Middleboxes and Tunneling

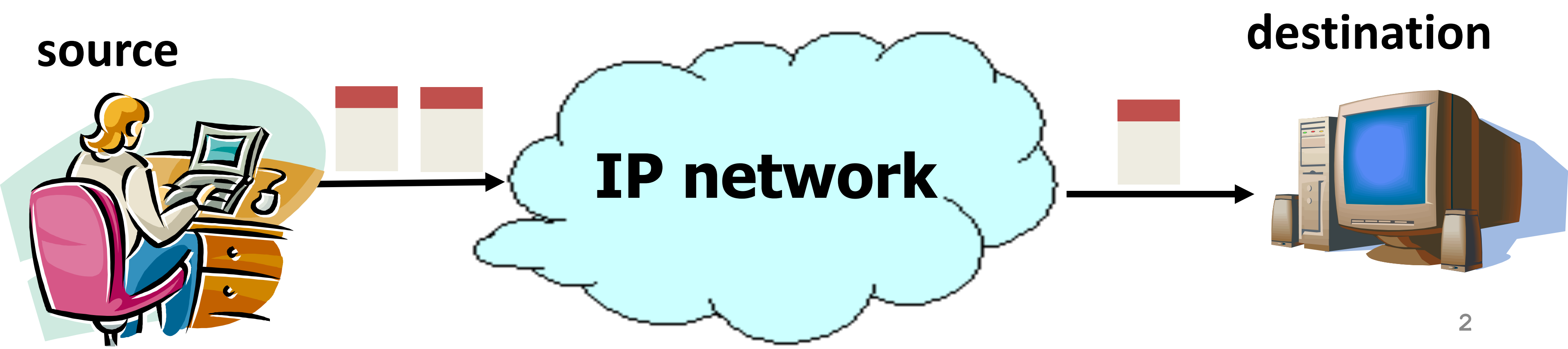
Kyle Jamieson

COS 461: Computer Networks

Lecture 8

Internet Ideal: Simple Network Model

- 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
  - Host changing address as it moves
- IP address depletion
  - Multiple hosts using the same address
- Security concerns
  - Detecting and blocking unwanted traffic
- Replicated services
  - Load balancing over server replicas
- Performance concerns
  - Allocating bandwidth, caching content, …
- Incremental deployment
  - New technology deployed in stages

Middleboxes BREAK the Simple Network Model

- Middleboxes are intermediaries
  - Interposed between communicating hosts
  - Often without knowledge of one or both parties
- Myriad uses
  - Address translators
  - Firewalls
  - Traffic shapers
  - Intrusion detection
  - Transparent proxies
  - Application accelerators

"An abomination!"

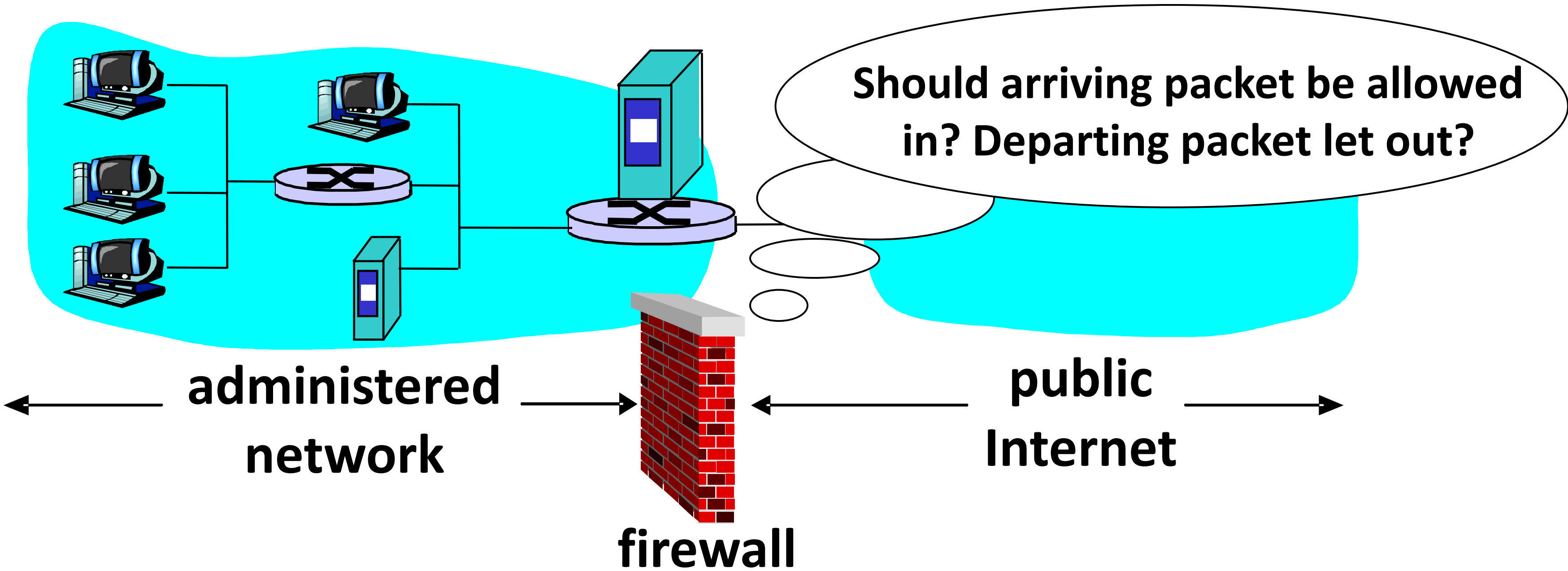
- **Violation of layering**
- **Hard to reason about**
- **Responsible for subtle bugs**

"A practical necessity!"

- **Solve real/pressing problems**
- **Needs not likely to go away**

Firewalls

Firewalls



- A firewall filters packet-by-packet, based on:
  - Source and destination IP addresses and port #'s
  - TCP SYN and ACK bits; ICMP message type
  - *Deep packet inspection* of packet contents (*DPI*)

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 headers, deciding whether to allow/deny
- Rule order matters
  - Once packet matches rule, drop/keep decision is made

Firewall Configuration Example

- Alice runs a network in 222.22/16, wants to **allow Bob's school** to access **only** certain hosts
  - **Bob** is on **111.11/16**
  - Alice's **designated hosts** are in **222.22.22/24**
- Alice **doesn't trust** Trudy, **inside Bob's network**
  - Trudy's hosts are in 111.11.11/24
- Alice doesn't want any other Internet traffic

Firewall Configuration Rules

1. Allow Bob's network in to special destinations
   - **ALLOW** (src=111.11/16, dst = 222.22.22/24)
2. Block Trudy's machines
   - **DENY** (src = 111.11.11/24, dst = 222.22/16)
3. Block world
   - **DENY** (src = 0/0, dst = 0/0)

- Order?

(Y) 3, 1 (M) 3, 1, 2 (C) 1, 3 (A) 2, 1, 3

Firewall Configuration Rules

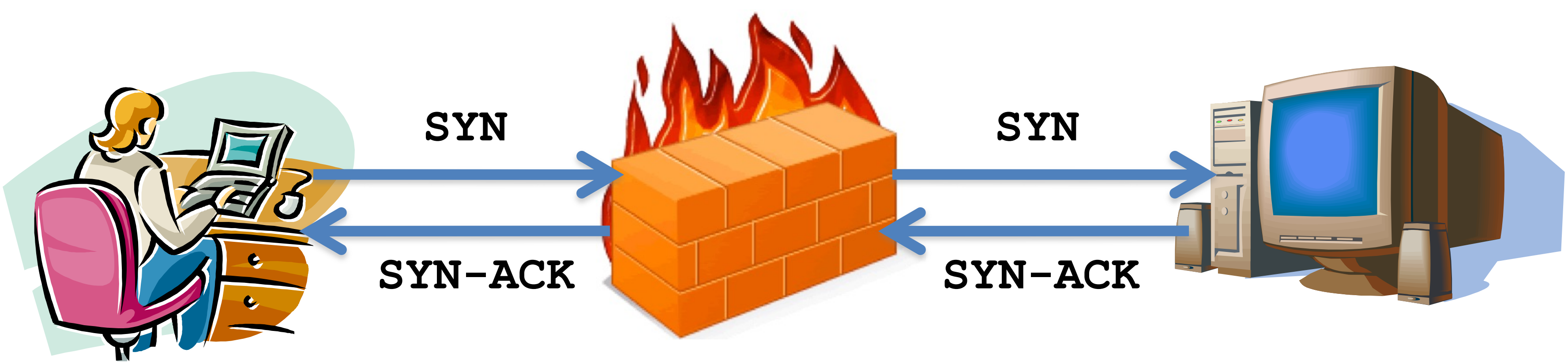
1. Allow Bob's network in to special destinations
   - **ALLOW** (src=111.11/16, dst = 222.22.22/24)
2. Block Trudy's machines
   - **DENY** (src = 111.11.11/24, dst = 222.22/16)
3. Block world
   - **DENY** (src = 0/0, dst = 0/0)

- Order?

(Y) 3, 1 (M) 3, 1, 2 (C) 1, 3 (A) 2, 1, 3

Stateful Firewall

- *Stateless* firewall:
  - Treats each packet independently
- *Stateful* firewall
  - Remembers connection-level information
  - E.g., client initiating connection with a server
  - … allows the server to send return traffic



A Variation: Traffic Management

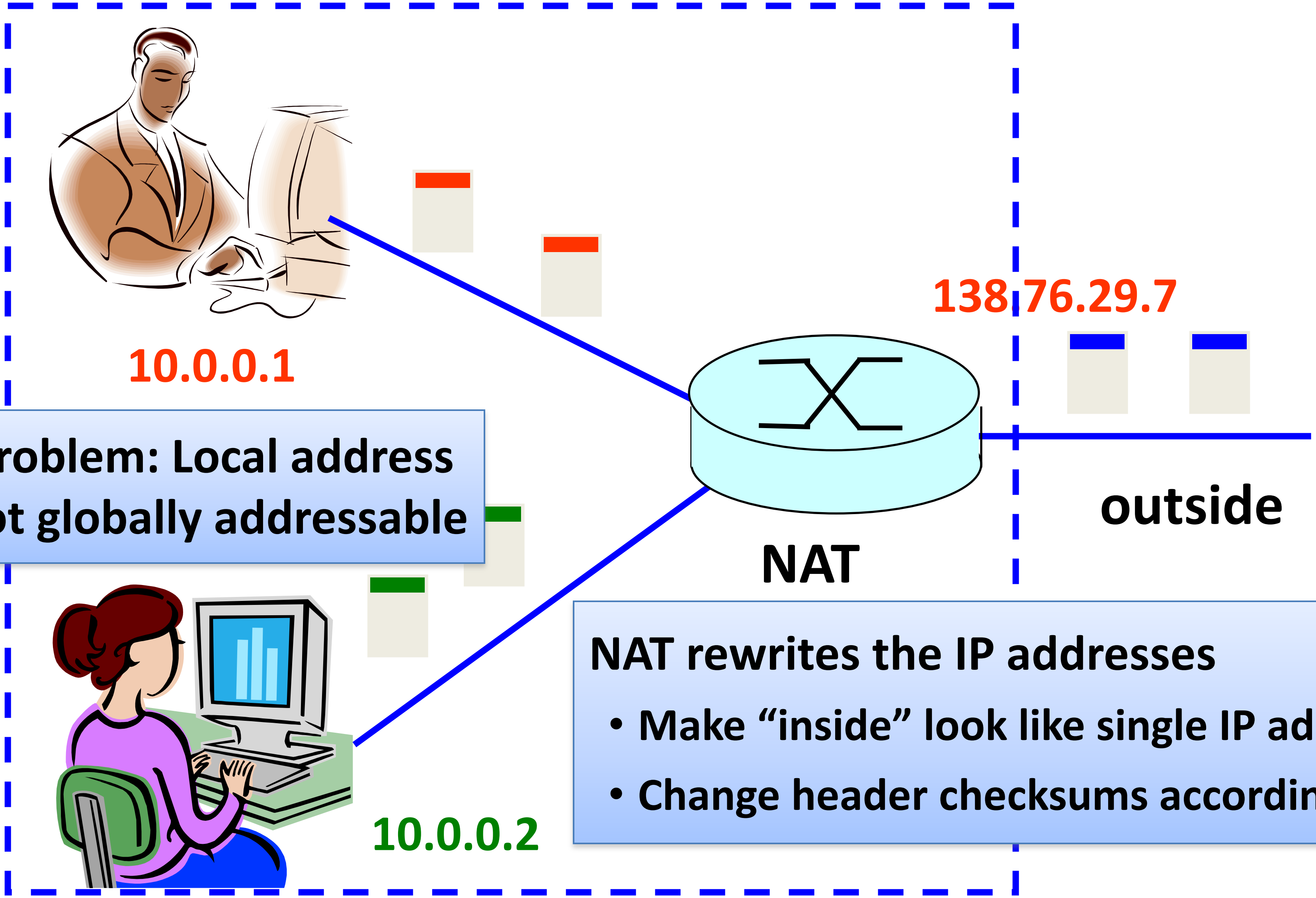
- Permit vs. deny is too binary a decision
  - Classify traffic using rules, handle classes differently
- Traffic shaping (rate limiting)
  - Limit the amount of bandwidth for certain traffic
- Separate queues
  - Use rules to group related packets
  - And then do weighted fair scheduling across groups

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 as a short-term remedy
  - Now: NAT is widely deployed, much more than IPv6

Network Address Translation

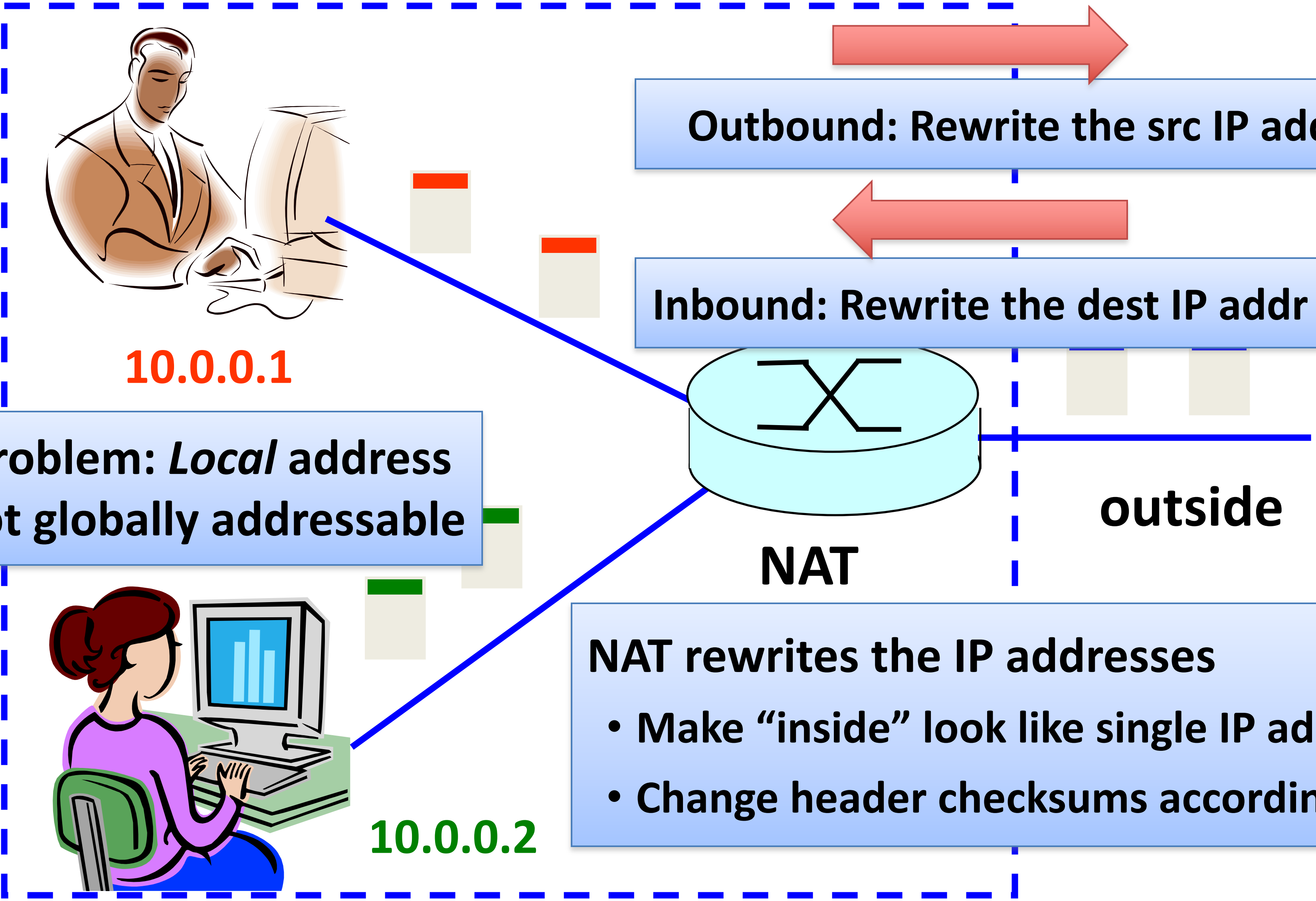


Problem: Local address not globally addressable

NAT rewrites the IP addresses

- **Make "inside" look like single IP addr**
- **Change header checksums accordingly**

Network Address Translation



Outbound: Rewrite the src IP addr

Inbound: Rewrite the dest IP addr

Problem: *Local* address not globally addressable

NAT rewrites the IP addresses

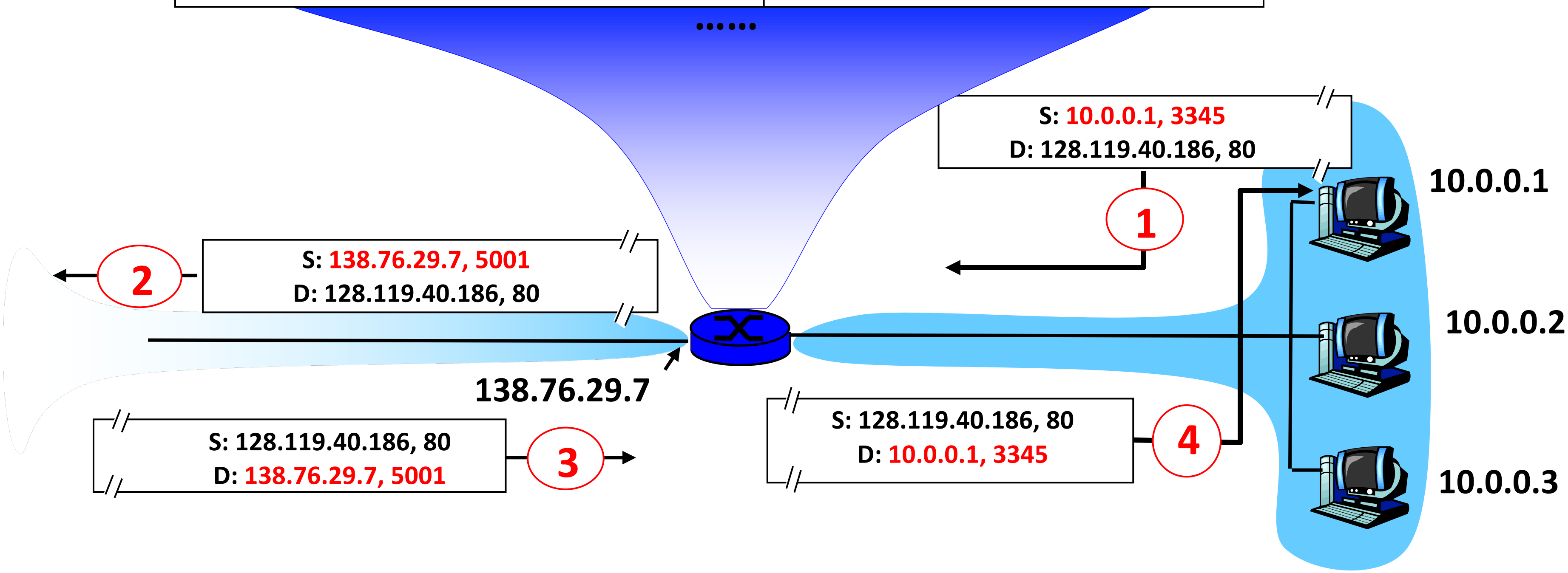
- **Make “inside” look like single IP addr**
- **Change header checksums accordingly**

Port-Translating NAT

- Two hosts communicate with same destination
  - Destination needs to differentiate the two
- Map outgoing packets
  - Change source IP address **and source port**
- Maintain a translation table
  - Map of (src addr, port #) to (local addr, old port #)
- Map incoming (reply) packets
  - Map (**dst** addr, port #) to (local addr, old port #)

Network Address Translation Example

NAT translation table	
WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345



Maintaining the Mapping Table

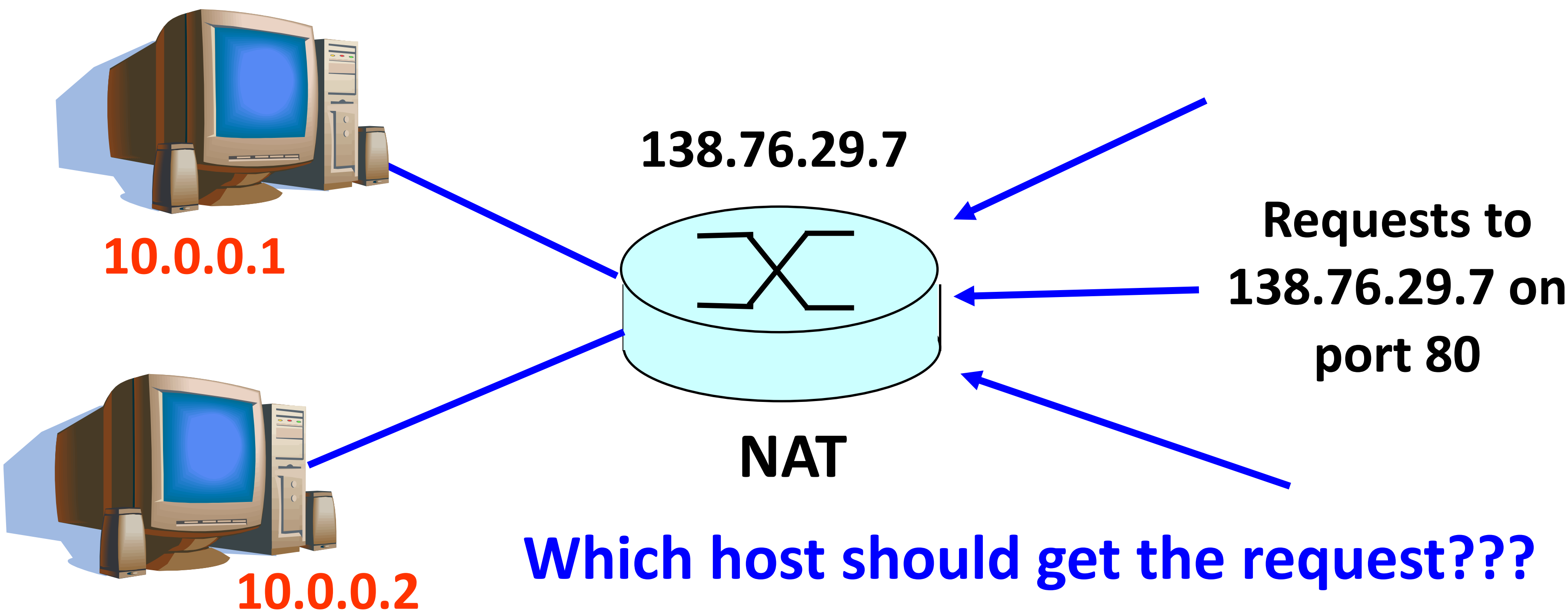
- Create an entry upon seeing an outgoing packet
  – Packet with new (local addr, source port) pair
- Eventually, need to delete entries to free up #'s
  – When? If no packets arrive before a timeout
  – (At risk of disrupting a temporarily idle connection)
- An example of "*soft state*"
  – *i.e.*,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

Where is NAT Implemented?

- Home wireless router
  - Integrates router, Wi-Fi, 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

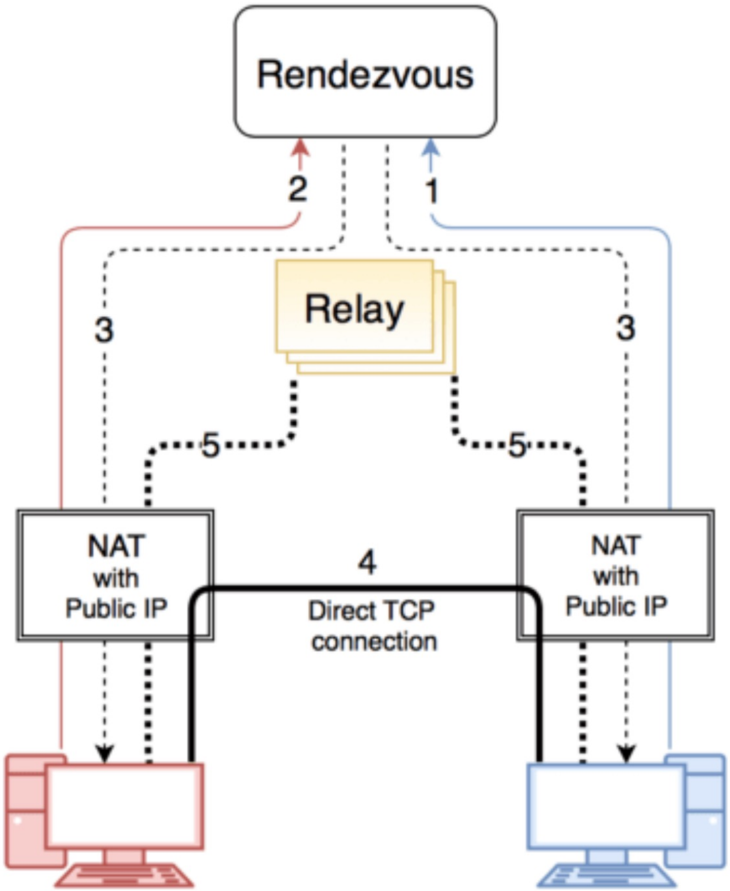
- Port numbers are meant to identify sockets
  - Yet, NAT uses them to identify end hosts
  - Makes it hard to run a server behind a NAT



Which host should get the request???

- **Explicit config at NAT for incoming conn's**

Not just servers: peer-to-peer traffic



1. Peer "registers" with rendezvous
2. Other peer contacts rendezvous
3. Rendezvous sends to each peer the others' IP:port
4. Try to connect in each direction. If one succeeds, done
5. Otherwise, proxy through relay

Principled Objections Against NAT

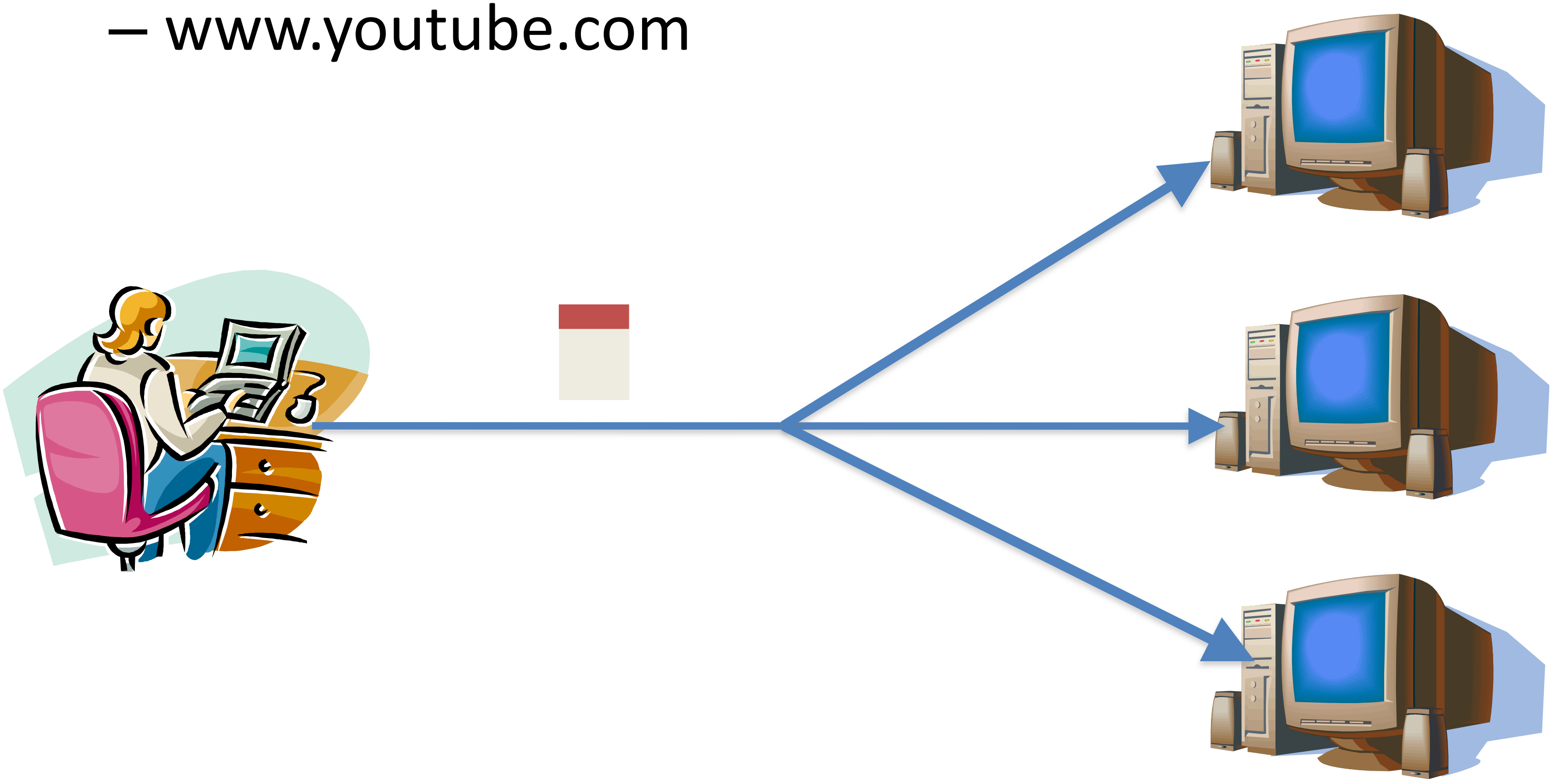
- Routers are not supposed to look at port #s
  - Network layer should care only about *IP* header
  - … and not be looking at the *port numbers* at all
- NAT violates the end-to-end argument
  - Network nodes should not modify the packets
- IPv6 is a cleaner solution
  - Better to migrate than to limp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

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

Load Balancers

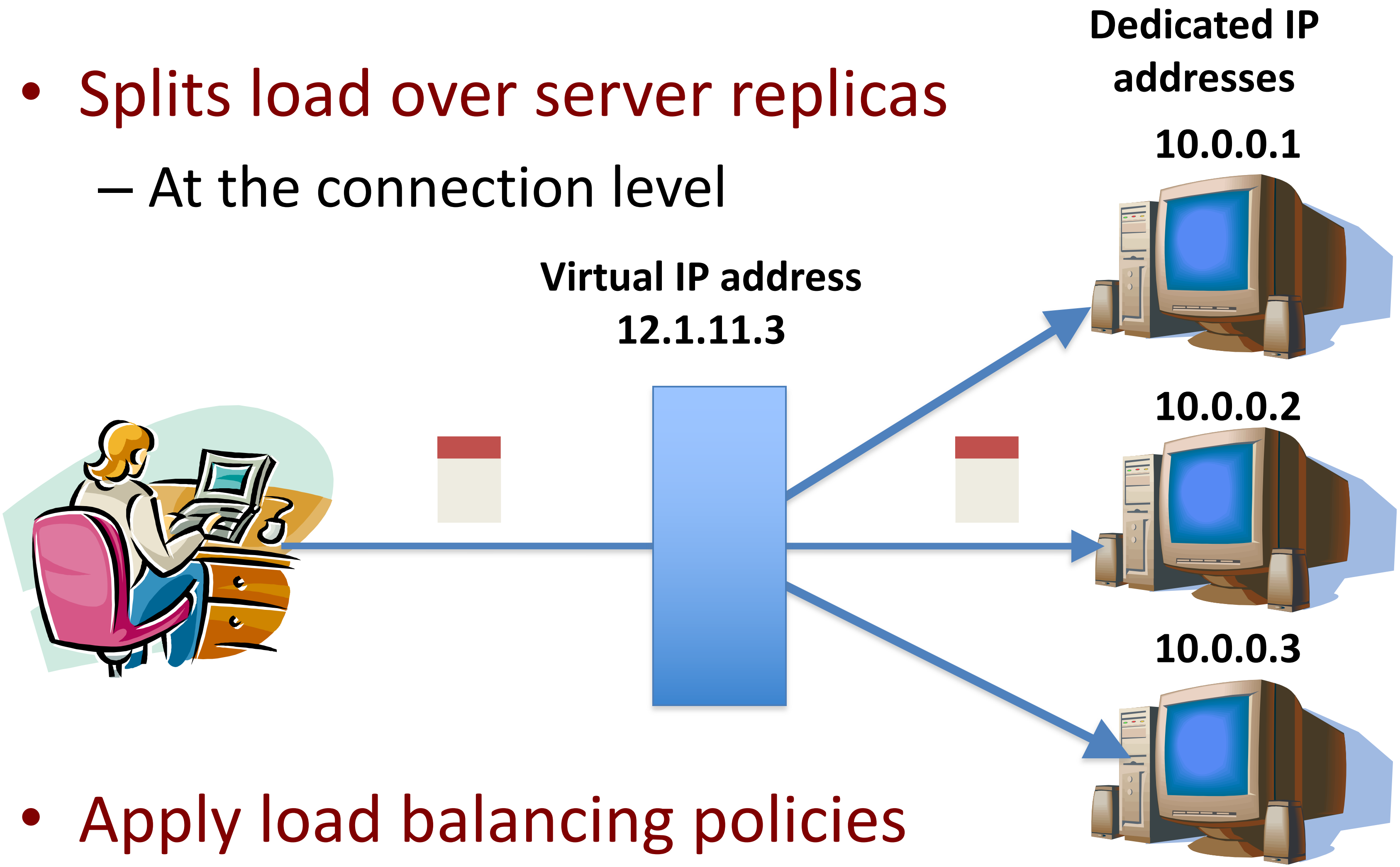
Replicated Servers

- One site, many servers
  - www.youtube.com



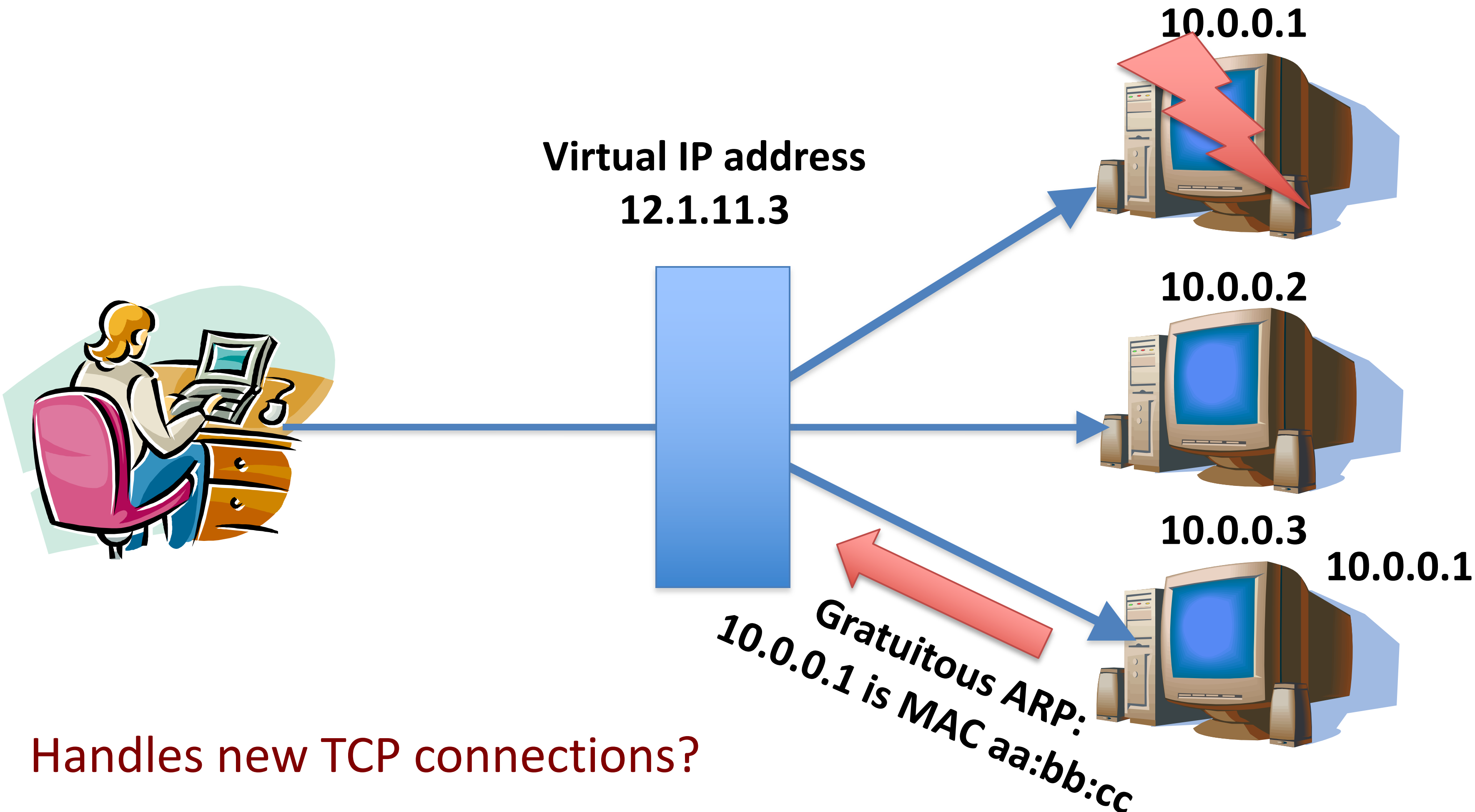
Load Balancer

- Splits load over server replicas
  - At the connection level



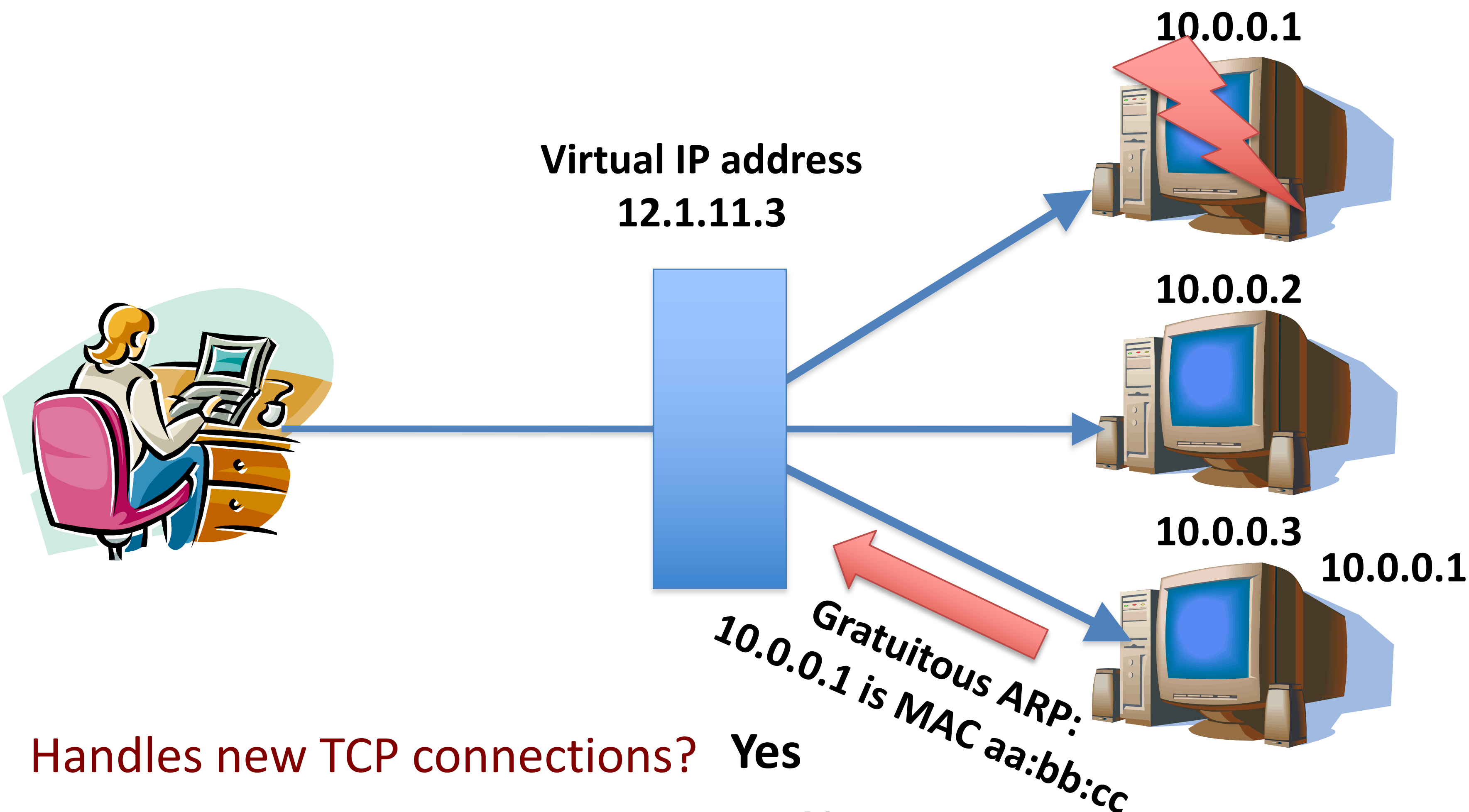
- Apply load balancing policies

Supports Layer-2 failover!



- Handles new TCP connections?
- Handles existing TCP connections?

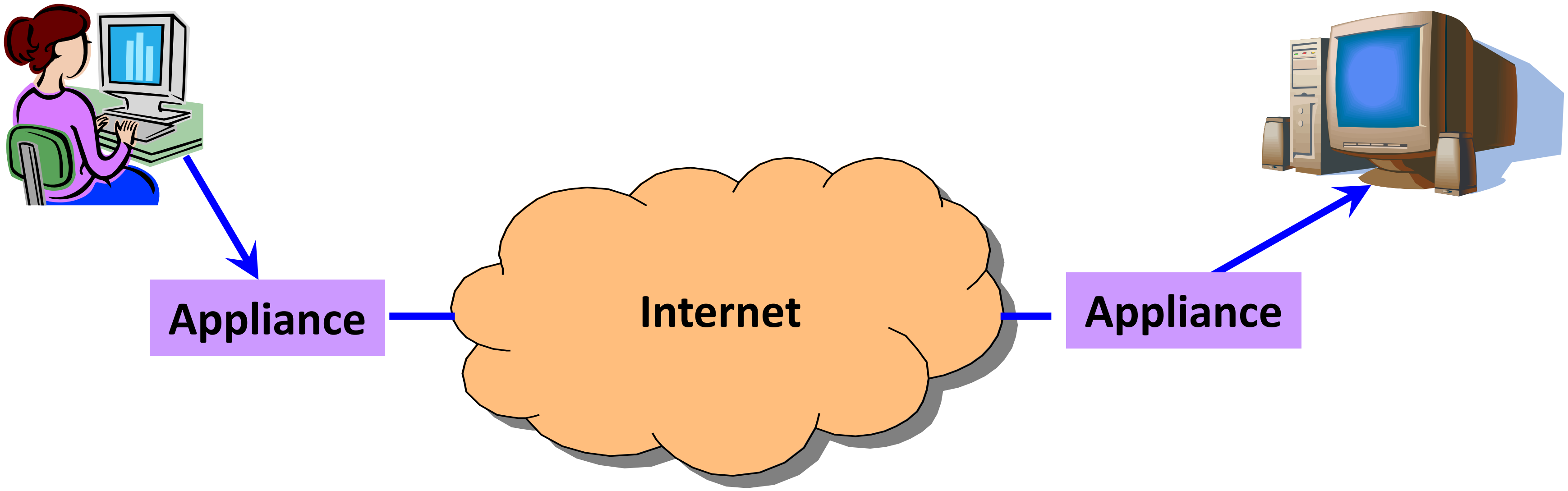
Supports Layer-2 failover!



- Handles new TCP connections? **Yes**
- Handles existing TCP connections? **No**

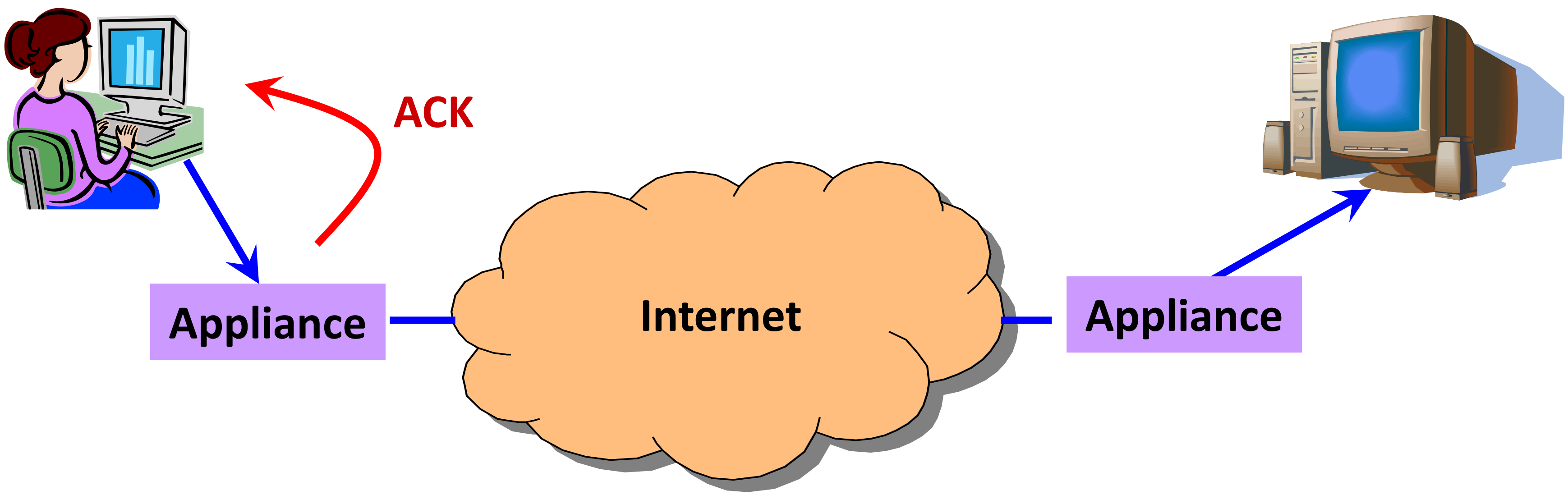
Wide-Area Accelerators

At Connection Point to the Internet



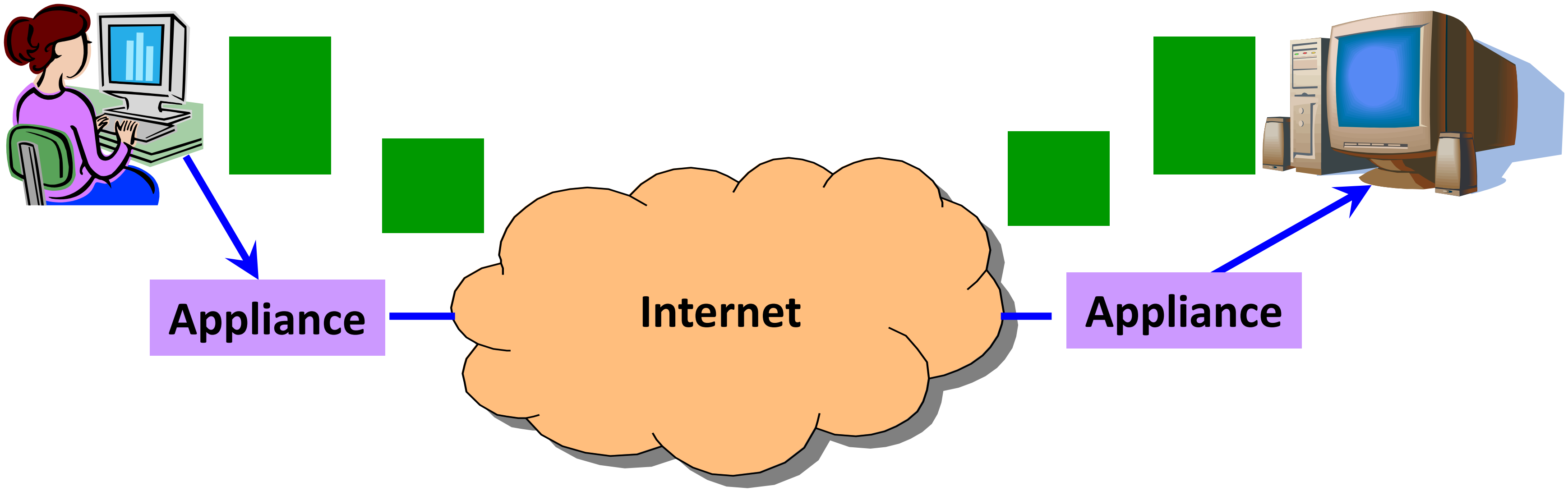
- Improve end-to-end performance
  – Through buffering, compression, caching, …
- Incrementally deployable
  – No changes to end hosts or the rest of the Internet

Example: Improve TCP Throughput



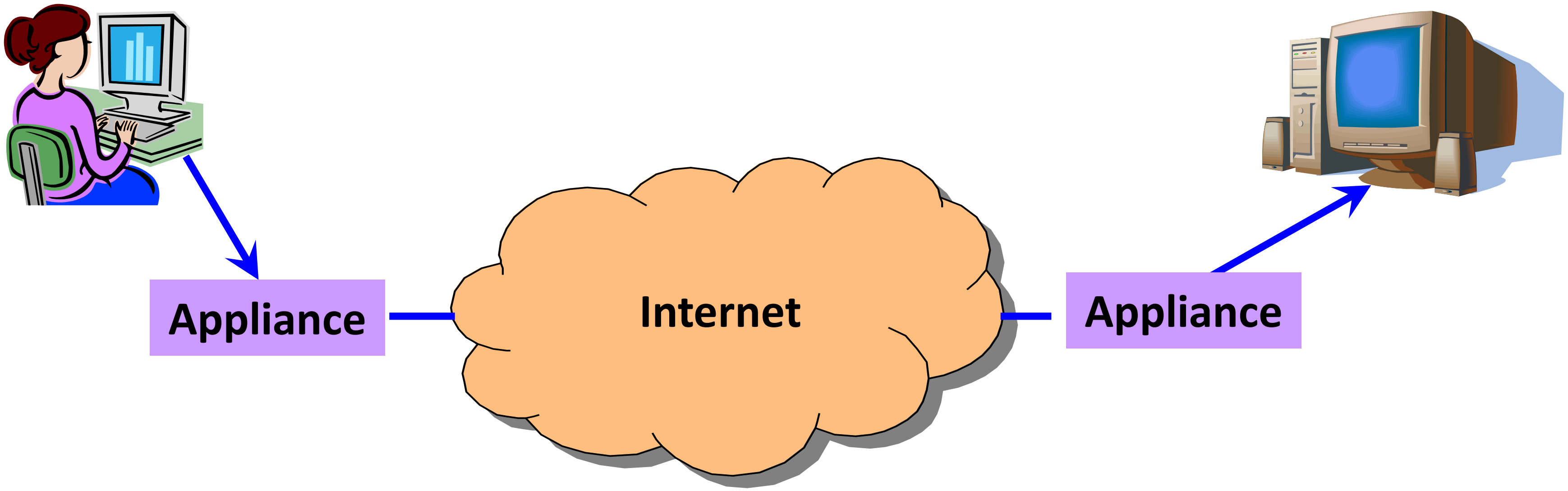
- Appliance with a lot of local memory
- Sends ACK packets quickly to the sender
- Overwrites receive window with a large value
- Or, even run a new and improved version of TCP

Example: Compression



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

Example: Caching

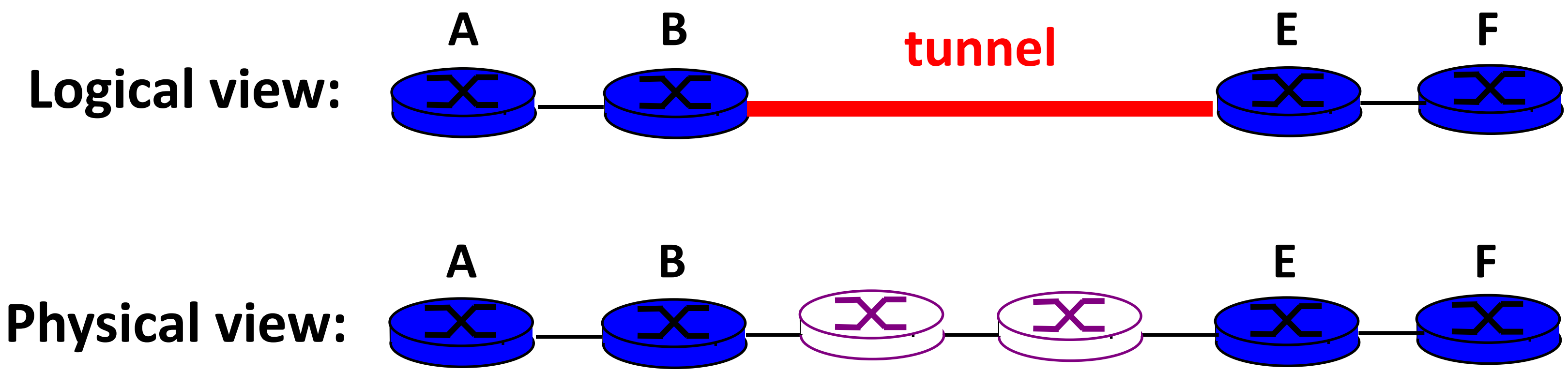


- Cache copies of the outgoing packets
- Check for sequences of bytes that match past data
- Just send a pointer to the past data
- And have the receiving appliance reconstruct

Tunneling

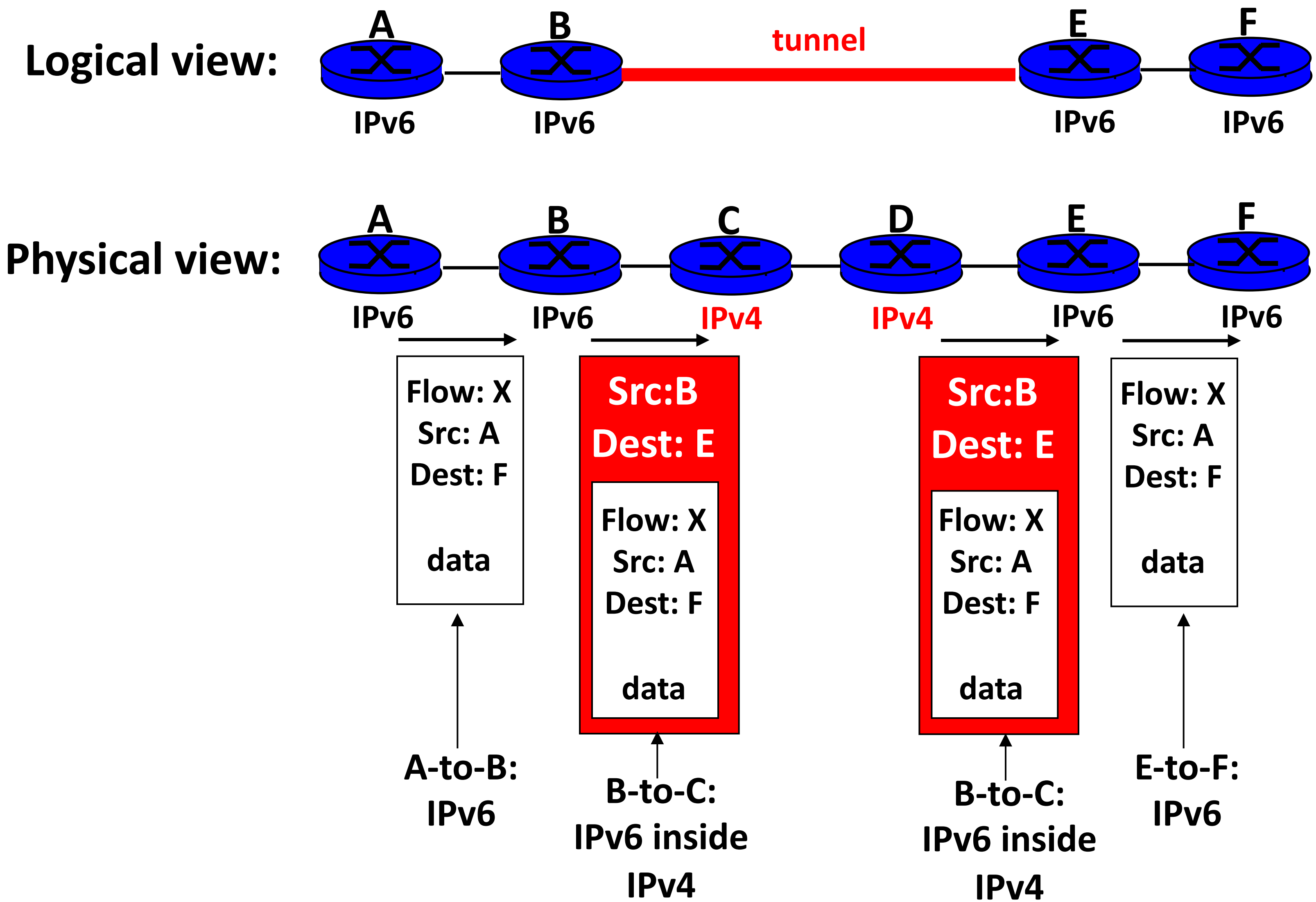
IP Tunneling

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

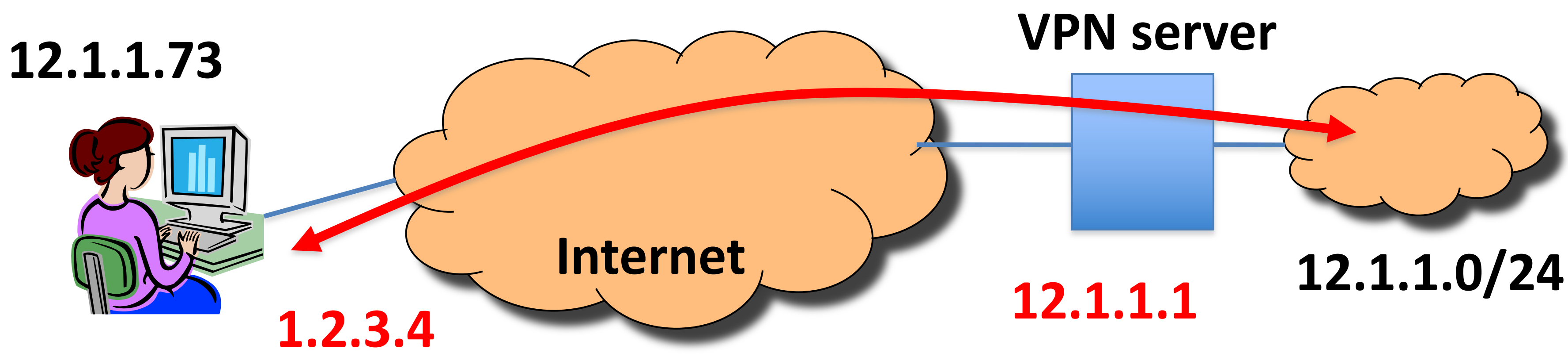


- Encapsulation of the packet inside IP datagram
  - Node B sends a packet to node E
  - … containing another packet as the payload

6Bone: Deploying IPv6 over IP4

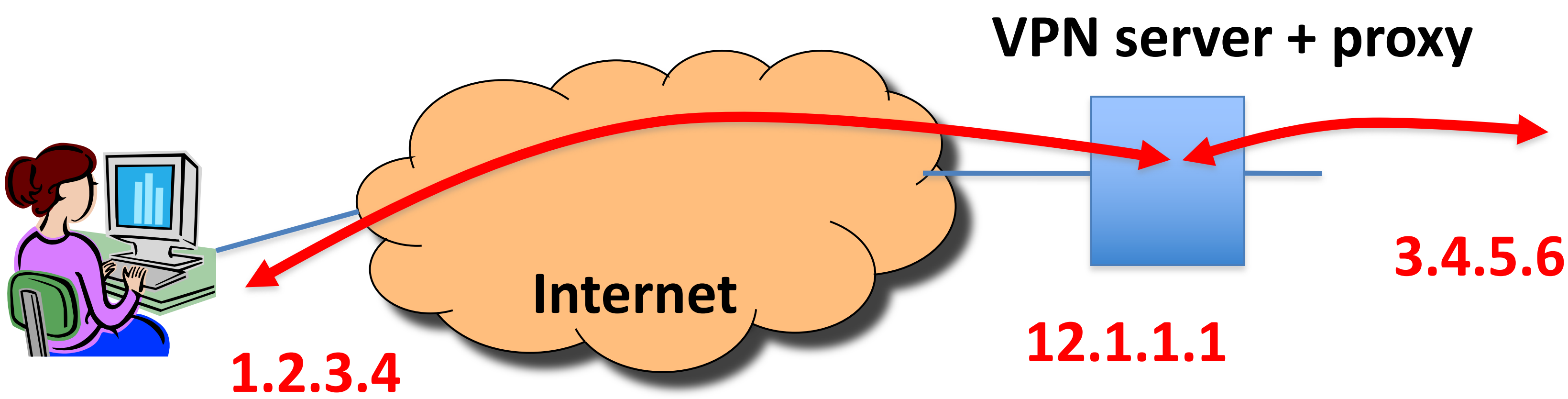


Remote Access Virtual Private Network



- Tunnel from user machine to VPN server
  - A “link” across the Internet to the local network
- Encapsulates packets to/from the user
  - Packet from 12.1.1.73 to 12.1.1.100
  - Inside a packet from 1.2.3.4 to 12.1.1.1
  - Interior packet can be point-to-point encrypted

"Commercial" VPNs



- Tunnel from user machine to VPN server
- VPN server NATs or TCP proxies traffic to origin sites
  - Traffic between client and VPN encrypted
  - VPN "anonymizes" the IP of client to rest of Internet, and can circumvent censorship on client-side
  - Client must fully trust VPN provider

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
  - Cannot assume network simply delivers packets