

Lab (1)

Firewall and NAT

Instructor: Prof. Phone Lin

Teacher Assistant: Chia-Peng Lee

Mobile Communications Networking LAB, CSIE, NTU

2019/02/27



Outline

❏ The Internet network layer

❏ IP datagram format

❏ IP addressing

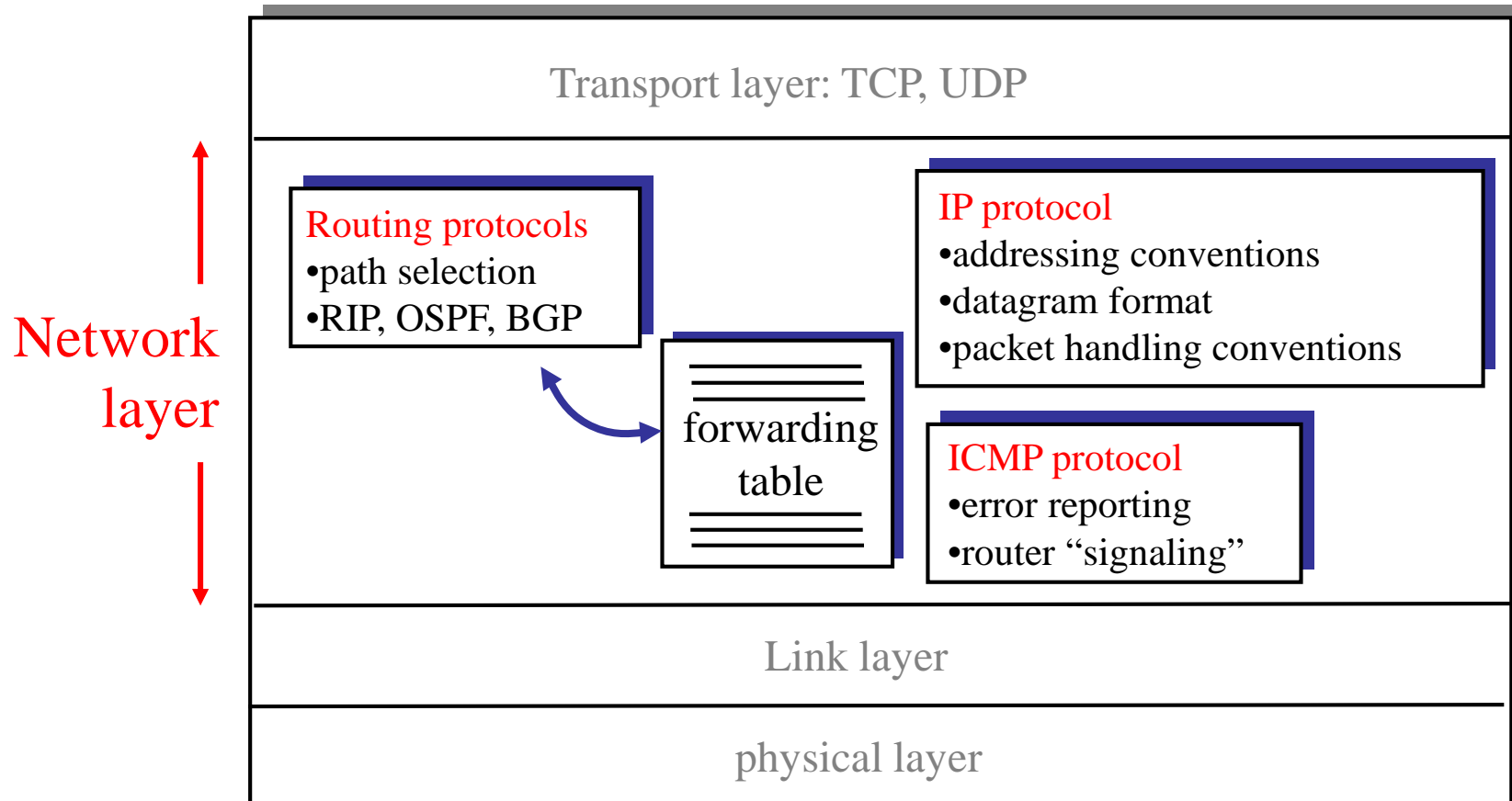
- DHCP
- NAT

❏ Network security

- Firewall

The Internet Network Layer

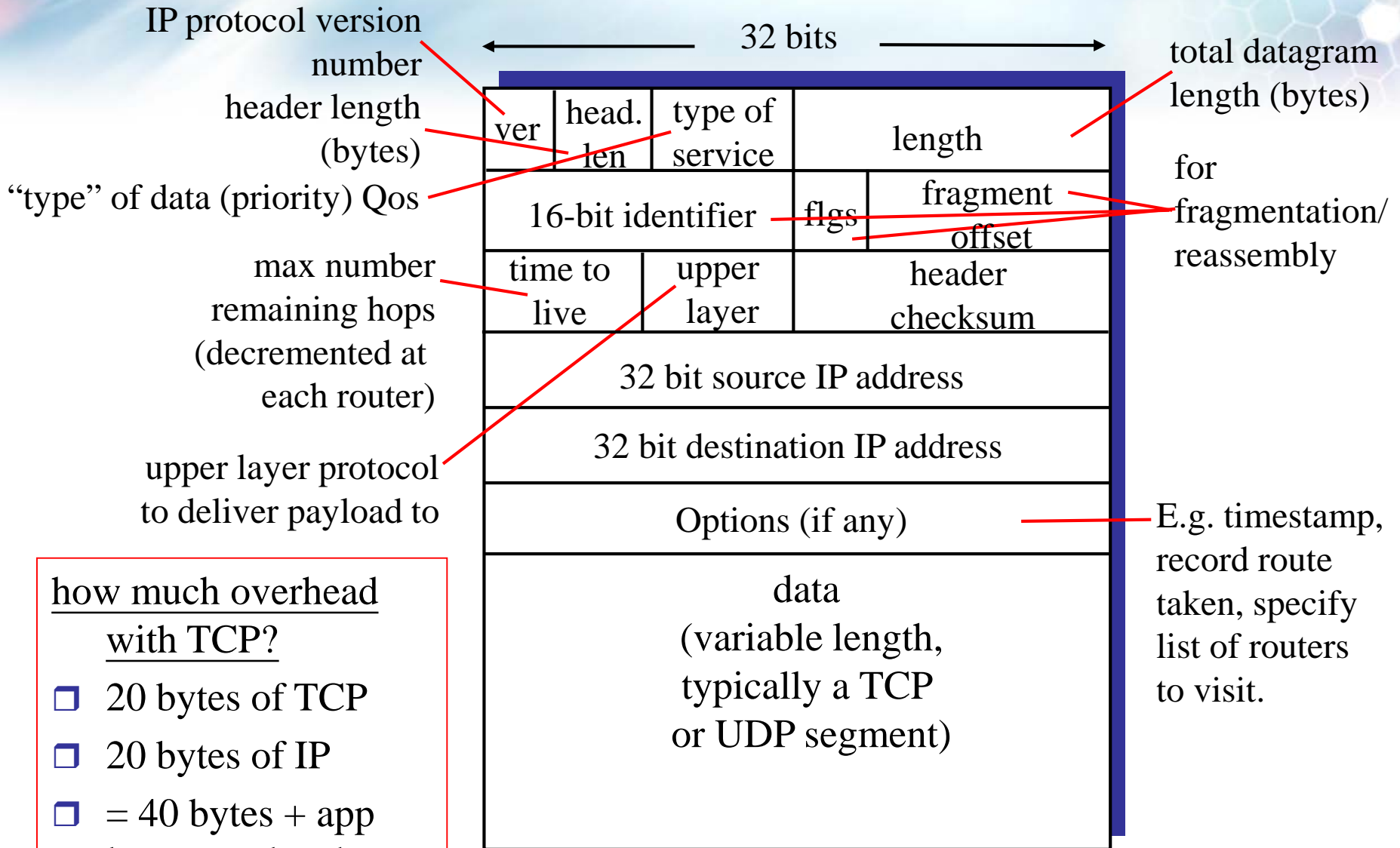
Host, router network layer functions:



Outline

- ❏ The Internet network layer
- ❏ IP datagram format
- ❏ IP addressing
 - DHCP
 - NAT
- ❏ Network security
 - Firewall

IP datagram format



how much overhead with TCP?

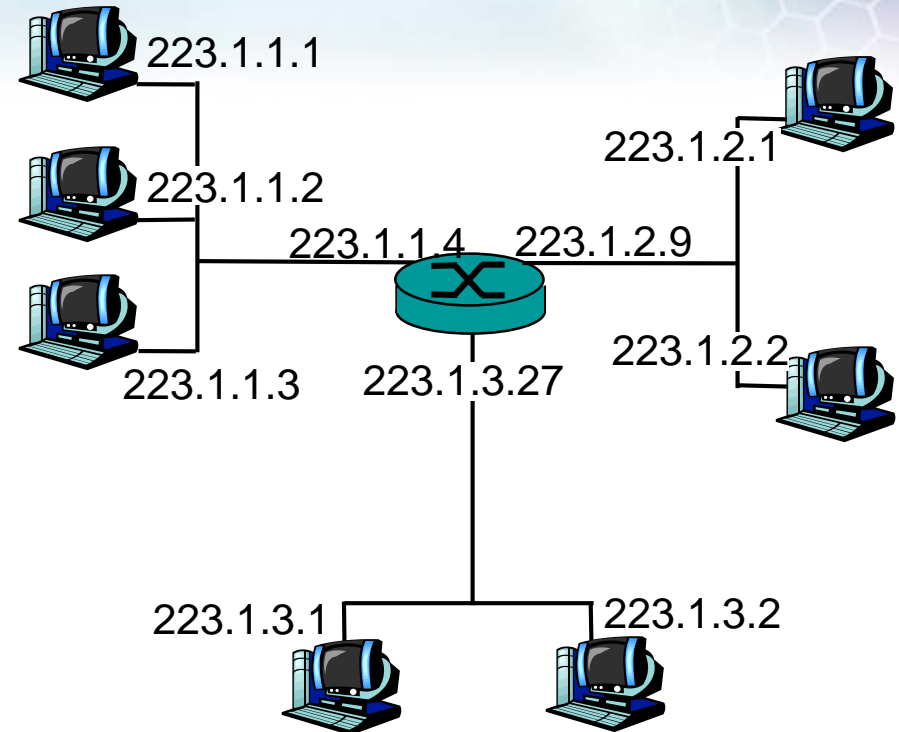
- ❑ 20 bytes of TCP
- ❑ 20 bytes of IP
- ❑ = 40 bytes + app layer overhead

Outline

- ❏ The Internet network layer
- ❏ IP datagram format
- ❏ IP addressing
 - DHCP
 - NAT
- ❏ Network security
 - Firewall

IP Addressing: introduction

- ❏ **IP address:** 32-bit identifier for host, router *interface*
- ❏ **interface:** connection between host/router and physical link
 - router's typically have multiple interfaces
 - host typically has one interface
 - IP addresses associated with each interface



$$223.1.1.1 = \underbrace{11011111}_{223} \underbrace{00000001}_1 \underbrace{00000001}_1 \underbrace{00000001}_1$$

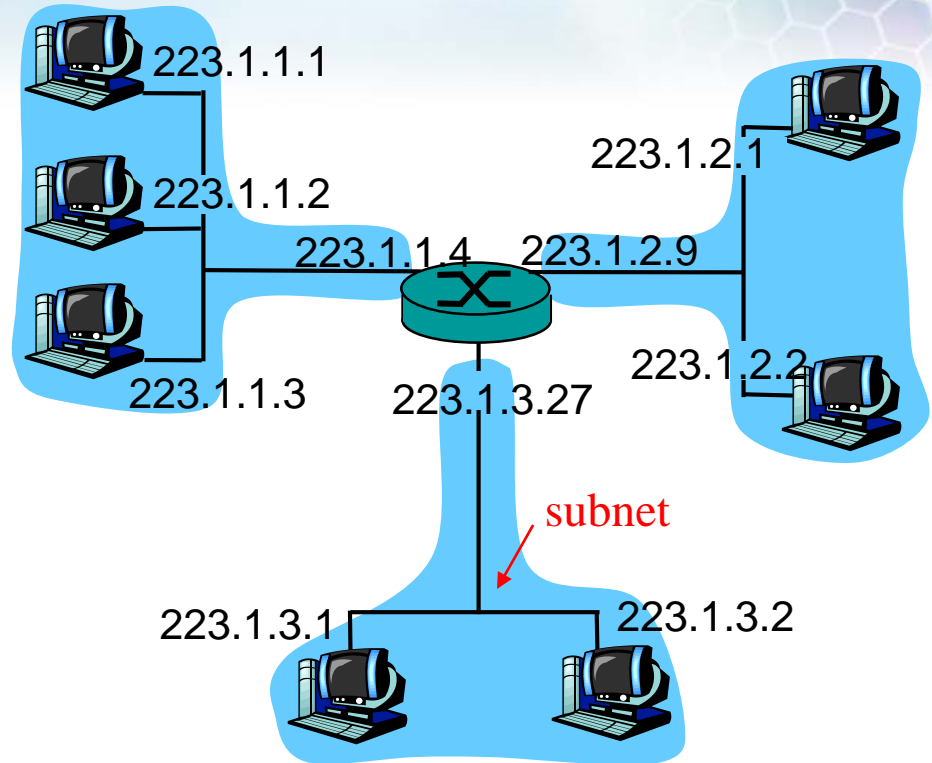
Subnets

❏ IP address:

- subnet part (high order bits)
- host part (low order bits)

❏ *What's a subnet ?*

- device interfaces with same subnet part of IP address
- can physically reach each other without intervening router

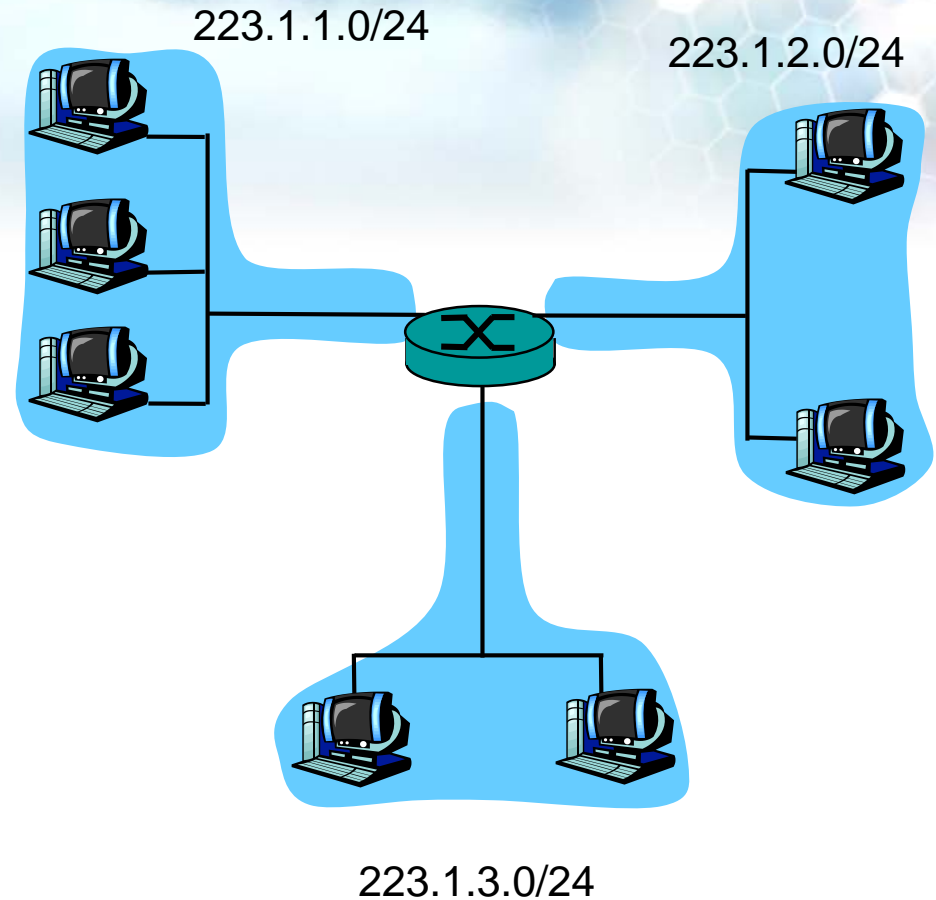


network consisting of 3 subnets

Subnets

Recipe

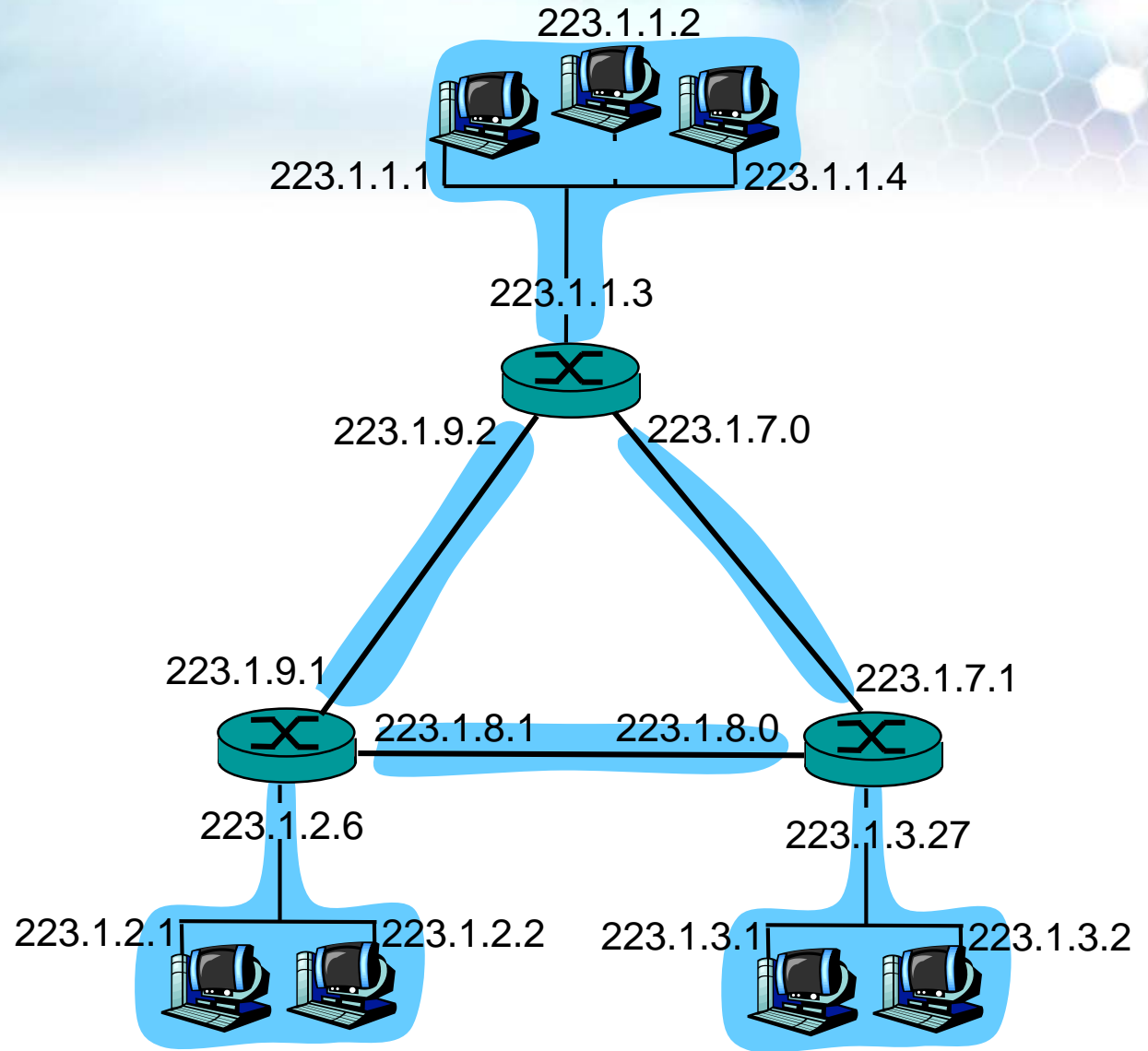
- ❏ To determine the subnets, detach each interface from its host or router, creating islands of isolated networks. Each isolated network is called a **subnet**.



Subnet mask: /24

Subnets

How many?



IP addressing: CIDR

CIDR: Classless InterDomain Routing

- subnet portion of address of arbitrary length
- address format: **a.b.c.d/x**, where x is # bits in subnet portion of address



200.23.16.0/23

Outline

- ❏ The Internet network layer
 - ❏ IP datagram format
 - ❏ IP addressing
 - DHCP
 - NAT
- ❏ Network security
 - Firewall

IP addresses: how to get one?

Q: How does a *host* get IP address?

- ❏ hard-coded by system admin in a file
 - Windows: control-panel->network->configuration->tcp/ip->properties
 - UNIX: /etc/rc.config
- ❏ **DHCP**: **D**ynamic **H**ost **C**onfiguration **P**rotocol: dynamically get address from as server
 - “plug-and-play”

DHCP: Dynamic Host Configuration Protocol

Goal: allow host to *dynamically* obtain its IP address from network server when it joins network

Can renew its lease on address in use

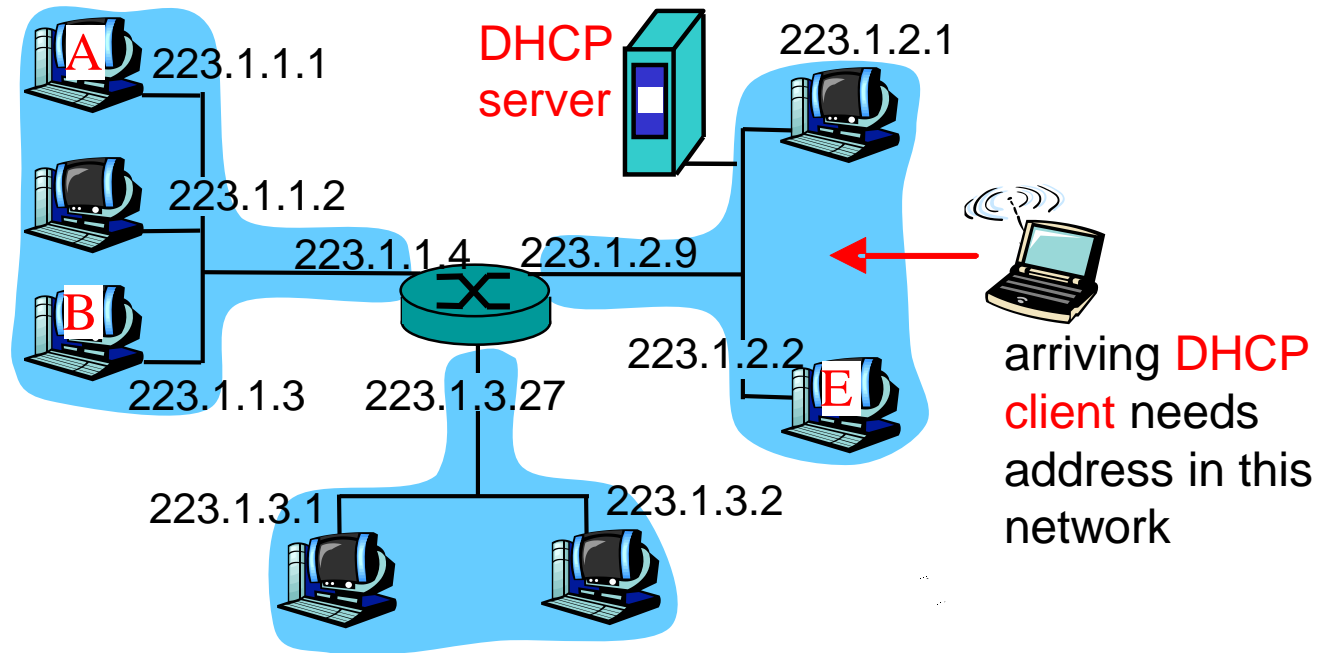
Allows reuse of addresses (only hold address while connected an “on”)

Support for mobile users who want to join network (more shortly)

DHCP overview:

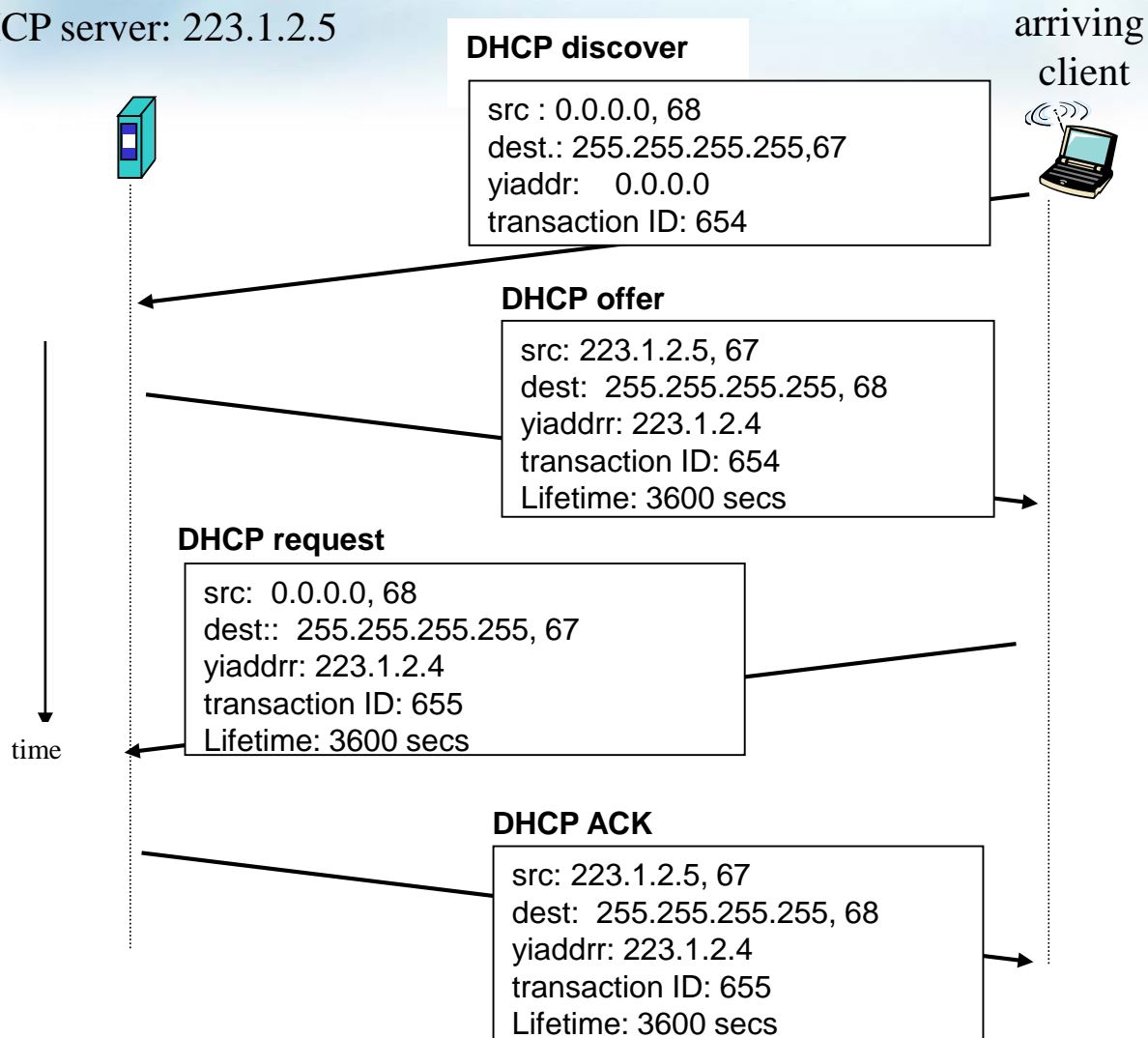
- host broadcasts “**DHCP discover**” msg [optional]
- DHCP server responds with “**DHCP offer**” msg [optional]
- host requests IP address: “**DHCP request**” msg
- DHCP server sends address: “**DHCP ack**” msg

DHCP client-server scenario



DHCP client-server scenario

DHCP server: 223.1.2.5

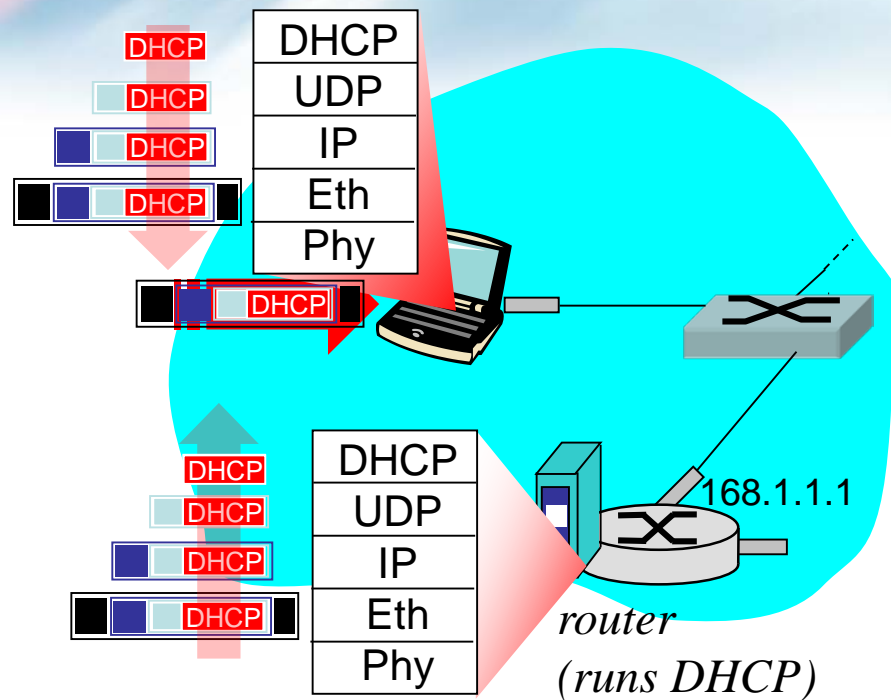


DHCP: more than IP address

DHCP can return more than just allocated IP address on subnet:

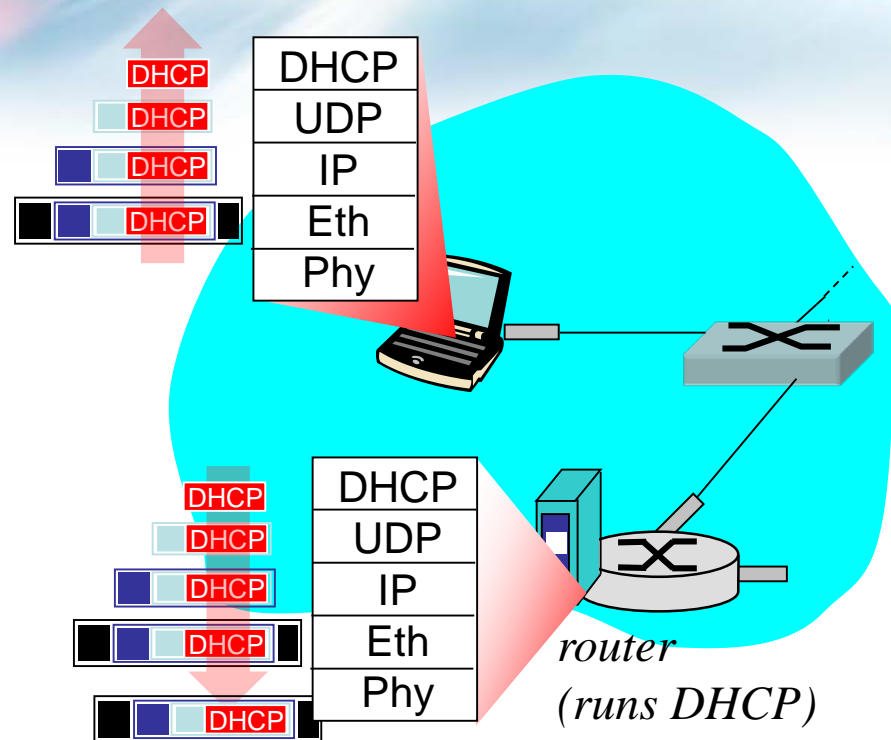
- address of first-hop router for client
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)

DHCP: example



- connecting laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP
- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet
- Ethernet frame broadcast (dest: FFFFFFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demux'ed to IP demux'ed, UDP demux'ed to DHCP

DHCP: example



- ❑ DCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- ❑ encapsulation of DHCP server, frame forwarded to client, demux'ing up to DHCP at client
- ❑ client now knows its IP address, name and IP address of DSN server, IP address of its first-hop router

DHCP: wireshark output (home LAN)

Message type: **Boot Request (1)**

Hardware type: Ethernet

Hardware address length: 6

Hops: 0

Transaction ID: 0x6b3a11b7

Seconds elapsed: 0

Bootp flags: 0x0000 (Unicast)

Client IP address: 0.0.0.0 (0.0.0.0)

Your (client) IP address: 0.0.0.0 (0.0.0.0)

Next server IP address: 0.0.0.0 (0.0.0.0)

Relay agent IP address: 0.0.0.0 (0.0.0.0)

Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)

Server host name not given

Boot file name not given

Magic cookie: (OK)

Option: (t=53,l=1) **DHCP Message Type = DHCP Request**

Option: (61) Client identifier

Length: 7; Value: 010016D323688A;

Hardware type: Ethernet

Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)

Option: (t=50,l=4) Requested IP Address = 192.168.1.101

Option: (t=12,l=5) Host Name = "nomad"

Option: (55) Parameter Request List

Length: 11; Value: 010F03062C2E2F1F21F92B

1 = Subnet Mask; 15 = Domain Name

3 = Router; 6 = Domain Name Server

44 = NetBIOS over TCP/IP Name Server

request

Message type: **Boot Reply (2)**

Hardware type: Ethernet

Hardware address length: 6

Hops: 0

Transaction ID: 0x6b3a11b7

Seconds elapsed: 0

Bootp flags: 0x0000 (Unicast)

Client IP address: 192.168.1.101 (192.168.1.101)

Your (client) IP address: 0.0.0.0 (0.0.0.0)

Next server IP address: 192.168.1.1 (192.168.1.1)

Relay agent IP address: 0.0.0.0 (0.0.0.0)

Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)

Server host name not given

Boot file name not given

Magic cookie: (OK)

Option: (t=53,l=1) DHCP Message Type = DHCP ACK

Option: (t=54,l=4) Server Identifier = 192.168.1.1

Option: (t=1,l=4) Subnet Mask = 255.255.255.0

Option: (t=3,l=4) Router = 192.168.1.1

Option: (6) Domain Name Server

Length: 12; Value: 445747E2445749F244574092;

IP Address: 68.87.71.226;

IP Address: 68.87.73.242;

IP Address: 68.87.64.146

Option: (t=15,l=20) Domain Name = "hsd1.ma.comcast.net."

reply

IP addresses: how to get one?

Q: How does *network* get subnet part of IP addr?

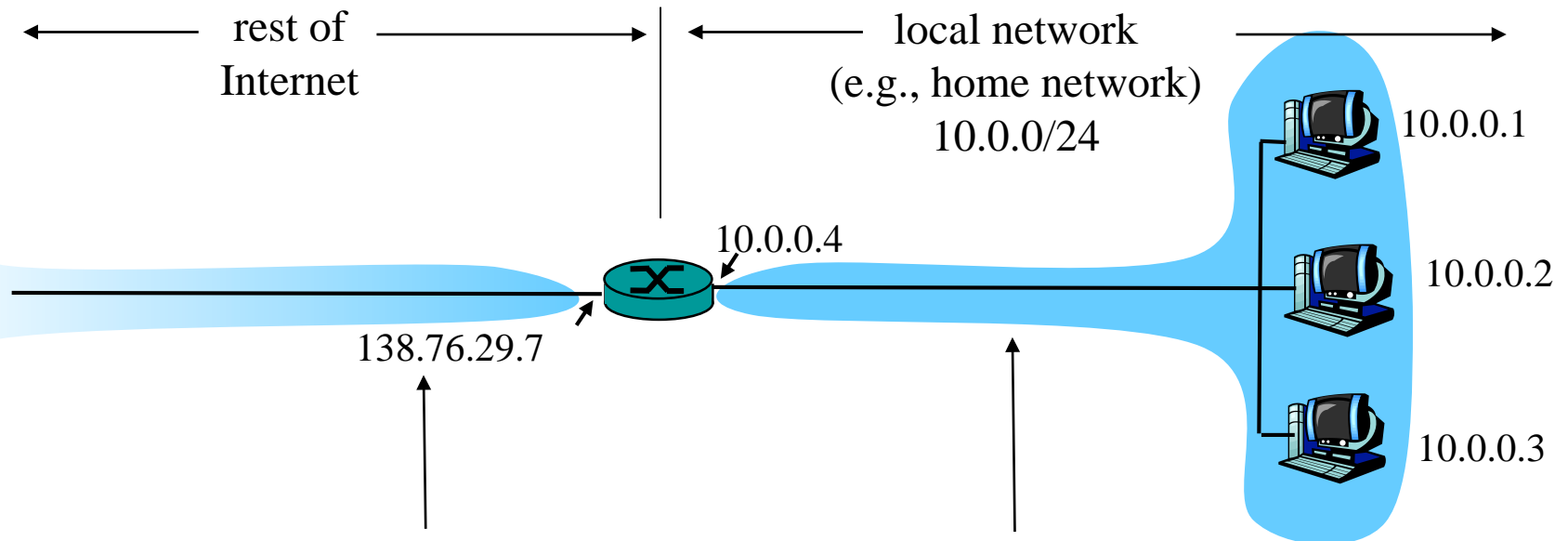
A: gets allocated portion of its provider ISP's address space

ISP's block	<u>11001000 00010111 00010000</u> 00000000	200.23.16.0/20
Organization 0	<u>11001000 00010111 00010000</u> 00000000	200.23.16.0/23
Organization 1	<u>11001000 00010111 00010010</u> 00000000	200.23.18.0/23
Organization 2	<u>11001000 00010111 00010100</u> 00000000	200.23.20.0/23
...
Organization 7	<u>11001000 00010111 00011110</u> 00000000	200.23.30.0/23

Outline

- ❏ The Internet network layer
 - ❏ IP datagram format
 - ❏ IP addressing
 - DHCP
 - NAT
- ❏ Network security
 - Firewall

NAT: Network Address Translation



All datagrams *leaving* local network have **same** single source NAT IP address: 138.76.29.7, different source port numbers

Datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

NAT: Network Address Translation

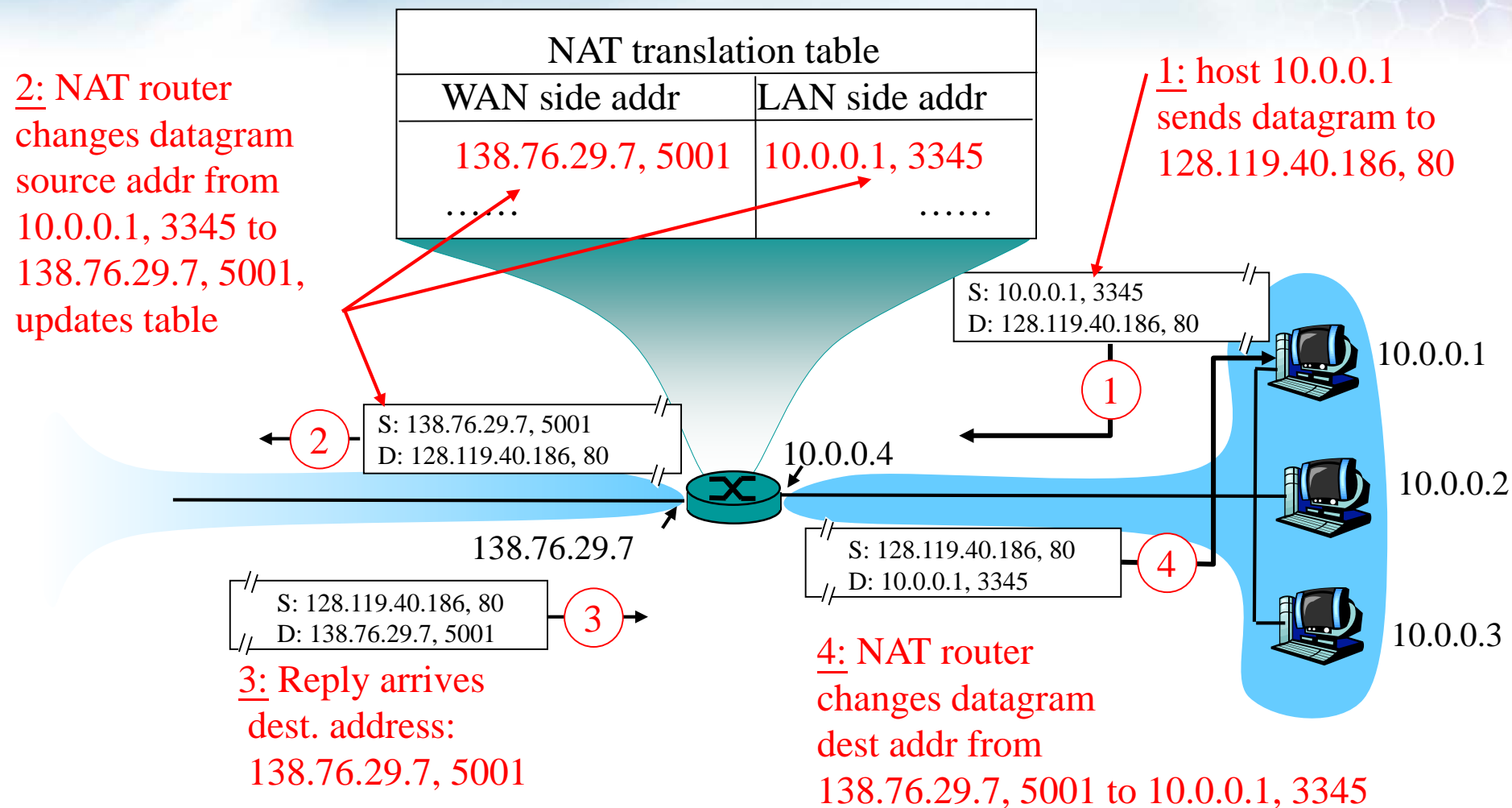
- ❏ **Motivation:** local network uses just one IP address as far as outside world is concerned:
 - range of addresses not needed from ISP: just one IP address for all devices
 - can change addresses of devices in local network without notifying outside world
 - can change ISP without changing addresses of devices in local network
 - devices inside local net not explicitly addressable, visible by outside world (a security plus).

NAT: Network Address Translation

Implementation: NAT router must:

- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
... remote clients/servers will respond using (NAT IP address, new port #) as destination addr.
- *remember (in NAT translation table)* every (source IP address, port #) to (NAT IP address, new port #) translation pair
- *incoming datagrams: replace* (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: Network Address Translation



NAT: Network Address Translation

❏ 16-bit port-number field:

- 60,000 simultaneous connections with a single LAN-side address!

❏ NAT is controversial!:

- routers should only process up to layer 3
- violates end-to-end argument
 - NAT possibility must be taken into account by app designers, eg, P2P applications
- address shortage should instead be solved by IPv6

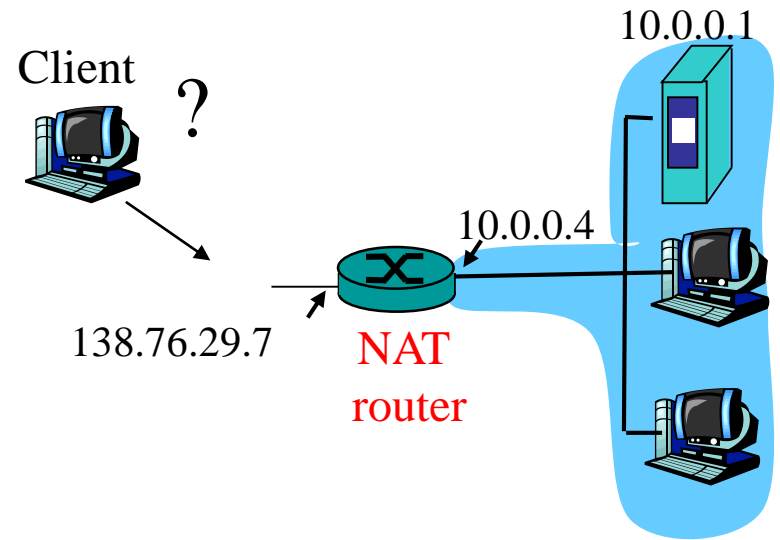
NAT traversal problem

❏ client wants to connect to server with address 10.0.0.1

- server address 10.0.0.1 local to LAN (client can't use it as destination addr)
- only one externally visible NATted address: 138.76.29.7

❏ solution 1: statically configure NAT to forward incoming connection requests at given port to server

- e.g., (138.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000

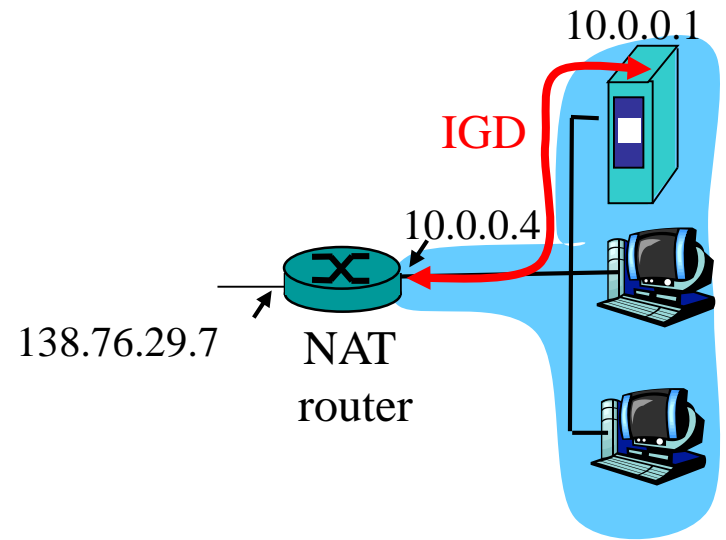


NAT traversal problem

❏ **solution 2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATted host to:**

- ❖ learn public IP address (138.76.29.7)
- ❖ add/remove port mappings (with lease times)

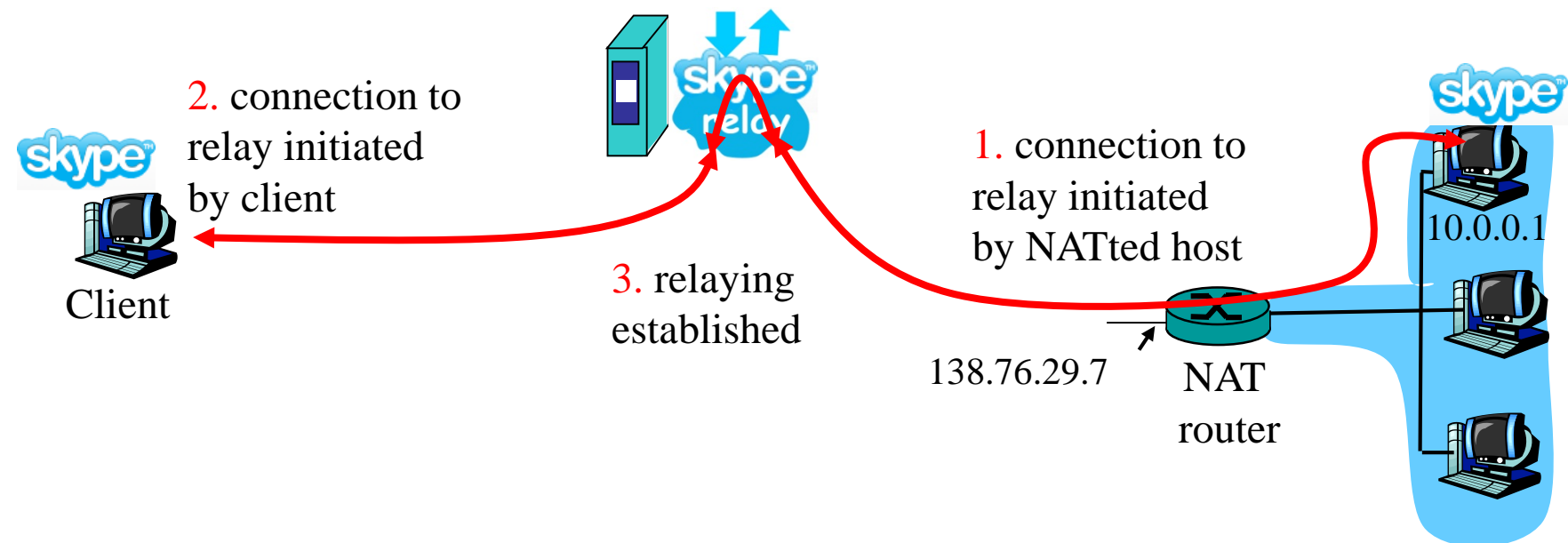
i.e., automate static NAT port map configuration



NAT traversal problem

☒ solution 3: relaying (used in Skype)

- NATed client establishes connection to relay
- External client connects to relay
- relay bridges packets between to connections



Outline

- ❏ The Internet network layer
- ❏ IP datagram format
- ❏ IP addressing
 - DHCP
 - NAT
- ❏ Network security
 - Firewall

What is network security?

Confidentiality: only sender, intended receiver should “understand” message contents

- sender encrypts message
- receiver decrypts message

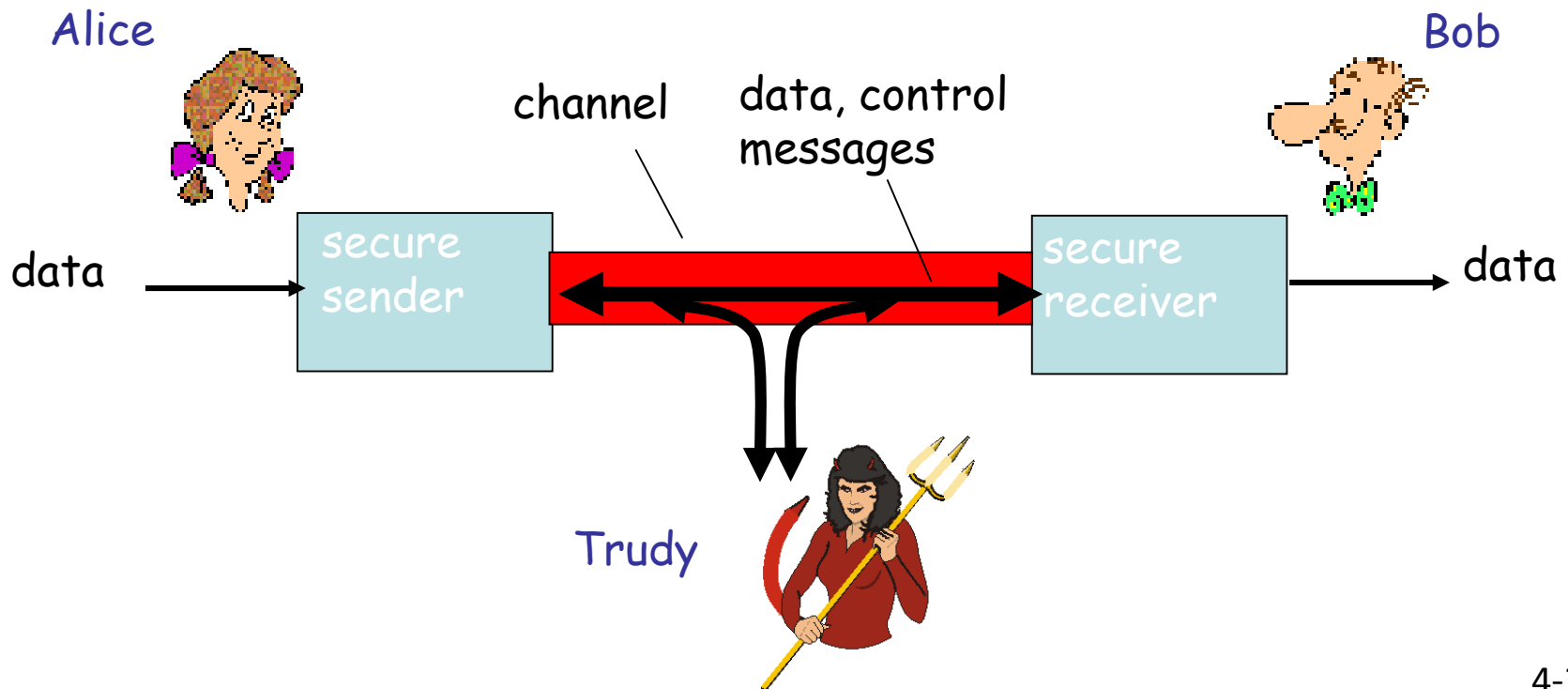
Authentication: sender, receiver want to confirm identity of each other (e.g., banking info)

Message Integrity: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection

Access and Availability: services must be accessible and available to users

Friends and enemies: Alice, Bob, Trudy

- ❏ Well-known in network security world
- ❏ **Bob**, **Alice** (lovers!) want to communicate “securely”
- ❏ **Trudy** (intruder) may **intercept**, **delete**, **add** messages



Who might Bob, Alice be?

- ❏ ... well, *real-life* Bobs and Alices!
- ❏ Web browser/server for electronic transactions (e.g., on-line purchases)
- ❏ on-line banking client/server
- ❏ DNS servers
- ❏ routers exchanging routing table updates
- ❏ other examples?

There are **bad** guys (and girls) out there!

Q: What can a “bad guy” do?

A: a lot!

- *eavesdrop*: intercept messages
- actively *insert* messages into connection
- *impersonation*: can fake (*spoof*) source address in packet (or any field in packet)
- *hijacking*: “take over” ongoing connection by removing sender or receiver, inserting himself in place
- *denial of service*: prevent service from being used by others (e.g., by overloading resources)

more on this later

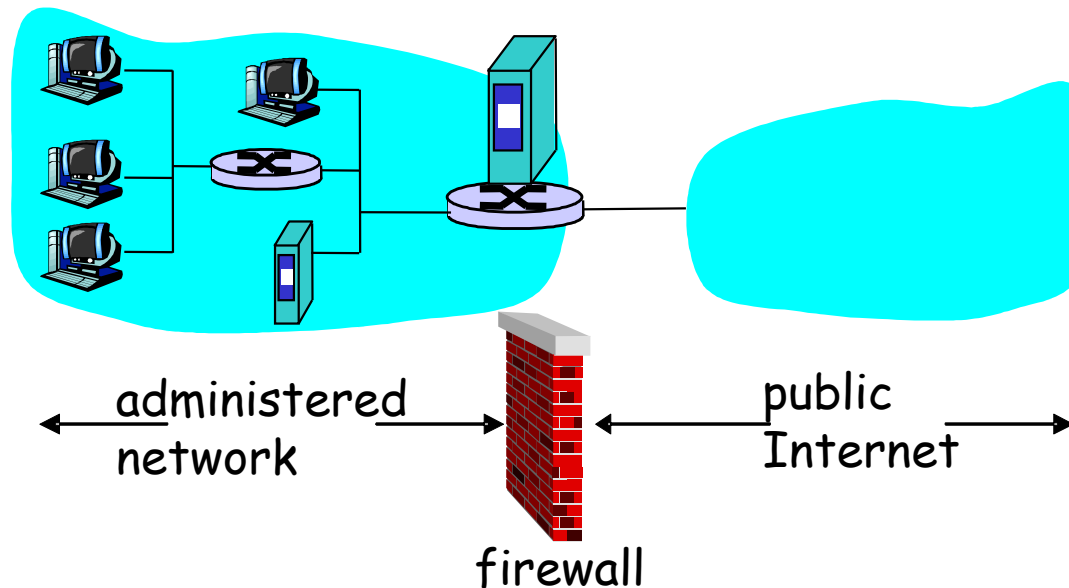
Outline

- ❏ The Internet network layer
 - ❏ IP datagram format
 - ❏ IP addressing
 - DHCP
 - NAT
- ❏ Network security
 - Firewall

Firewalls

firewall

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



Firewalls: Why

prevent denial of service attacks:

- SYN flooding: attacker establishes many bogus TCP connections, no resources left for "real" connections

prevent illegal modification/access of internal data.

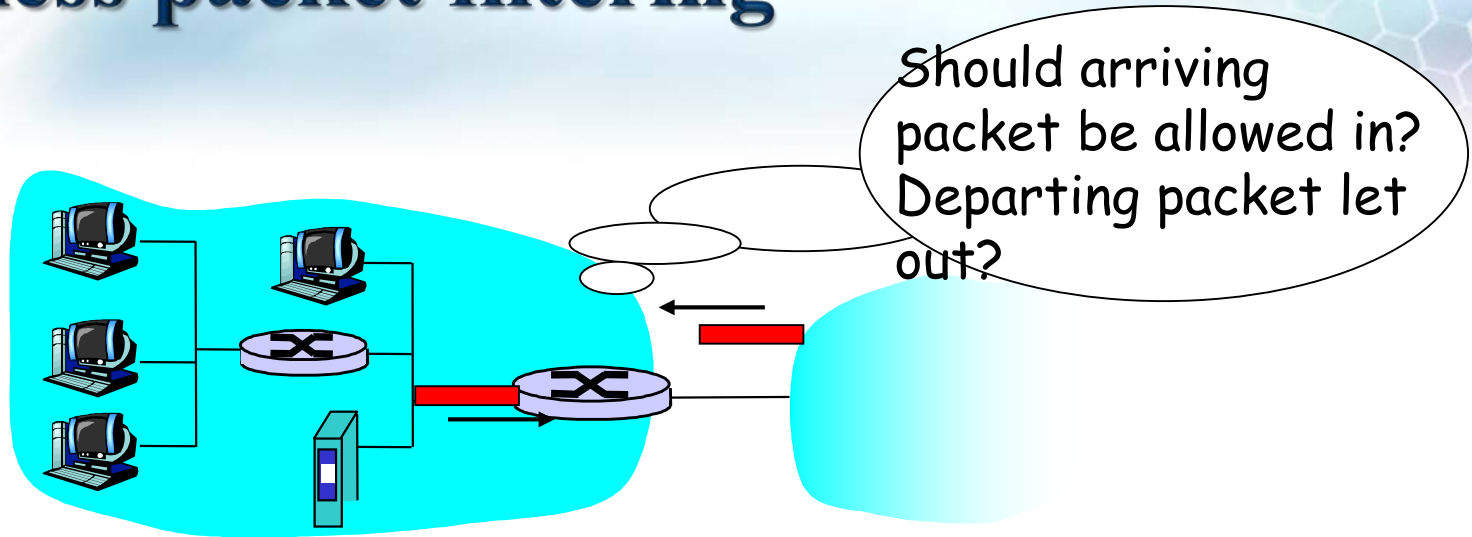
- e.g., attacker replaces CIA's homepage with something else

allow only authorized access to inside network (set of authenticated users/hosts)

three types of firewalls:

- stateless packet filters
- statefull packet filters
- application gateways

Stateless packet filtering



- ❏ internal network connected to Internet via **router firewall**
- ❏ router **filters packet-by-packet**, decision to forward/drop packet based on:
 - source IP address, destination IP address
 - TCP/UDP source and destination port numbers
 - ICMP message type
 - TCP SYN and ACK bits

Stateless packet filtering: example

❏ **example 1: block incoming and outgoing datagrams with IP protocol field = 17 and with either source or dest port = 23.**

- all incoming, outgoing UDP flows and telnet connections are blocked.

❏ **example 2: Block inbound TCP segments with ACK=0.**

- prevents external clients from making TCP connections with internal clients, but allows internal clients to connect to outside.

Stateless packet filtering: more examples

<u>Policy</u>	<u>Firewall Setting</u>
No outside Web access.	Drop all outgoing packets to any IP address, port 80
No incoming TCP connections, except those for institution's public Web server only.	Drop all incoming TCP SYN packets to any IP except 130.207.244.203, port 80
Prevent Web-radios from eating up the available bandwidth.	Drop all incoming UDP packets - except DNS and router broadcasts.
Prevent your network from being used for a smurf DoS attack.	Drop all ICMP packets going to a "broadcast" address (eg 130.207.255.255).
Prevent your network from being tracerouted	Drop all outgoing ICMP TTL expired traffic

Access Control Lists

- **ACL:** table of rules, applied top to bottom to incoming packets:
(action, condition) pairs

action	source address	dest address	protocol	source port	dest port	flag bit
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----
deny	all	all	all	all	all	all

Statefull packet filtering

❏ stateless packet filter: heavy handed tool

- admits packets that “make no sense,” e.g., dest port = 80, ACK bit set, even though no TCP connection established:

action	source address	dest address	protocol	source port	dest port	flag bit
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK

- ❏ *Statefull packet filter*: track status of every TCP connection
 - track connection setup (SYN), teardown (FIN): can determine whether incoming, outgoing packets “makes sense”
 - timeout inactive connections at firewall: no longer admit packets

Stateful packet filtering

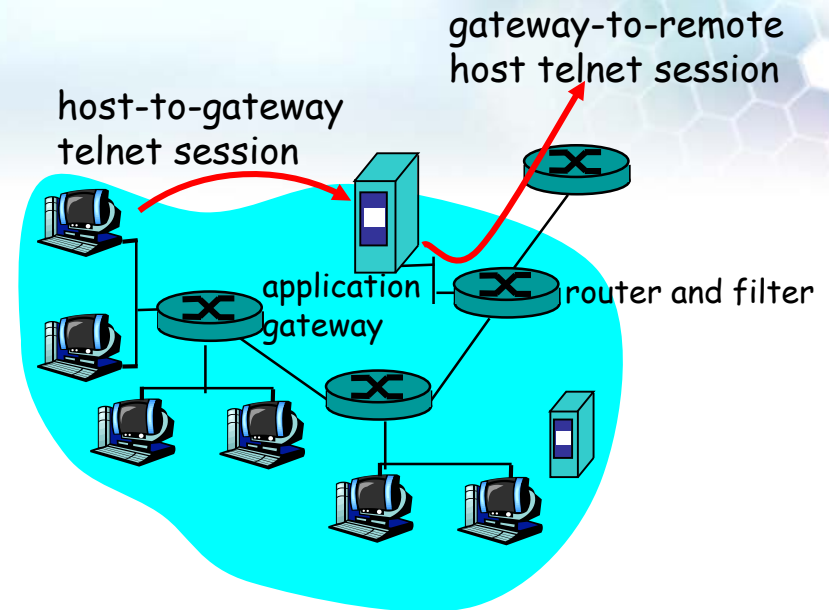
- ACL augmented to indicate need to check connection state table before admitting packet

action	source address	dest address	proto	source port	dest port	flag bit	check conxion
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any	
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK	X
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---	
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----	X
deny	all	all	all	all	all	all	

Application gateways

- ❏ filters packets on application data as well as on IP/TCP/UDP fields.
- ❏ example: allow select internal users to telnet outside.

1. require all telnet users to telnet through gateway.
2. for authorized users, gateway sets up telnet connection to dest host.
Gateway relays data between 2 connections
3. router filter blocks all telnet connections not originating from gateway.



Limitations of firewalls and gateways

- ❖ **IP spoofing**: router can't know if data “really” comes from claimed source
- ❖ if multiple app's. need special treatment, each has own app. gateway.
- ❖ client software must know how to contact gateway.
 - e.g., must set IP address of proxy in Web browser
- ❖ filters often use all or nothing policy for UDP.
- ❖ tradeoff: **degree of communication with outside world, level of security**
- ❖ many highly protected sites still suffer from attacks.

Intrusion detection systems

☒ packet filtering:

- operates on TCP/IP headers only
- no correlation check among sessions

☒ *IDS: intrusion detection system*

- *deep packet inspection*: look at packet contents (e.g., check character strings in packet against database of known virus, attack strings)
- *examine correlation* among multiple packets
 - port scanning
 - network mapping
 - DoS attack

Intrusion detection systems

- multiple IDSs: different types of checking at different locations

