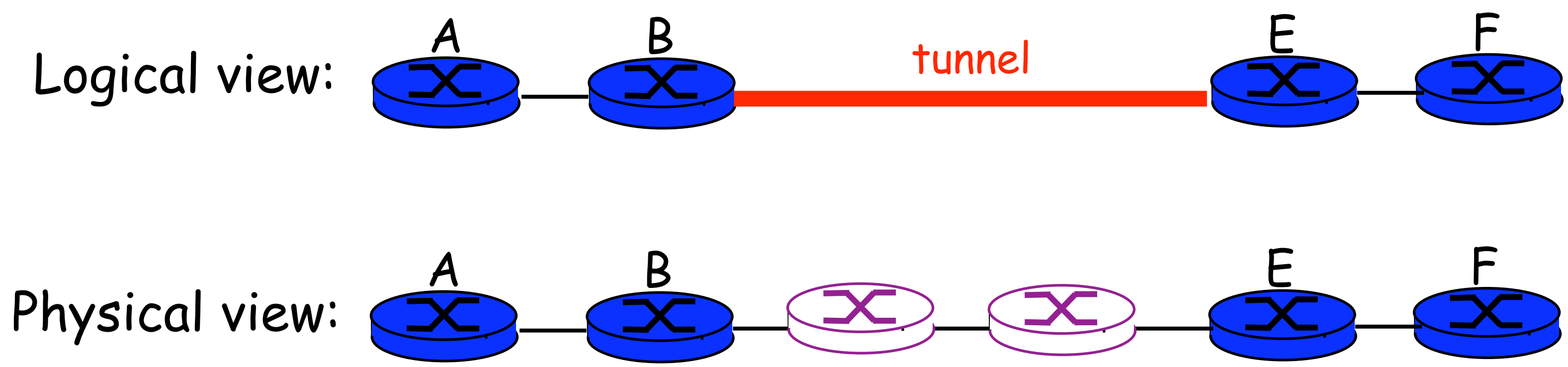


Overlay Networks

- 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

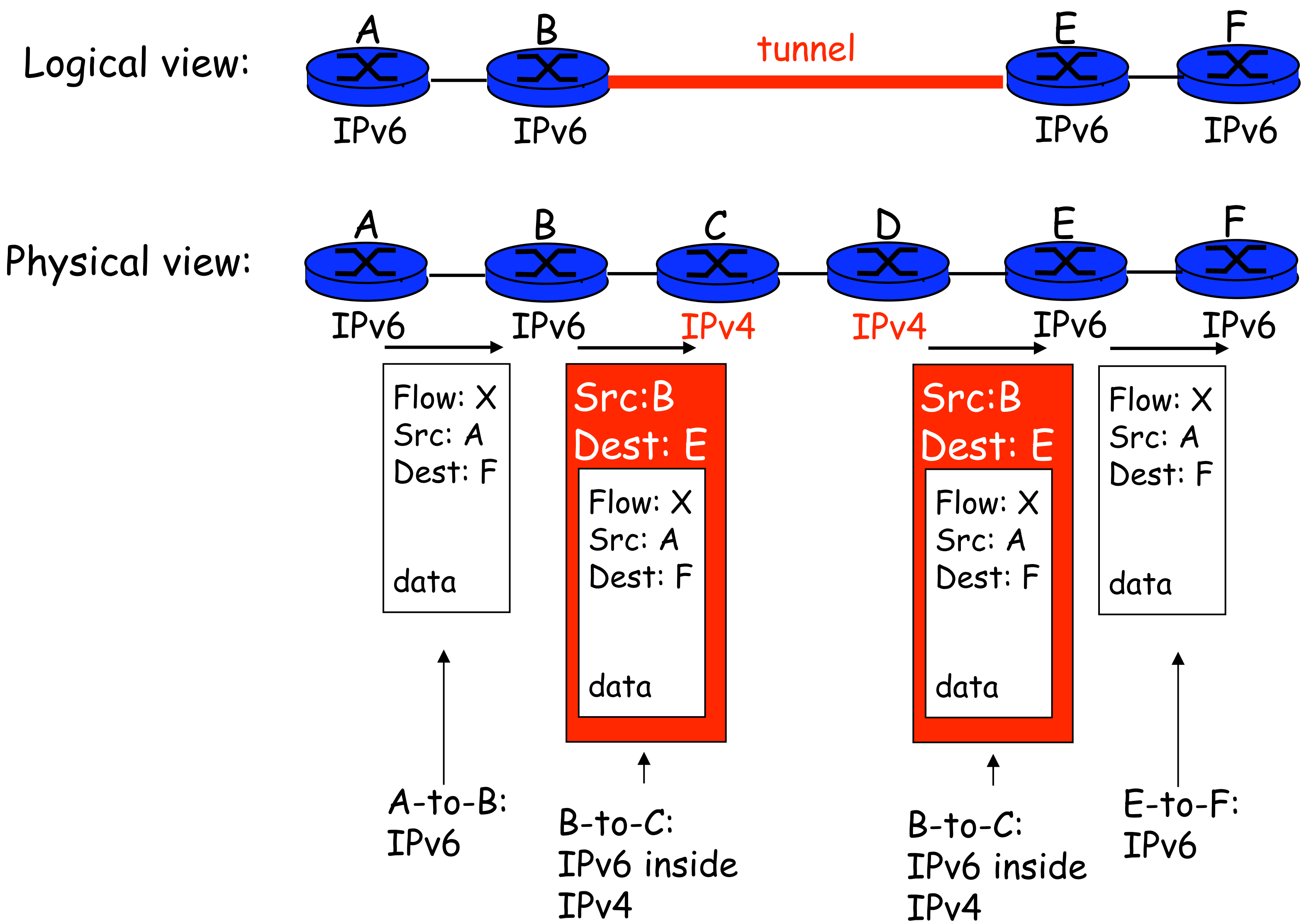
IP Tunneling to Build Overlay Links

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



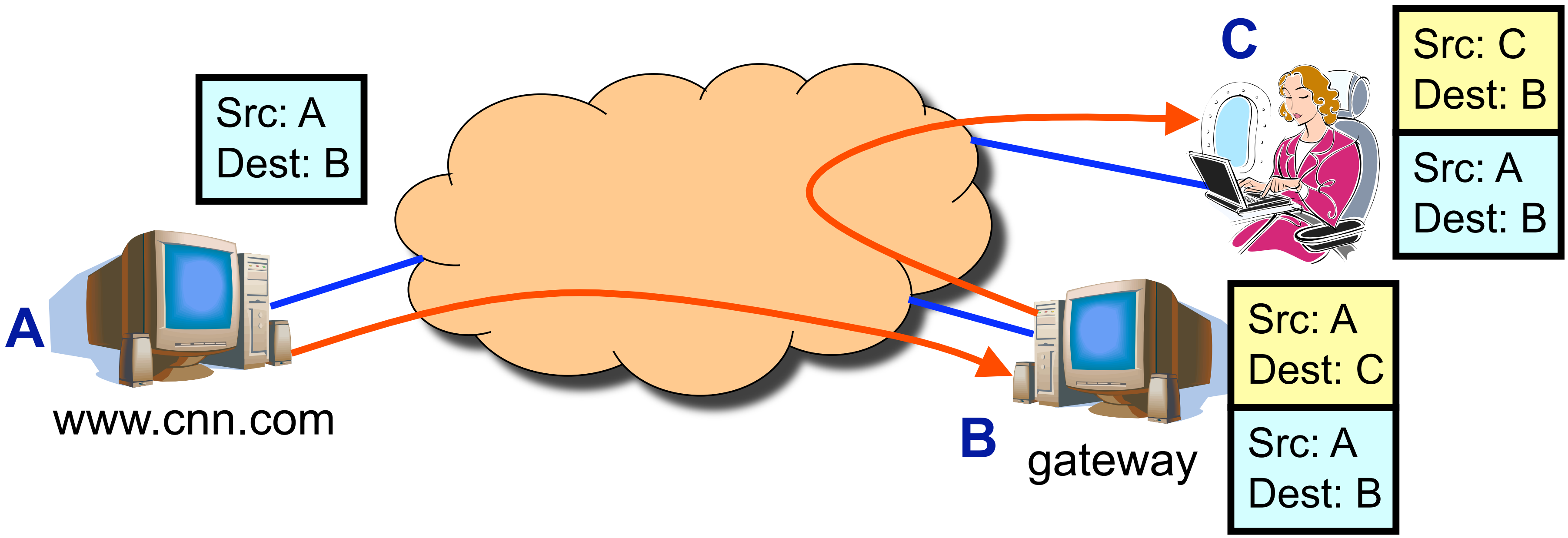
- Encapsulation of the packet inside an IP datagram
  - Node B sends a packet to node E
  - … containing another packet as the payload

6Bone: Deploying IPv6 over IP4



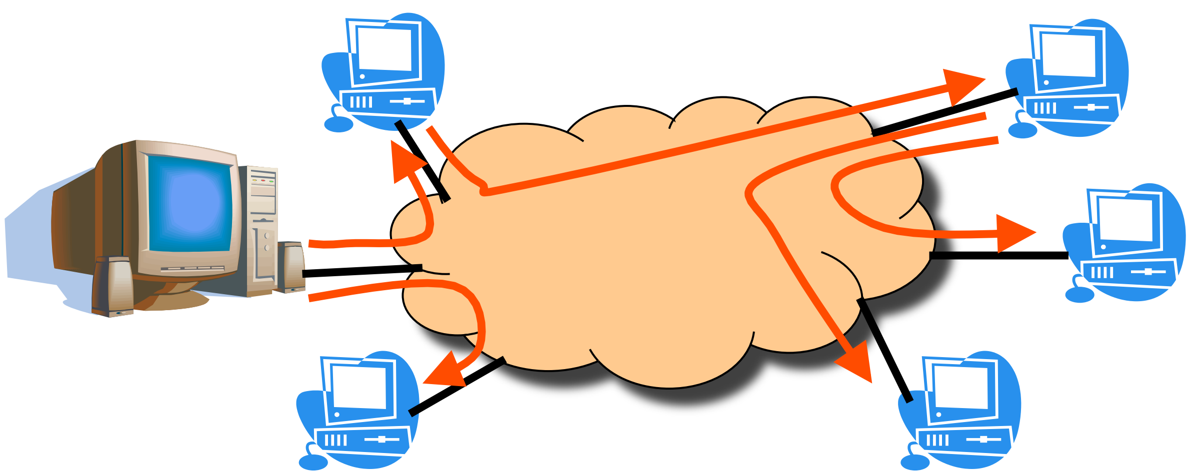
Communicating With Mobile Users

- A mobile user changes locations frequently
  - So, the IP address of the machine changes often
- The user wants applications to continue running
  - So, the change in IP address needs to be hidden
- Solution: fixed gateway forwards packets
  - Gateway has fixed IP address and keeps track of mobile addr



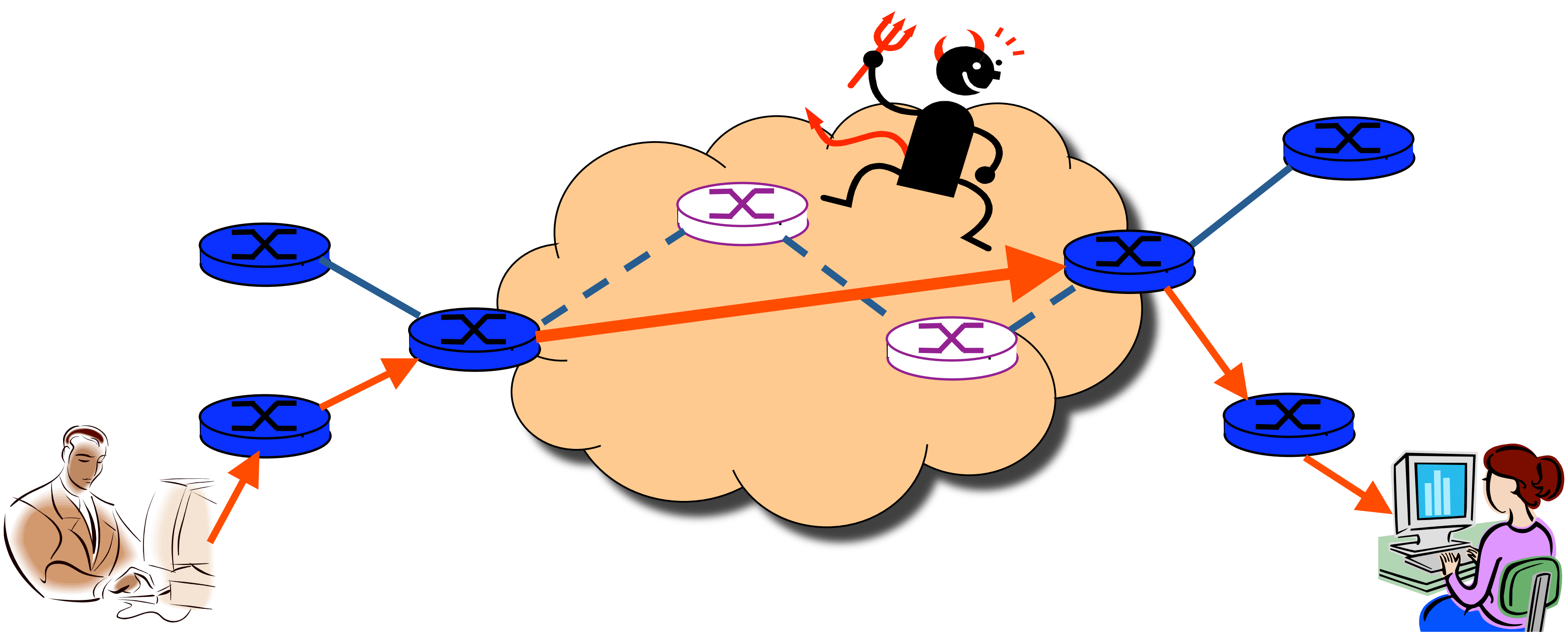
MBone: Multicast Backbone

- IP Multicast
  - One packet, many receivers on same IP (multicast) address
- A catch-22 for deploying multicast
  - Router vendors wouldn't support, since they weren't sure
  - And, since it didn't exist, nobody was using it
- Idea: software implementing multicast protocols
  - And unicast tunnels to traverse non-participants



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



Tunneling under IP Networks

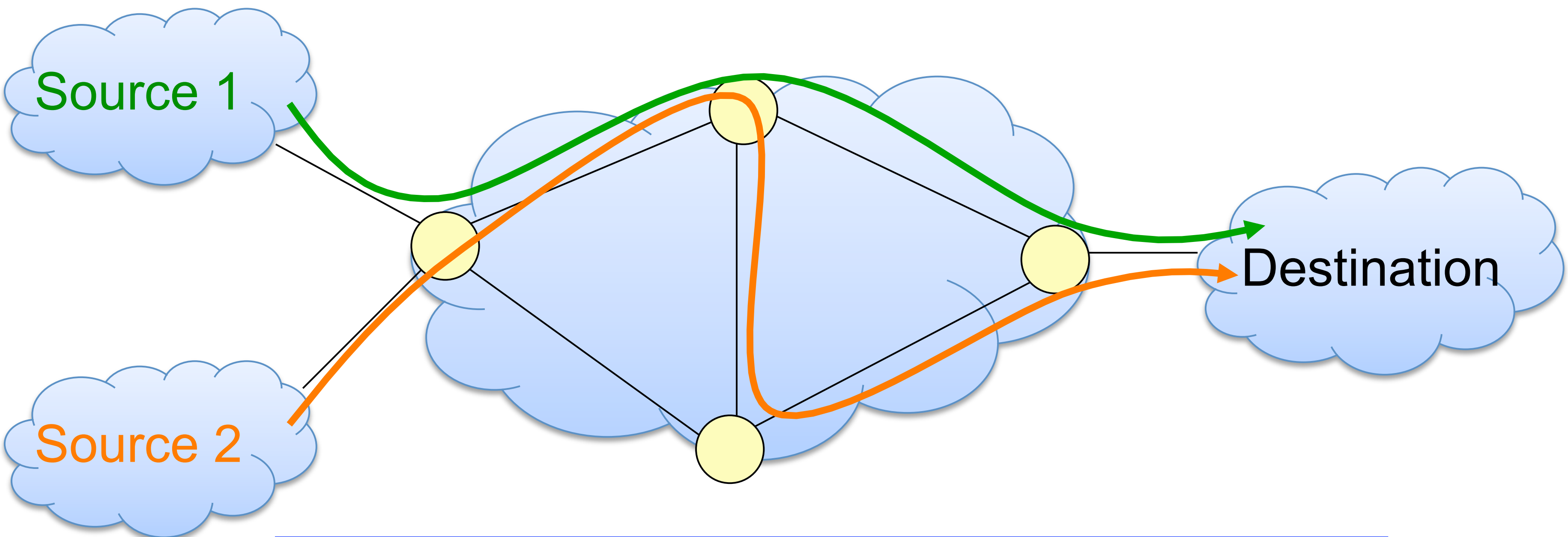
Introducing

Multi-Protocol Label Switching

(MPLS)

MPLS Overview

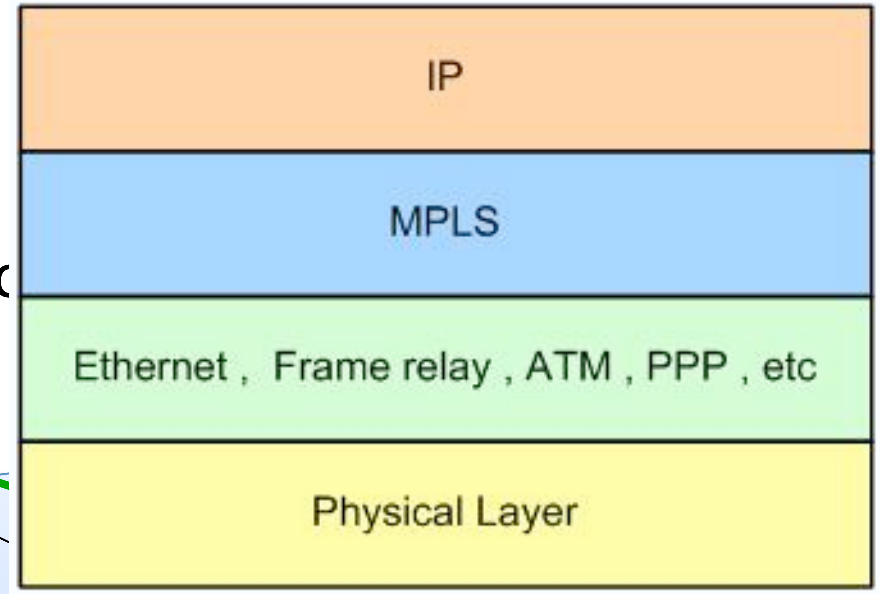
- **Main idea:** Virtual circuit
  - Packets forwarded based only on circuit identifier

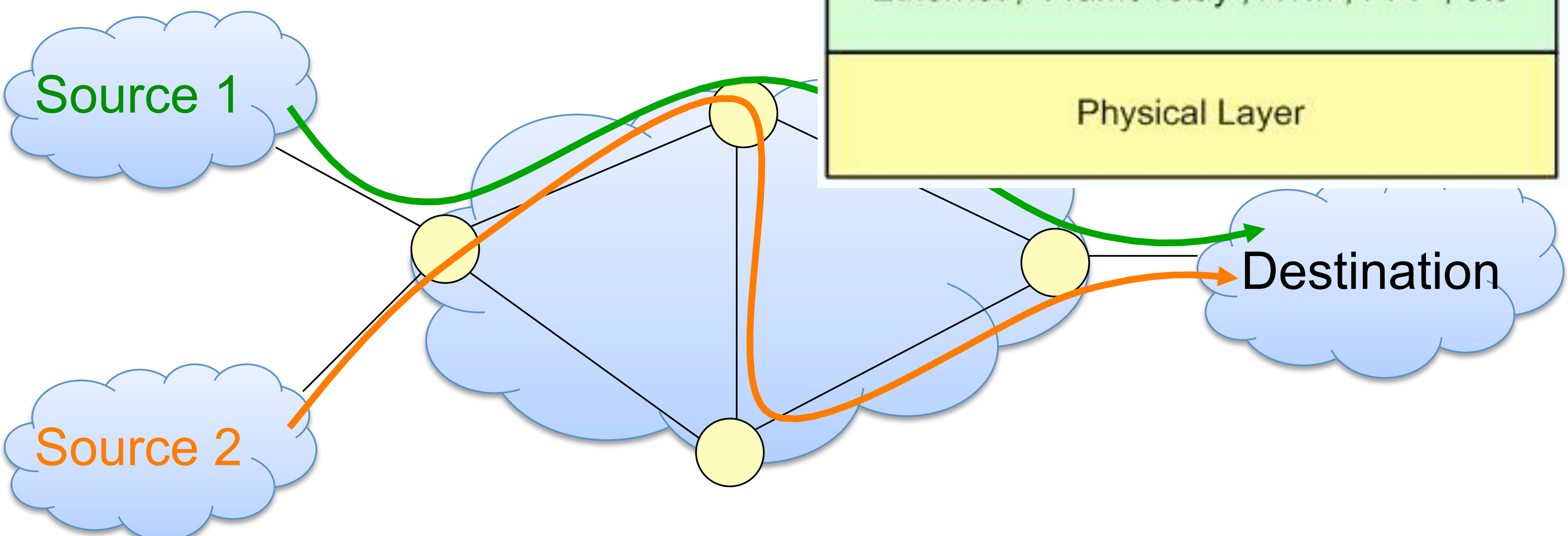


Router can forward traffic to the same destination on different interfaces/paths.

MPLS Overview

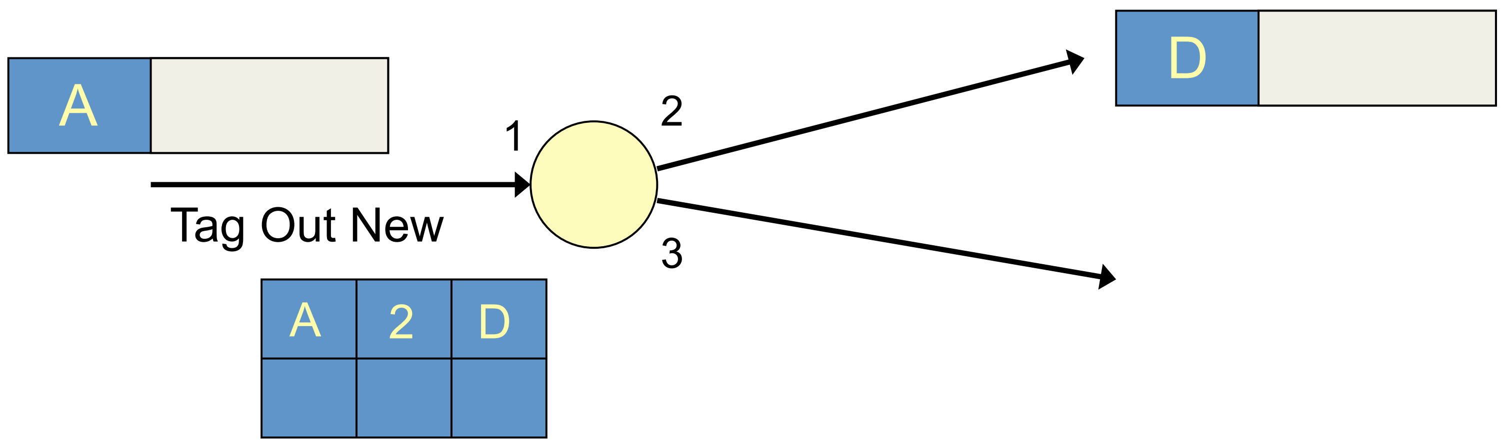
- **Main idea:** Virtual circuit
  - Packets forwarded based c





Router can forward traffic to the same destination on different interfaces/paths.

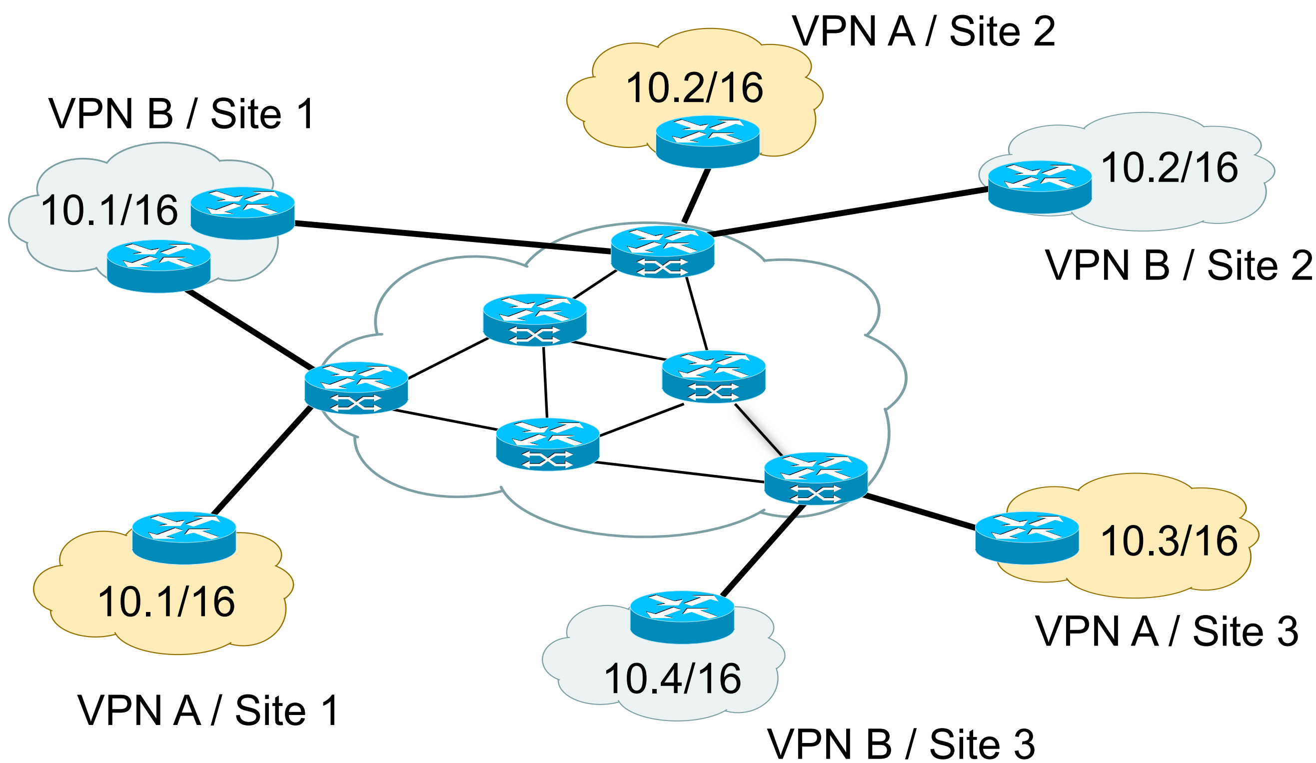
Circuit Abstraction: Label Swapping



- **Label-switched paths:** Paths "named" by label at ingress
- At each hop, MPLS routers:
  - Use label to determine outgoing interface, new label
  - Thus, push/pop/swap MPLS headers that encapsulate IP
- **Label distribution protocol:** disseminate signaling info
- Initially from concern with longest-prefix-match speed
  - Now use in other applications, e.g., intra-AS traffic management

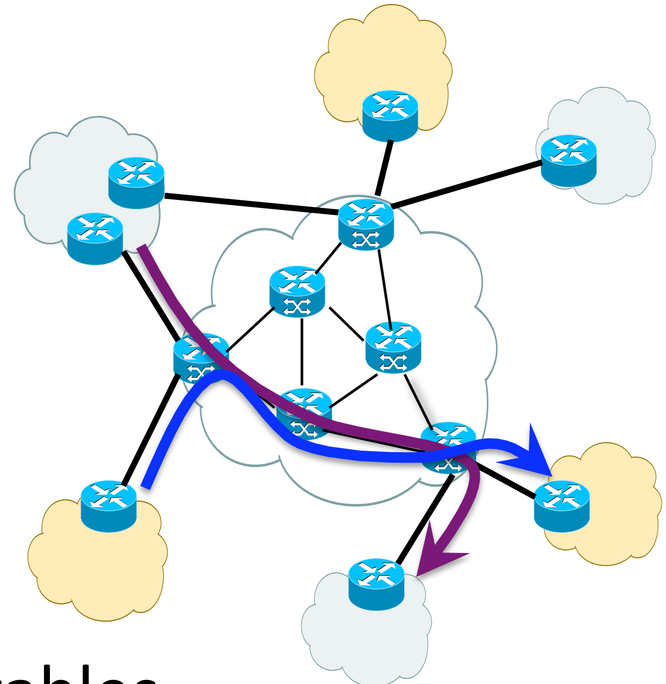
Private communication over a public network

Layer 3 Virtual Private Networks



- **Isolation:** Multiple logical networks over a single, shared physical infrastructure
- **Tunneling:** Keeping routes out of the core

High-Level Overview of Operation



- IP packets arrive at provider edge router (PE)
- Destination IP looked up in forwarding table
  - Multiple "virtual" forwarding tables
  - Ingress port associated with one virtual forw. table
- Datagram sent to customer's network using tunneling (i.e., an MPLS label-switched path)

Conclusions

- Middleboxes address important problems
  - Getting by with fewer IP addresses
  - Blocking unwanted traffic
  - Making fair use of network resources
  - Improving end-to-end performance

- Middleboxes cause problems of their own
  - No longer globally unique IP addresses
  - No longer can assume network simply delivers packets