Data Communication and Computer Networks

6. Network Layer
PART-A

Dr. Aiman Hanna

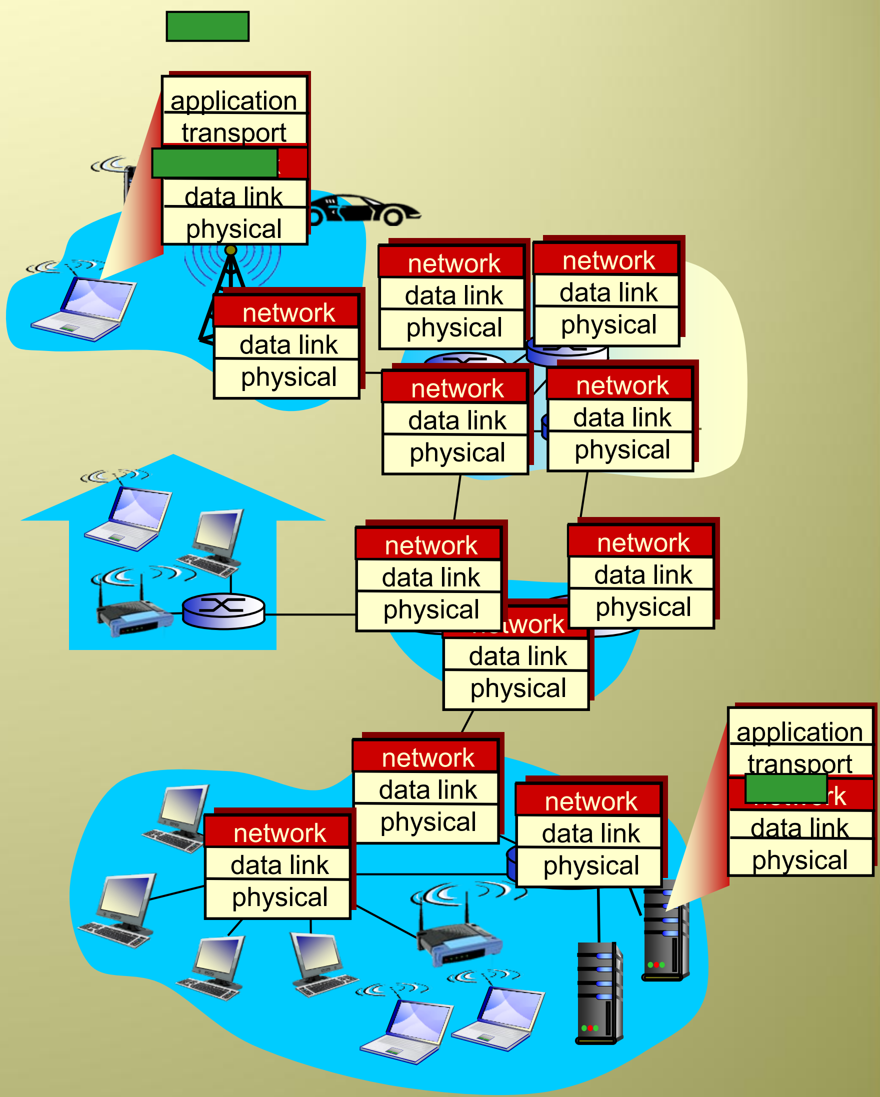
Department of Computer Science & Software Engineering
Concordia University, Montreal, Canada

These slides have mainly been extracted, modified and updated from original slides of :
Computer Networking: A Top Down Approach, 6th edition Jim Kurose, Keith Ross
Addison-Wesley, 2013

Additional materials have been extracted, modified and updated from:
Understanding Communications and Networking, 3e by William A. Shay 2005

Network layer

- transport segment from sending to receiving host
- on sending side encapsulates segments into datagrams
- on receiving side, delivers segments to transport layer
- network layer protocols in *every* host, router
- router examines header fields in all IP datagrams passing through it



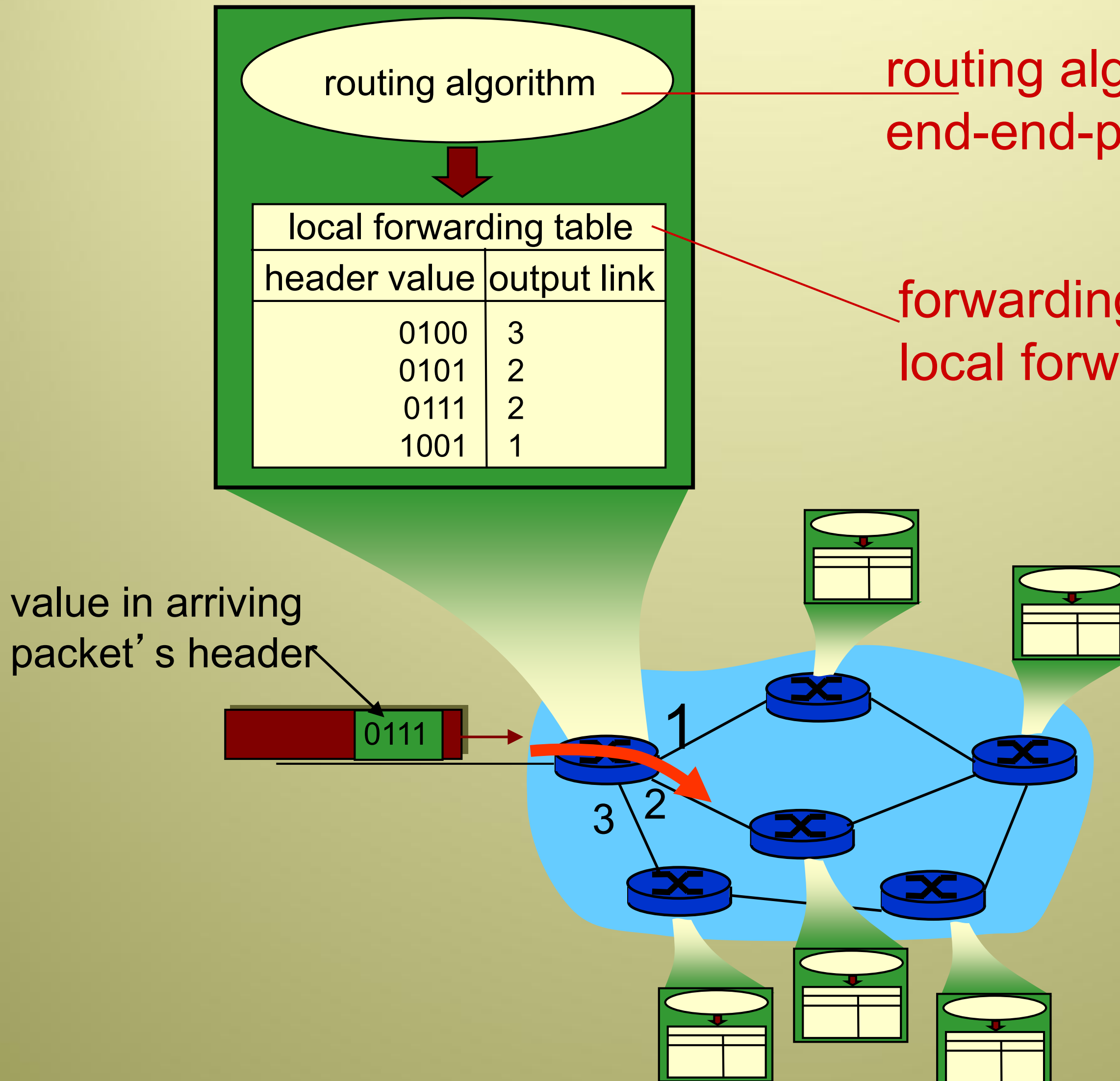
Two key network-layer functions

- *forwarding:* move packets from router's input to appropriate router output
  - involves a single router
- *routing:* determine route taken by packets from source to dest.
  - involves all routers from source to distination
  - *routing algorithms*

analogy:

- *routing:* process of planning trip from source to dest
- *forwarding:* process of getting through single interchange

Interplay between routing and forwarding



routing algorithm determines end-end-path through network

forwarding table determines local forwarding at this router

Connection setup

- 3rd important function in *some* network architectures
- before datagrams flow, two end hosts *and* intervening routers establish virtual connections
  - routers get involved
- network vs transport layer connection service:
  - *network:* between two hosts (may also involve intervening routers in case of VCs)
  - *transport:* between two processes

Network service model

Q: What is the *service model* for "channel" transporting datagrams from sender to receiver?

possible questions in relation to services provided to Transport layer:

- Is **delivery guaranteed?**
- Is **delivery time/delay guaranteed** (i.e. packets must be delivered in less than *x* msec)?

possible questions in relation to congestion:

- Does the network layer provide feedback about network **congestion?**

Network service model

Q: What is the *service model* for "channel" transporting datagrams from sender to receiver?

services related to flow of packets:

- ***in-order delivery***: will packets be delivered in the same order they are sent?
- ***guaranteed minimal BW***: is there a guarantee of minimum bandwidth to flow (i.e. guaranteed host-to-host delay as long as the sender transmits below the allowable bit rate)?
- ***guaranteed maximum jitter***: is delay between consecutive transmissions same as delay between receptions?

services related to security:

- ***security***: would the network encrypt/decrypt?

Network service model

A: in general, there is only a partial list of services that a network layer can provide

- the Internet’s network layer provides a single service: **Best-effort**
  - packets may arrive out of order,
  - no guarantee on delivery time, and actually
  - no guarantee of delivery at all!
  - no feedback on congestion as well

- other networks implemented service models that go beyond the Internet's *best-effort* service

Connection, connection-less service

- *datagram* network provides ***network-layer connectionless*** service
- *virtual-circuit* network provides ***network-layer connection*** service
- analogous to TCP/UDP connection-oriented / connectionless transport-layer services, but:
  - *service:* host-to-host
  - *no selection:* network provides one or the other
  - *implementation:* in network core

Virtual circuits

"source-to-dest path behaves much like telephone circuit" (just without the pure dedication)

- performance-wise
- network actions along source-to-dest path

- call setup for each call *before* data can flow, and teardown at the end
- each packet carries VC identifier (not destination host address)
- *every* router on source-dest path maintains "state" for each passing connection
- link, router resources (bandwidth, buffers) may be *allocated* to VC (dedicated resources = predictable service)

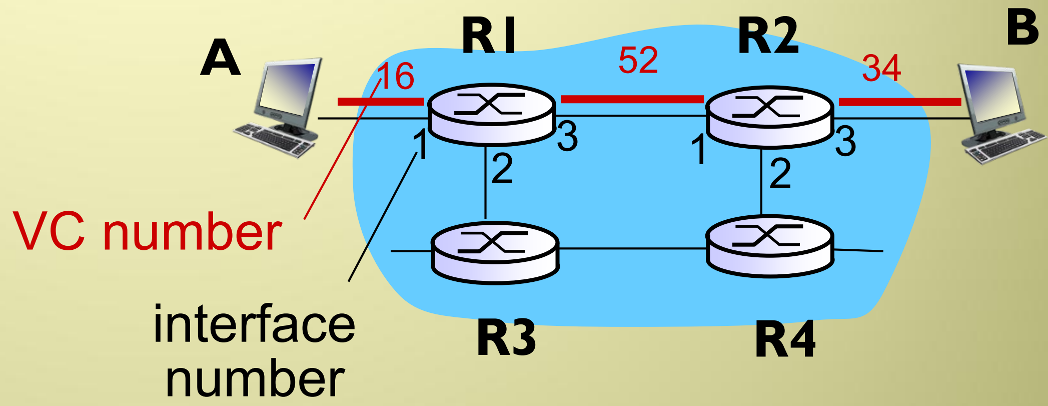
Virtual Circuit (VC) implementation

a VC consists of:

1. *path:* a series of links and routers between source and destination
2. *VC numbers:* one number for each link along path
3. *entries in forwarding tables* in routers along path

- packet belonging to VC carries VC number (rather than dest address) in its header
- VC number can be changed on each link
  - new VC number comes from forwarding table

VC forwarding table



forwarding table in northwest router (R1):

Incoming interface	Incoming VC #	Outgoing interface	Outgoing VC #
1	16	3	52
2	63	1	18
3	7	2	17
1	97	3	87
…	…	…	…

VC routers maintain connection state information!

Virtual circuits

There are 3 main phases in VC

1) **VC setup:**

- transport layer contacts network layer with receiver address
- network determines the path (series of links and routers of the VC)
- network assigns a VC number to the determined links and update the router table with these entries
- the network may also **reverse resources** during this phase

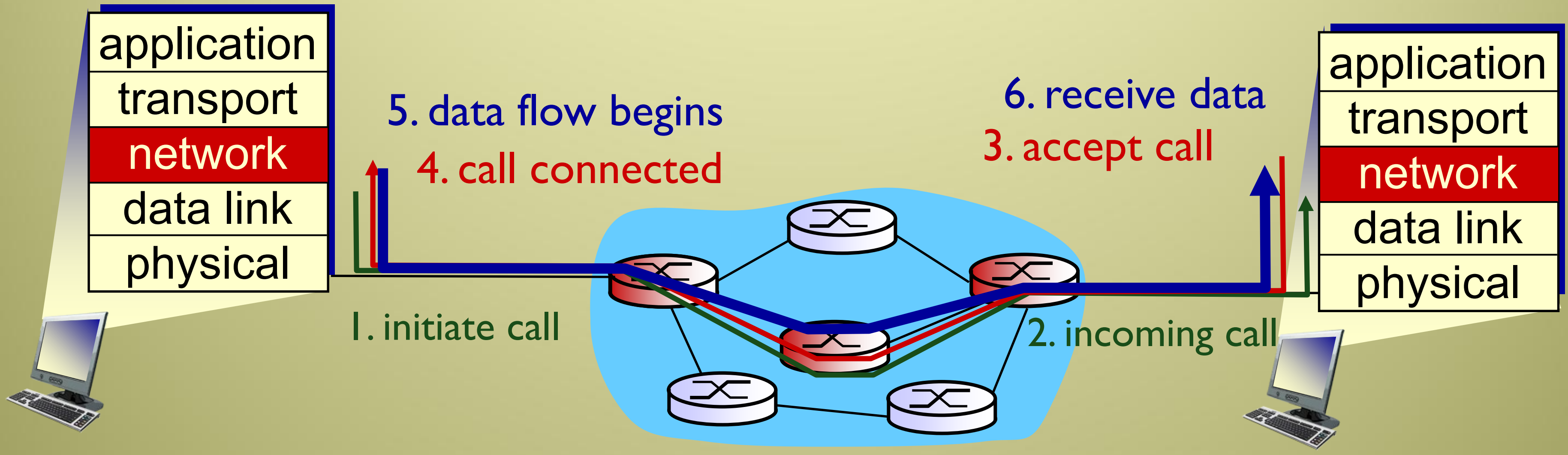
2) **VC teardown:**

- transport inform network that the VC is to be terminated
- network informs the end system at the other side, removes all assigned VC numbers and update tables

Virtual circuits: signaling protocols

3) **data transfer:**

Once VC is established, packets can start flowing along the VC

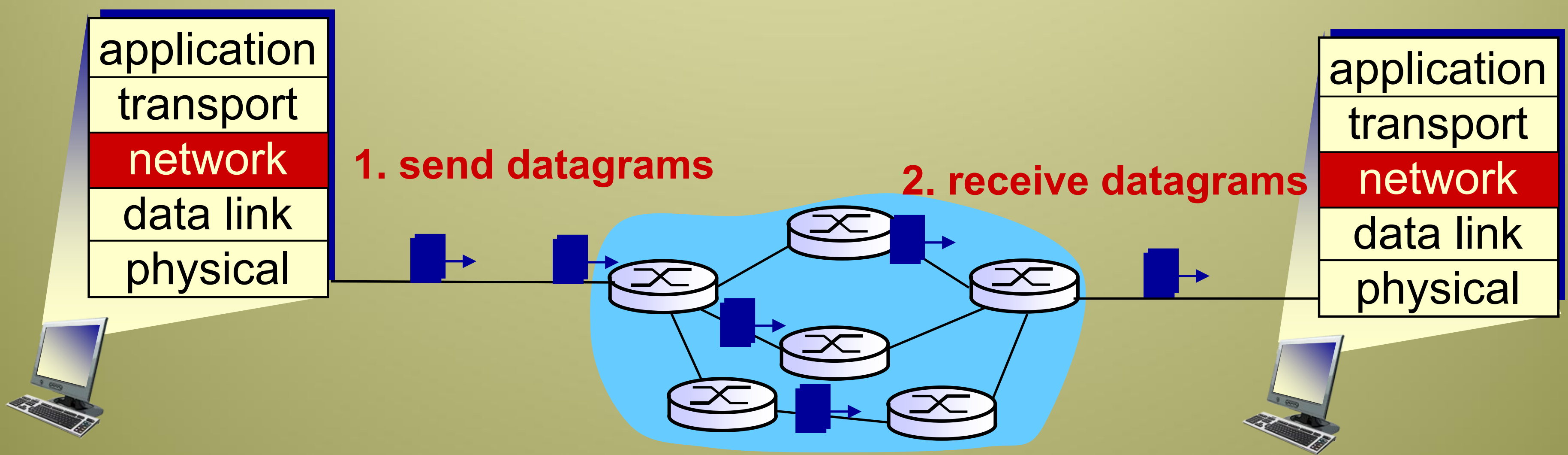


NOTE:

- Note: VC (setup, maintain, teardown VC) is used in ATM, frame-relay, X.25
- not used in today's Internet

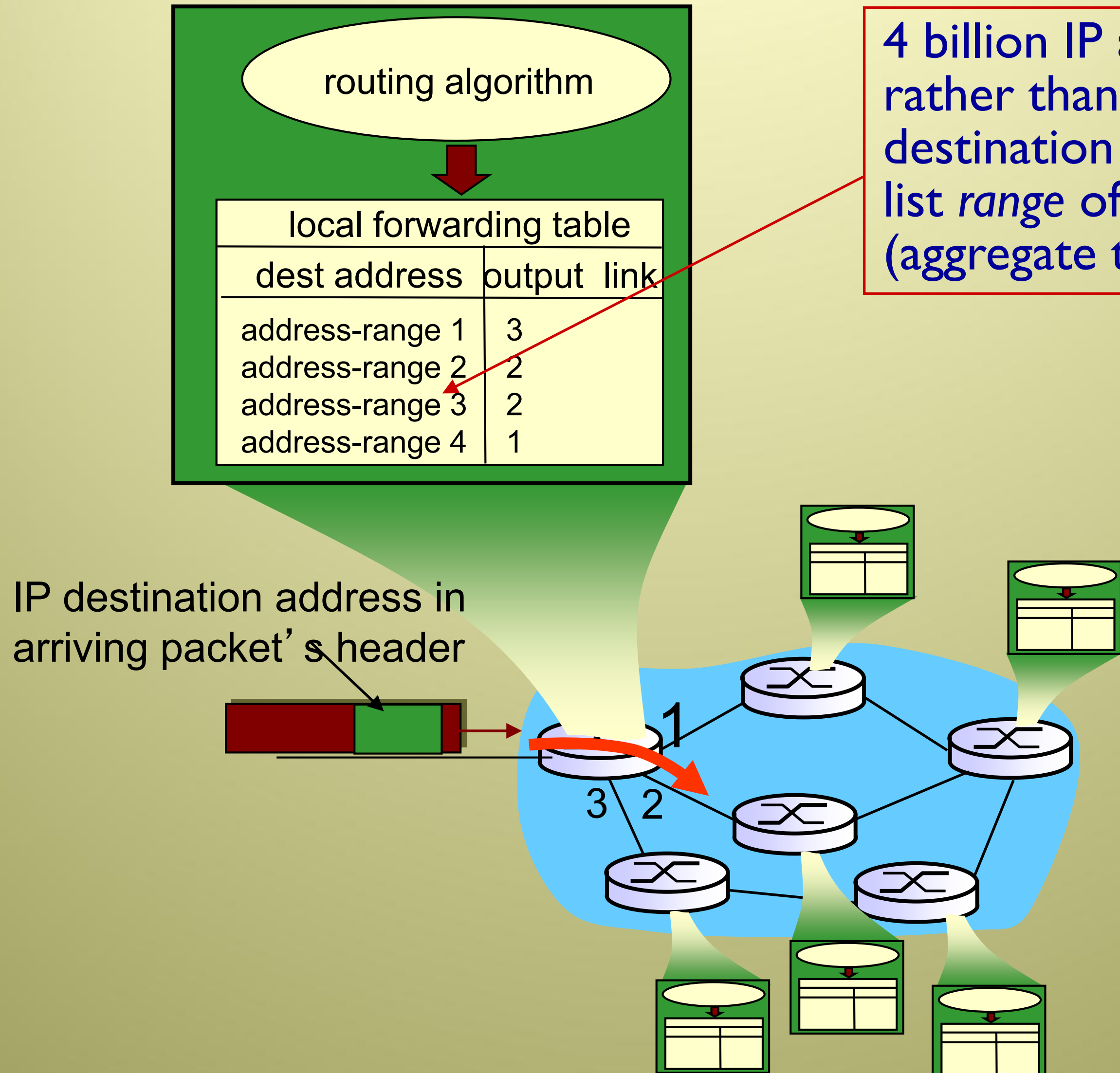
Datagram networks

- no call setup at network layer
- routers: no state about end-to-end connections
  - no network-level concept of "connection"
- packets forwarded using destination host address



Datagram forwarding table

4 billion IP addresses, so rather than list individual destination address list *range* of addresses (aggregate table entries)



Datagram forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 **through** 11001000 00010111 00010111 11111111	**0**
11001000 00010111 00011000 00000000 **through** 11001000 00010111 00011000 11111111	**1**
11001000 00010111 00011001 00000000 **through** 11001000 00010111 00011111 11111111	**2**
Otherwise	**3**

Q: can we avoid the insertion of these many (4 billion) entries?

Datagram forwarding table

A: yes; we can replace them with only 4 entries; just let the router matches the **prefix** of the addresses

Prefix Match	Link Interface
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
Otherwise	3

Q: but what happens if ranges do not divide up so nicely?

Longest prefix matching

longest prefix matching

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Prefix Match	Link Interface
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
Otherwise	3

examples:

DA: 11001000 00010111 00010110 10100001 **which interface?**

DA: 11001000 00010111 00011000 10101010 **which interface?**

Datagram or VC network: why?

Internet (datagram)

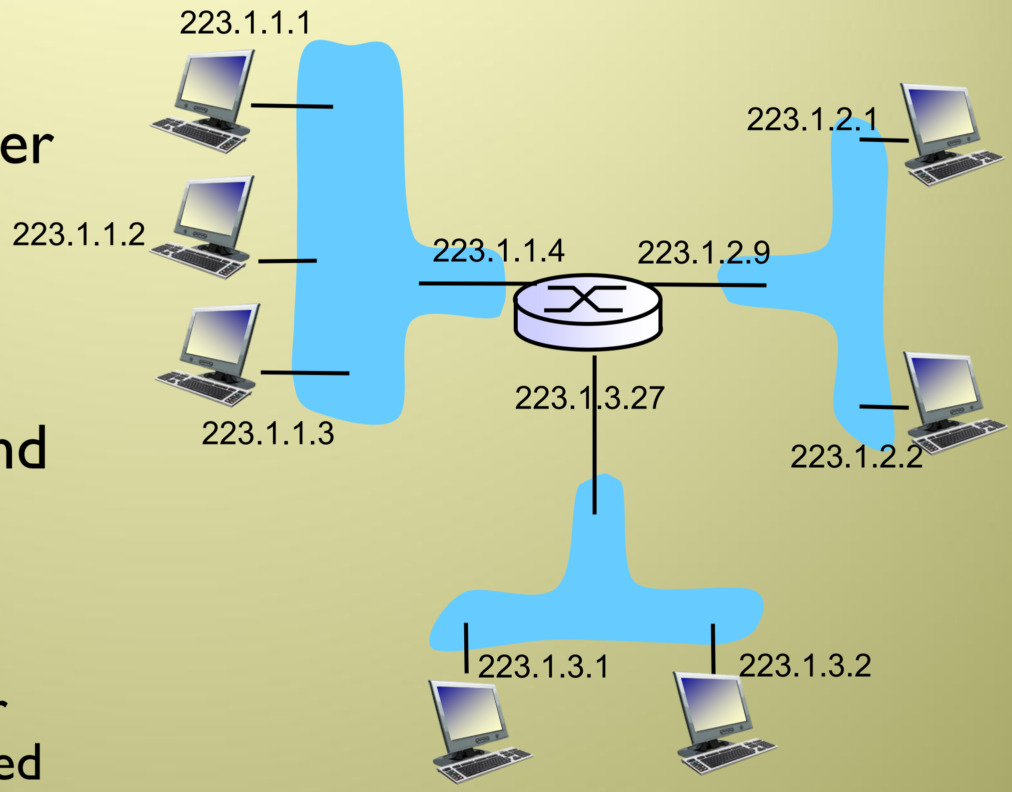
- data exchange among computers
  - "elastic" service, no strict timing requirement
- many link types
  - different characteristics
  - uniform service difficult
- "smart" end systems (computers)
  - can adapt, perform control, error recovery
  - ***simple inside network, complexity at "edge"***

ATM (VC)

- evolved from telephony
- human conversation:
  - strict timing, reliability requirements
  - need for guaranteed service
- "dumb" end systems
  - i.e. rotary telephones
  - ***complexity inside network***

IP addressing: introduction

- *IPv4 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 or two interfaces (e.g., wired Ethernet, wireless 802.11)
- *IP addresses associated with each interface*

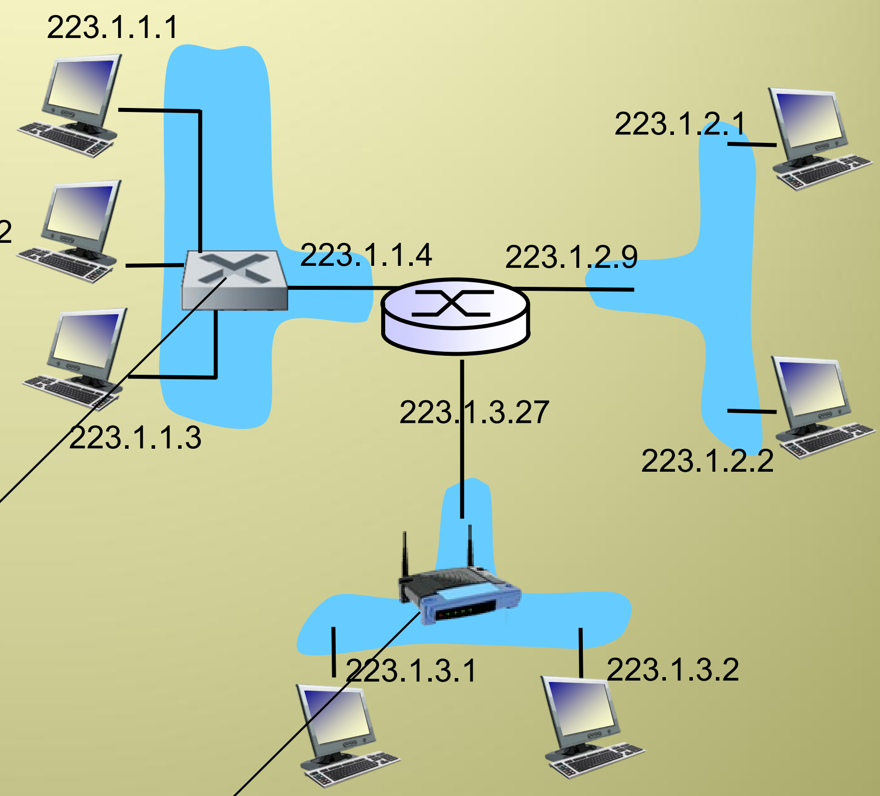


$$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$$

IP addressing: introduction

Q: how are interfaces actually connected?

A: the details will be discussed later! However, for now:



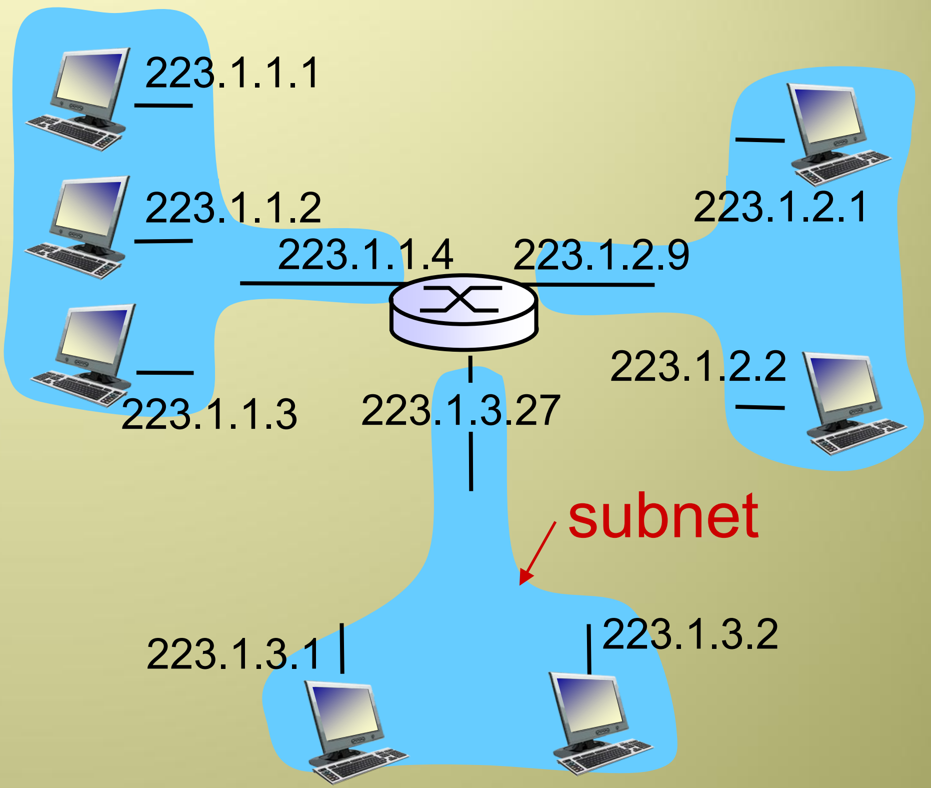
A: wired Ethernet interfaces connected by Ethernet switches

A: wireless WiFi interfaces connected by WiFi base station

For now: don't need to worry about how one interface is connected to one another (with no intervening router)

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*

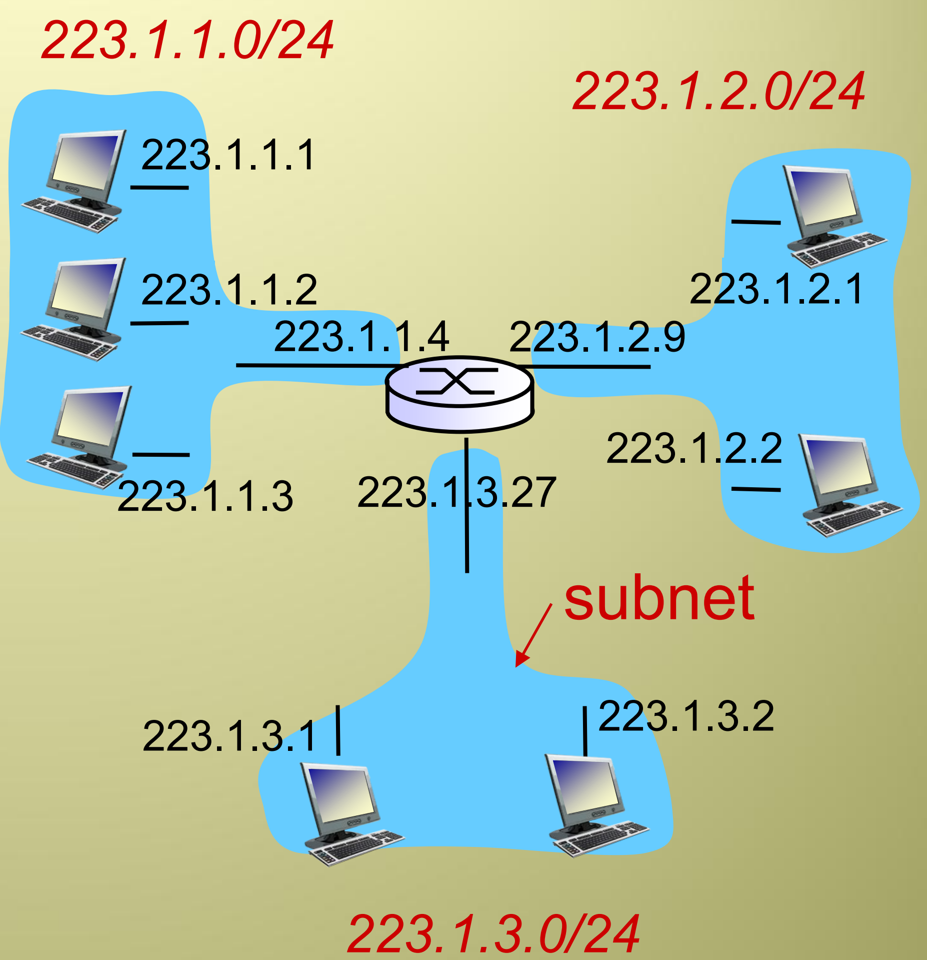


network consisting of 3 subnets

Subnets

recipe

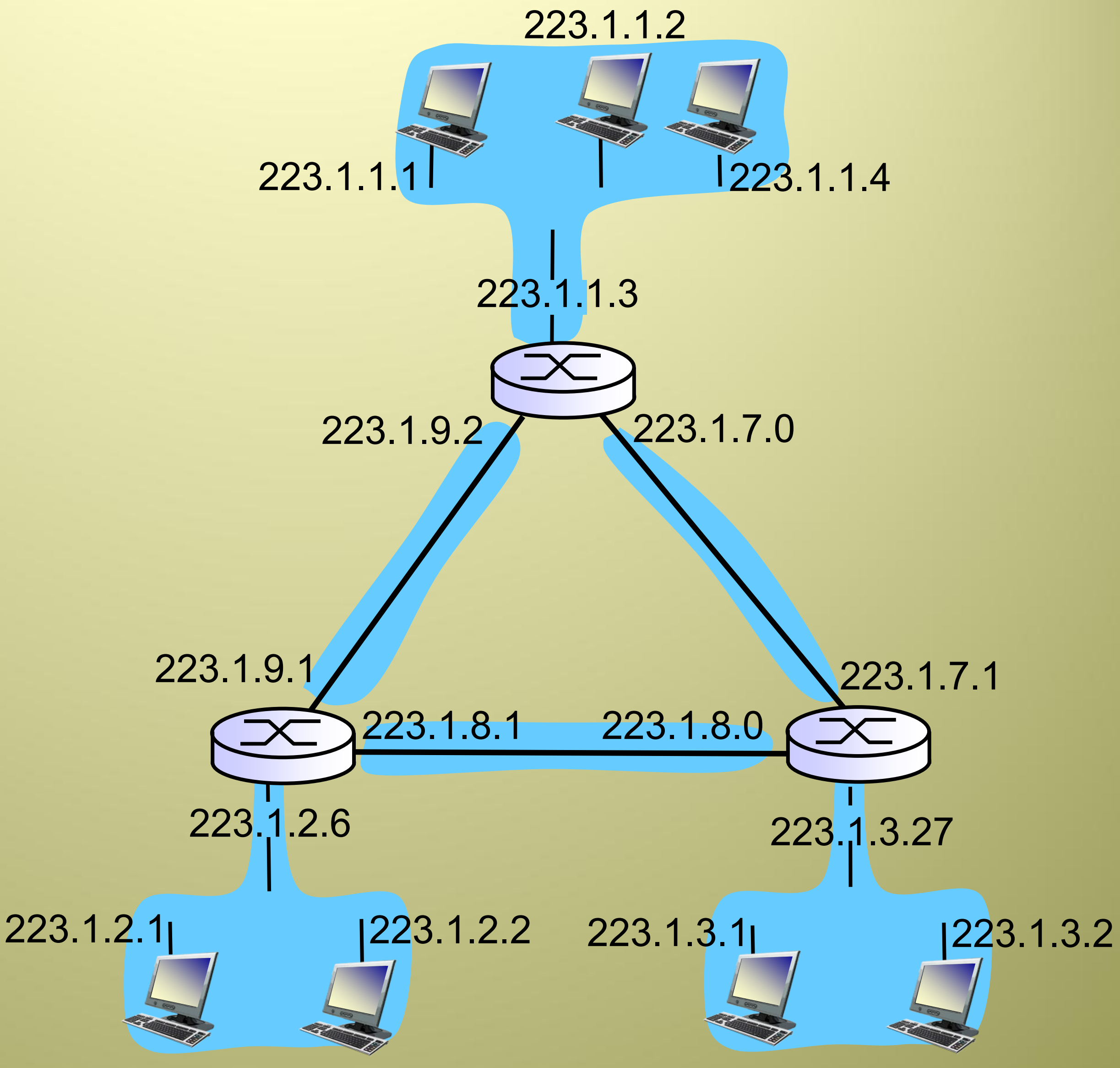
- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- each isolated network is called a *subnet*



subnet mask: /24

Subnets

how many?



Internet Addressing

- textual addresses are given to hosts
- for example, concordia.ca is 132.205.7.63
- DNS takes care of the translation between the textual representation and the IP addresses

Internet Addressing

- but now, which bits represent a network number and which represent a local ID?

- this depends on the type of address; IP recognizes several classification of Internet addresses depending on the size of the organization's network

classification	byte1	byte2	byte3	byte4	# of NWs	max nodes
Class A	0nnnnnnn	xxxxxxxx	xxxxxxxx	xxxxxxxx	2^7 (127)	2^{24} (>16 million)
Class B	10nnnnnn	nnnnnnnn	xxxxxxxx	xxxxxxxx	2^{14} (16,364)	2^{16} (65,536)
Class C	110nnnnn	nnnnnnnn	nnnnnnnn	xxxxxxxx	2^{21} (2,097,152)	2^8 (256)
Class D multicast	1110 followed by a 28-bit multicast address					
Class E Reserved	1111; reserved					
n's represent bits in the network number; x's represent bits in the local identifier						

Internet Addressing

- Example:

  Which class does Concordia.ca belong (assume IP address: 132.205.7.63)? Which class does Nasa.gov belong (Assume IP address: 64.37.246.3)?

- Concordia.ca is 132.205.7.63, which is **10**000100.11001101.00000111.00111111 which is class B

- Nasa.gov has address: 64.37.246.3, which is

  01000000.00100101.11110110.00000011

  So it is class A

Classless Addresses

- the older version of IP, IPv4, has address depletion problem
- internet addresses are 32-bit, so there is only finite number of addresses
- $2^{32} \approx 4.3$ billion; isn't that enough?
- what if an organization has 1000 computers?
- two solutions exist:
  - use more bits for addresses (more than 32)
  - use ***Classless InterDomain Routing*** (**CIDR**)

CIDR

- specifies a group of addresses that do not fall into any of the predefined classes
- each address in the group can still be interpreted as a network number followed by a local identifier
- commonly used to allocate multiple class C addresses
- for example, if a network has 1000 computers, CIDR allocates 4 consecutive Class C addresses to that network

CIDR

- Example:

Class C	Bit representation	Address range
211.195.8.0	1101011-11000011-00001000-xxxxxxxx	211.195.8.0 to 211.195.8.255
211.195.9.0	1101011-11000011-00001001-xxxxxxxx	211.195.9.0 to 211.195.9.255
211.195.10.0	1101011-11000011-00001010-xxxxxxxx	211.195.10.0 to 211.195.10.255
211.195.11.0	1101011-11000011-00001011-xxxxxxxx	211.195.11.0 to 211.195.11.255

CIDR

- further, a router can extract the network number (in this case it is 211.195.8.0), which is the 1st address of this network

- this can be done via logical AND operation between a 32-bit subnet mask (255.255.252.0) and an IP address

IP Address	1101011-11000011-**000010**xx-xxxxxxxx
AND with subnet mask	11111111-11111111-11111100-00000000
Network number	1101011-11000011-00001000-00000000 (which is, 211.195.8.0)

CIDR

- in effect, CIDR groups several smaller networks together and visualize them as a single large network; this is referred to as ***supernetting***

- advantages?

- yet, there is another issue: each IP packet has the destination IP address, where the first 3 bits determine whether this is class A, B or C

- the rest of the bits in the address are then extracted to determine the network number

CIDR

- determining the network number is straight forward when that number of bits is fixed
- with CIDR, this is not the case
- the router must know the number of network bits
- to allow that, the usual representation of an address (*w.x.y.z*) is replaced by (*w.x.y.z/m*), where *m* is the number of bits in the network ID
- for the previous example, that will be: 211.195.8.0/22, which means that the network number has 22 bits long

Obtaining an Address

- prior to 1999, all network addresses were managed by **IANA**, ***Internet Assigned Numbers Authority***
- in 1999, the Internet Corporation for Assigned Names and Numbers (ICANN) assumed that responsibility and others related to ***Domain Name System*** (**DNS**)
- after all, isn't IP addresses are hard to memorize?!
  - DNS is needed

Obtaining an Address

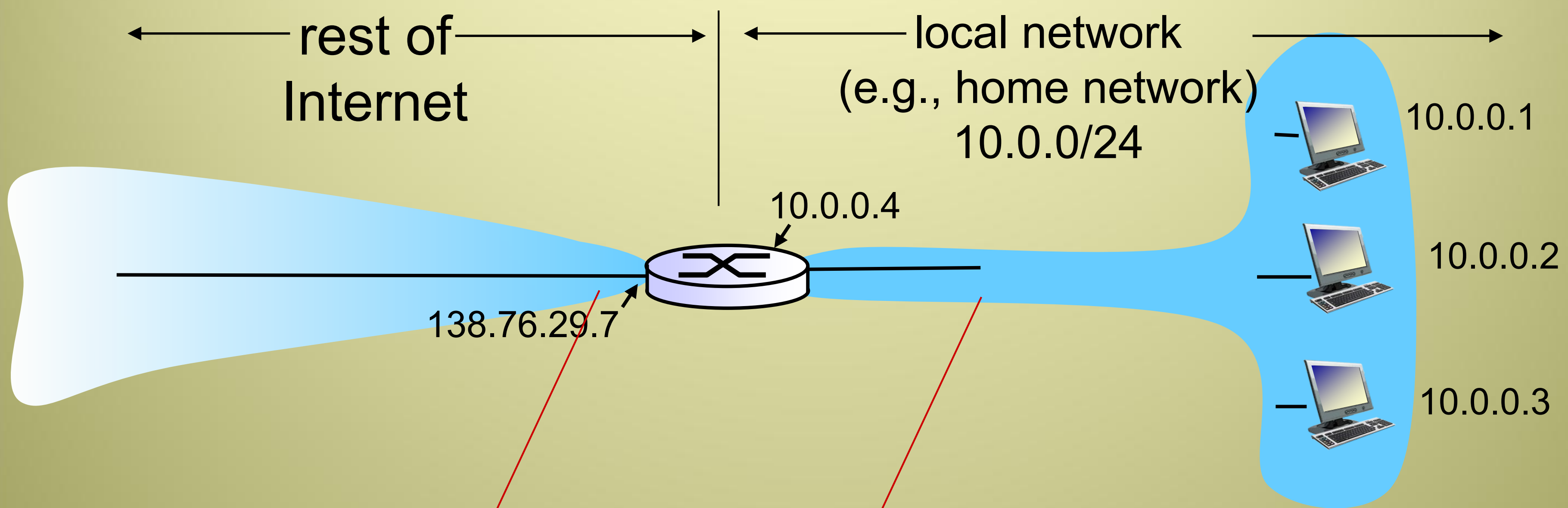
- a computer connected to a company network, or connected to an ISP, will get an IP address assigned
- the machine may have a static IP address, or it may dynamically require an IP address from the server
- the server runs a protocol called ***Dynamic Host Configuration Protocol* (DHCP)**, which allocates the machine one of the available IP addresses it maintains
- try *ipconfig* command

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: Dynamic Host Configuration Protocol: dynamically get address from as server
  - "plug-and-play"

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

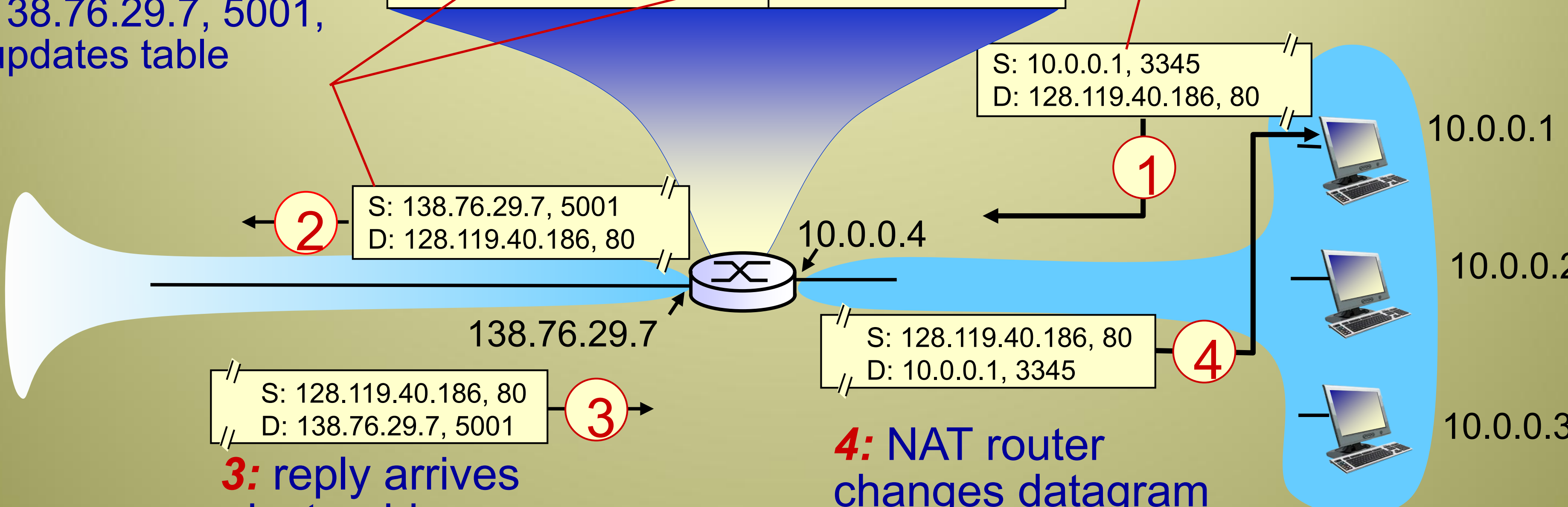
- *incoming datagrams: replace* (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: network address translation

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

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80

2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table



3: reply arrives dest. address: 138.76.29.7, 5001

4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345

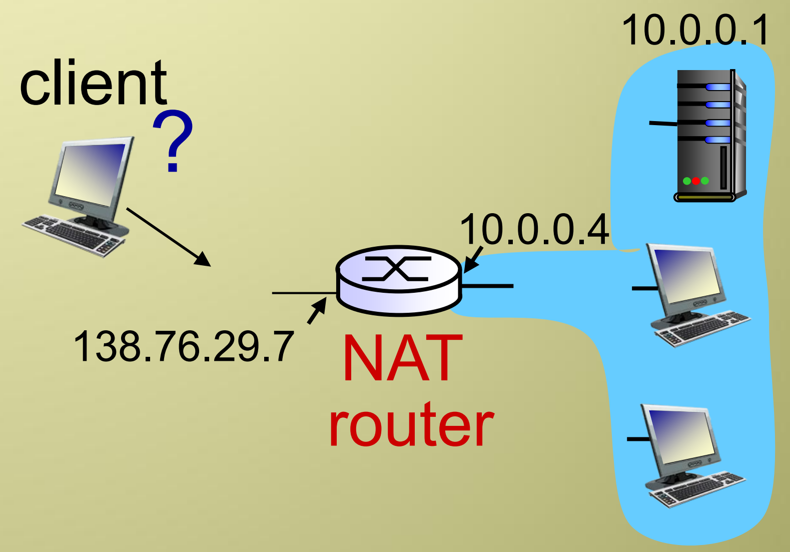
NAT: network address translation

- 16-bit port-number field:
  - 60,000 simultaneous connections with a single LAN-side address!

- NAT is controversial:
  1) routers should only process up to layer 3
  2) violates end-to-end argument
     - NAT possibility must be taken into account by app designers, e.g., P2P applications
  3) port numbers are meant for addressing processes and not addressing hosts
  4)address shortage should instead be solved by IPv6

NAT traversal problem

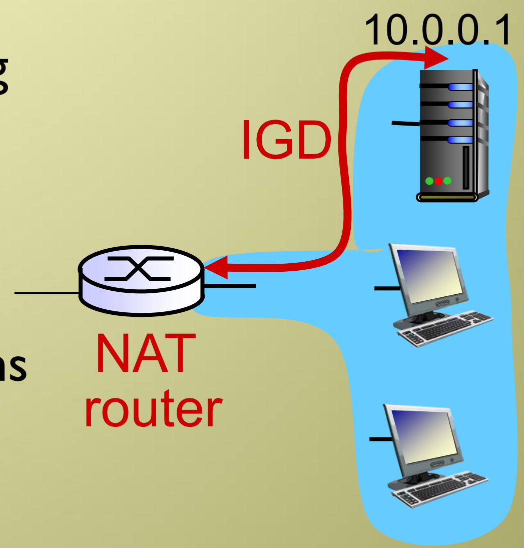
- Yet; there is another major problem with NAT
  - It interferes with P2P applications
- client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - only one externally visible NATed address: 138.76.29.7



- *solution1:* statically configure NAT to forward incoming connection requests at given port to server
  - e.g., (123.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000

NAT traversal problem

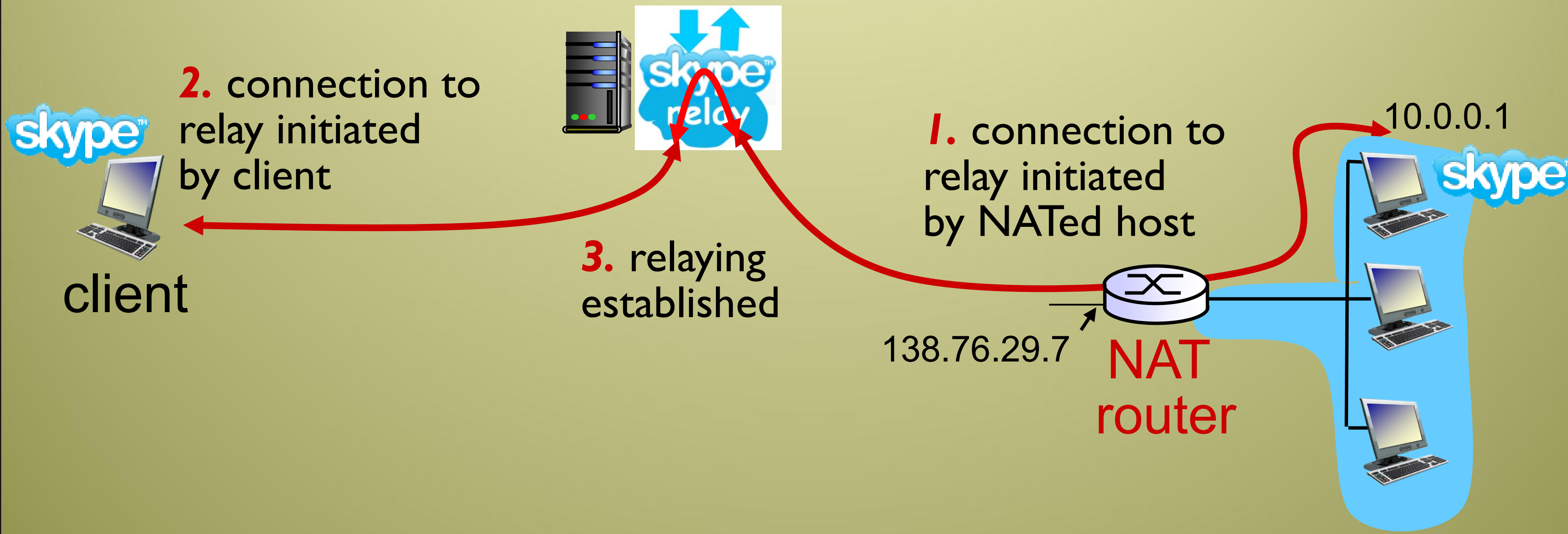
- *solution 2:* **UPnP -** Universal Plug and Play, Internet Gateway Device (IGD) Protocol.
- both the host and NAT must be UPnP compatible
- application running in a host requests NAT mapping between its *(private IP address, private port #* and *public IP address, public port number*)
- if NAT accepts request, it creates the mapping and informs the applications of it public information
- Application advertises its public to other applications communicating with it



- **Example**:
  - host with private IP 10.0.0.1 running BitTorrent at private port 3345
  - application asks NAT to create a "hole" that maps its private info (10.0.0.1, 3345) to its public info (138.23.9.45, 5001), where the public port # 5001 is chosen by the application
  - host advertises to its BitTorrent tracker its public info
  - External hosts use this public information, which are then translated by NAT to connect correctly with the NATed host

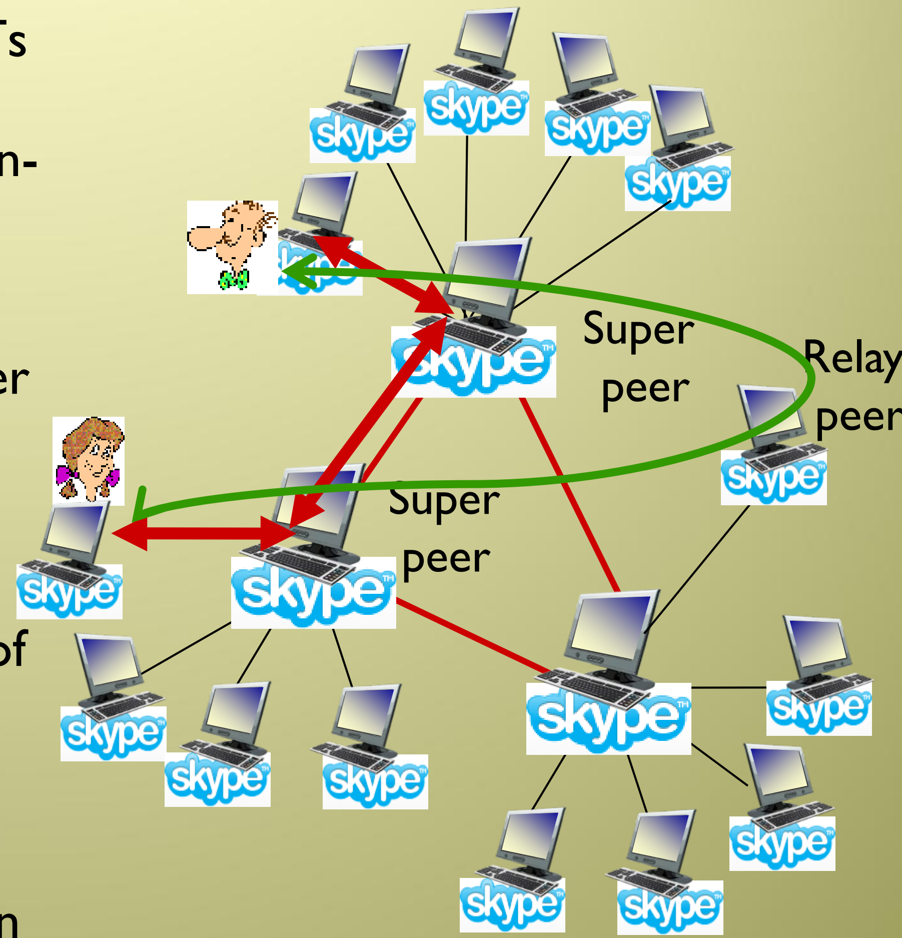
NAT traversal problem

- *solution 3:* relaying (used in Skype)
  - NATed client establishes connection to relay
  - external client connects to relay
  - relay bridges packets between two connections



Skype: peers as relays

- relay solution:
  - both Alice & Bob are behind NATs
    - Alice signs in → get assigned a non-NATed super peer, where she initiates a session with that super peer
    - session allows Alice and super peer to exchange message
    - Bob does the same
    - when Alice needs to call Bob, her super peer informs Bob's super peer, which in turns informs Bob of the incoming call
    - if Bob accepts call, the two super peers assign a third non-NATed super peer, called the Relay Peer, whose job is to relay data between Alice and Bob



- Alice send voice packets to the relay, which forwards them to Bob, and vise versa