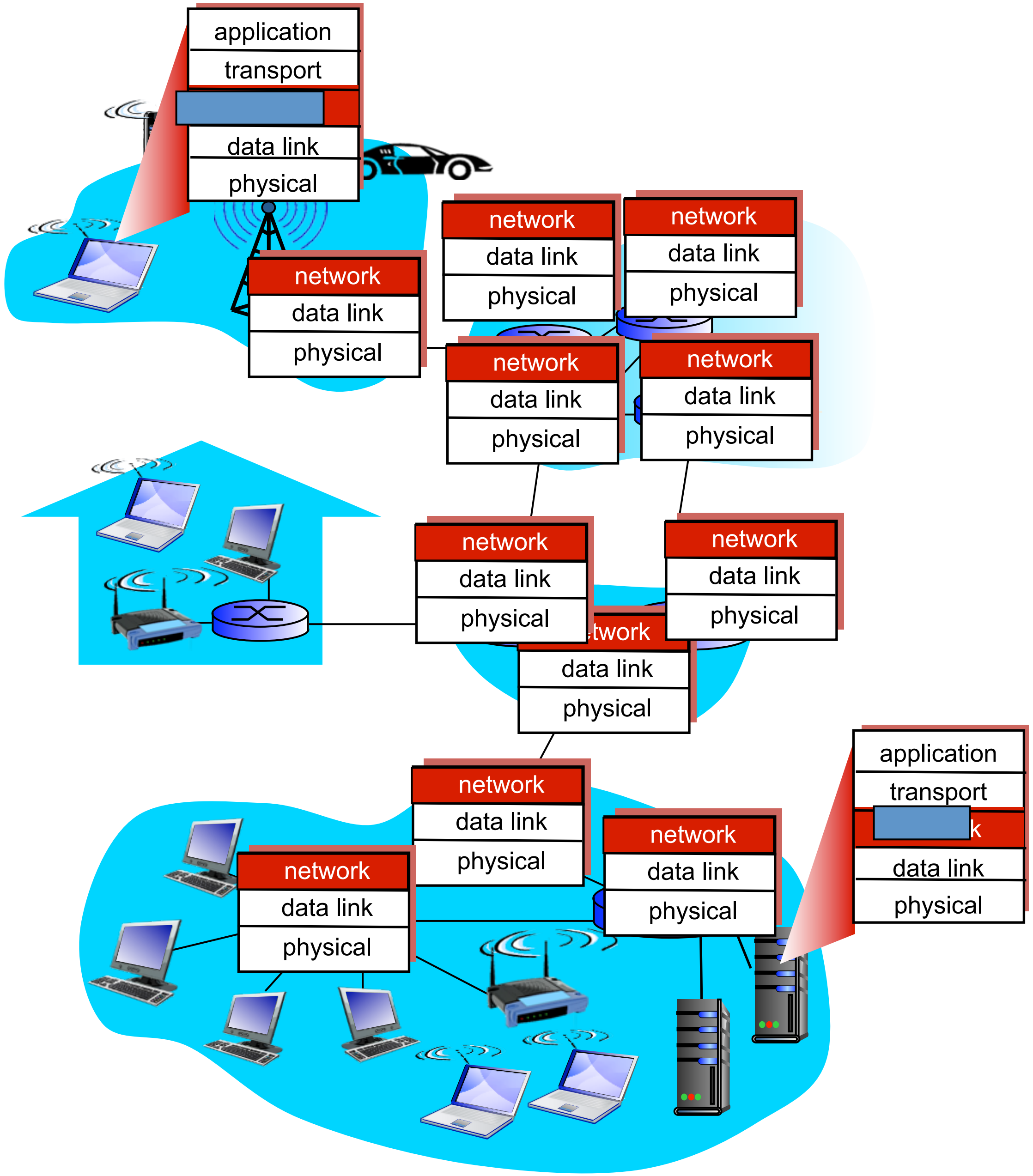
The Network Layer

three important functions:

- *path determination:* route taken by packets from source to dest. ***Routing algorithms***
- *forwarding:* move packets from routers input to appropriate router output
- *call setup:* some network architectures require router call setup along path before data flows



Network layer: data plane, control plane

Data plane

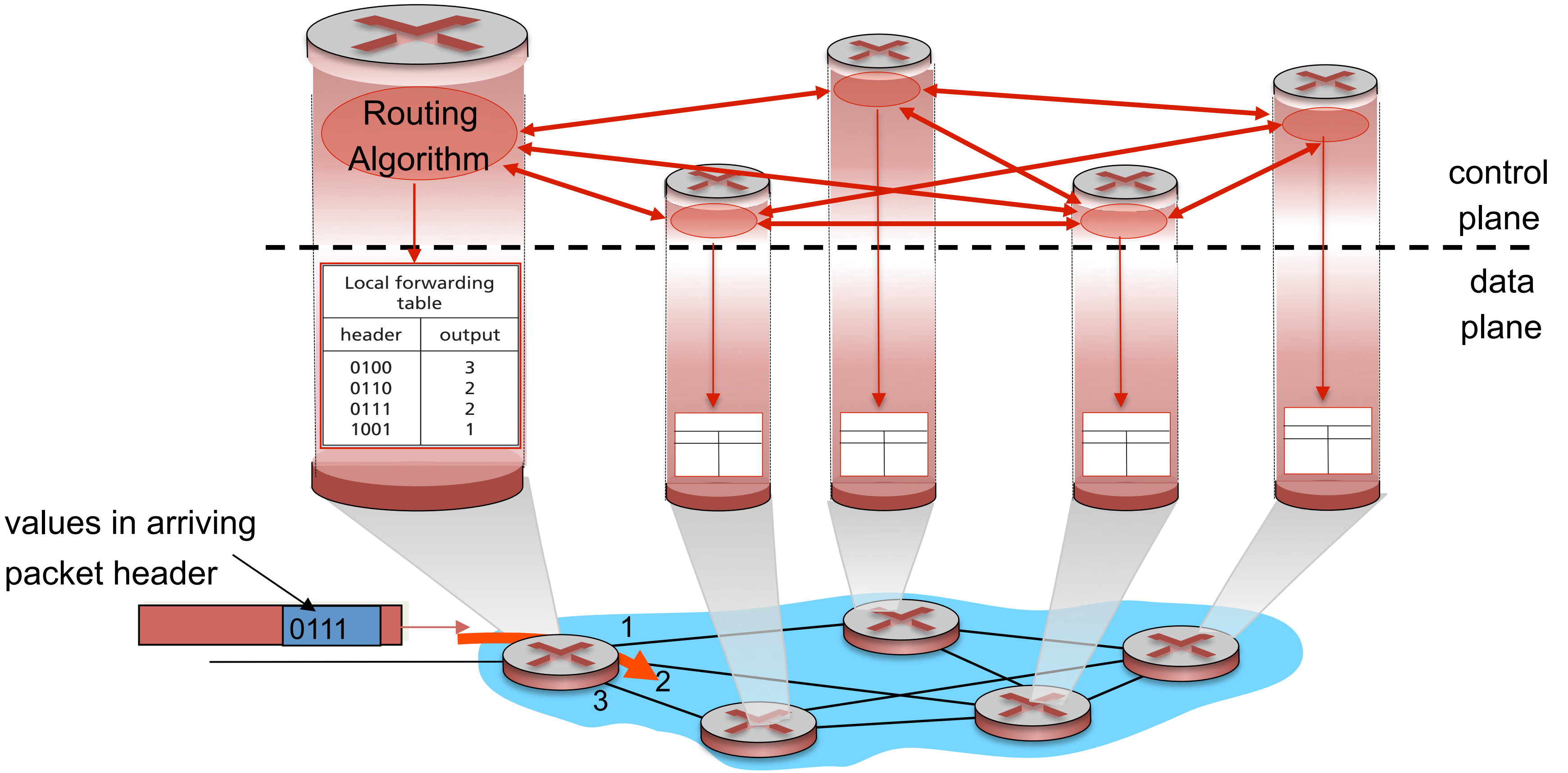
- local, per-router function
- determines how datagram arriving on router input port is forwarded to router output port
- forwarding function

Control plane

- network-wide logic
- determines how datagram is routed among routers along end-end path from source host to destination host
- two control-plane approaches:
  - *traditional routing algorithms:* implemented in routers
  - *software-defined networking (SDN)*: implemented in (remote) servers

Per-router control plane

Individual routing algorithm components *in each and every router* interact in the control plane

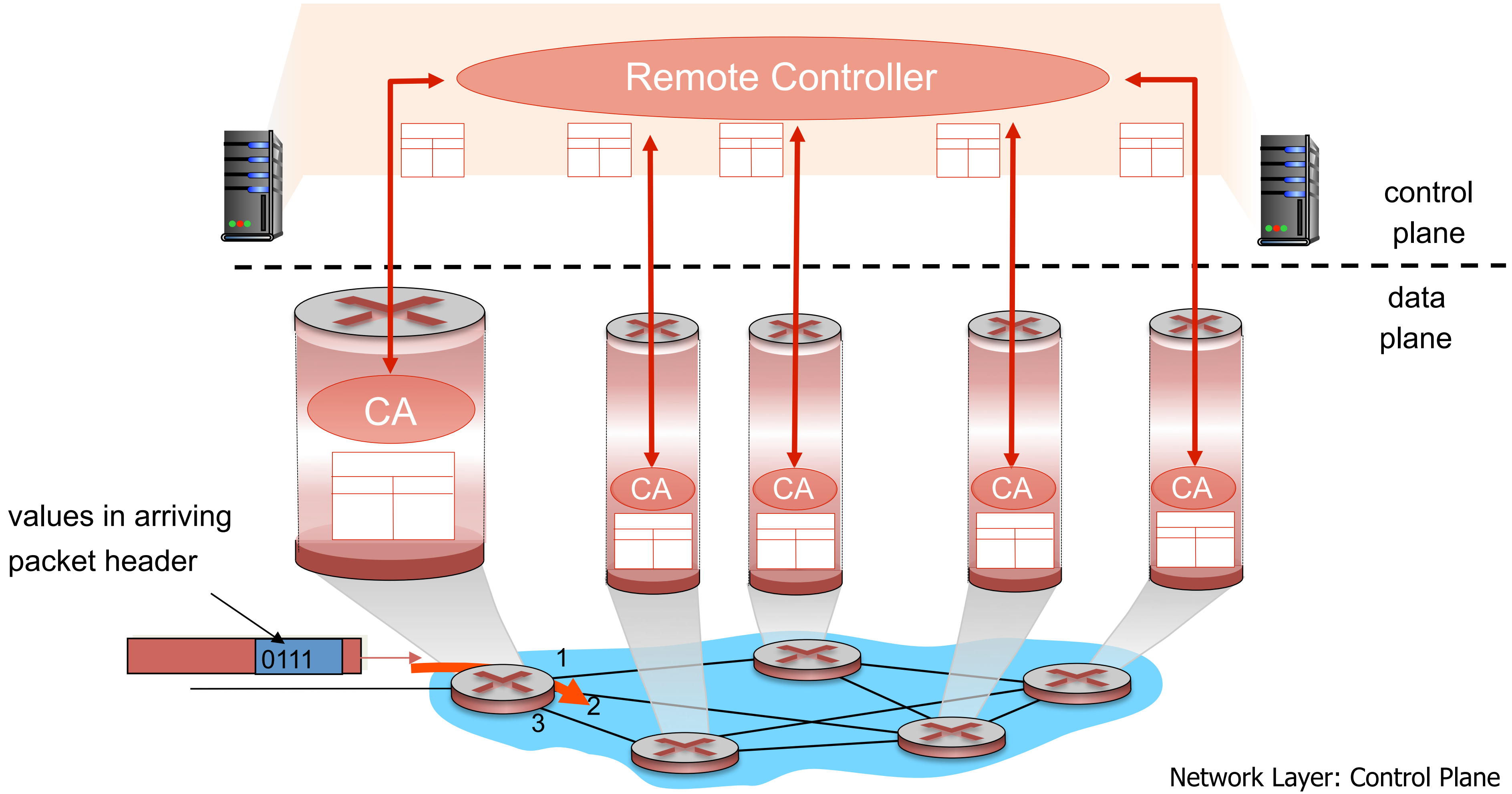


Local forwarding table

header	output
0100	3
0110	2
0111	2
1001	1

Logically centralized control plane

A distinct (typically remote) controller interacts with local control agents (CAs)



Service Models…

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

- guaranteed bandwidth?
- preservation of inter-packet timing (no jitter)?
- loss-free delivery?
- in-order delivery?
- congestion feedback to sender?

service abstraction

The most important abstraction provided by network layer:

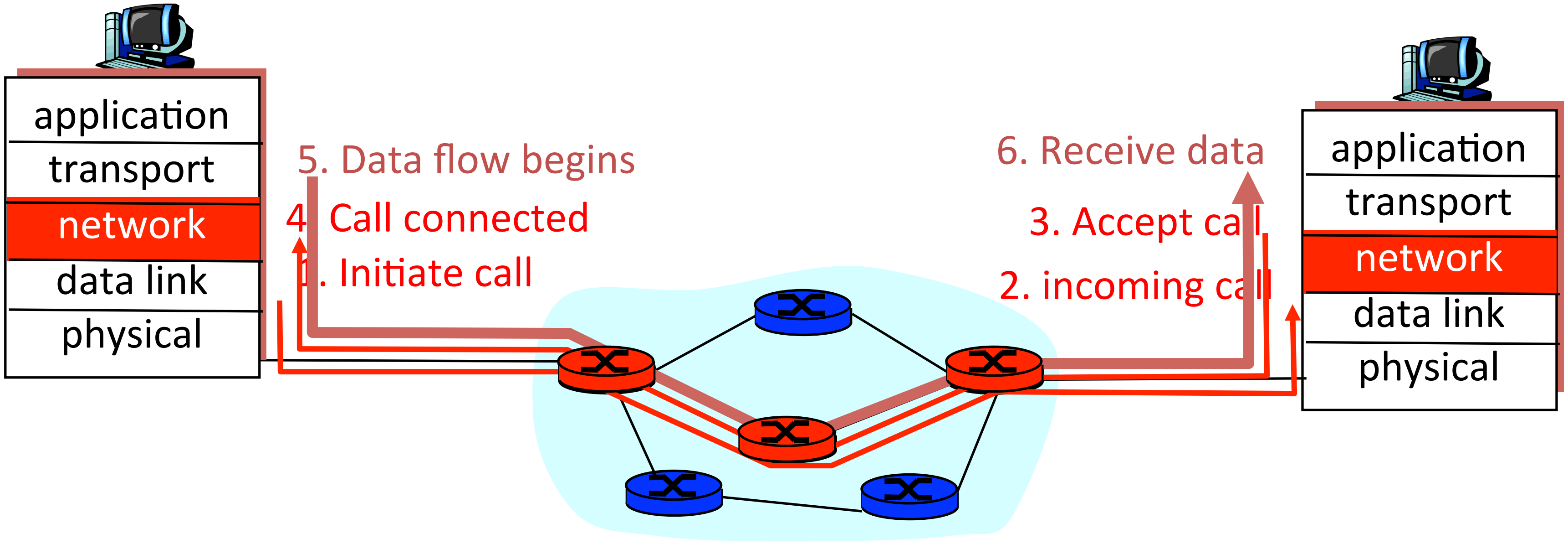
?

virtual circuit
or
datagram?

? ?

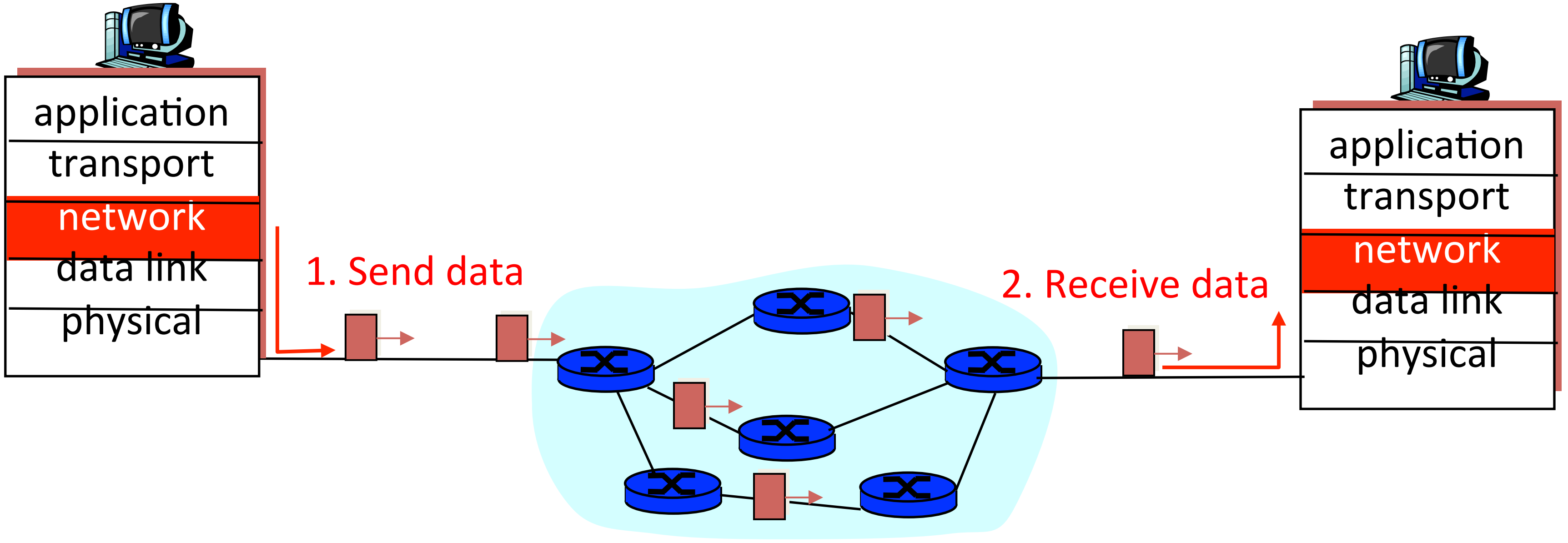
Virtual circuits: signaling protocols

- used to setup, maintain teardown VC
- used in ATM, frame-relay, X.25
- not used in today’s Internet



Datagram networks: the Internet model

- no call setup at network layer
- routers: no state about end-to-end connections
  - no network-level concept of "connection"
- packets typically routed using destination host ID
  - packets between same source-dest pair may take different paths



Datagram or VC network: why?

Internet (DataGram)

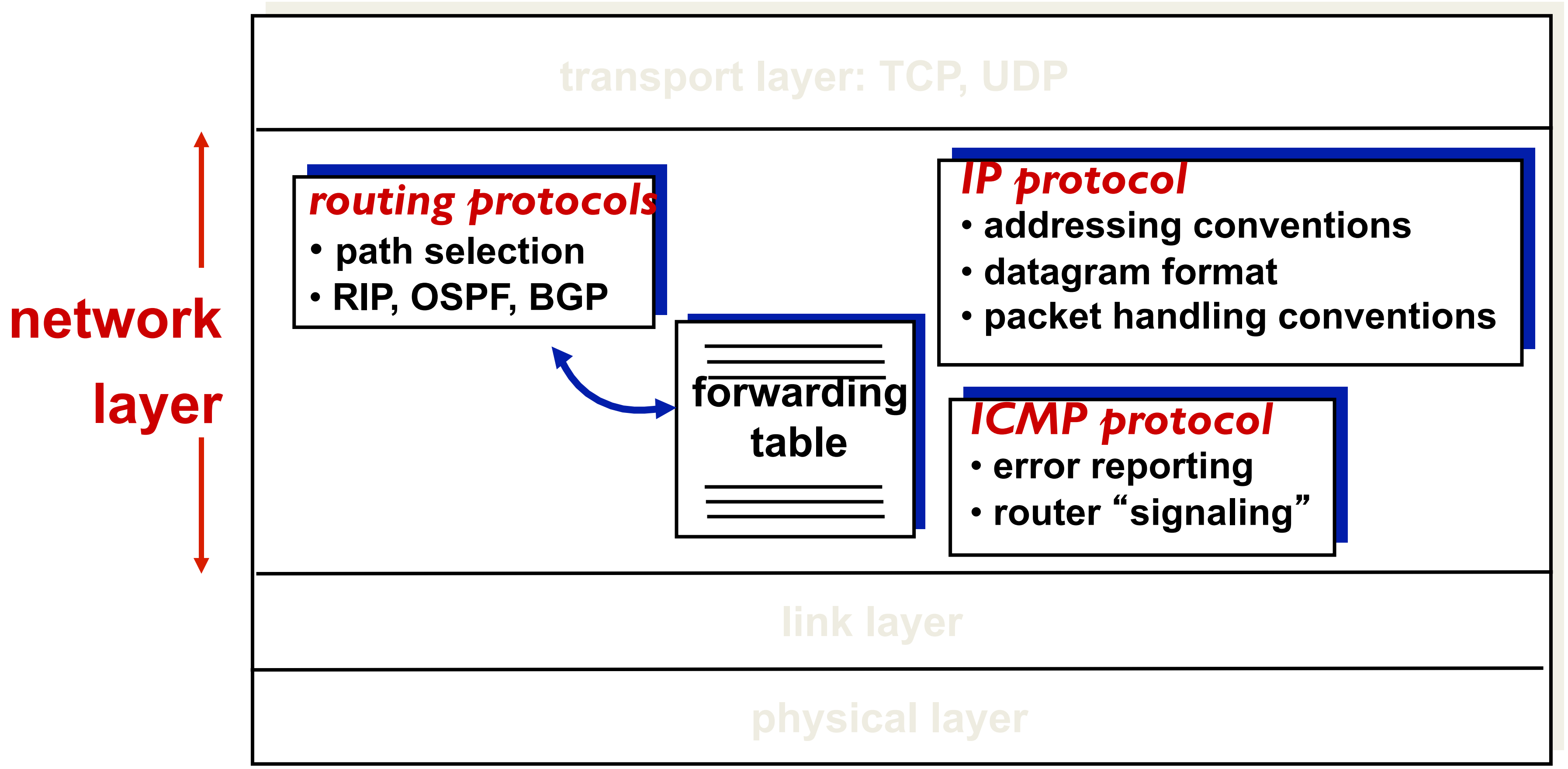
- data exchange among computers
  - “elastic” service, no strict timing req.
- “smart” end systems (computers)
  - can adapt, perform control, error recovery
  - simple inside network, complexity at “edge”
- many link types
  - different characteristics
  - uniform service difficult

ATM (Virtual Circuit)

- evolved from telephony
- human conversation:
  - strict timing, reliability requirements
  - need for guaranteed service
- “dumb” end systems
  - telephones
  - complexity inside network

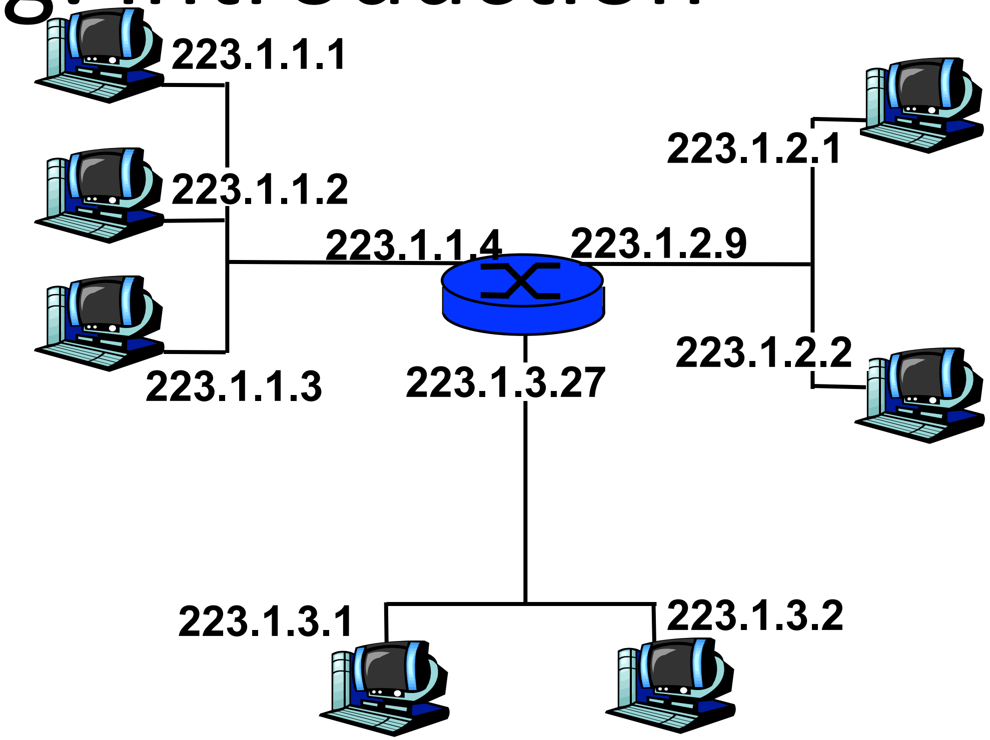
The Internet network layer

host, router network layer functions:



IP Addressing: introduction

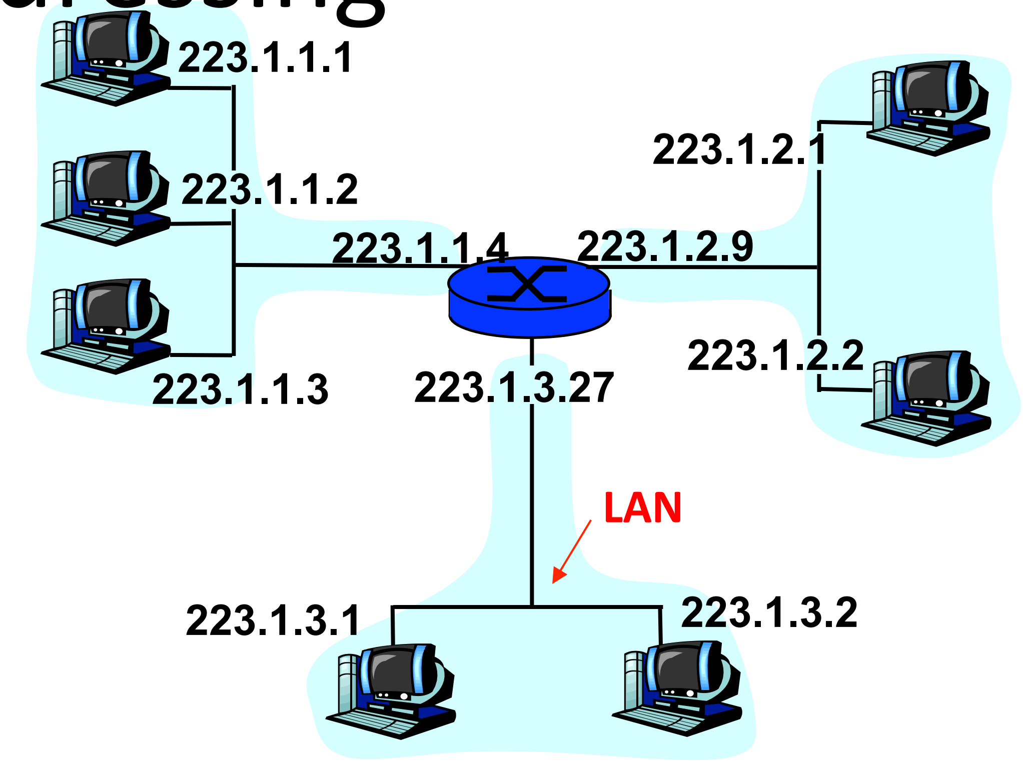
- IP address: 32-bit identifier for host, router *interface*
- *interface:* connection between host, router and physical link
  - routers typically have multiple interfaces
  - host may have multiple interfaces
  - IP addresses associated with interface, not host, router



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

IP Addressing

- IP address:
  - network part (high order bits)
  - host part (low order bits)
- *What's a network ?* (from IP address perspective)
  - device interfaces with same network part of IP address
  - can physically reach each other without intervening router

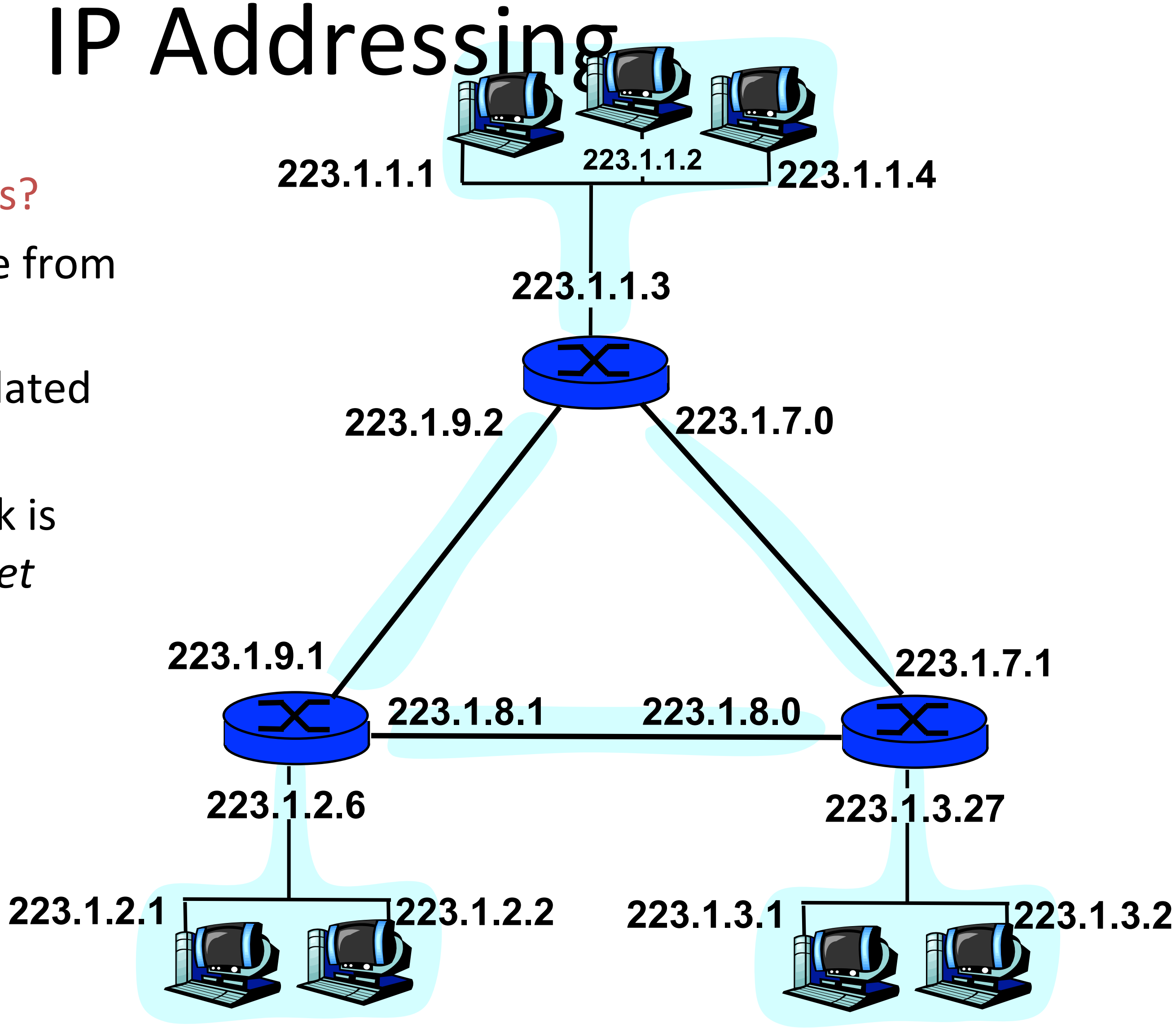


network consisting of 3 IP networks (for IP addresses starting with 223, first 24 bits are network address)

IP Addressing

How to find the networks?

- Detach each interface from router, host
- create "islands of isolated networks"
- each isolated network is also known as a *subnet*

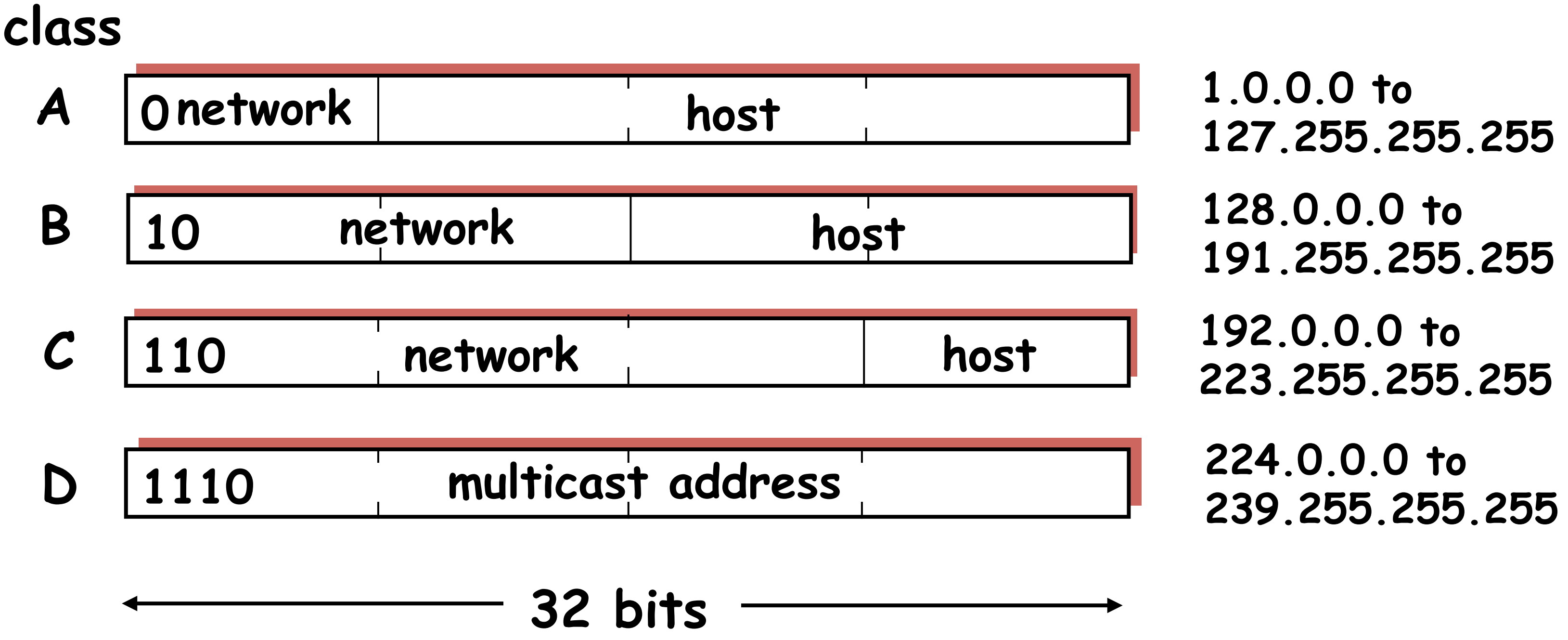


Interconnected system consisting of six networks

IP Addresses

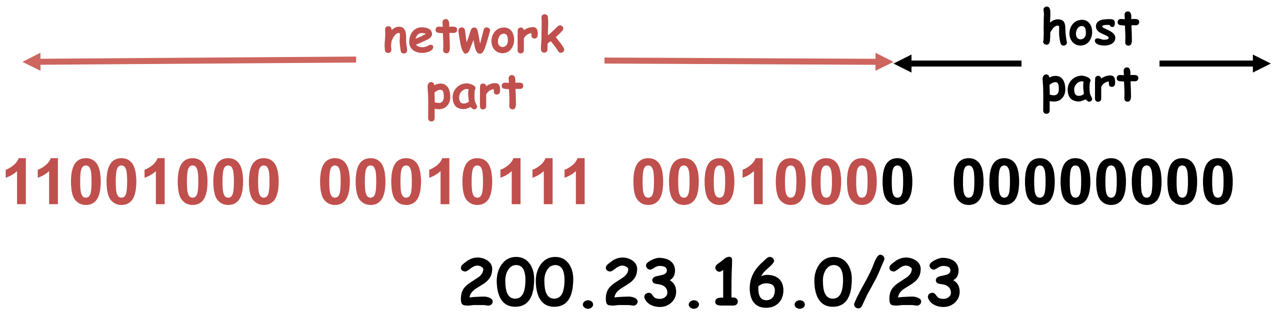
given notion of “network”, let’s re-examine IP addresses:

“class-full” addressing:



IP addressing: CIDR

- classful addressing:
  - inefficient use of address space, address space exhaustion
    e.g., class B net allocated enough addresses for 65K hosts, even if only 2K hosts in that network
  - There are many class C network numbers — some 2 million
  - Allocate them (instead of class B numbers) in variable size blocks
    eg. 2000 addresses could be allocated by 8 contiguous class C networks) instead of a class B network.
- CIDR: Classless InterDomain Routing
  - network portion of address of arbitrary length
  - address format: a.b.c.d/x, where x is # bits in network portion of address



11001000 00010111 00010000 00000000

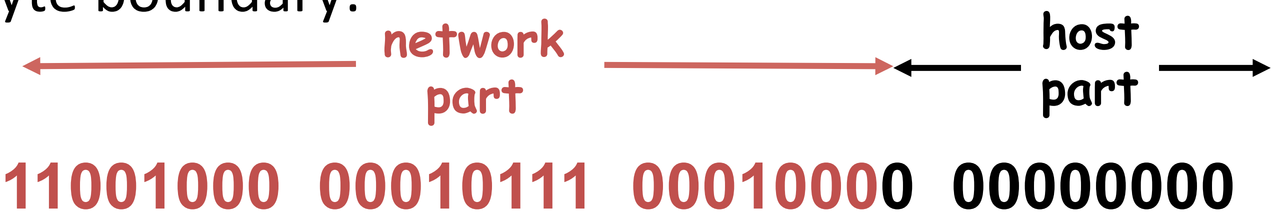
200.23.16.0/23

Broadcast and Multicast

- Most of the IP addresses that are used have a network section and a host section.
  - The larger the network section, the more networks that you can have.
  - The larger the host section, the more hosts you can have.
- There are some special IP addresses and ranges that are used for specific purposes in a network.
- The broadcast address:
  - The broadcast address for any network sends the packet to EVERY node in the network.
  - Before CIDR, you could just set the host bits to all 1s.
  - The broadcast address in 10.x.x.x is 10.255.255.255
  - The broadcast address in 192.168.1.x is 192.168.1.255
  - This works neatly if you are using classes – but what about CIDR?

Broadcast in CIDR

- A CIDR address such as 200.23.16.0/23 does not break neatly over a byte boundary:



network part

host part

11001000 00010111 00010000 00000000

- In this case, we still set the host part to all 1s, but this means that the broadcast address for network 200.23.16.0 is actually 200.23.17.255!
- Network identifiers have a host part of all 0s, broadcast addresses have host part of all 1s. This means that you can't use the first or last host address in any range.
- In the case above, by moving the boundary, we have more addresses than we would have in a class C, but far fewer than in a class B. We should have 512 but we lose the first (network) and the last (broadcast) – 510.

Unicast, Broadcast and Multicast

- Unicast addresses send datagrams to a single destination.
- Broadcast addresses are used to send a datagram to every other host in a given network.
- Multicast addresses are used to send datagrams to a group of interested parties.
- How do you show interest?
  - The hosts and routers use Internet Group Management Protocol (IGMP) to communicate their desire to join the multicast network.
  - The router uses the source address to determine data stream direction
    - The source is considered to be *upstream*
    - The router finds all the registered *downstream* interfaces and sends the packets out through the interfaces, routing AWAY from the source.
    - This is also called *reverse path forwarding*.

IP Address Allocation

- Allocation is based on *geographical* zone (to simplify routing):
  194.0.0.0 to 195.255.255.255 – Europe
  198.0.0.0 to 199.255.255.255 – North America
  200.0.0.0 to 201.255.255.255 – Central and South America
  202.0.0.0 to 203.255.255.255 – Asia and Pacific
  - Each region has ~32 million addresses
  - Numbers 204.0.0.0 to 223.255.255.255 (some 320 million numbers) are still in reserve.

IP addresses: how to get one?

- Hosts (host portion):
- hard-coded by system admin in a file
- DHCP: Dynamic Host Configuration Protocol: dynamically get address: “plug-and-play”
  - host broadcasts “DHCP discover” msg
  - DHCP server responds with “DHCP offer” msg
  - host requests IP address: “DHCP request” msg
  - DHCP server sends address: “DHCP ack” msg

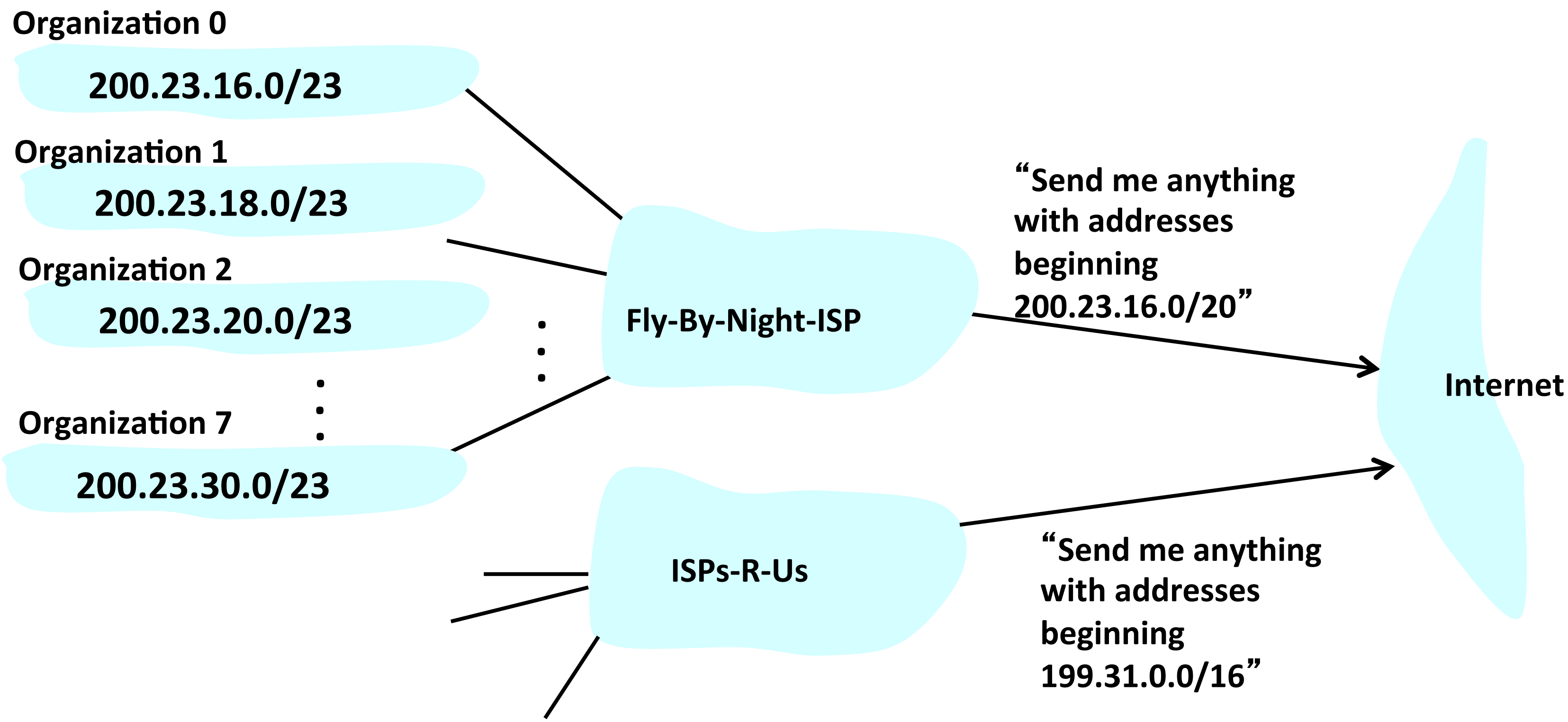
IP addresses: how to get one?

- Network (network portion):
- get allocated portion of ISP's address space:

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

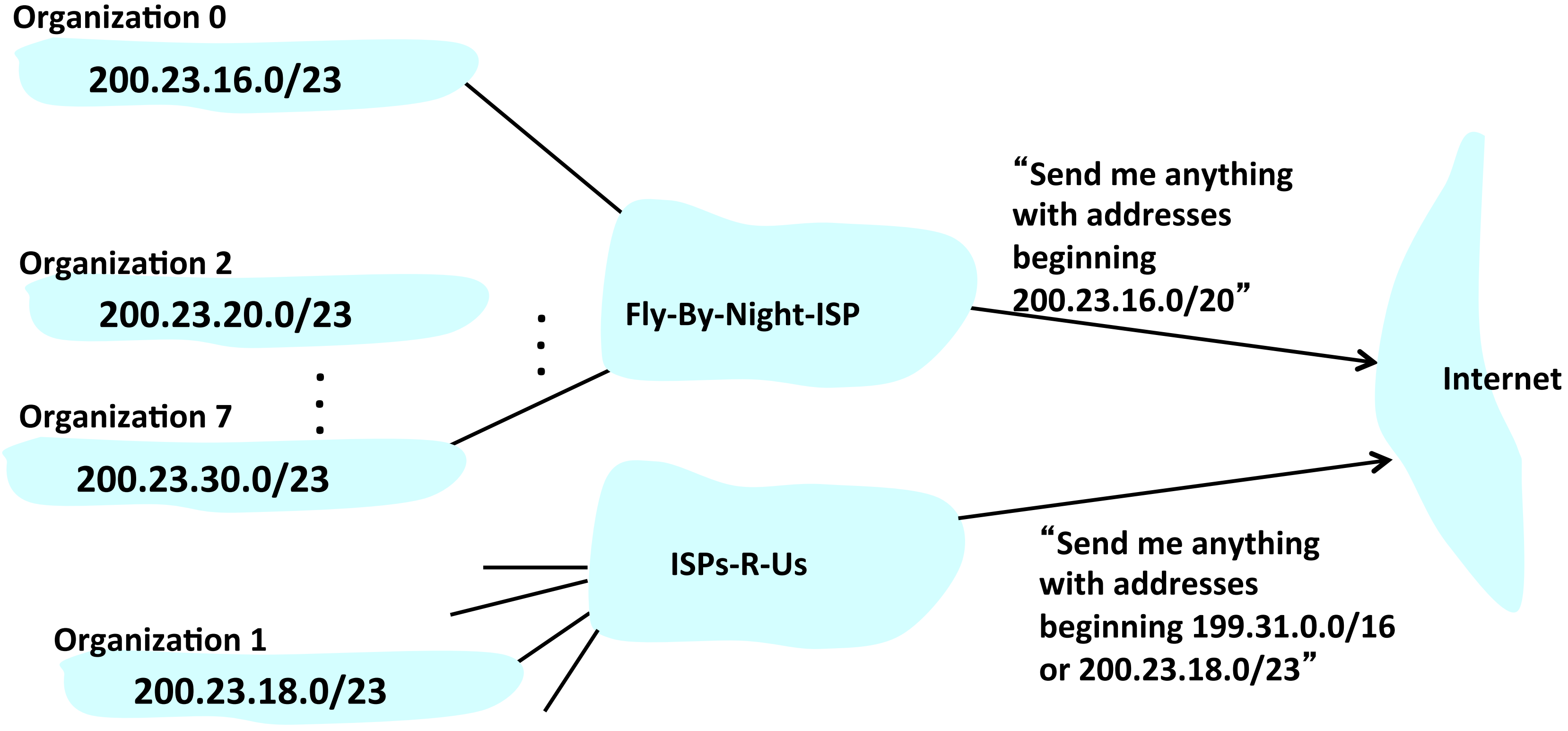
Hierarchical addressing: route aggregation

Hierarchical addressing allows efficient advertisement of routing information:



Hierarchical addressing: more specific routes

ISPs-R-Us has a more specific route to Organization 1



Getting a datagram from source to dest.

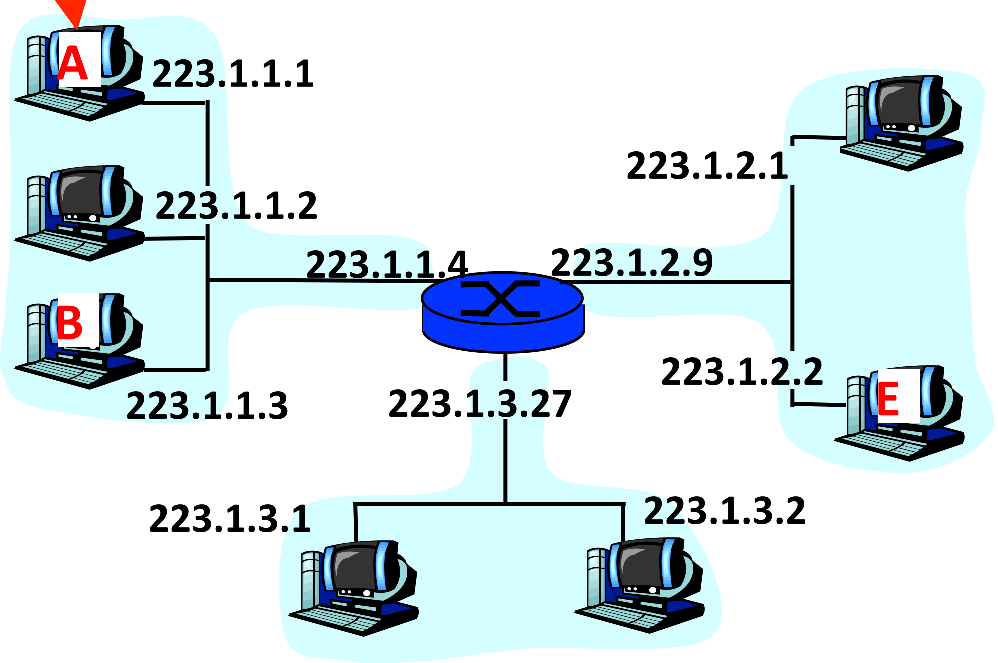
- IP datagram:

misc fields	source IP addr	dest IP addr	data

- **datagram remains unchanged, as it travels source to destination**
- **addr fields of interest here**

routing table in A

Dest. Net.	next router	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2



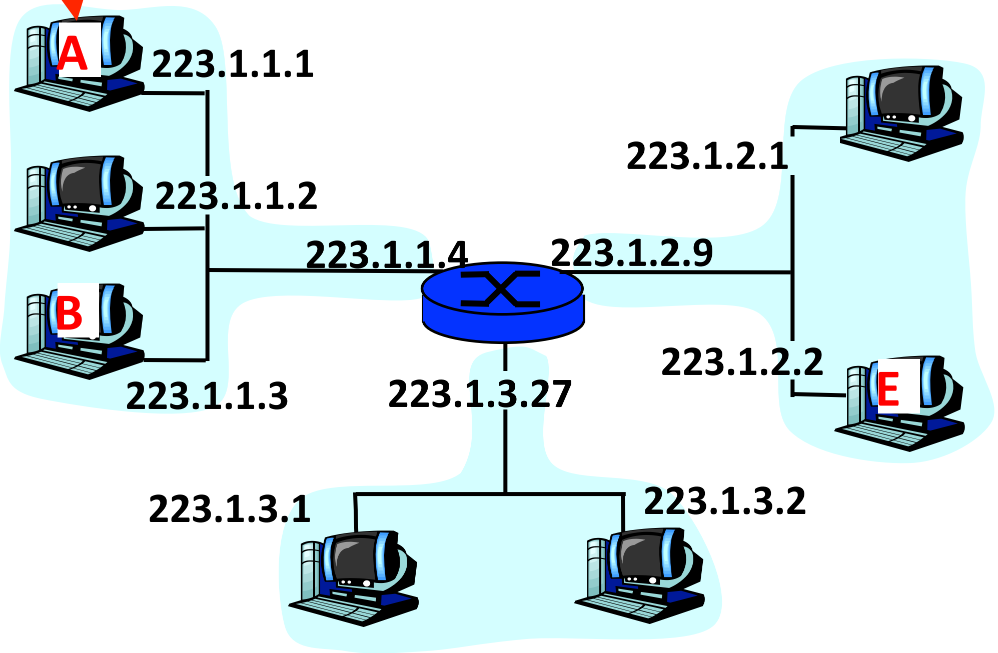
Getting a datagram from source to dest.

misc fields	223.1.1.1	223.1.1.3	data

Dest. Net.	next router	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2

Starting at A, given IP datagram addressed to B:

- look up net. address of B
- find B is on same net. as A
- link layer will send datagram directly to B inside link-layer frame
  - B and A are directly connected



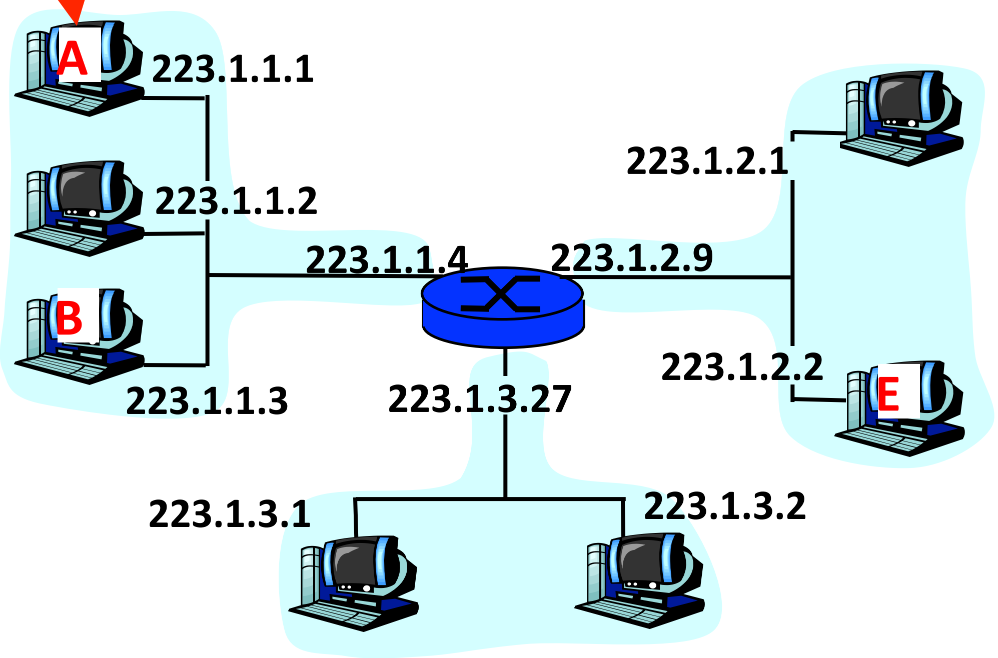
Getting a datagram from source to dest.

misc fields	223.1.1.1	223.1.2.2	data

Starting at A, dest. E:

- look up network address of E
- E on *different* network
  - A, E not directly attached
- routing table: next hop router to E is 223.1.1.4
- link layer sends datagram to router 223.1.1.4 inside link-layer frame
- datagram arrives at 223.1.1.4
- continued…..

Dest. Net.	next router	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2



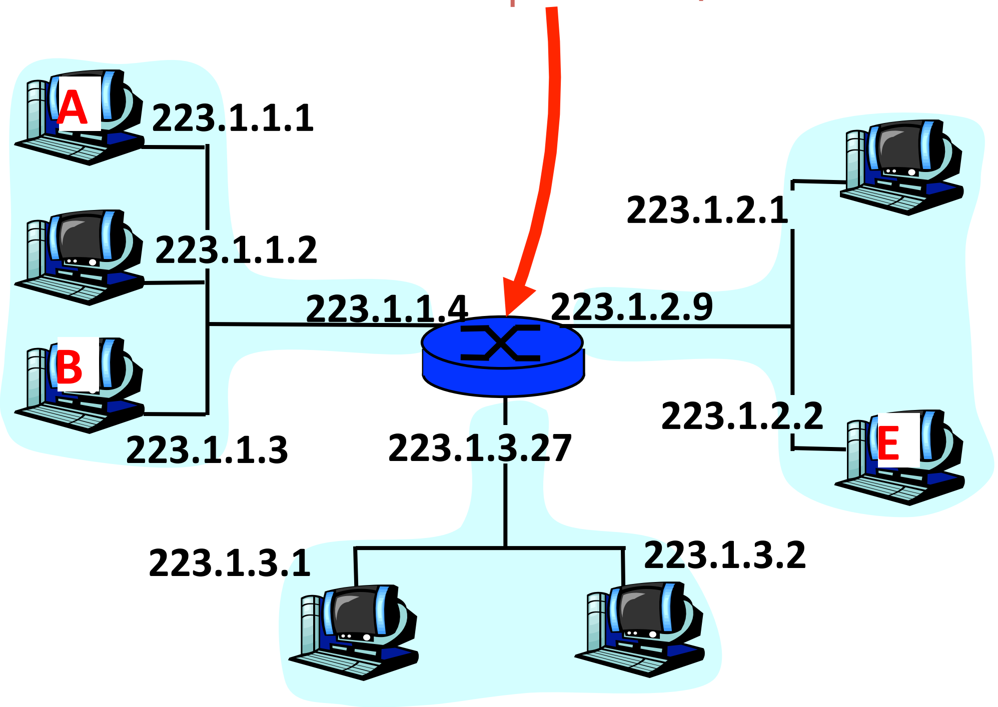
Getting a datagram from source to dest.

misc fields	223.1.1.1	223.1.2.2	data

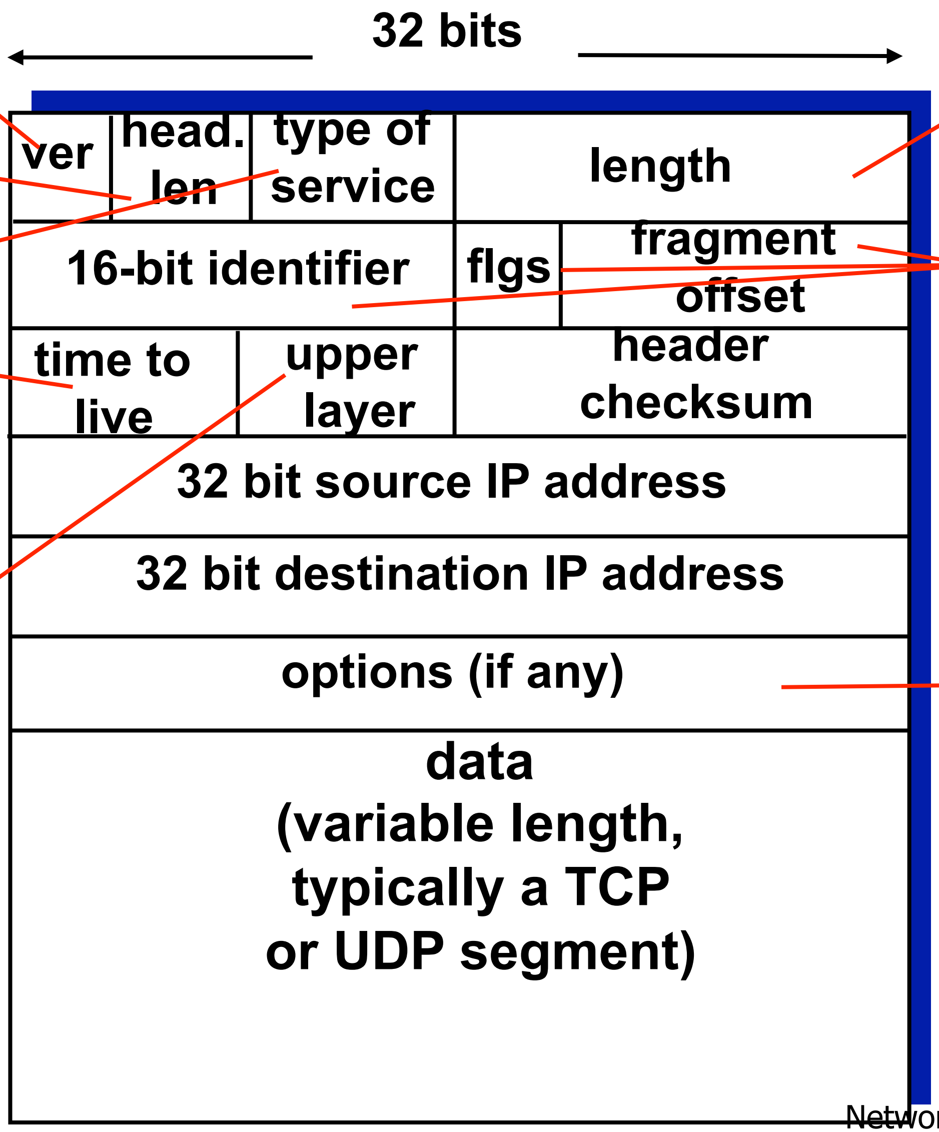
Dest. network	next router	Nhops	interface
223.1.1	-	1	223.1.1.4
223.1.2	-	1	223.1.2.9
223.1.3	-	1	223.1.3.27

Arriving at 223.1.4, destined for 223.1.2.2

- look up network address of E
- E on *same* network as router's interface 223.1.2.9
  - router, E directly attached
- link layer sends datagram to 223.1.2.2 inside link-layer frame via interface 223.1.2.9
- datagram arrives at 223.1.2.2!!! (hooray!)



IP datagram format



32 bits

ver | head. len | type of service | length

16-bit identifier | flgs | fragment offset

time to live | upper layer | header checksum

32 bit source IP address

32 bit destination IP address

options (if any)

data (variable length, typically a TCP or UDP segment)

IP protocol version number

header length (bytes)

"type" of data

max number remaining hops (decremented at each router)

upper layer protocol to deliver payload to

total datagram length (bytes)

for fragmentation/ reassembly

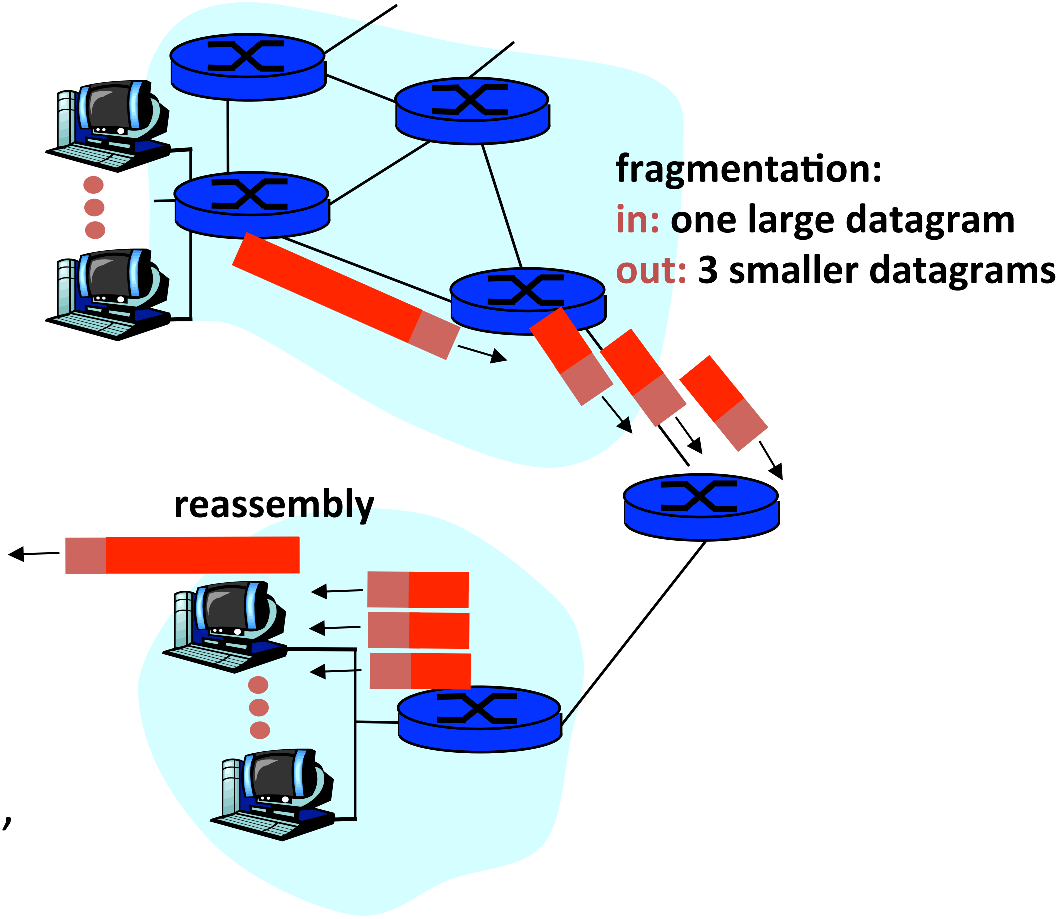
e.g. timestamp, record route taken, specify list of routers to visit.

how much overhead?

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

IP Fragmentation & Reassembly

- network links have MTU (max.transfer size) - largest possible link-level frame.
  - different link types, different MTUs
- large IP datagram divided (“fragmented”) within net
  - one datagram becomes several datagrams
  - “reassembled” only at final destination
  - IP header bits used to identify, order related fragments



IPv6

- IP was designed in the 1970's to support researchers and the military. The global use today wasn't predicted.
- Early design decisions led to predictions in the early 1990's of the "death of the Internet" due to
  - Shortage of Internet protocol (IP) addresses
  - Routing table explosion
  - Shortage of network numbers
- New capabilities needed:
  - Improved security (or even *some* security!)
  - Support for QoS (real-time services)
  - Better multicasting support
  - Mobile computing
- Two paths could be taken
  - Retrofit larger addresses and functionality onto IPv4
  - Develop a new version of IP

- **Early proposals**
  - **CLNP (Connectionless Network Protocol)**
  - **SIP (Simple IP)**
  - **Pip(Paul's Internet protocol)**
  - **SIPP (SIP plus Pip) became IPv6 or IPng**

Expanding the address space

- By 1996, all of the class A networks, 62% of class B and 37% of class C networks were allocated. Predicted to run out of addresses in 2008.
- The temporary solution is Classless InterDomain Routing (CIDR)
- With the push towards IP enabled devices this is likely to be only a temporary patch. IPv6 aims for a more permanent solution.
- How big is "big enough" for the address space?
  - Current 32 bit space allows *billions* of addresses.
  - Inefficiencies of allocation are almost inevitable.
  - May want to identify experimental networks in a different address spaces.
  - Can we predict future use? Are we setting ourselves up to need IPv7?
- Should the address be variable length?

IPv6 addressing

- 128 bits was a compromise
  - Fixed addresses are easier to manage and program.
- There is nearly 1 IP address for every molecule on the earth’s surface.
  - Even with ***in***efficient allocation there should be at least 1000 addresses per square metre.
- 128 bit addresses are written in hex: x:x:x:x:x:x:x:x
  - Each x is 16-bits = 4 hex digits
  - Leading zeros are not required
  - Sequence of zero fields given by "::”
- Examples:

  1080:0:0:0:8:800:200C:417A = 1080::8:800:200C:417A

  FF01:0:0:0:0:0:0:43 = FF01::43

  0:0:0:0:0:0:0:1 = ::1

IPv6 Routing

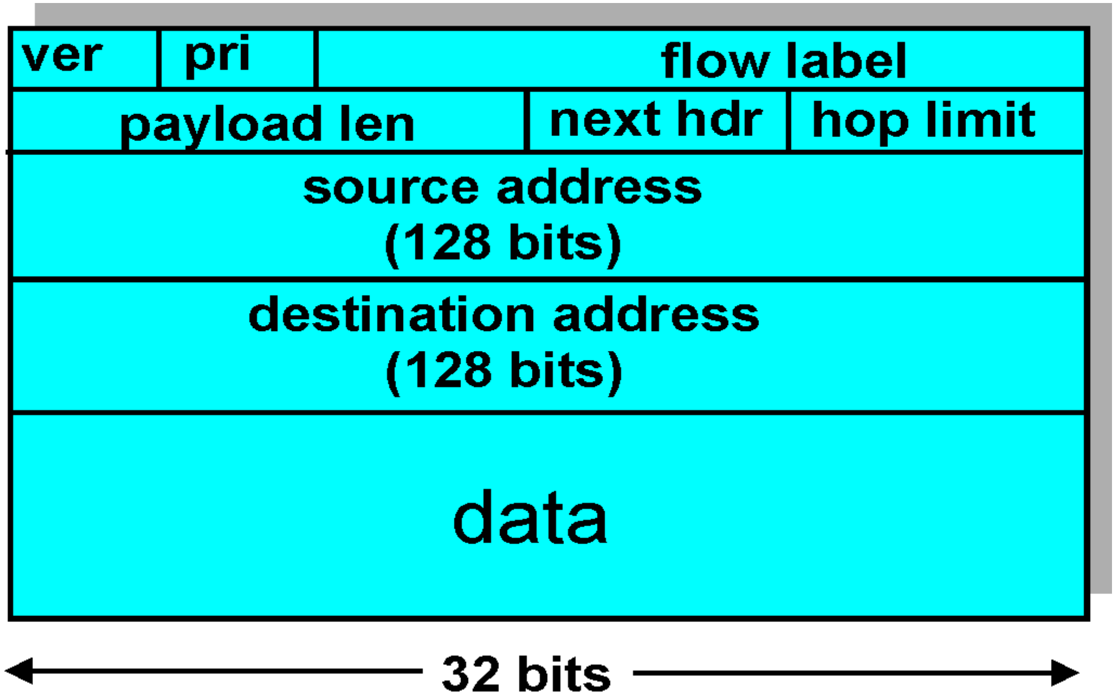
- Each interface has an address (but possibly more than one):
  - Unicast – identify single interface
  - Anycast (or "cluster")– identify one out of number of interfaces ie. any one will do (eg. nearest host)
  - Multicast – identify set of interfaces, all of which are to receive message
- Intention in IPv6 addresses is to support ***hierarchical routing***
  → prefixes will **identify** registries, providers etc.
- Example: hierarchical organisation:

s	n	m	128-s-n-m
subscriber prefix	area id	subnet id	interface id

- **This will help routing tables**
- Other possibilities:
  - global provider-based unicast address
  - geographic-based unicast address

IPv6 Header

- IPv6 8 fields in base header vs 13 fields in IPv4
  - Faster processing
  - Simpler management
  - More flexibility



ver | pri | flow label

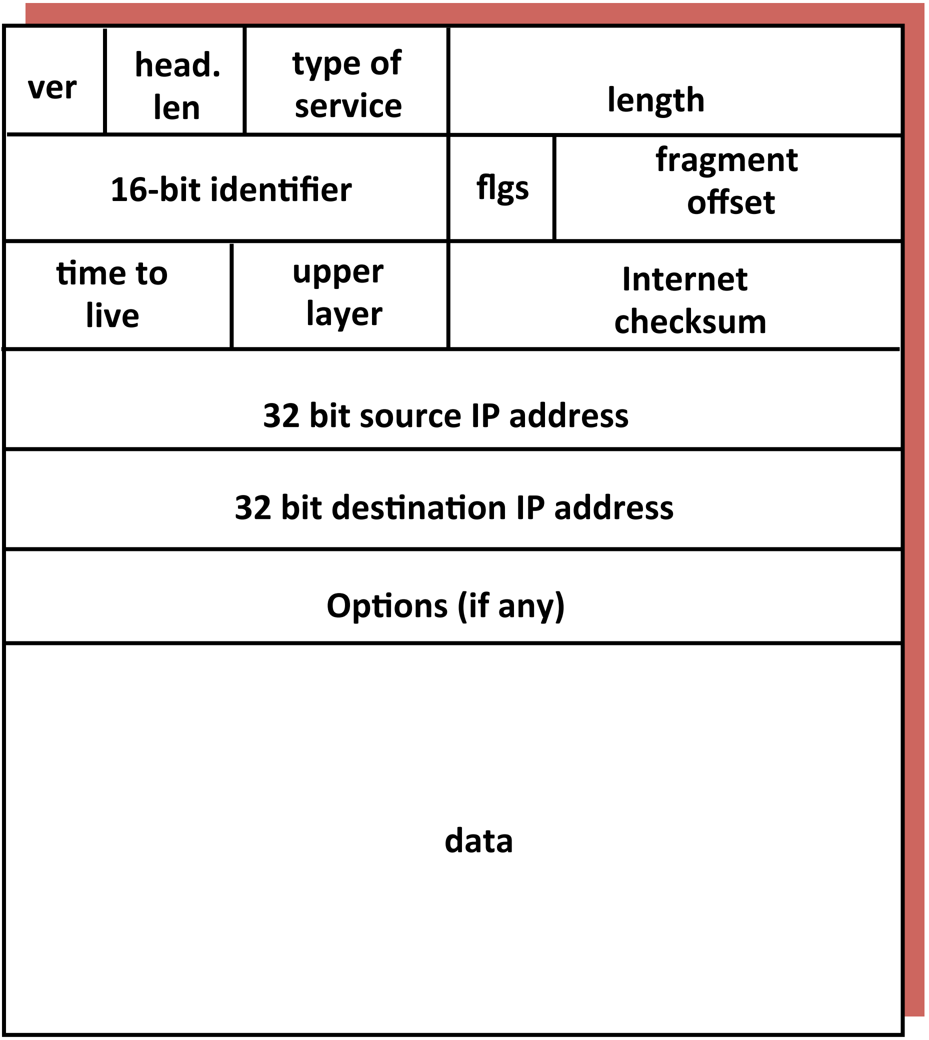
payload len | next hdr | hop limit

source address (128 bits)

destination address (128 bits)

data

32 bits



ver | head. len | type of service | length

16-bit identifier | flgs | fragment offset

time to live | upper layer | Internet checksum

32 bit source IP address

32 bit destination IP address

Options (if any)

data

IPv4

Extension Headers

- This base header is followed by a number of optional ***extension headers***
  - Still allows flexibility
- Each header specifies code of next header/data component
- Extension headers commonly specify:
  - type of header and length
  - response if the router can’t process the header — ignore the header, skip the packet, skip the packet and report an error

Extension Header	Description
Hop-by-hop Options	Misc. information for routers
Routing	Full, or partial, route to follow
Fragmentation	Management of datagram fragments
Authorisation	Verification of Sender’ s Identity
Encrypted Security Payload	Information about encrypted Contents
Destination Options	Additional Info for destination

Fragmentation

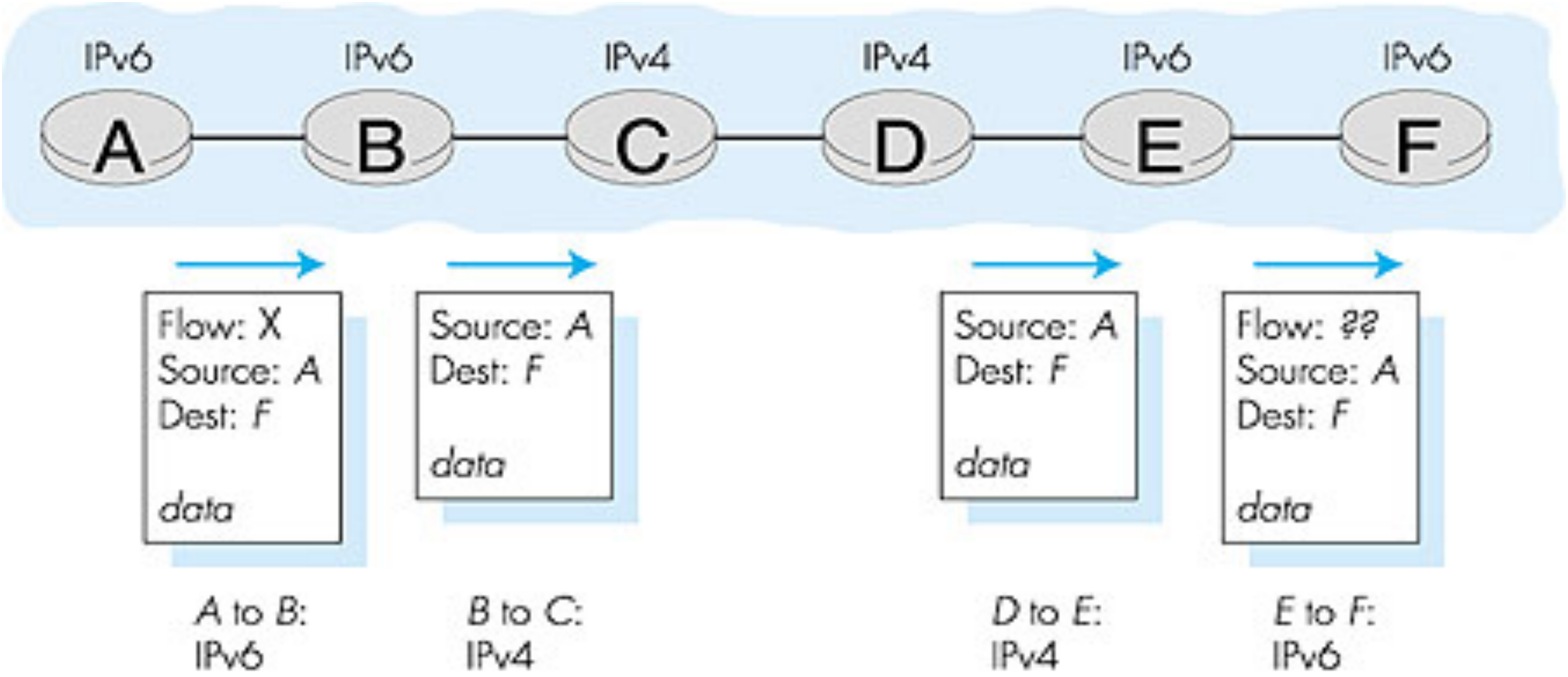
- Fragmentation information is no longer in every header, but only in special extension headers
- Fragmentation is no longer performed at intermediate routers
  - The source host should choose datagram size so fragmentation is not necessary
  - Source host needs to run "path MTU discovery”

  e.g. send sequence of datagram sizes to target until they don't arrive

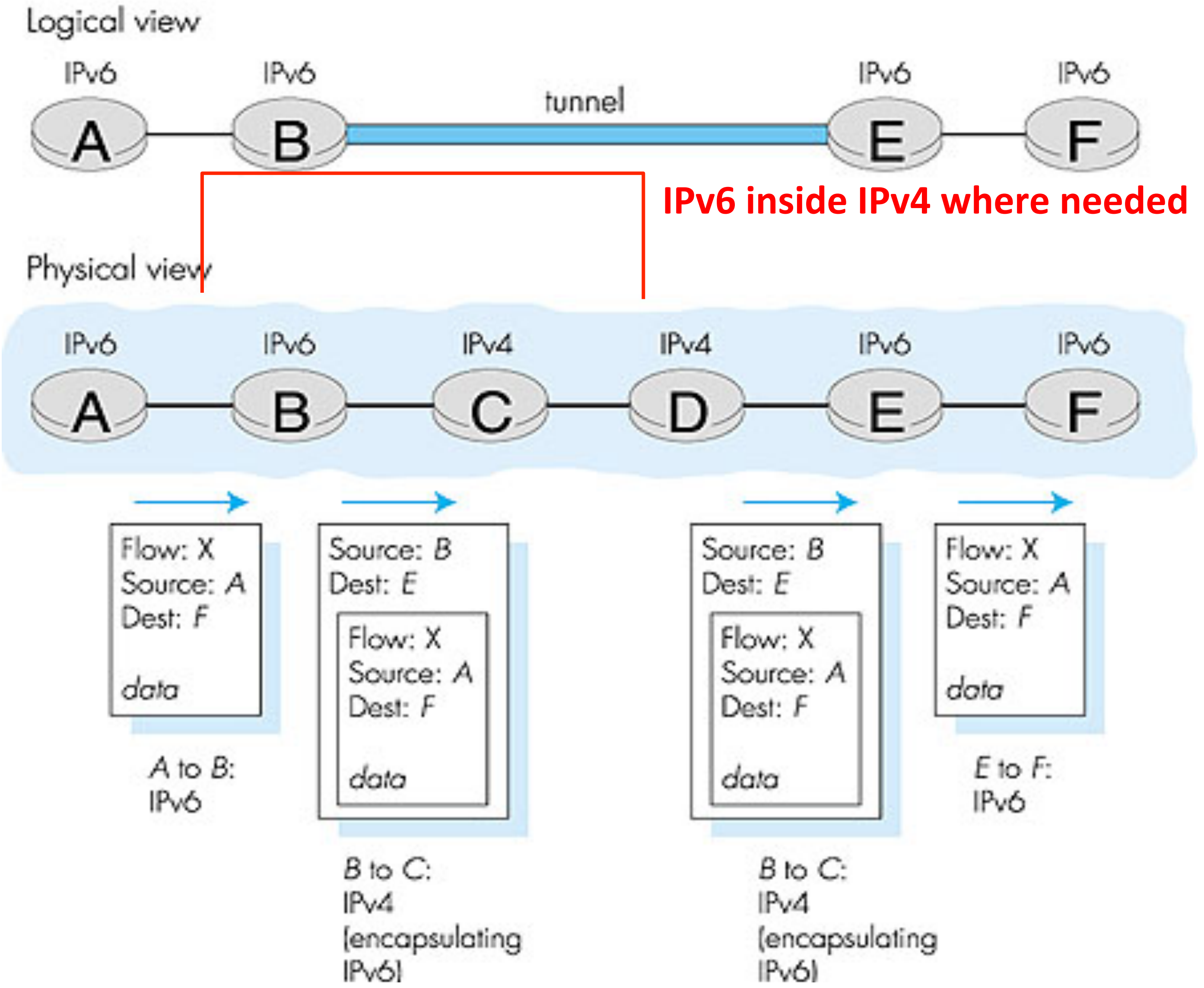
Transition From IPv4 To IPv6

- Not all routers can be upgraded simultaneously
  - no “flag days”
  - How will the network operate with mixed IPv4 and IPv6 routers?
- Two proposed approaches:
  - *Dual Stack*: some routers with dual stack (v6, v4) can “translate” between formats
  - *Tunneling:* IPv6 carried as payload in IPv4 datagram among IPv4 routers
- Changes in networking technologies *always* proceed by evolution, rather than revolution

Dual Stack Approach



Tunneling



IPv6 inside IPv4 where needed

Tunneling

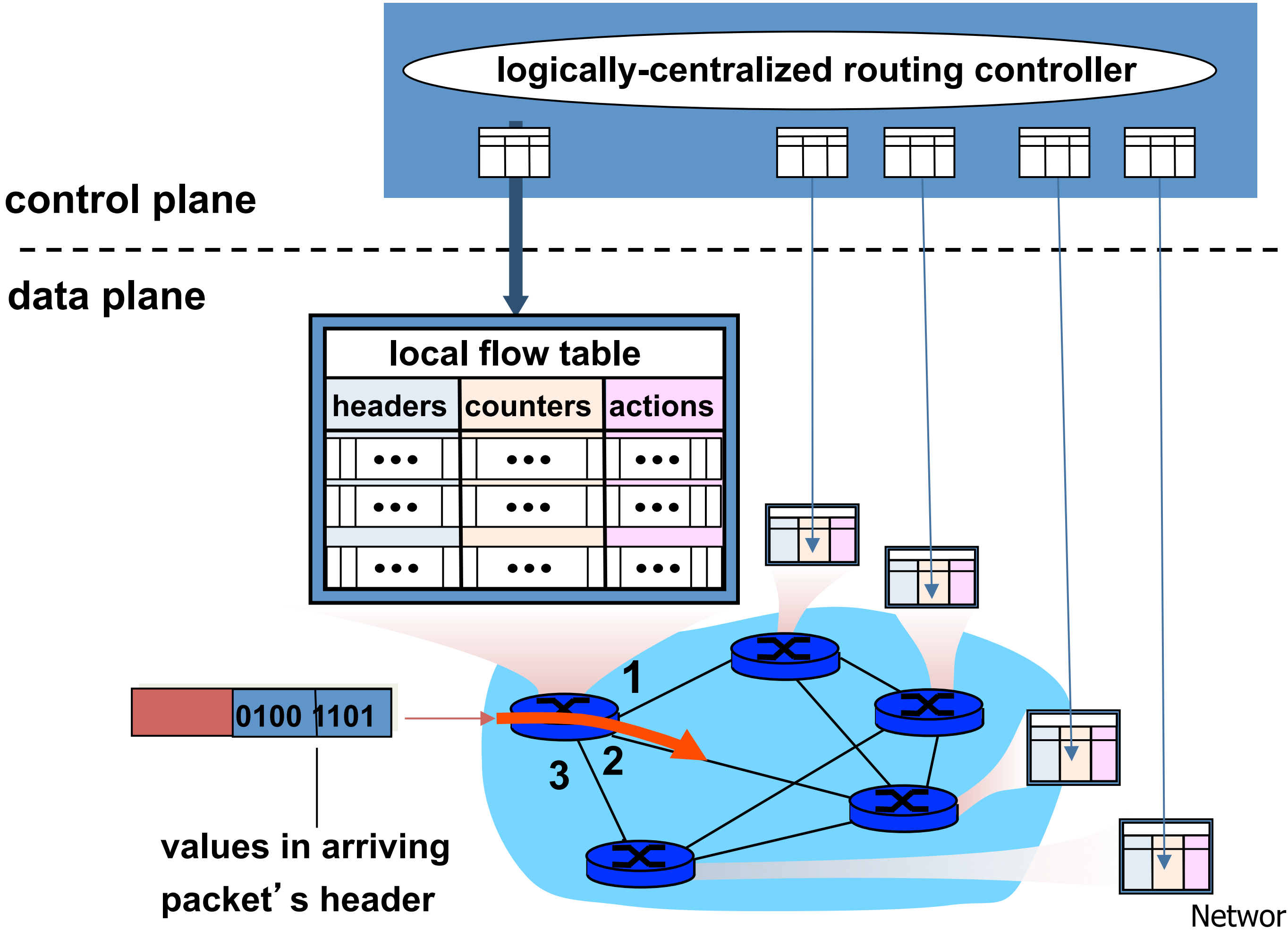
- Tunneling is a generically useful technique in networking…
  - Note that IPv6 payloads could be anything!
  - We could tunnel arbitrary protocols
- This is how experimentation is done –
  - New protocols are tunneled across existing infrastructure
  - Can spread new protocols across the network this way
  - Can link in “new” devices or “old” devices this way
  - Can implement secure networks this way (see later…)

IPv6 Summary

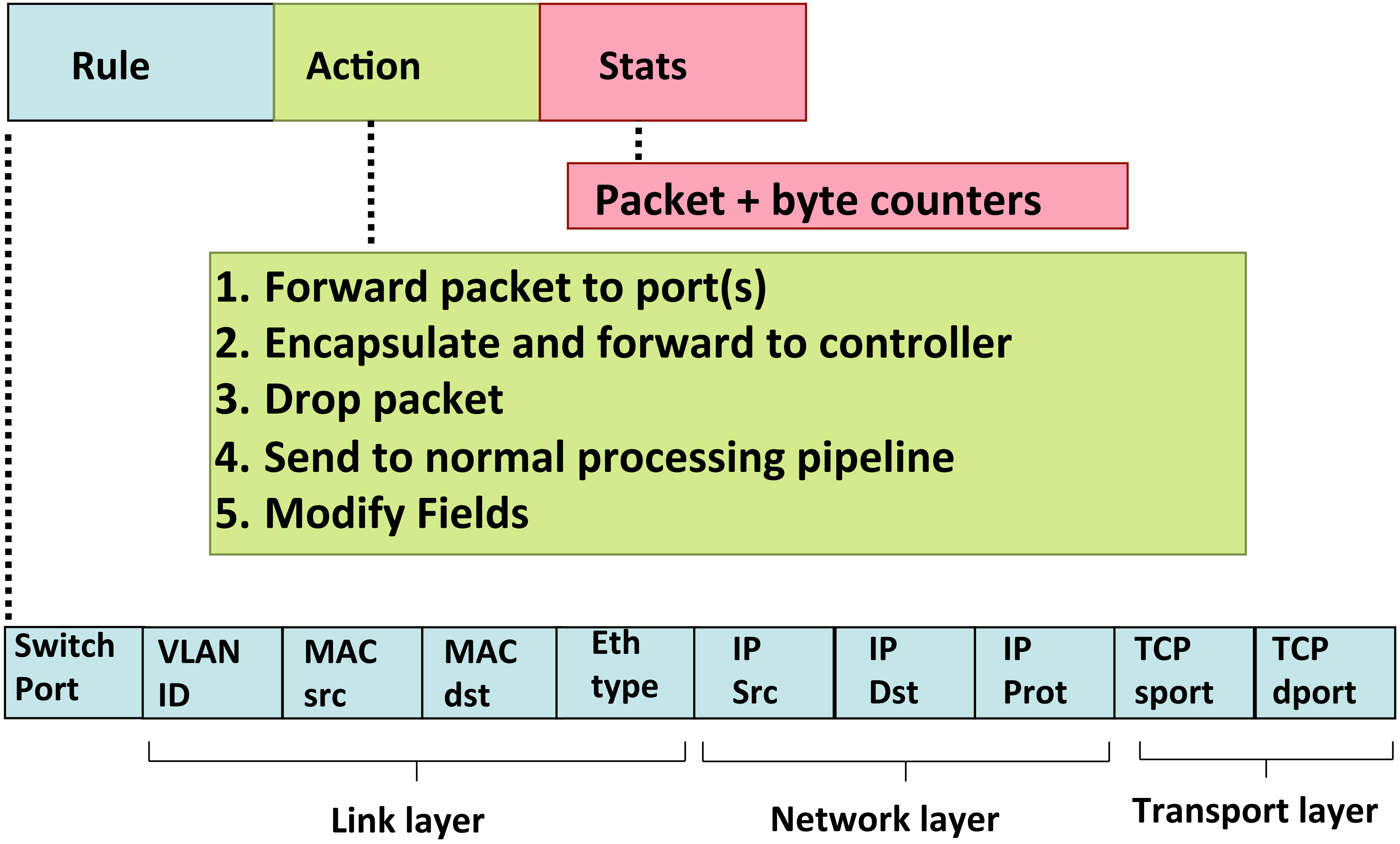
- What has changed
  - Simpler, fixed length header
    - No fragmentation in routers
    - Options in extension headers
    - No checksum
  - 128 bit (IPv4 32 bit) addresses, *with hierarchy*
  - Additional support for
    - Multicast and anycast routing
    - Security
    - Mobile hosts and autoconfiguration
    - Real Time applications

Generalized Forwarding and SDN

Each router contains a *flow table* that is computed and distributed by a *logically centralized* routing controller



OpenFlow: Flow Table Entries



Rule	Action	Stats

Packet + byte counters

1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline
5. Modify Fields

Switch Port	VLAN ID	MAC src	MAC dst	Eth type	IP Src	IP Dst	IP Prot	TCP sport	TCP dport

	Link layer				Network layer			Transport layer	

Examples

Destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	51.6.0.8	*	*	*	port6

IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6

Firewall:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	*	*	*	*	22	drop

do not forward (block) all datagrams destined to TCP port 22

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	128.119.1.1	*	*	*	*	drop

do not forward (block) all datagrams sent by host 128.119.1.1