Network Control, Con't

CS 161 - Computer Security

Profs. Vern Paxson & David Wagner

TAs: John Bethencourt, Erika Chin, Matthew Finifter, Cynthia Sturton, Joel Weinberger

http://inst.eecs.berkeley.edu/~cs161/

Feb 19, 2010

Focus of Today's Lecture

- Finish discussion of packet-filter firewalls
- The general notion of *reference monitors*
- Firewall limitations
- Virtual Private Networks (VPNs)
- Application proxies
- Network Address Translation (NAT)

Problem: Outbound Connections Fail

```
1.allow tcp *:* -> 1.2.3.4:25
2.allow tcp 1.2.3.0/24:* -> *:*
3.drop   *  *:* -> *:*
```

- Inside host opens TCP connection to port 80 on external machine:
  - Initial SYN packet passed through by rule 2
  - SYN+ACK packet coming back is dropped
    - Fails rule 1 (not destined for port 25)
    - Fails rule 2 (source not inside host)
    - Matches rule 3 ⇒ DROP
- Fix?
  - In general, we need to distinguish between 2 kinds of inbound pkts
    - Allow inbound packets associated with an outbound connection
    - Restrict inbound packets associated with an inbound connection
  - How do we tell them apart?
    - Approach #1: remember previous outbound connections (takes **state**)
    - Approach #2: leverage details of how TCP works

Inbound vs. Outbound Connections

- Key TCP feature: ACK bit set on all packets except first
  - Plus: TCP receiver disregards packets with ACK set if they don't belong to an existing connection
- Solution ruleset:

```
1.allow tcp *:* -> 1.2.3.4:25
2.allow tcp 1.2.3.0/24:* -> *:*
3.allow tcp *:* -> 1.2.3.0/24:* only if ACK bit set
4.drop   *  *:* -> *:*
```

  - Rules 1 and 2 allow traffic in either direction for inbound connections to port 25 on machine `1.2.3.4`
  - Rules 2 and 3 allow outbound connections to any port

How This Ruleset Protects

```
1.allow tcp *:* -> 1.2.3.4:25
2.allow tcp 1.2.3.0/24:* -> *:*
3.allow tcp *:* -> 1.2.3.0/24:* only if ACK bit set
4.drop   *   *:* -> *:*
```

- Suppose external attacker tries to exploit vulnerability in SMB (TCP port 445):
  - = Attempts to open an inbound TCP connection to internal SMB server
- Attempt #1: Sends SYN packet to server
  - Packet lacks ACK bit ⇒ no match to Rules 1-3, dropped by Rule 4
- Attempt #2: Sends SYN+ACK packet to server
  - Firewall permits the packet due to Rule 3
  - But then dropped by server's TCP stack (since ACK bit set, but isn't part of existing connection)

Security Principle: *Reference Monitors*

- Firewalls embody useful principles that are applicable elsewhere in computer security
  - Optimized for enforcing particular kind of *access control policy*
  - Chokepoint notion makes enforcement possible
- A key conceptual approach to access control: *reference monitor*
  - Examines every request to access a controlled resource (an *object*) and determines whether to allow request

Reference Monitor Security Properties

- *Always invoked*
  - *Complete mediation* property: all security-relevant operations must be mediated by RM
  - RM should be invoked on every operation controlled by access control policy
- *Tamper-resistant*
  - Maintain RM integrity (no code/state tampering)
- *Verifiable*
  - Can verify RM operation (correctly enforces desired access control policy)
    - Requires extremely **simple** RM
    - Can't verify correctness for systems with any appreciable degree of complexity

Considering Firewalls as Reference Monitors

- Always invoked?
  - Place Packet Filter on chokepoint link for all internal-external communications
  - Packets only forwarded across link if firewall explicitly decides to do so after inspection

Potential Problems?

- What if a user hooks up an unsecured wireless access point to their internal machine?
- Anyone who drives by with wireless-enabled laptop can gain access to internal network
  - Bypasses packet filter!
- To use a firewall safely, must ensure we've covered all links between internal and external networks with firewalls
  - Set of links known as the *security perimeter*

RM Property: *Tamper-Resistant*

- Do not allow management access to firewall other than from specific hosts
  - I.e., firewall itself needs firewalling
- Must secure storage & propagation of configuration data
- Must also protect firewall's physical security

RM Property: *Verifiable*

- Current practice:
  - Packet filter software too complex for feasible systematic verification …
- Result:
  - *Bugs* that allowed attackers to defeat intended security policy by sending unexpected packets that packet filter doesn't handle quite the way it should

Subverting Firewalls

- In addition, packet filters have a fundamentally limited semantic model
  - They lack a full understanding of the meaning of the traffic they carry
    - In part because operate only at layers 3 & 4; not 7
- One method of subversion: **abuse ports**
  - Who says that e.g. port 22/tcp = SSH?
    - Why couldn't it be say Skype or BitTorrent?
    - Just requires that client & server agree on app proto

Hiding on Other Ports

- Method #1: use port allocated to another service (how can this be detected?)
- Method #2: **tunneling**
  - Encapsulate one protocol inside another
  - Receiver of "outer" protocol *decapsulates* interior tunneled protocol to recover it
  - Pretty much any protocol can be tunneled over another (with enough effort)
- E.g., tunneling IP over SMTP
  - Just need a way to code an IP datagram as an email message (either mail body or just headers)

Example: Tunneling IP over Email

From: doesnt-matter@bogus.com
To: my-buddy@tunnel-decapsulators.R.us
Subject: Here's my IP datagram

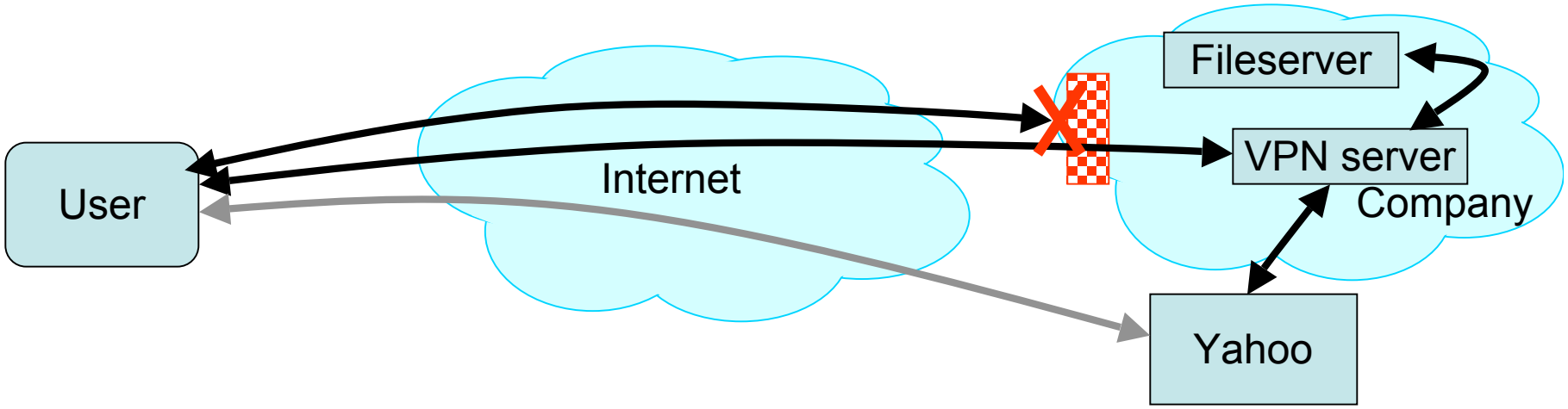
IP-header-version: 4
IP-header-len: 5
IP-ID: 11234
IP-src: 1.2.3.4
IP-dst: 5.6.7.8
IP-payload: 0xa144bf2c0102…

Program receives this legal email and **builds** an IP packet corresponding to description in email body …
… **injects** it into the network
How can a firewall detect this??

Tunneling, con't

- E.g., IP-over-ICMP:
  - Encode an IP datagram as the payload of a "ping" packet
- E.g., Skype-over-HTTP:
  - Encode Skype message in URL of requests or header fields (or cookies) of replies
- Note #1: to tunnel, the sender and receiver must both cooperate
- Note #2: tunneling has many legitimate uses too
  - E.g., *overlay* networks that forward packets along paths different from what direct routing would pick
  - E.g., Virtual Private Networks (VPNs)
    - Make a remote machine look like it's local to its home network
    - Tunnel encrypts traffic for privacy & to prevent meddling

Secure External Access to Inside Machines



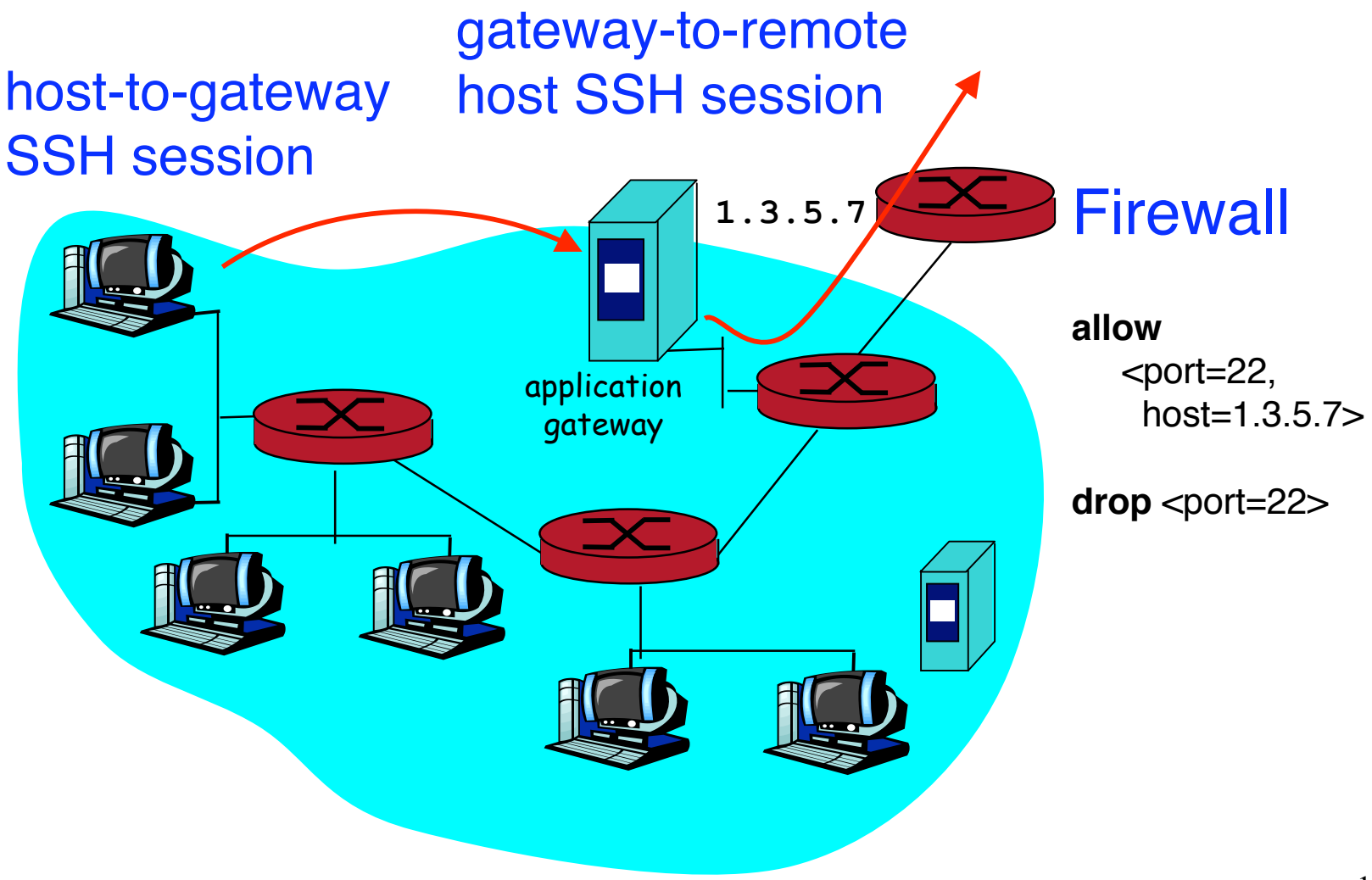
- Often need to provide secure remote access to a network protected by a firewall
  - Remote access, telecommuting, branch offices, …
- Create secure channel (*Virtual Private Network*, or VPN) to tunnel traffic from outside host/network to inside network
  - Provides Authentication, Confidentiality, Integrity
  - However, also raises perimeter issues

    (Try it yourself at http://www.net.berkeley.edu/vpn/)

Application Proxies

- Can more directly control applications by requiring them to go through a proxy for external access
  - Proxy doesn't just forward, but acts as an application-level middleman
- Example: SSH gateway
  - Require all SSH in/out of site to go through gateway
  - Gateway logs authentication, inspects decrypted text
  - Site's firewall configured to prohibit any other SSH access

SSH Gateway Example



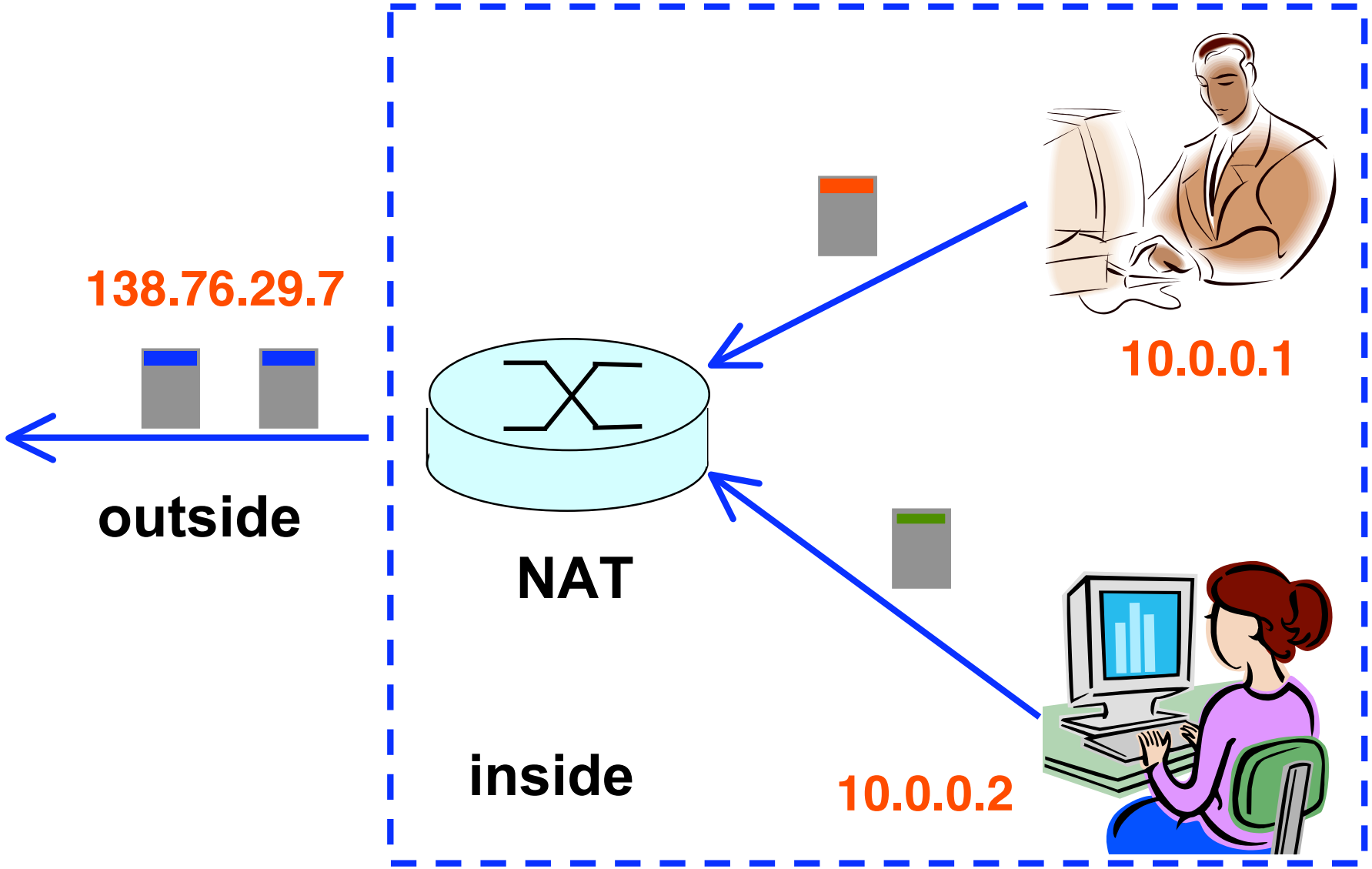
Application Proxies

- Can more directly control applications by requiring them to go through a proxy for external access
  - Proxy doesn't just forward, but acts as an application-level middleman
- Example: SSH gateway
  - Require all SSH in/out of site to go through gateway
  - Gateway logs authentication, inspects decrypted text
  - Site's firewall configured to prohibit any other SSH access
- Provides a powerful degree of monitoring/control, but costs significant resources
  - Need to run extra server(s) per app
  - Each server requires careful hardening

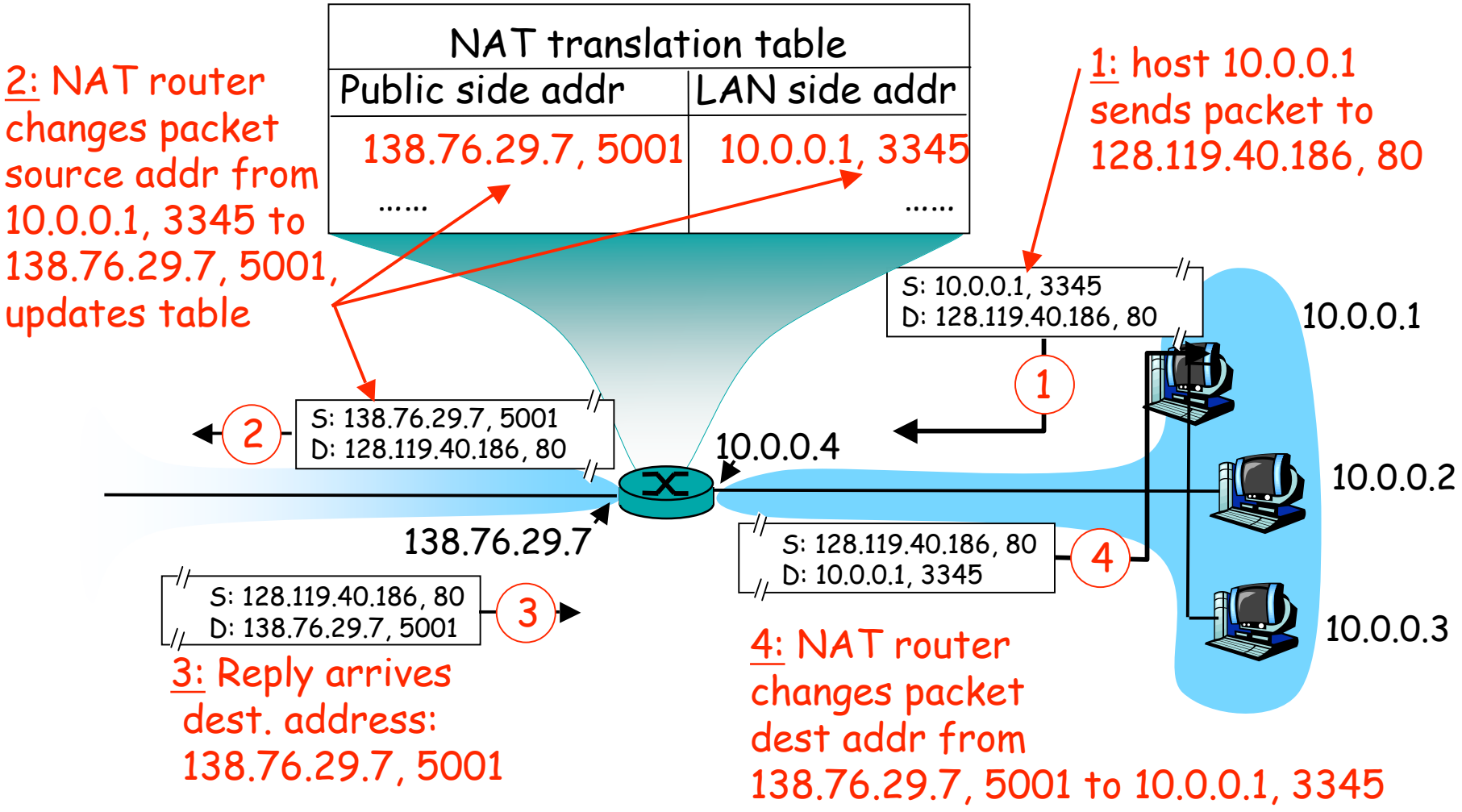
Network Control: NATs

- To conserve global Internet addresses, network operators often give their systems private addresses
  - Usually numbered out of 10.0.0.0/8 or 192.168.0.0/16
  - These addresses will not work for reaching the hosts from external Internet locations
    - Internet routers lack paths for them
- Hosts communicate externally by having their traffic go through a Network Address Translator (NAT)
  - Active, in-path network element
- The NAT translates (maps) private addresses to a public address
  - Also maps TCP/UDP ports

Multiple Private Addresses Using One Public Address via NAT



NAT Translation Table



Security Implications of NAT

- If an external packet arrives for which the NAT doesn't have a mapping in its table, it (necessarily) discards it
  - Thus, as a *side effect* a NAT prevents probing of services offered by internal systems
  - (Unless operator explicitly sets up an exception)
- NATs change IP headers (addresses) and transport headers (ports)
  - Thus, any mechanism we might use to ensure layer 3/4 packet **integrity** will complain that packet has been meddled with
  - (⇒ operator *convenience* from using NAT is at odds with providing basic security guarantees)