Middleboxes

Reading: Section 8.4

COS 461: Computer Networks

Spring 2009 (MW 1:30-2:50 in COS 105)

Michael Freedman

Teaching Assistants: Wyatt Lloyd and Jeff Terrace

http://www.cs.princeton.edu/courses/archive/spring09/cos461/

First order business

- Log into your account on labpc-01.cs.princeton.edu

- Type: `chmod –R 700 <cos461>`

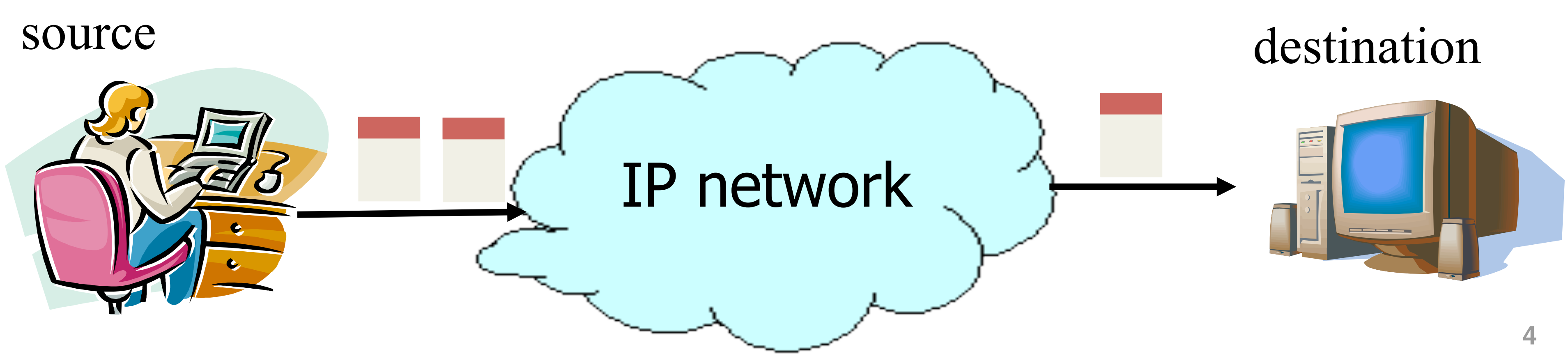
- If you don't have a computer today, do this by tonight. (We'll be checking tomorrow.)

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

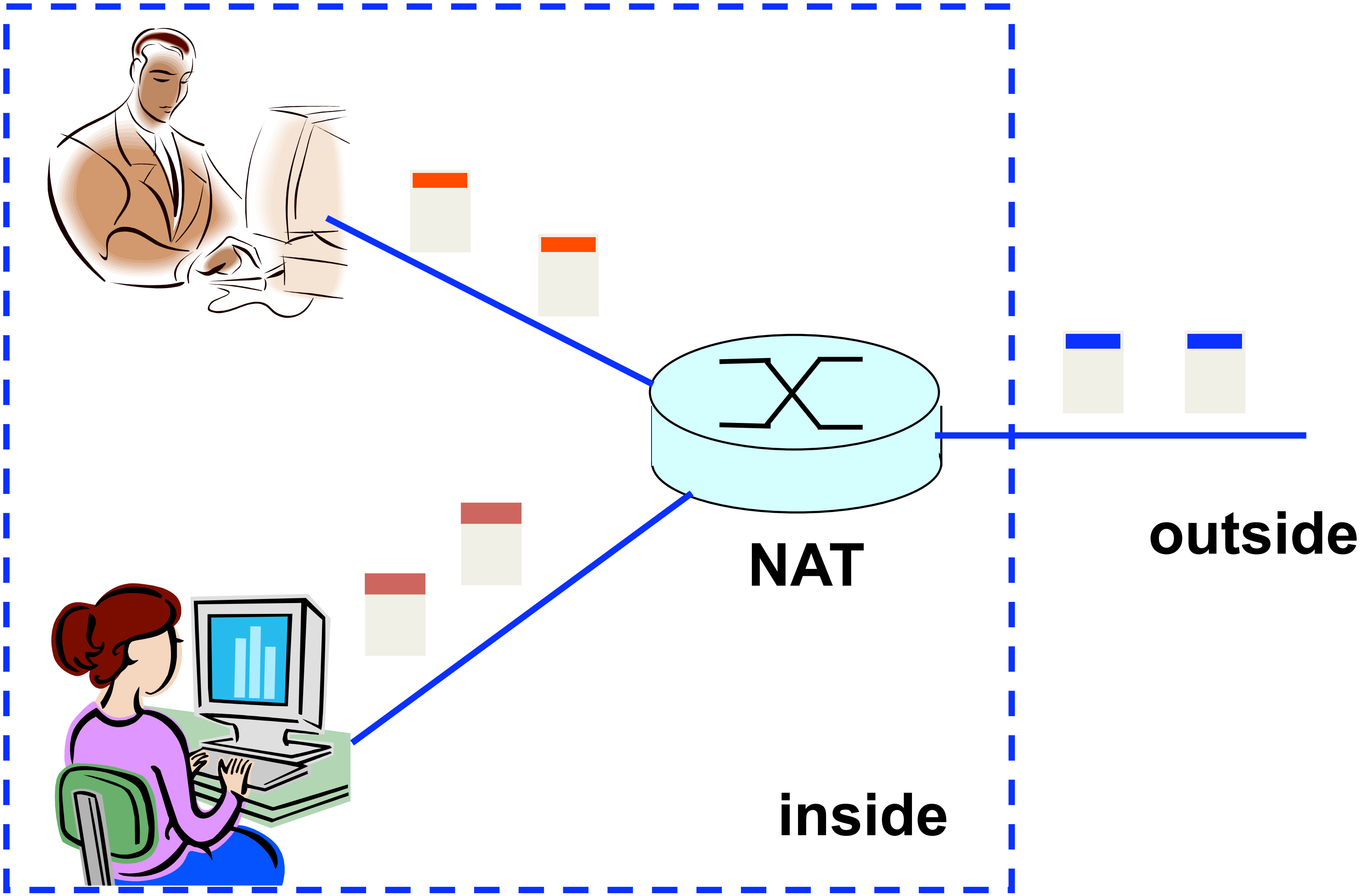
- Would they arise in *any* edge-empowered network, even if redesigned from scratch?

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 ☺

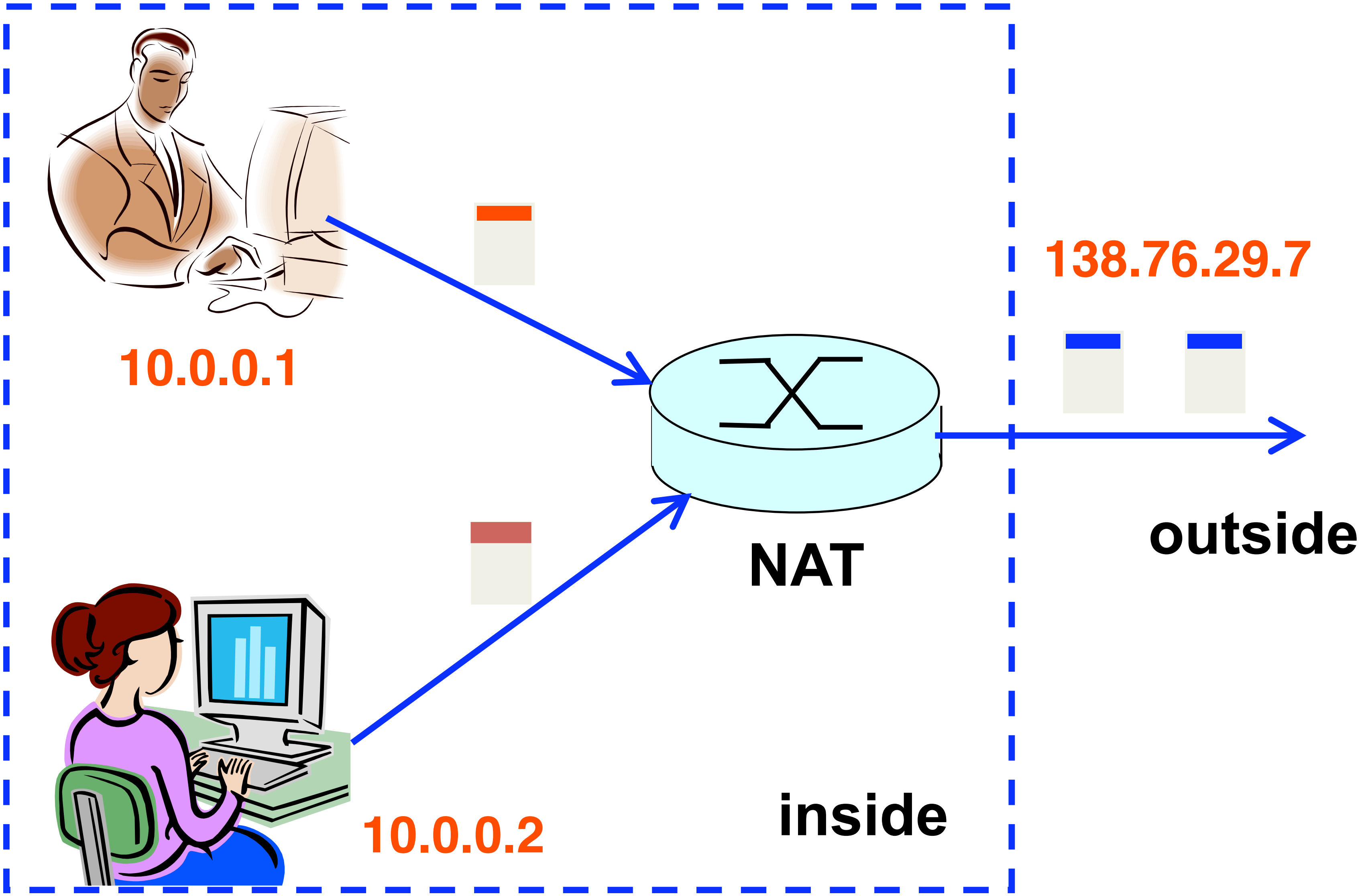
Active Component in the Data Path



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



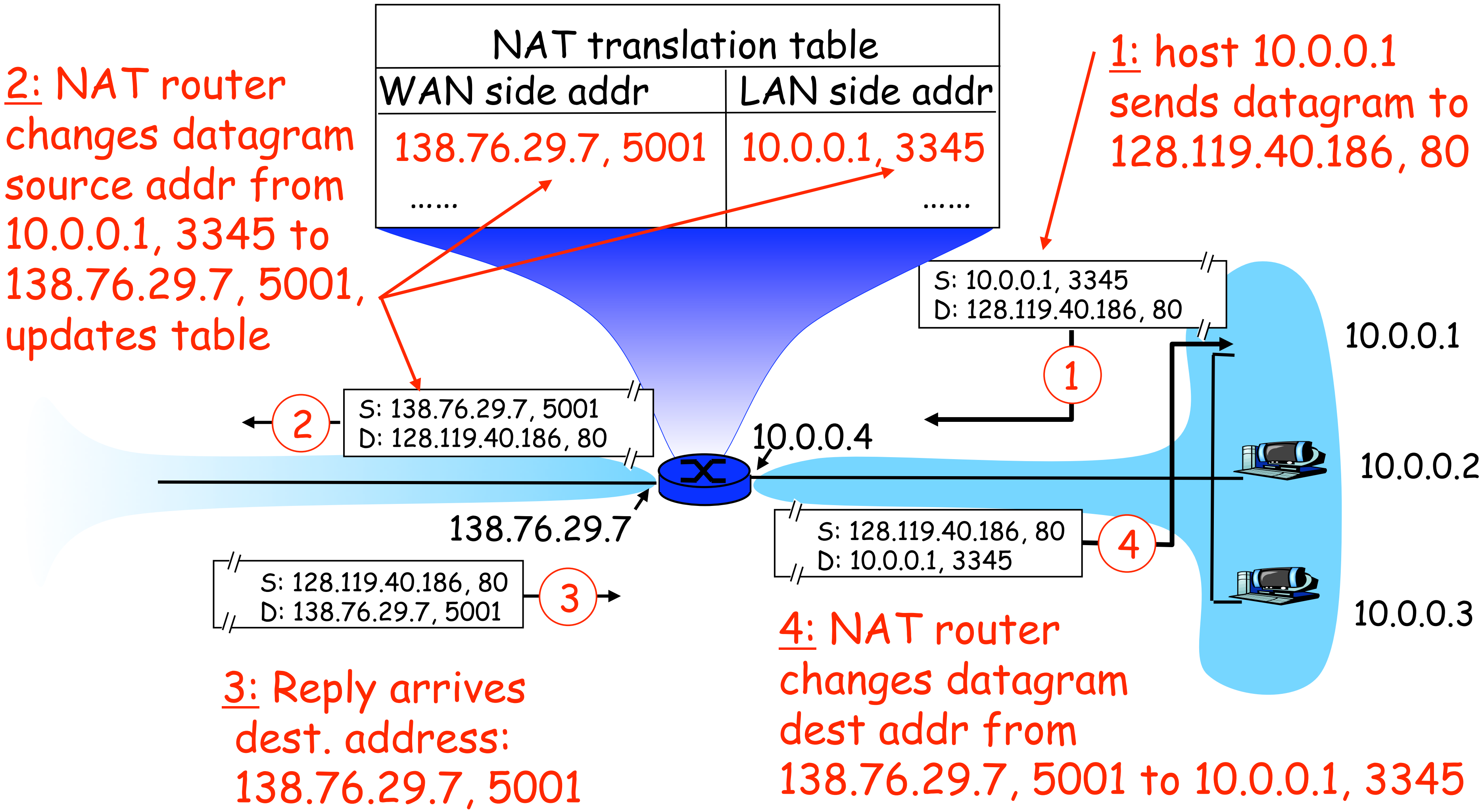
What if Both Hosts Contact Same Site?

- 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 (src addr, port #) to (NAT addr, 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



Maintaining the Mapping Table

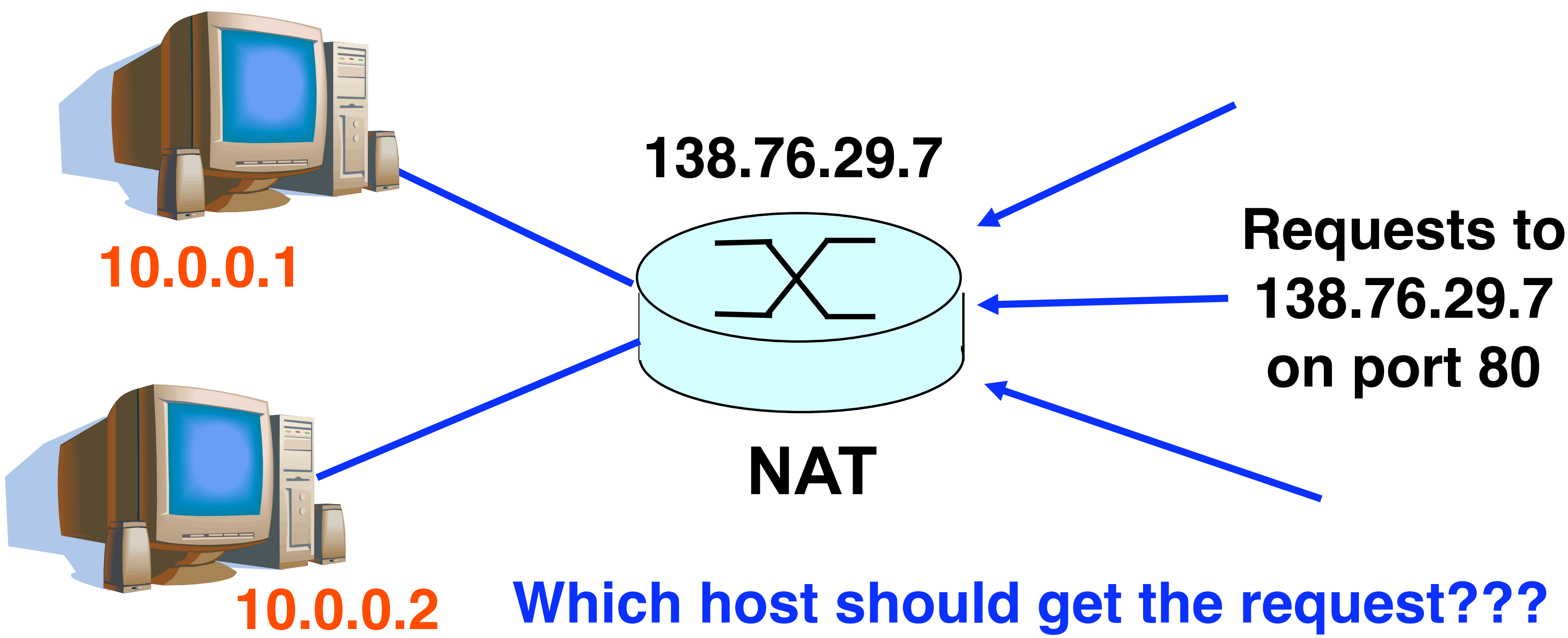
- Create an entry upon seeing a packet
  – Packet with new (source addr, source port) pair
- Eventually, need to delete the map entry
  – But when to remove the binding?
- If no packets arrive within a time window
  – … then delete the mapping to free up the port #s
  – At risk of disrupting a temporarily 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



Which host should get the request???

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>

- More challenging for P2P applications
  - Especially if *both* peers are behind NAT boxes
- Though solutions are possible here as well
  - Existing work-arounds (e.g., in Skype)
  - Ongoing work on "NAT traversal" techniques

Principled Objections Against NAT

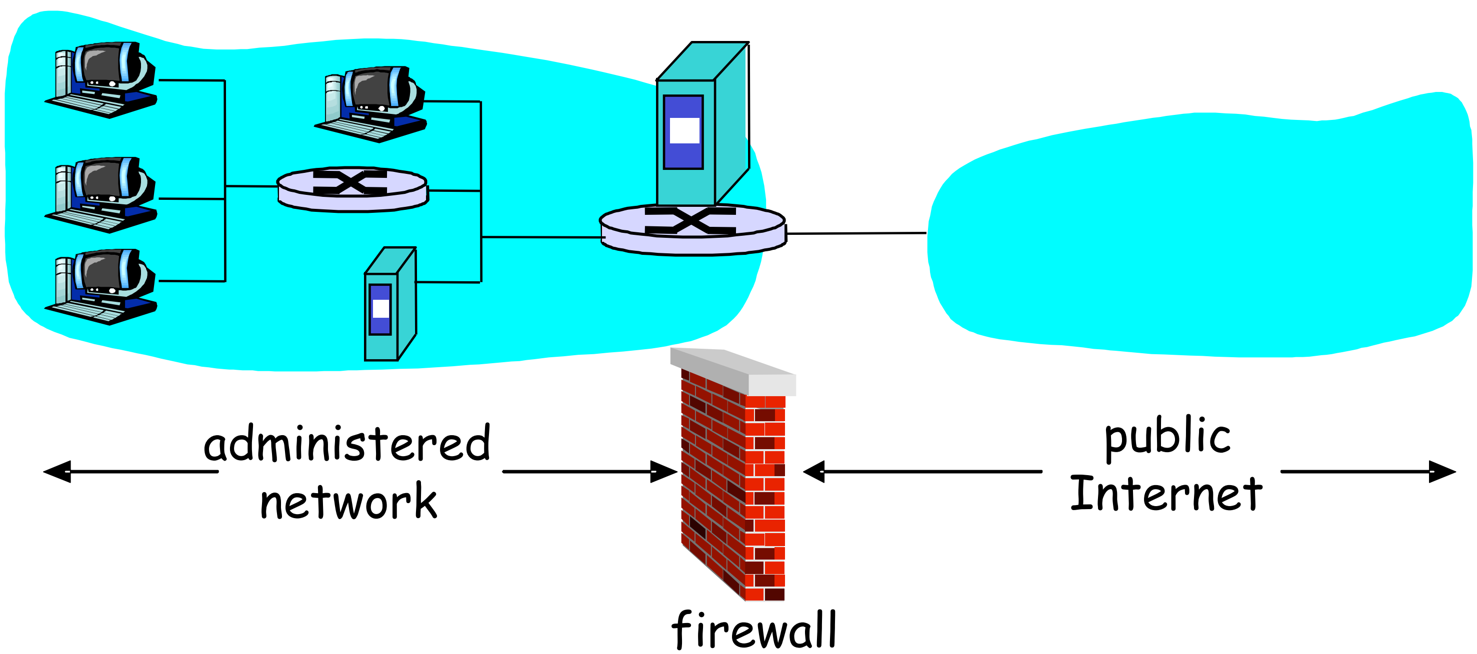
- Routers are not supposed to look at port #s
  - Network layer should care *only* about IP header
  - … and *not* be looking at the port numbers at all
- NAT violates the *end-to-end* argument
  - Network nodes should not modify the packets
- IPv6 is a cleaner solution
  - Better to migrate than to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



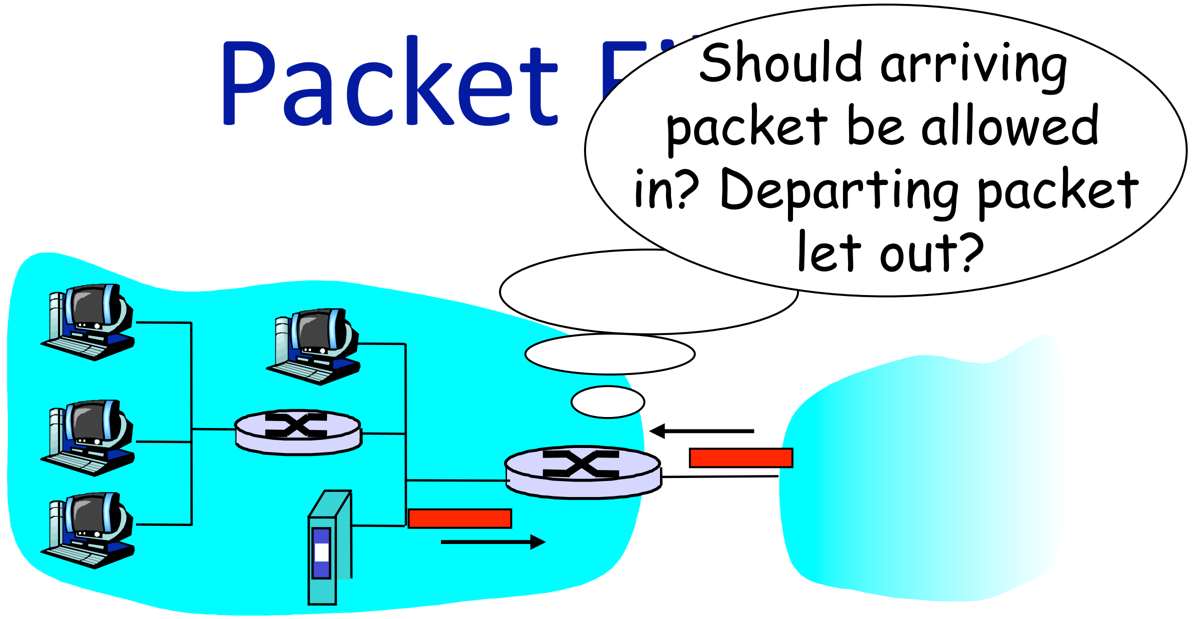
Internet Attacks: Denial of Service

- 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 F



- 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
- Block all packets with TCP port of Counterstrike


SNS @ Princeton Computer Science
United States | Counter Strike Source Server View Counter Strike Source Server List
COUNTER STRIKE SOURCE
Dedicated Counter Strike Servers from $9.95/mo - Click for info!
Server Info
24 seconds ago
SERVER SUMMARY (Manage Game Server)
Game Server: SNS @ Princeton Computer Science
Game Type: Counter Strike Source
IP Address: 128.112.139.199 Port: 27015 Status: Alive
Added On: Feb 19, 2009 Owner: None (claim ownership)
Favorite: Login to add this to your favorite game servers.
JOIN
CLAN INFORMATION
No clan info is available. Are you the game server owner?
If so, click here to add your clan's information to this page!
SERVER RANKING
20th
Game Server Rank:
9351st (20th Percentile) - Counter Strike Source Servers
Highest (past month): 9351st Lowest (past month): 10055th
Game Server Page Views: 29
CURRENT MAP
de_prodigy
Is this image bad or incorrect?
Upload a new one!
Last Map: cs_havana
PLAYER STATS (View All)
Current Players: 0 / 32
Average (past month): 2
SERVER BANNERS
SERVER NAME:
SNS @ Princeton Computer Science
IP ADDRESS: PORT: SERVER STATUS:
128.112.139.199 27015 Online
CURRENT MAP: PLAYERS: SERVER RANK:
de_prodigy 0/32 9351st (20th pctile)
OF PLAYERS (past 24 hours)
GRAPH DATA
NOT AVAILABLE
GAMETRACKER
www.gametracker.com
Free Voice Servers
Host COD5 Servers
Left 4 Dead Hosting
Cheap Ventrilo Hosting
Counter Strike Servers
Teamspeak Servers
Instant Game Servers
Download Ventrilo
Server Blog
There are no blogs for this server.
View Server Blog
Add Blog Post
Historical Data

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, 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

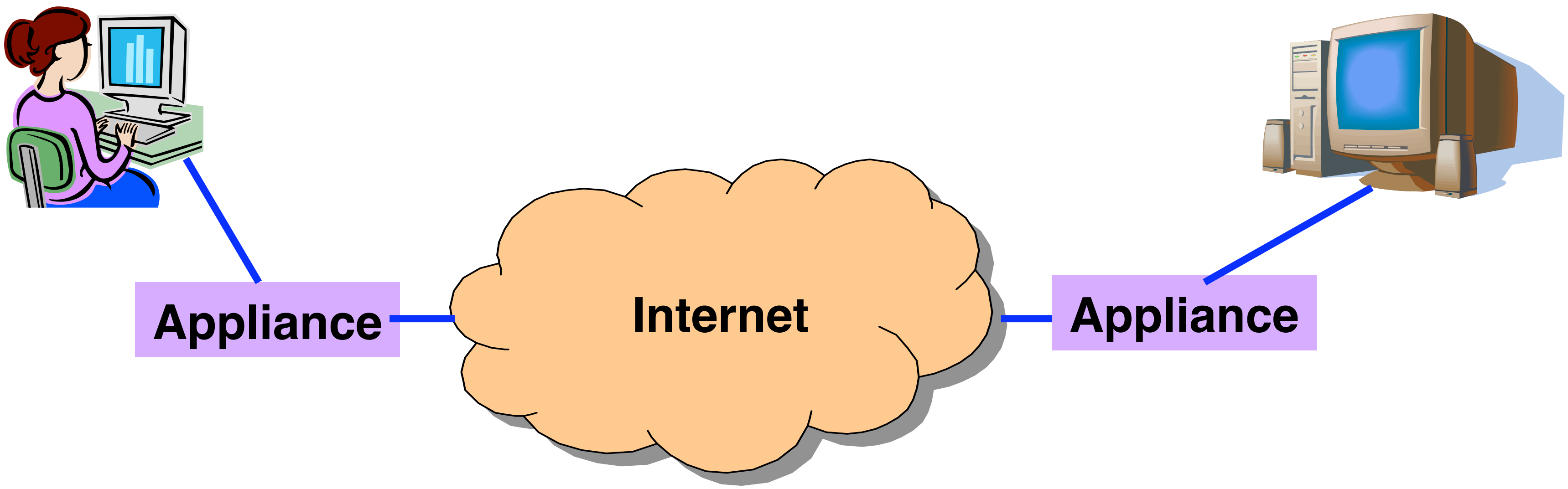
- Per-packet handling
  - Must inspect every packet
  - Challenging on very high-speed links
- Complex filtering rules
  - May have large # of rules
  - May have very complicated rules
- Location of firewalls
  - Complex firewalls near the edge, at low speed
  - Simpler firewalls in the core, at higher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

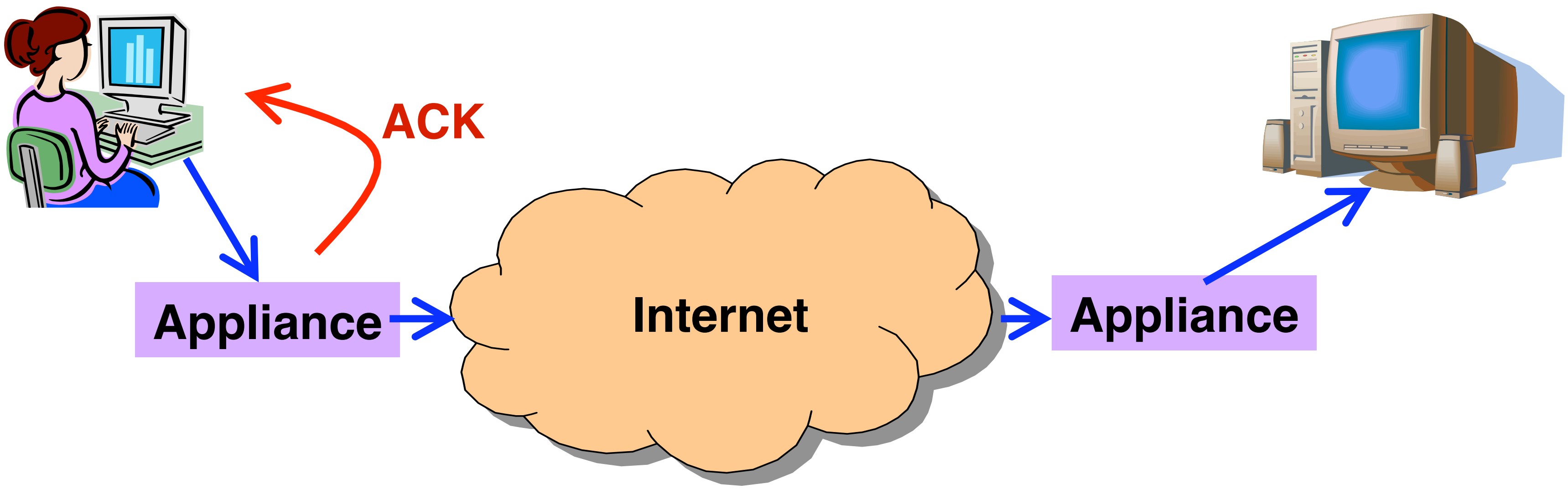
LAN Appliances
aka WAN Accelerators
aka Application Accelerators

At Connection Point to the Internet



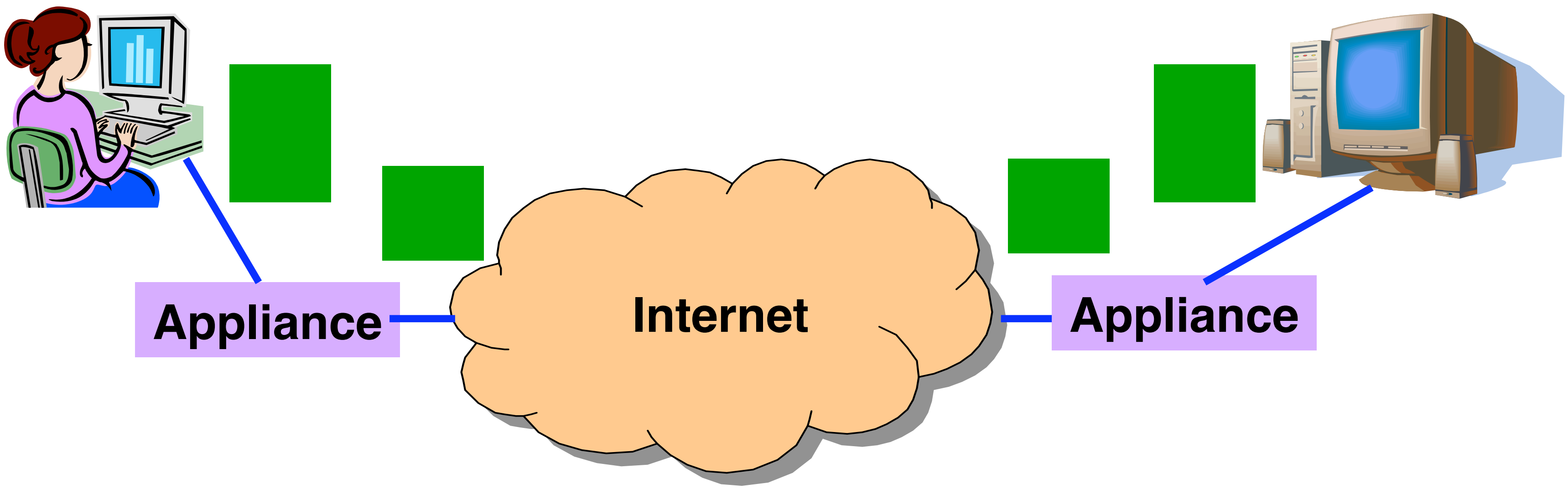
- Improve performance between edge networks
  - E.g., multiple sites of the same company
  - Through buffering, compression, caching, …
- Incrementally deployable
  - No changes to the end hosts or the rest of the Internet
  - Inspects the packets as they go by, and takes action

Example: Improve TCP Throughput



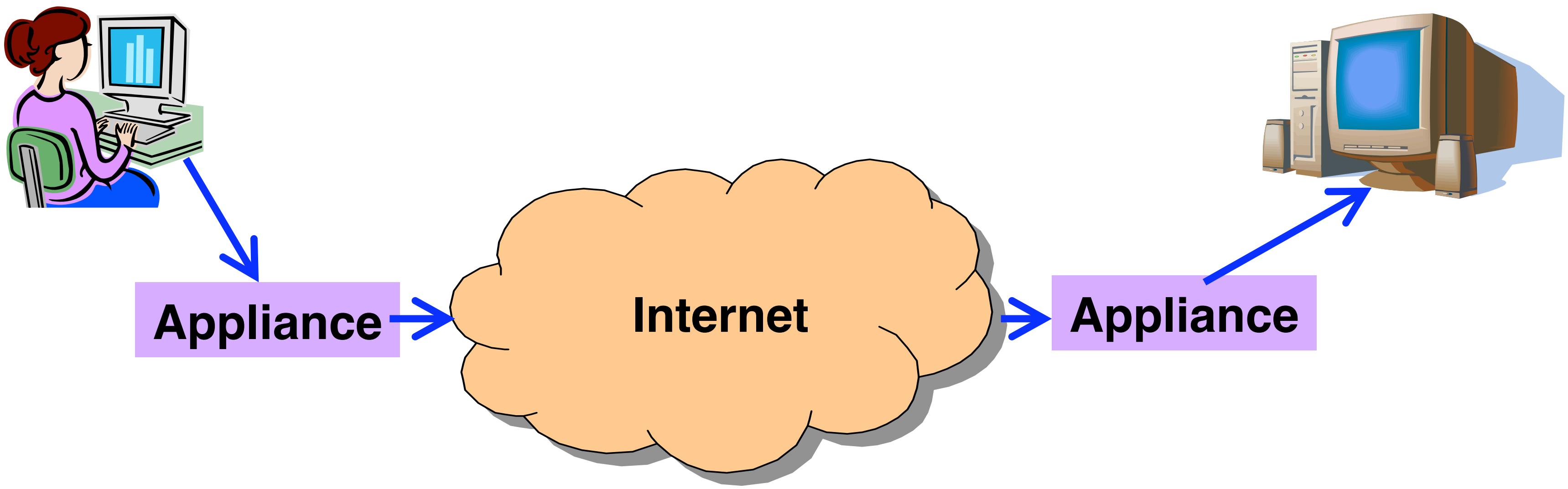
- Appliance with a lot of local memory
- Sends ACK packets quickly to the sender
- Overwrites receive window with a large value
- Or, even run a new and improved version of TCP

Example: Compression



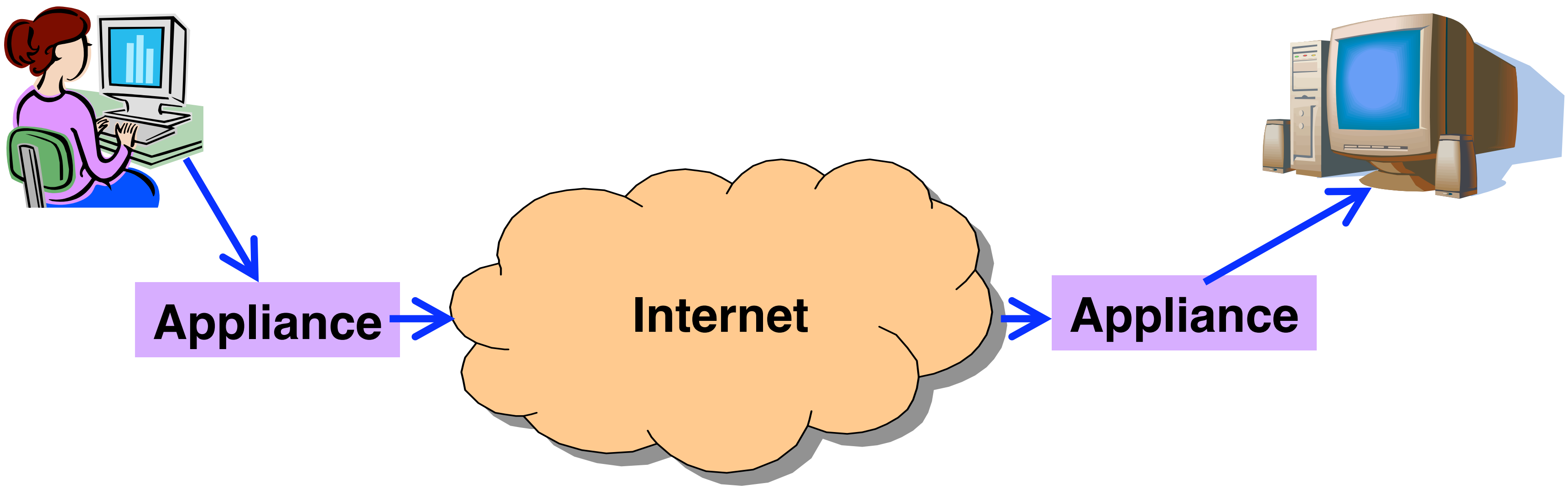
- Compress the packet
- Send the compressed packet
- Uncompress at the other end
- Maybe compress across successive packets

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
- Next class
  - Repeaters/hubs and bridges/switches
  - Reading: Section 3.2