



Middleboxes

Reading: Ch. 8.4

Goals of Today's Class

- Network-layer principles
 - Globally unique identifiers and simple packet forwarding
 - Middleboxes as a way to violate these principles
- Network Address Translation (NAT)
 - Multiple machines behind a single public address
 - Private addresses behind the NAT box
- Firewalls
 - Discarding unwanted packets
- LAN appliances
 - Improving performance and security
 - Using a middlebox at sending and receiving sites

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 suspicious or unwanted packets
 - Detecting suspicious traffic
- **Performance concerns**
 - Controlling how link bandwidth is allocated
 - Storing popular content near the clients

Middleboxes

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

Two Views of Middleboxes

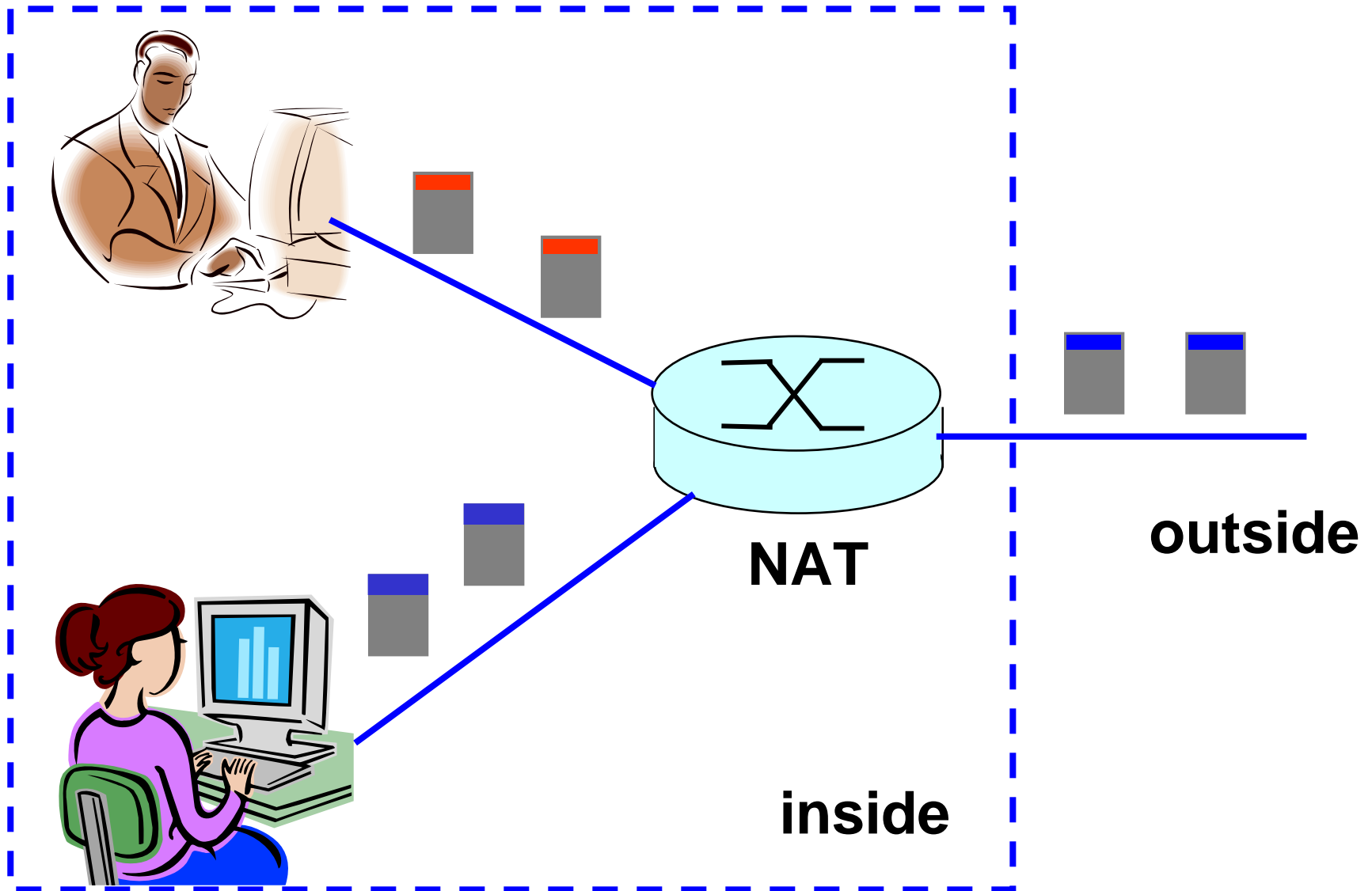
- An abomination
 - Violation of layering
 - Cause confusion in reasoning about the network
 - Responsible for many subtle bugs
- A practical necessity
 - Solving real and pressing problems
 - Needs that are not likely to go away
- Would they arise in *any* edge-empowered network, even if redesigned from scratch?

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 temporary relief
 - Intended as a short-term remedy
 - Now, NAT are very widely deployed
 - ... much moreso than IPv6 😊

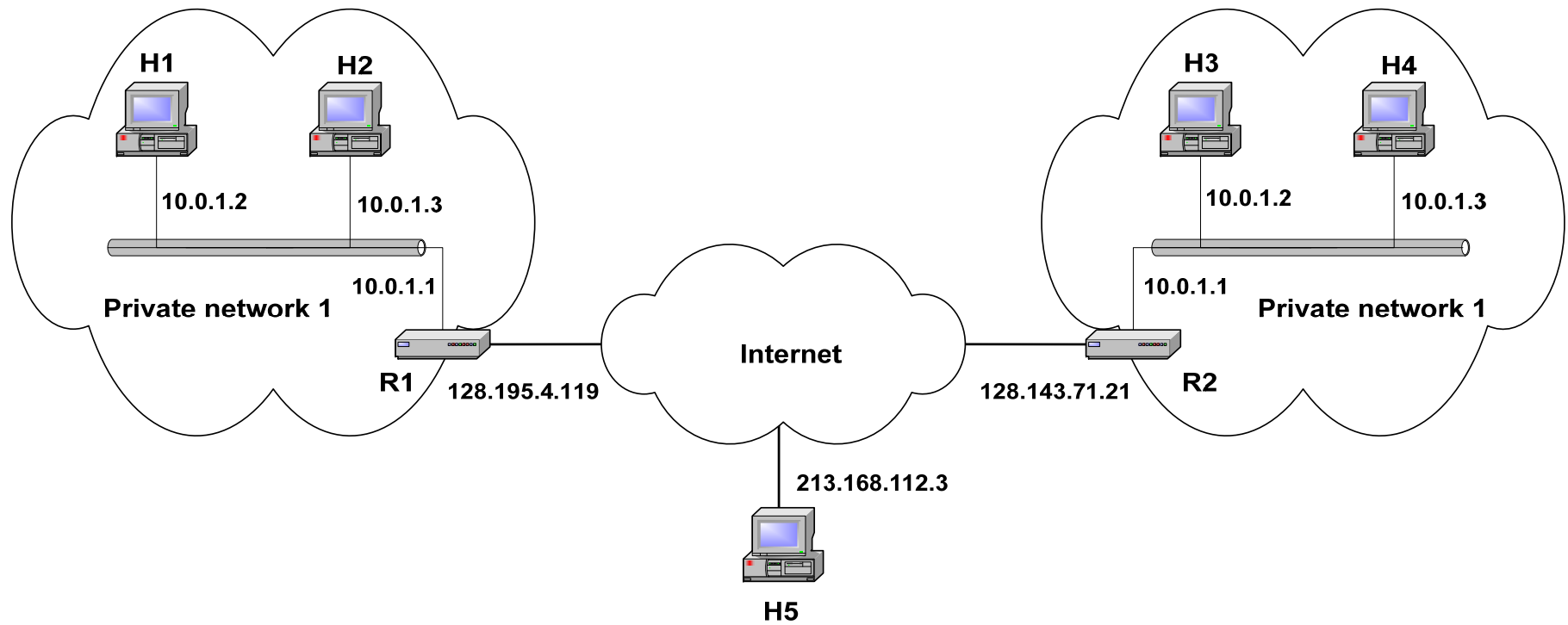
Active Component in the Data Path



Private Network

- *Private IP* network is an IP network that is not directly connected to the Internet
- IP addresses in a private network can be assigned arbitrarily.
 - Not registered and not guaranteed to be globally unique
- Generally, private networks use addresses from the following experimental address ranges (*non-routable addresses*):
 - 10.0.0.0 – 10.255.255.255
 - 172.16.0.0 – 172.31.255.255
 - 192.168.0.0 – 192.168.255.255

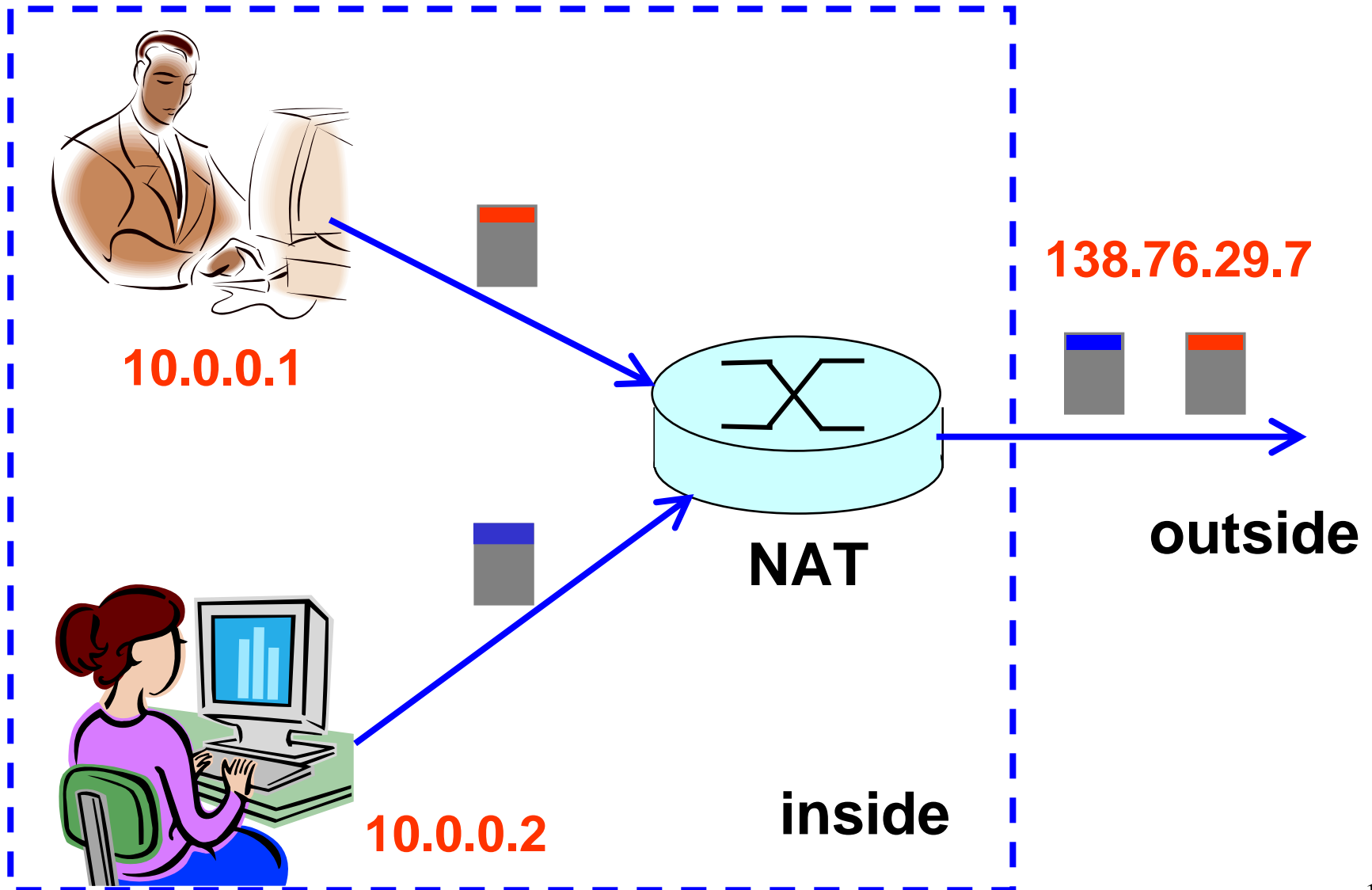
Private Addresses



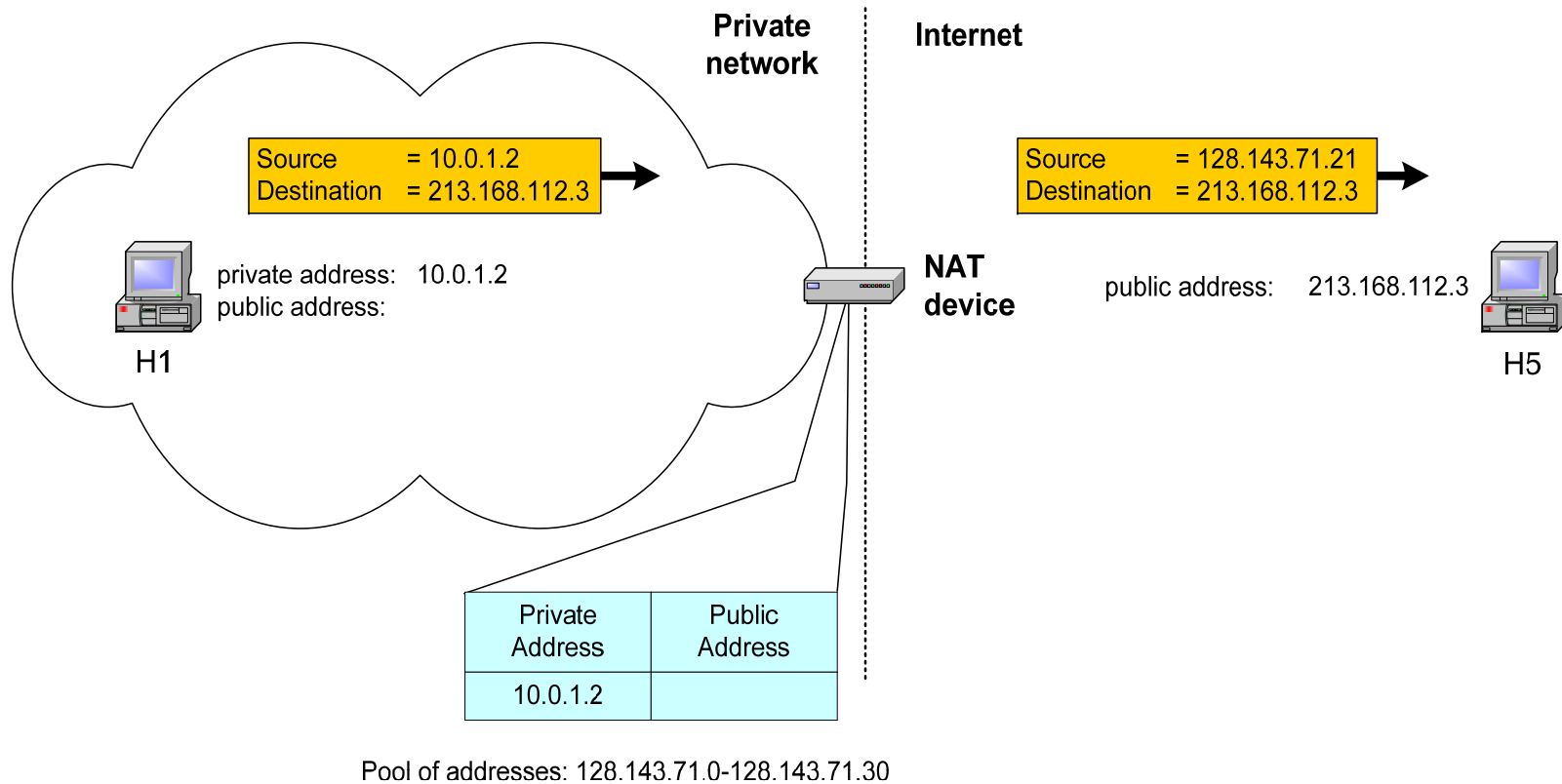
IP Header Translators

- Local network addresses not globally unique
 - E.g., private IP addresses (in 10.0.0.0/8)
- NAT box rewrites the IP addresses
 - Make the “inside” look like a single IP address
 - ... and change header checksums accordingly
- Outbound traffic: from inside to outside
 - Rewrite the source IP address
- Inbound traffic: from outside to inside
 - Rewrite the destination IP address

Using a Single Source Address



Basic Operation of NAT



- NAT manages a pool of global addresses (1 or more)
- Maintains an address translation table

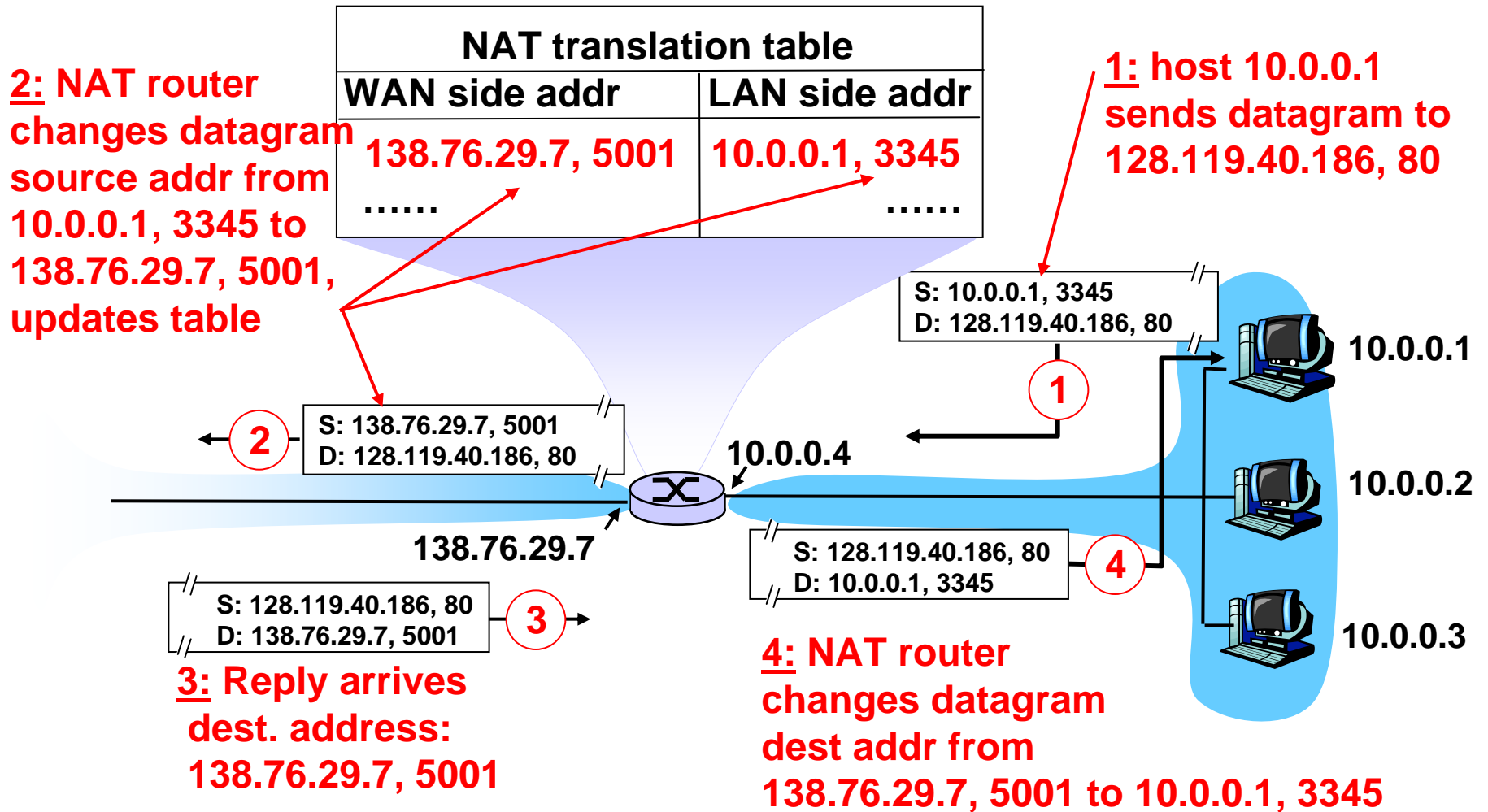
What if Both Hosts Contact Same Site?

- Suppose hosts contact the same destination
 - E.g., both hosts open a socket with local port 3345 to destination 128.119.40.186 on port 80
- NAT gives packets same source address
 - All packets have source address 138.76.29.7
- Problems
 - Can destination differentiate between senders?
 - Can return traffic get back to the correct hosts?

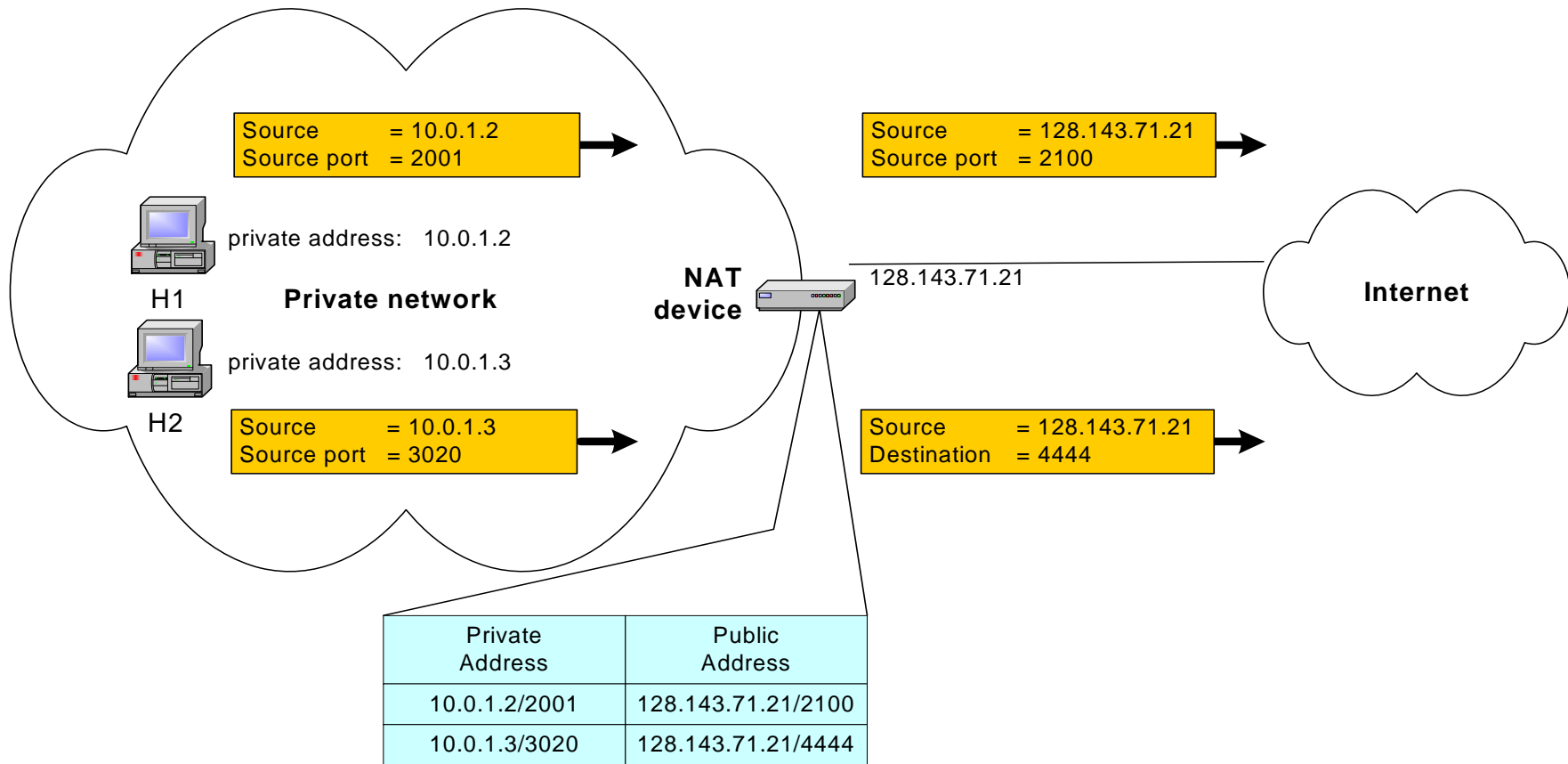
Port-Translating NAT

- **Map outgoing packets**
 - Replace source address with NAT address
 - Replace source port number with a new port number
 - Remote hosts respond using (NAT address, new port #)
- **Maintain a translation table**
 - Store map of (source address, port #) to (NAT address, new port #)
- **Map incoming packets**
 - Consult the translation table
 - Map the destination address and port number
 - Local host receives the incoming packet

Network Address Translation Example



IP Masquerading



Maintaining the Mapping Table

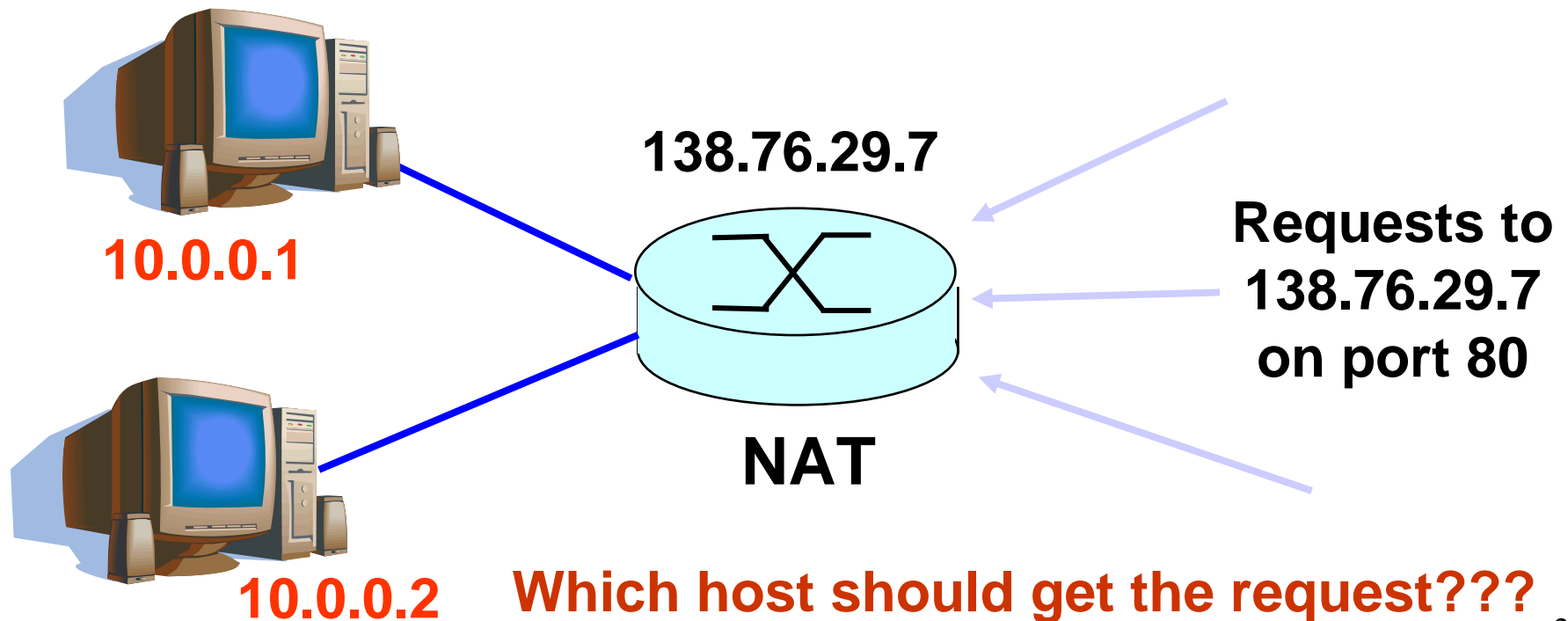
- Create an entry upon seeing a packet
 - Packet with new (source addr, source port) pair
- Eventually, need to delete the map entry
 - But when to remove the binding?
- If no packets arrive within a time window
 - ... then delete the mapping to free up the port #s
 - At risk of disrupting a temporarily 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
 - ... and have a bunch of hosts hiding behind it
- Campus or corporate network
 - NAT at the connection to the Internet
 - Share a collection of public IP addresses
 - Avoid complexity of renumbering end hosts and local routers when changing service providers

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



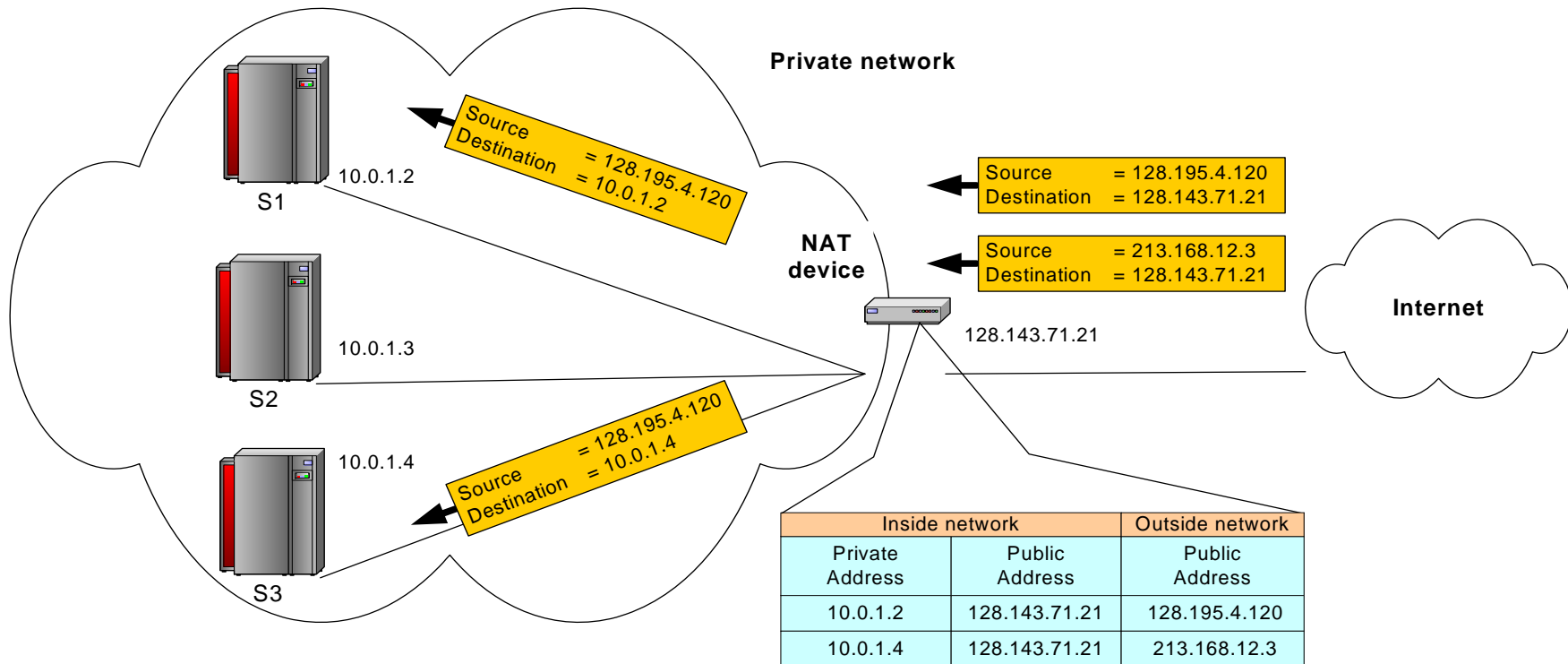
Running Servers Behind NATs

- Running servers is still possible
 - Admittedly with a bit more difficulty
- By explicit configuration of the NAT box
 - E.g., internal service at <dst 138.76.29.7, dst-port 80>
 - ... mapped to <dst 10.0.0.1, dst-port 80>

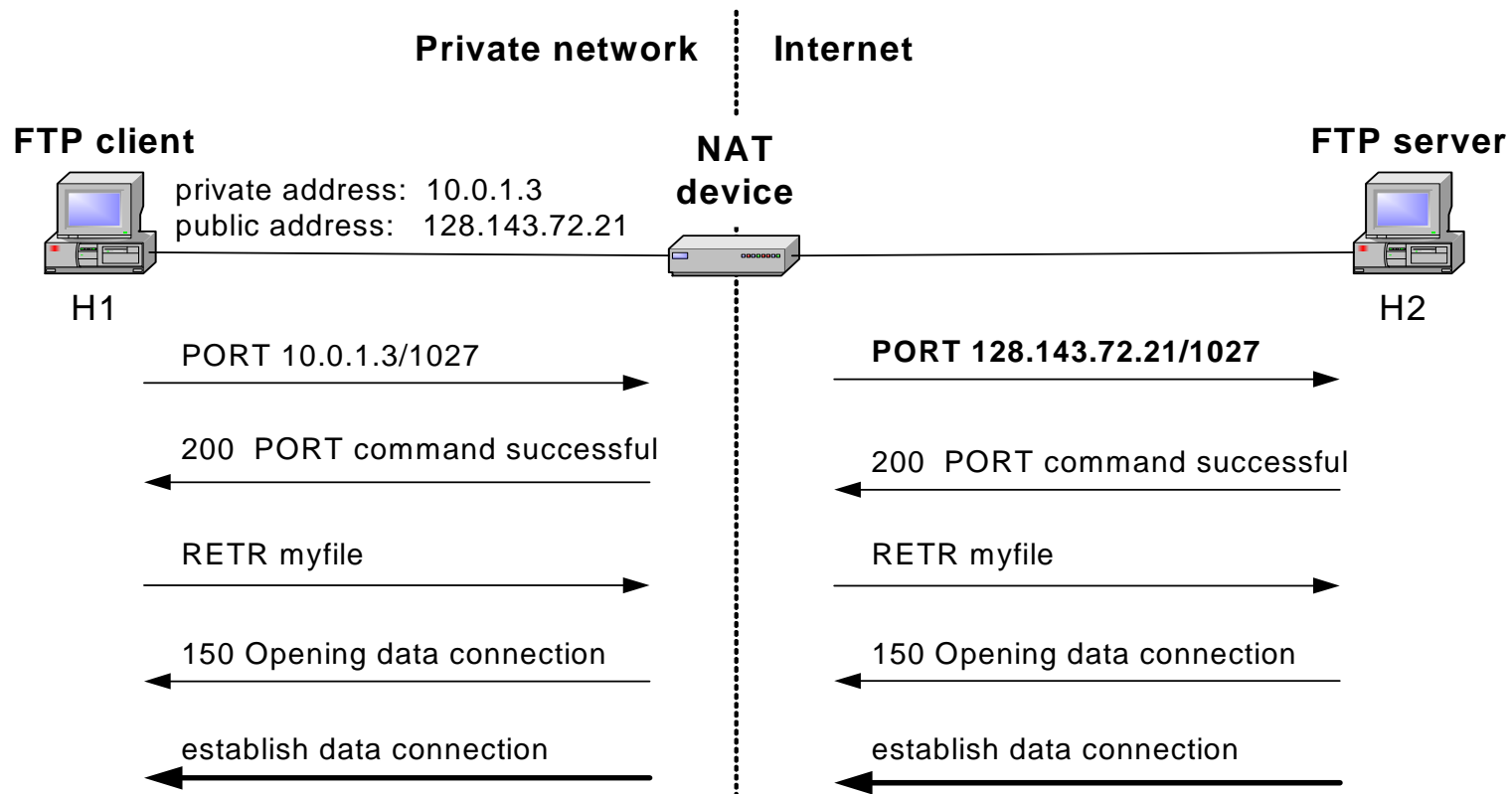
Load Balancing of Servers

- **Scenario:** Balance the load on a set of identical servers, which are accessible from a single IP address
- **NAT solution:**
 - Here, the servers are assigned private addresses
 - NAT device acts as a proxy for requests to the server from the public network
 - The NAT device changes the destination IP address of arriving packets to one of the private addresses for a server
 - A sensible strategy for balancing the load of the servers is to assign the addresses of the servers in a round-robin fashion.

Load Balancing of Servers



Example: NAT and FTP



- NAT device with FTP support

Principled Objections Against NAT

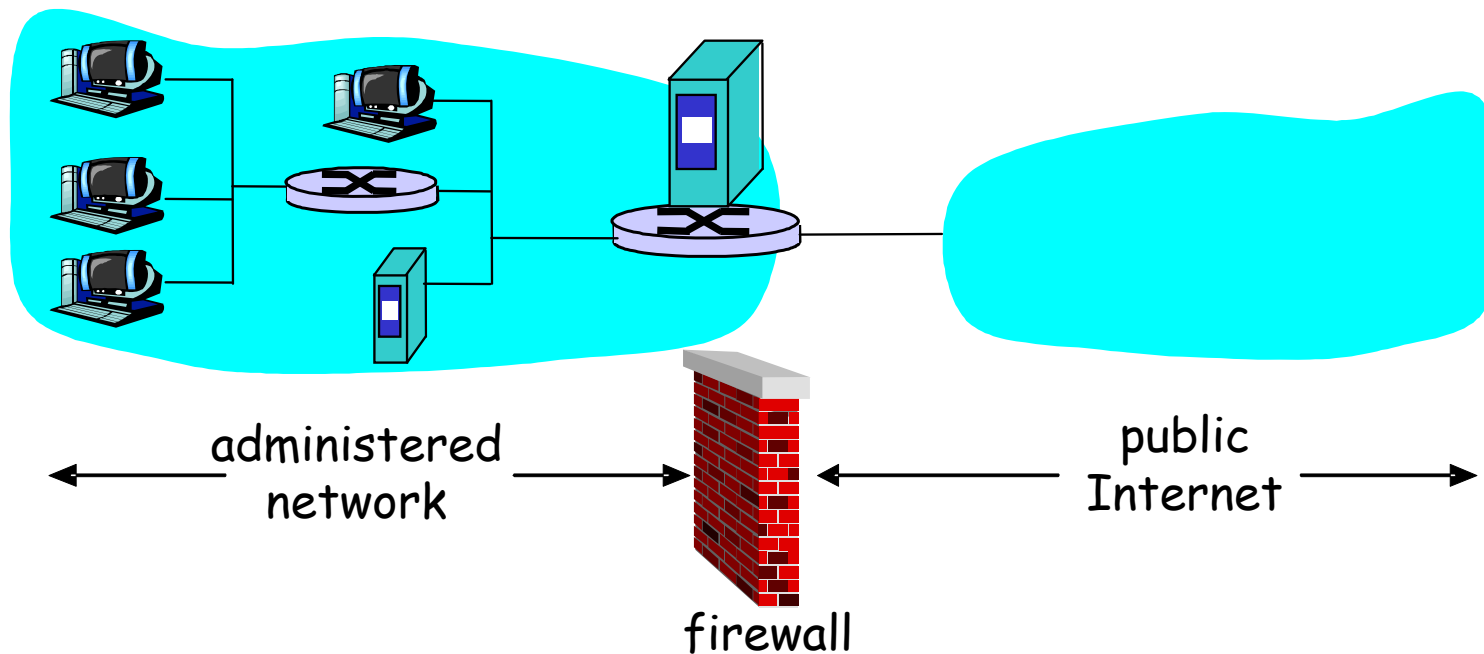
- Routers are not supposed to look at port #s
 - Network layer should care *only* about IP header
 - ... and *not* be looking at the port numbers at all
- NAT violates the *end-to-end* argument
 - Network nodes should not modify the packets
- IPv6 is a cleaner solution
 - Better to migrate than to limp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

**That's what you get when you design a network
that puts power in the hands of end users! 😊**

Firewalls

Firewalls

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



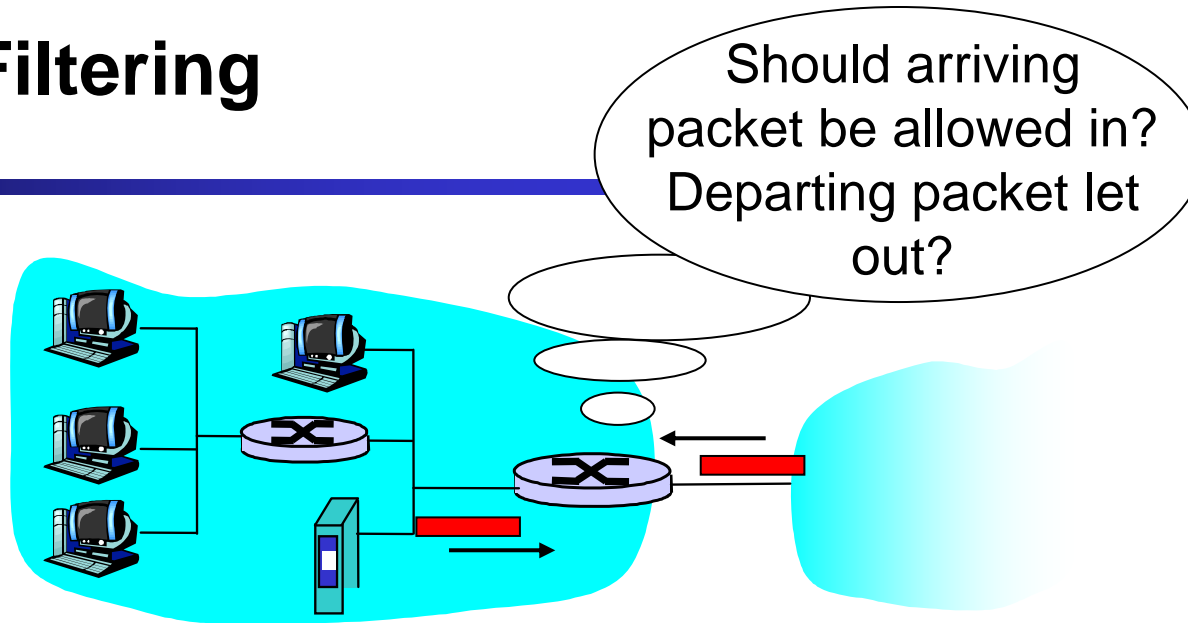
Internet Attacks: Denial of Service

- 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



- Internal network connected to Internet via **firewall**
- Firewall **filters packet-by-packet**, based on:
 - Source IP address, destination IP address
 - TCP/UDP source and destination port numbers
 - ICMP message type
 - TCP SYN and ACK bits

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

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 the 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, a guy 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 the 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

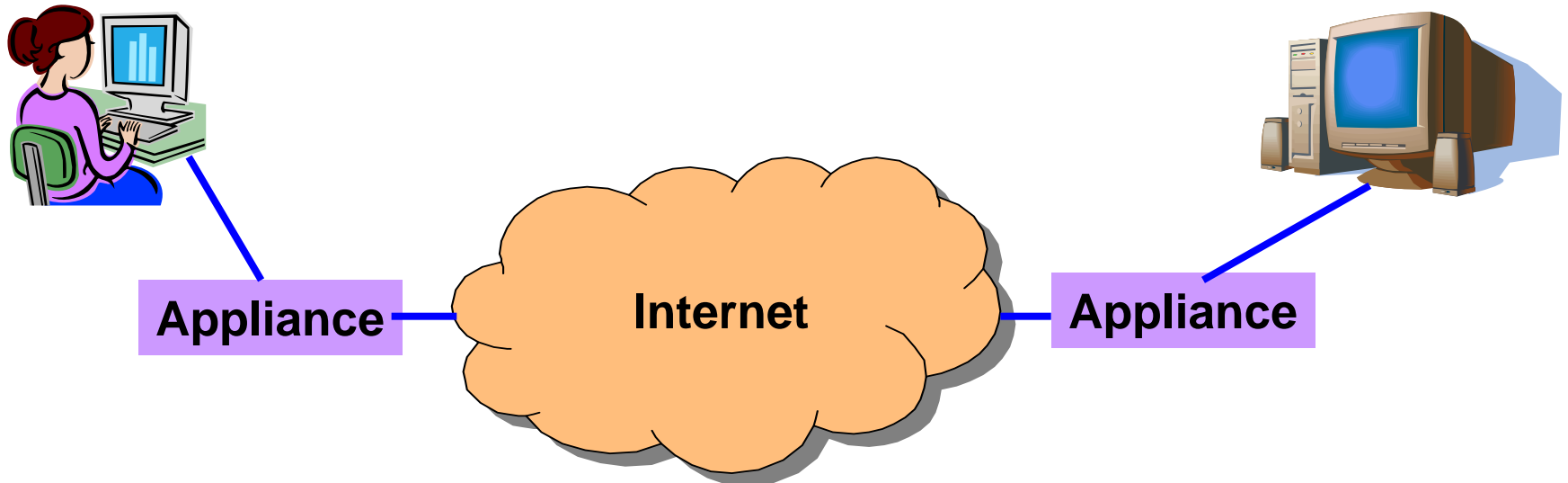
Clever Users Subvert Firewalls

- 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

LAN Appliances

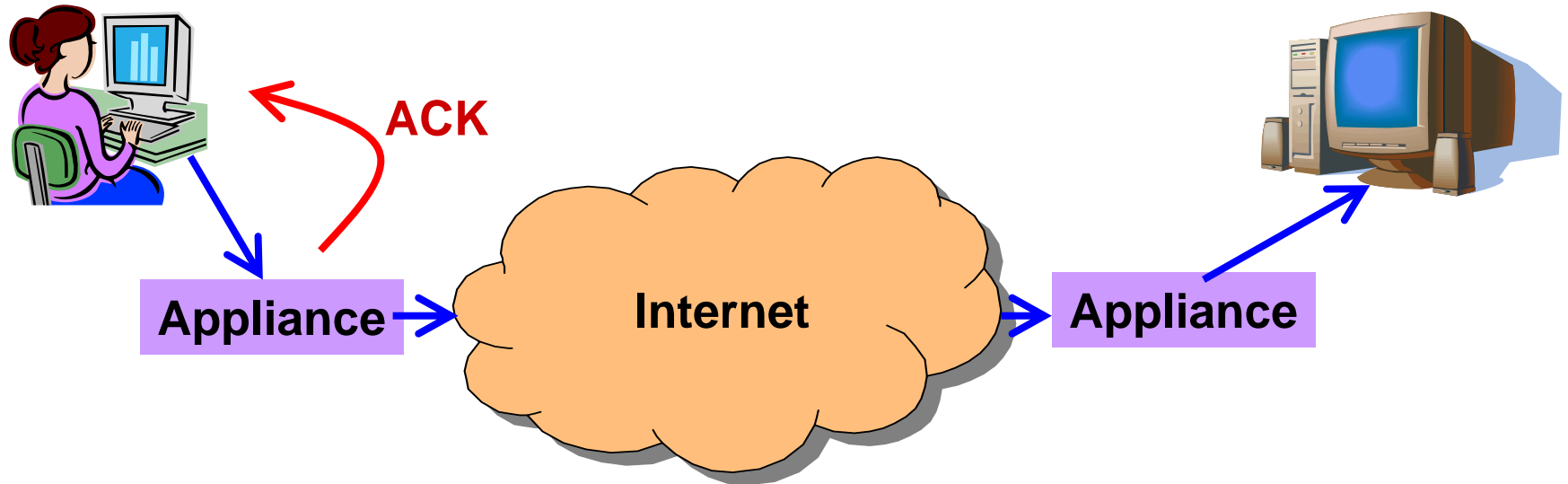
aka WAN Accelerators
aka Application Accelerators

At Connection Point to the Internet



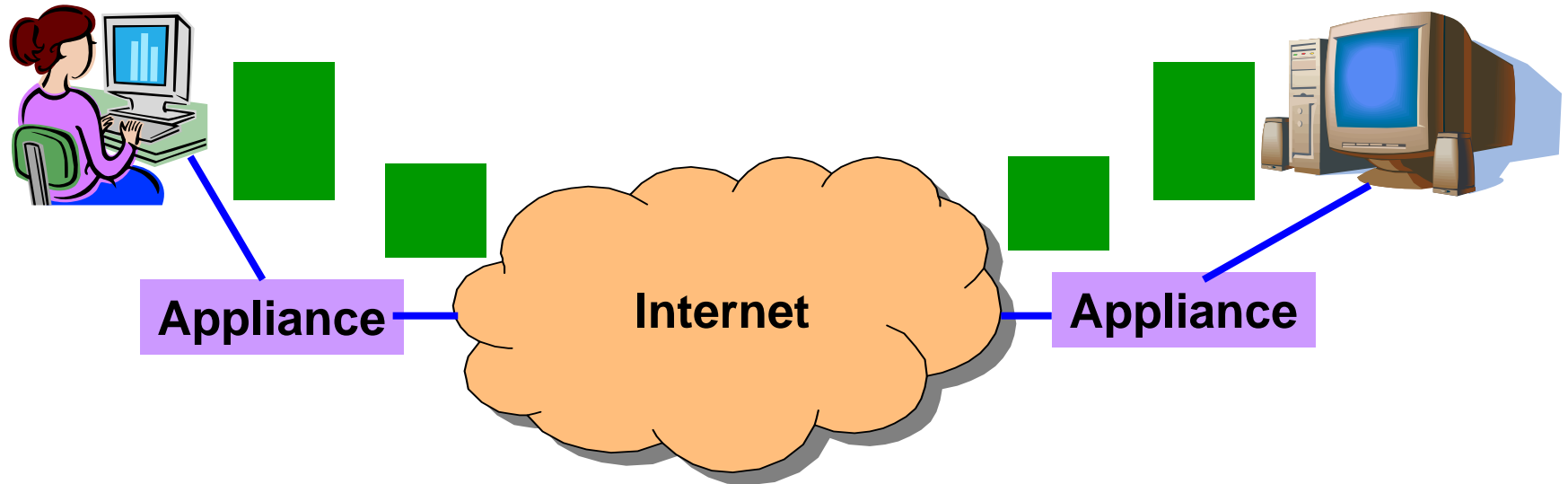
- Improve performance between edge networks
 - E.g., multiple sites of the same company
 - Through buffering, compression, caching, ...
- Incrementally deployable
 - No changes to the end hosts or the rest of the Internet
 - Inspects the packets as they go by, and takes action

Example: Improve TCP Throughput



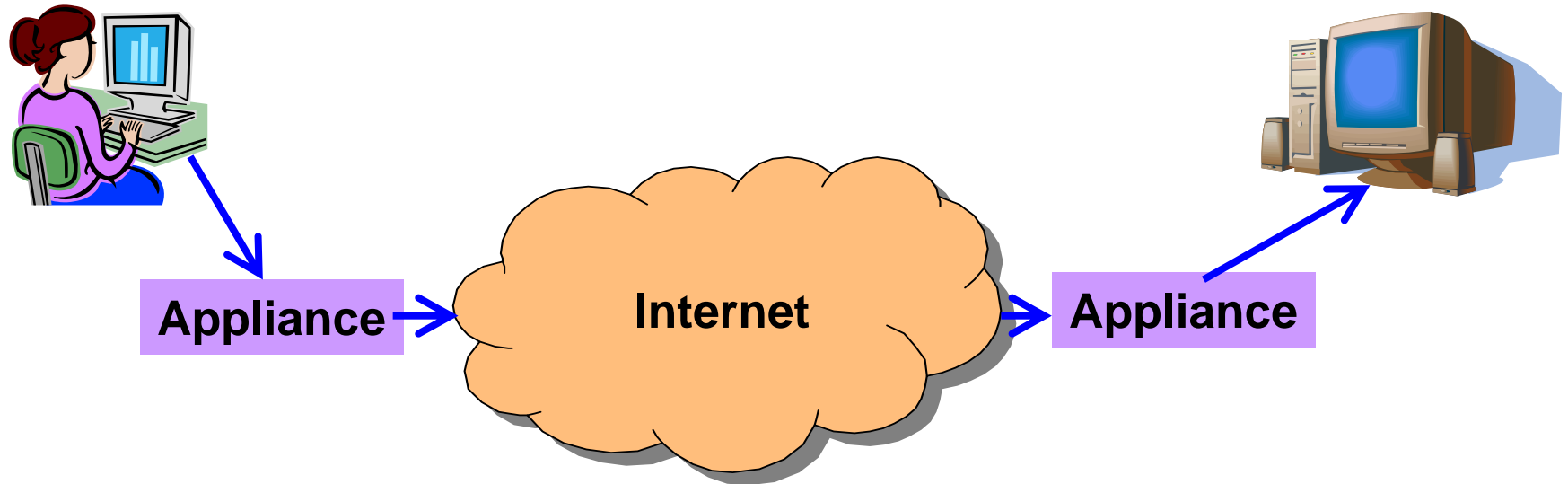
- Appliance with a lot of local memory
- Sends ACK packets quickly to the sender
- Overwrites the receive window with a large value
- Or, even run a new and improved version of TCP

Example: Compression



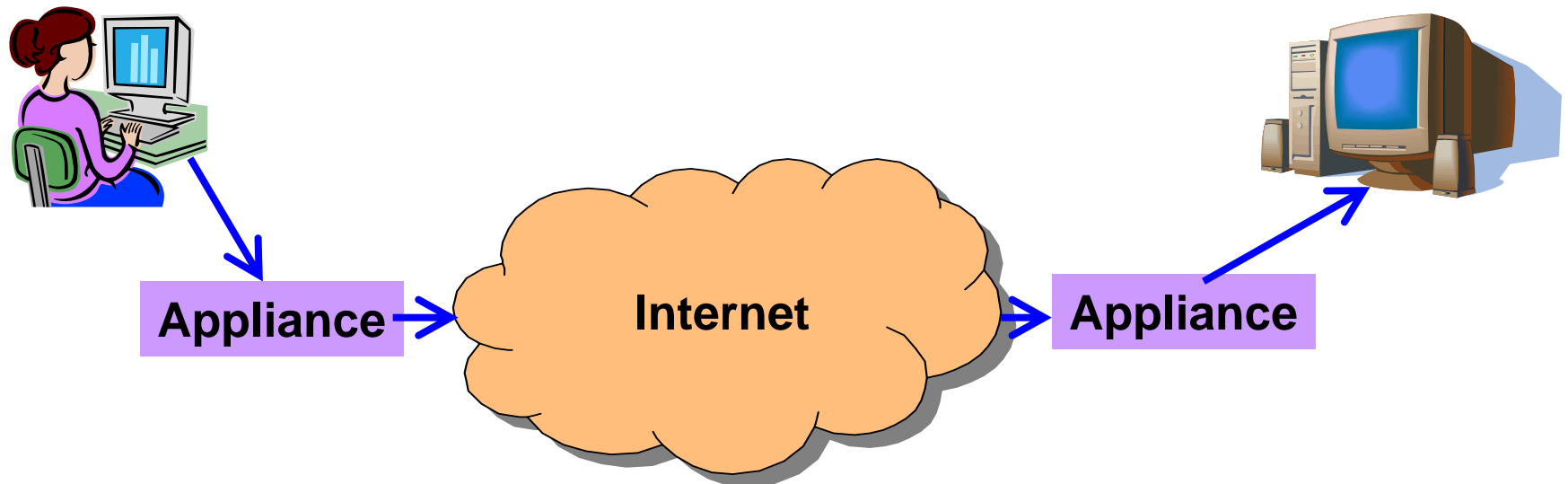
- Compress the packet
- Send the compressed packet
- Uncompress at the other end
- Maybe compress across successive packets

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

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