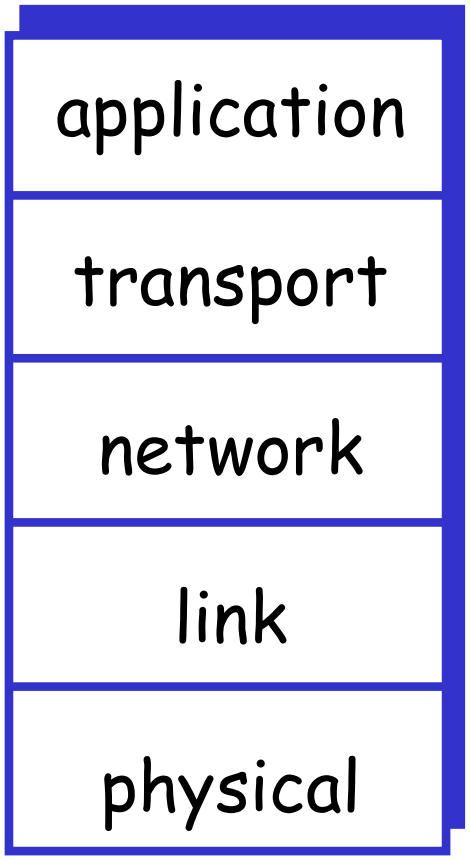
Internet Protocol Stack

- Application: supporting network applications
  - FTP, SMTP, HTTP
- Transport: data transfer between processes
  - TCP, UDP
- Network: routing of datagrams from source to destination
  - IP, routing protocols
- Link: data transfer between neighboring network elements
  - Ethernet, WiFi
- Physical: bits "on the wire"
  - Coaxial cable, optical fibers, radios



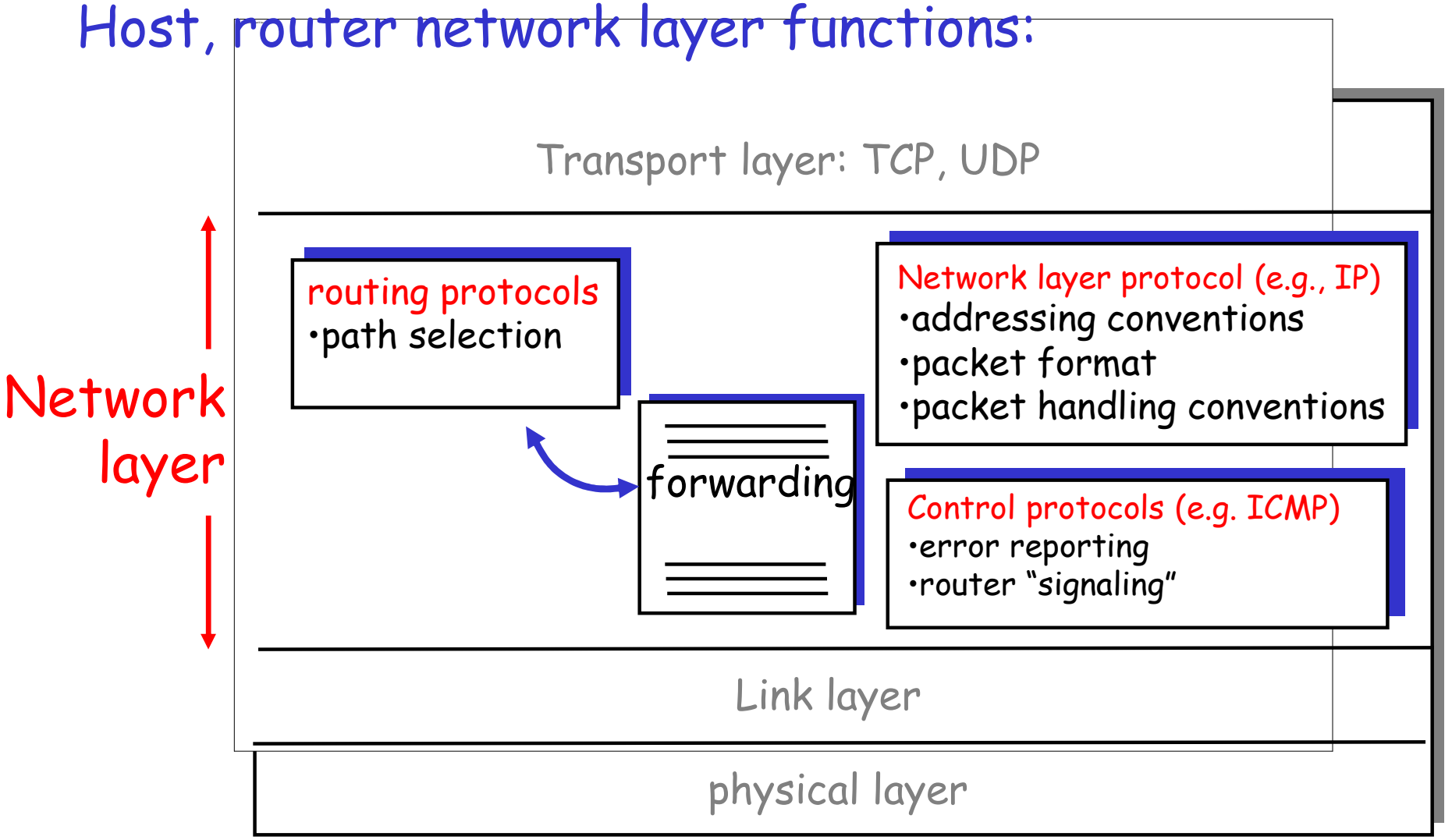
Network Layer and Mobile IP

Outline

- IP addresses
- Virtual circuit vs. datagrams
- Routing algorithms
  - Link state
  - Distance vector
- Mobile IP
  - Architecture
- Encapsulation

Network Layer in Internet: Big Picture

Host, router network layer functions:

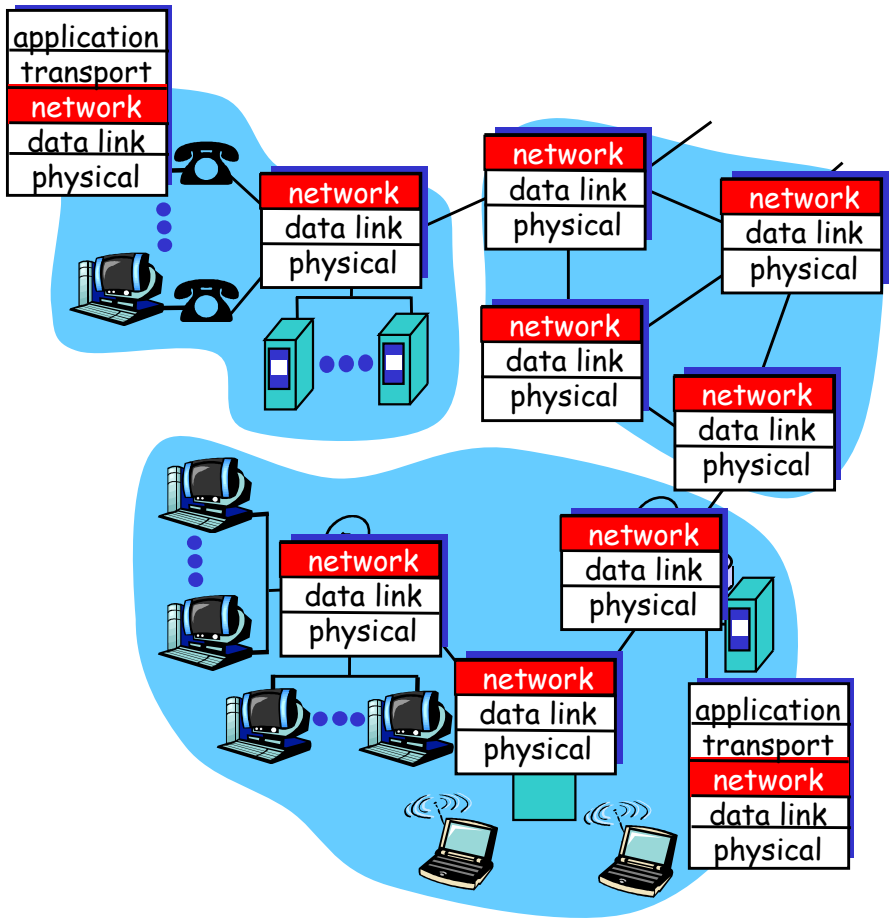


Network layer functions

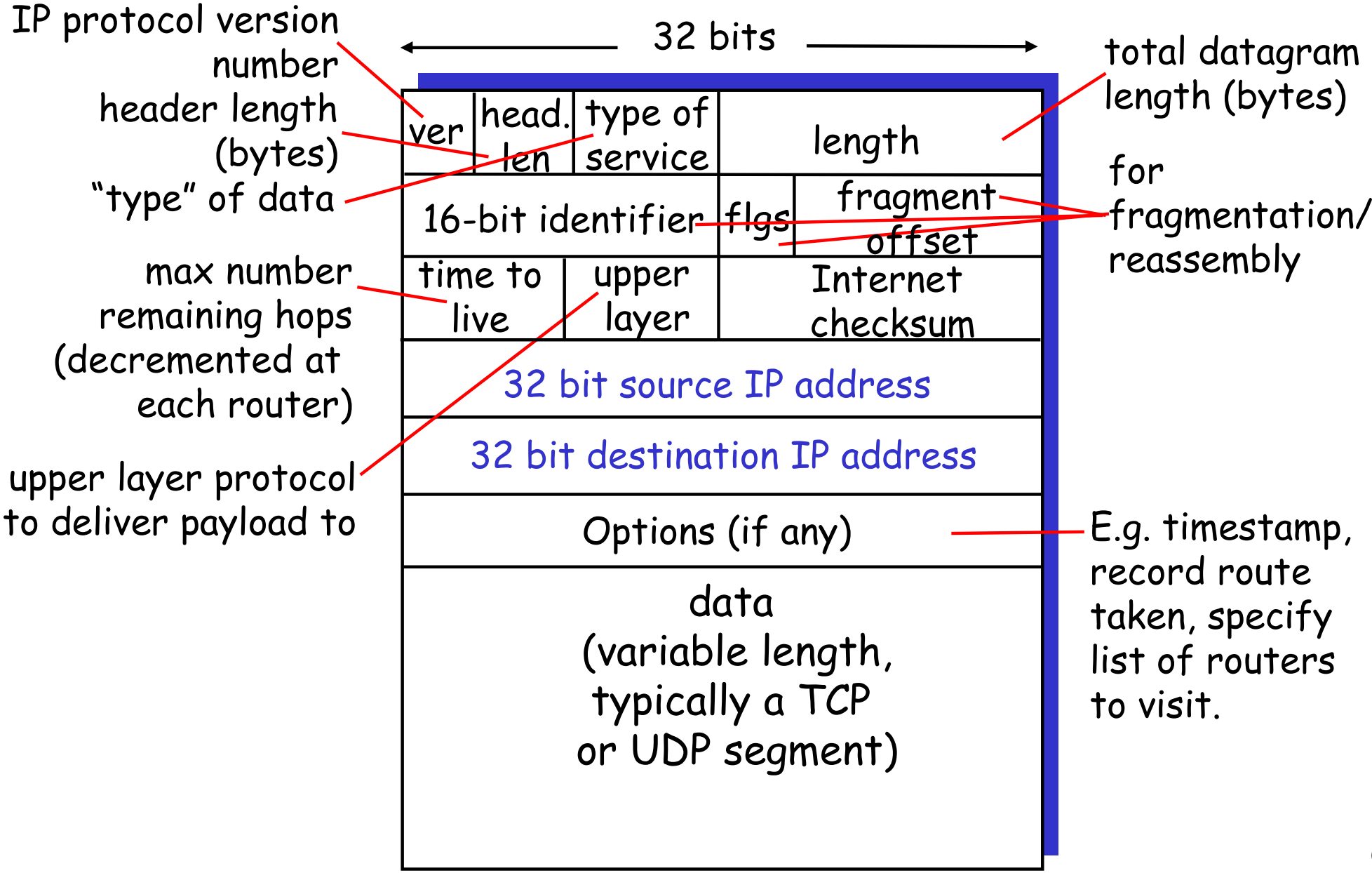
- transport packet from sending to receiving hosts
- network layer protocols in *every* host, router

three important functions:

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

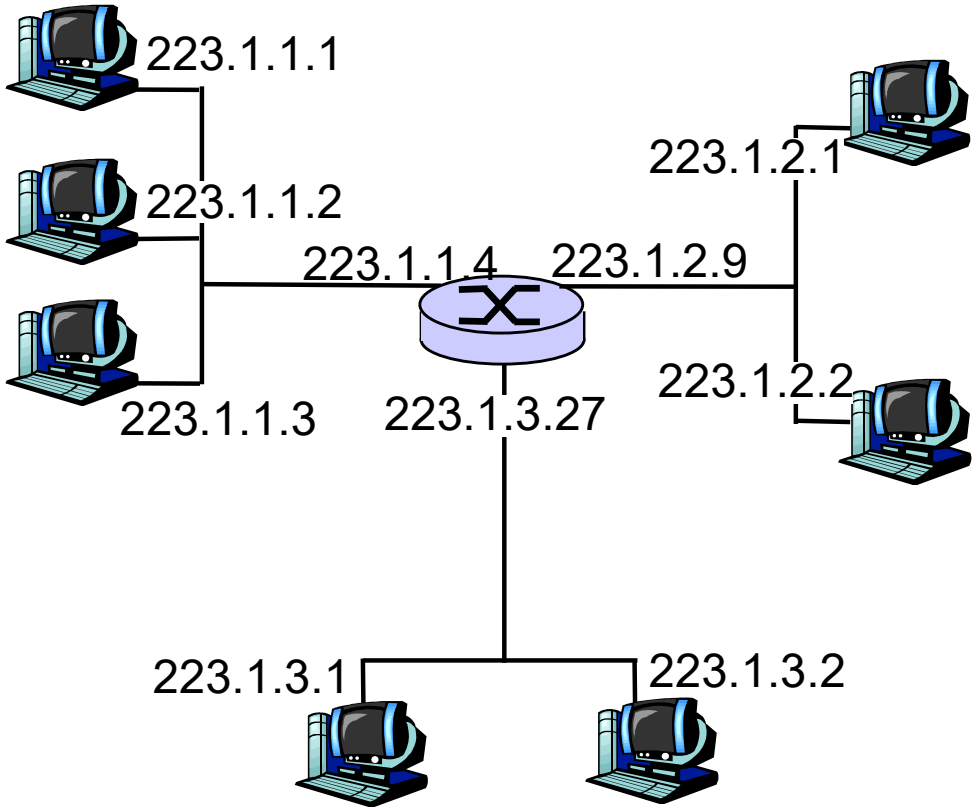


IP Datagram Format



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

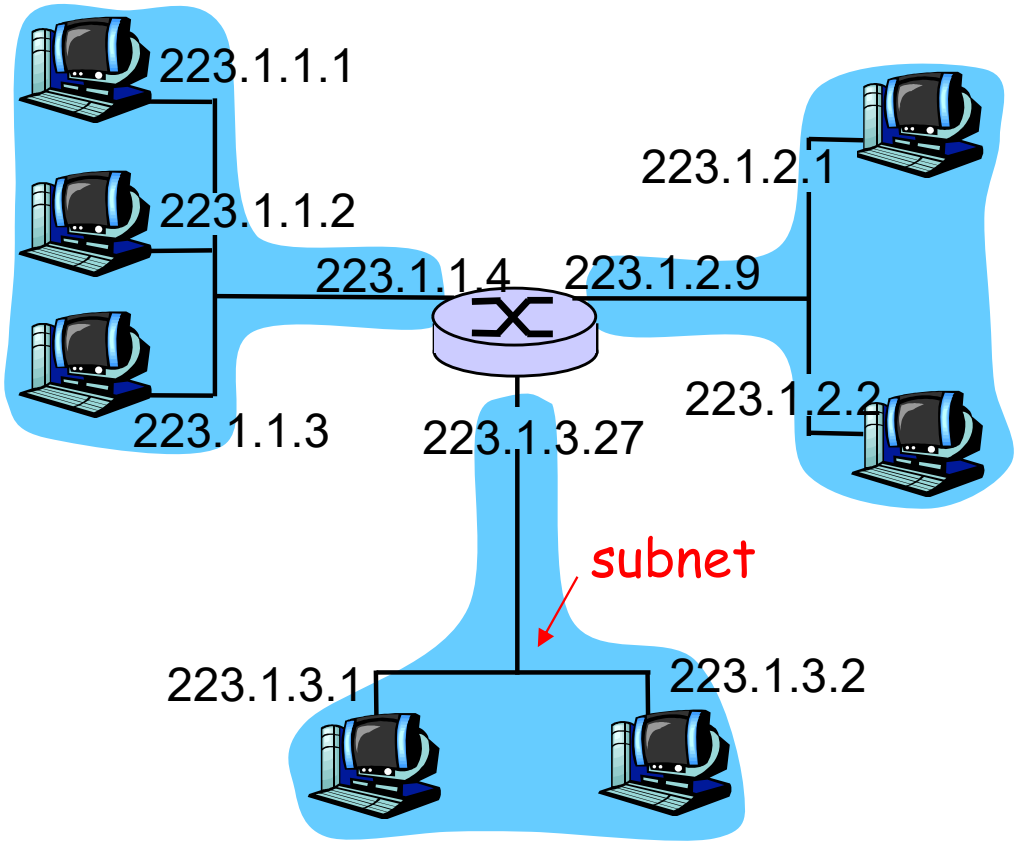


223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

Subnets

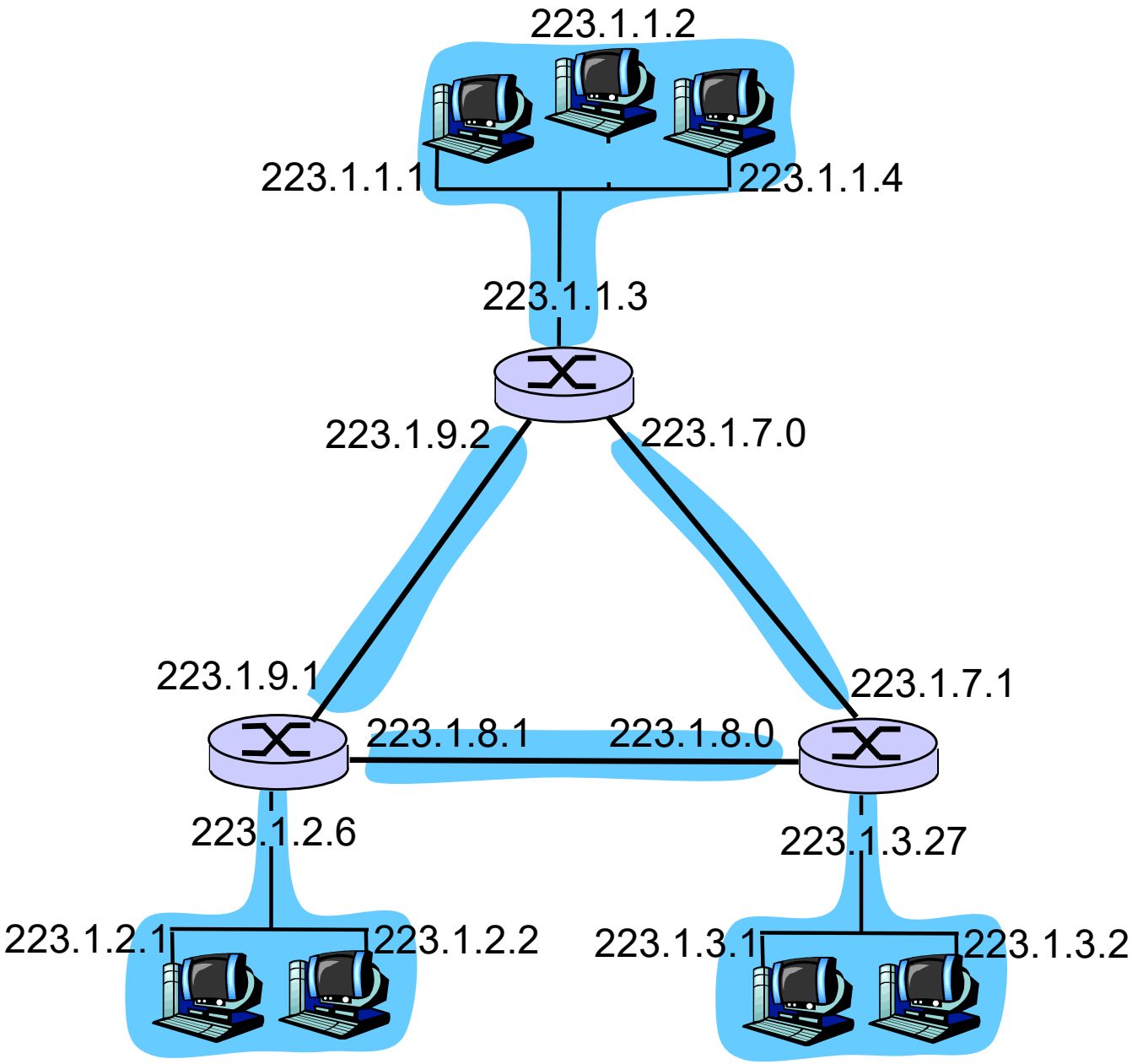
- IP address:
  - subnet part (high order bits)
  - host part (low order bits)
- *What's a subnet?*
  - device interfaces with same subnet part of IP address
  - can physically reach each other without intervening router



network consisting of 3 subnets

Subnets

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

IP addresses: how to get one?

Q: How does *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 server

IP addresses: how to get one?

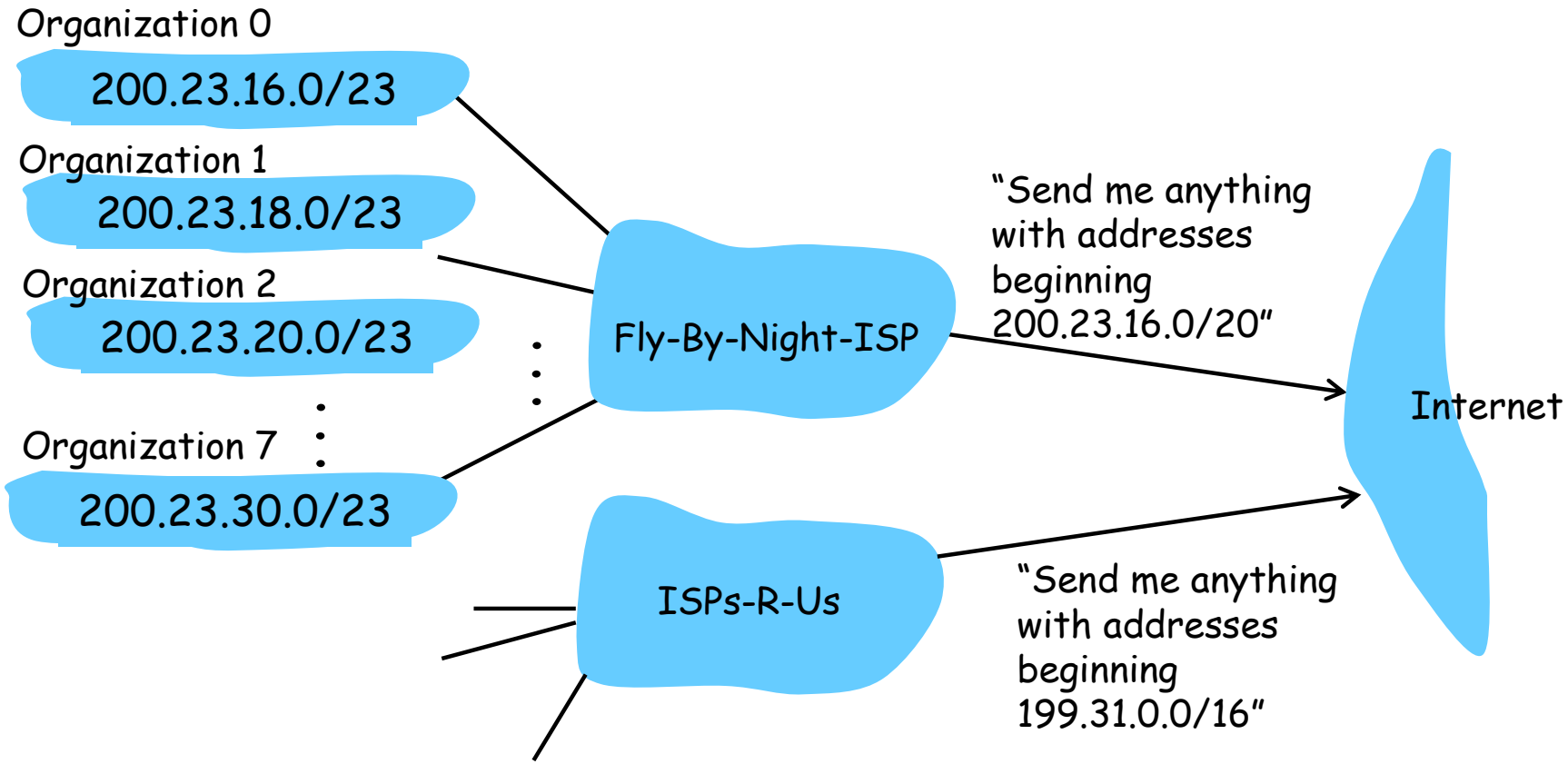
Q: How does *network* get subnet part of IP addr?

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

ISP's block	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

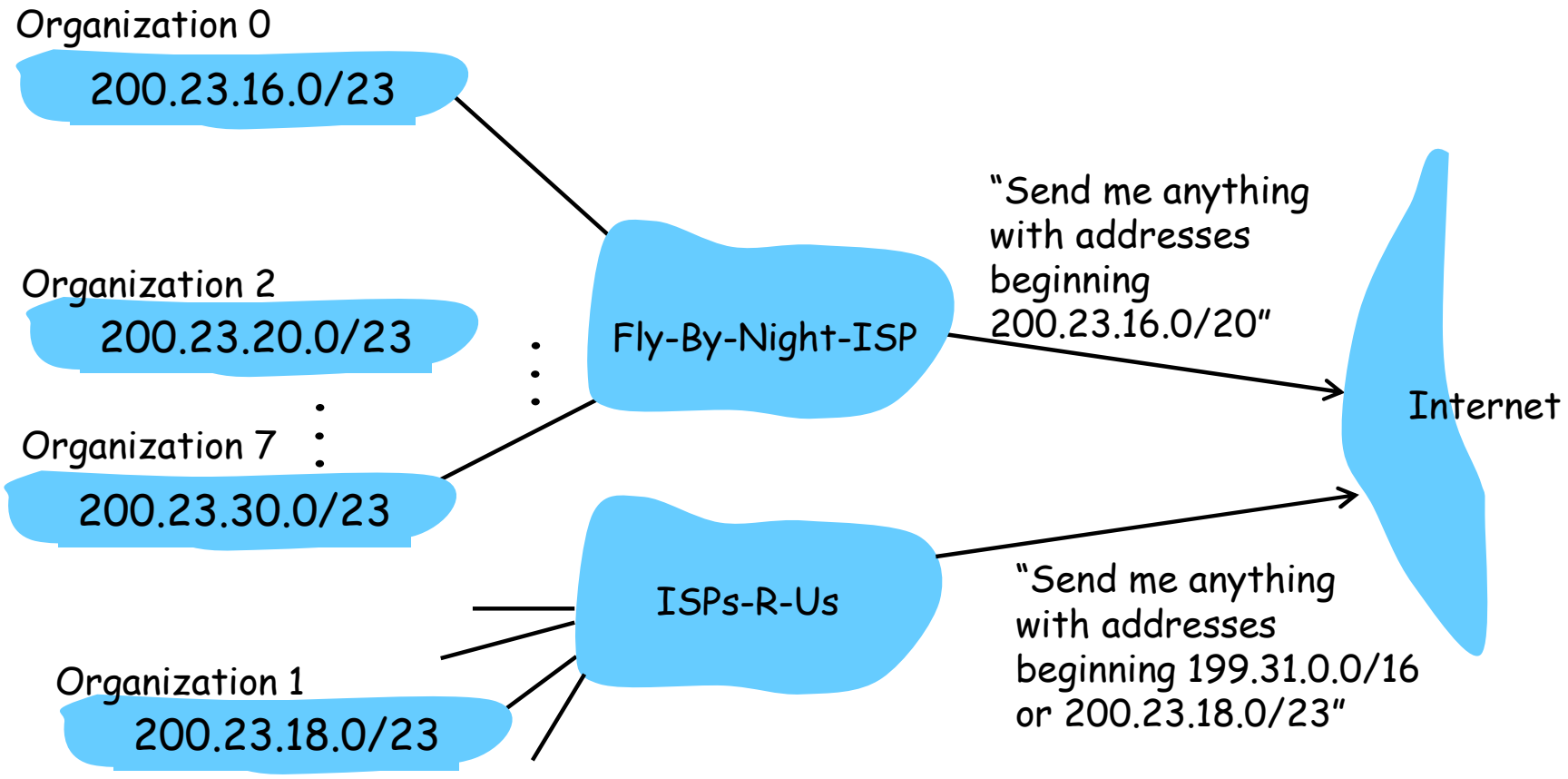
Hierarchical addressing: route aggregation

Hierarchical addressing allows efficient advertisement of routing information:



Hierarchical addressing: more specific routes

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



IP addressing: the last word...

Q: How does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers

- allocates addresses
- manages DNS
- assigns domain names, resolves disputes

Network service model

CRUCIAL question!

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

service abstraction

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

The most important abstraction provided by network layer:

virtual circuit or datagram?

Virtual circuits

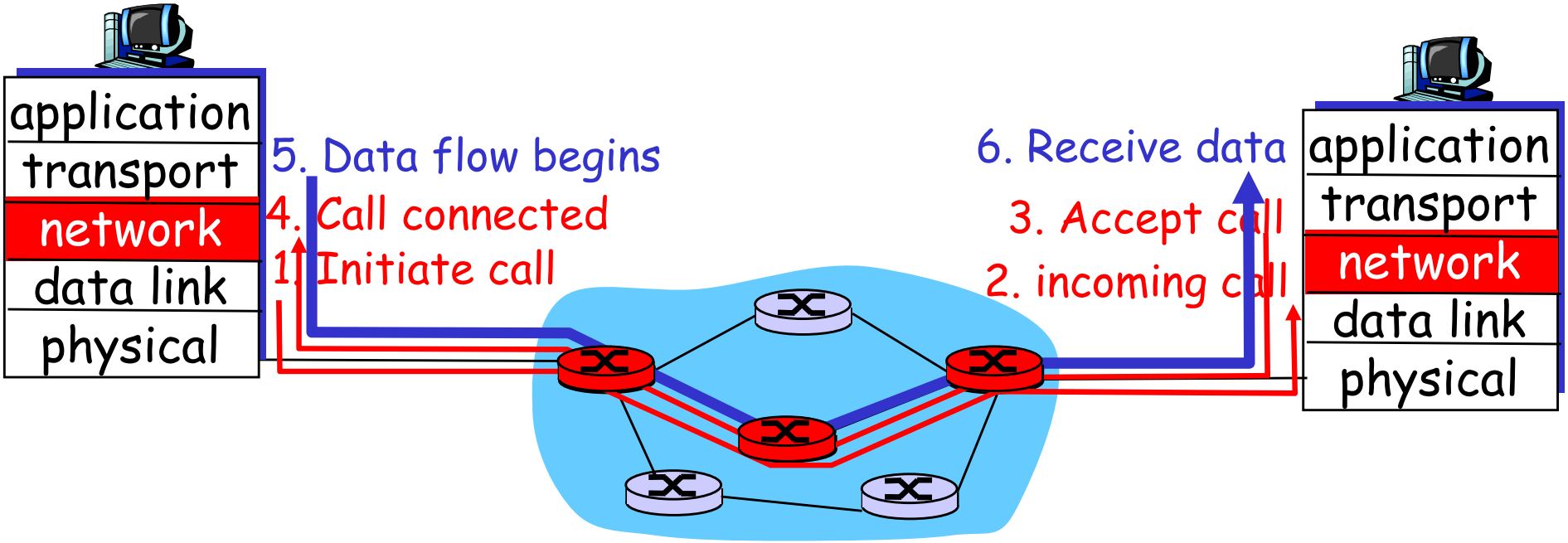
"source-to-dest path behaves much like telephone circuit"

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

- call setup, teardown for each call *before* data can flow
- each packet carries VC identifier (not destination host ID)
- *every* router on source-dest path maintains "state" for each passing connection
  - transport-layer connection only involved two end systems
- link, router resources (bandwidth, buffers) may be *allocated* to VC
  - to get circuit-like perf.

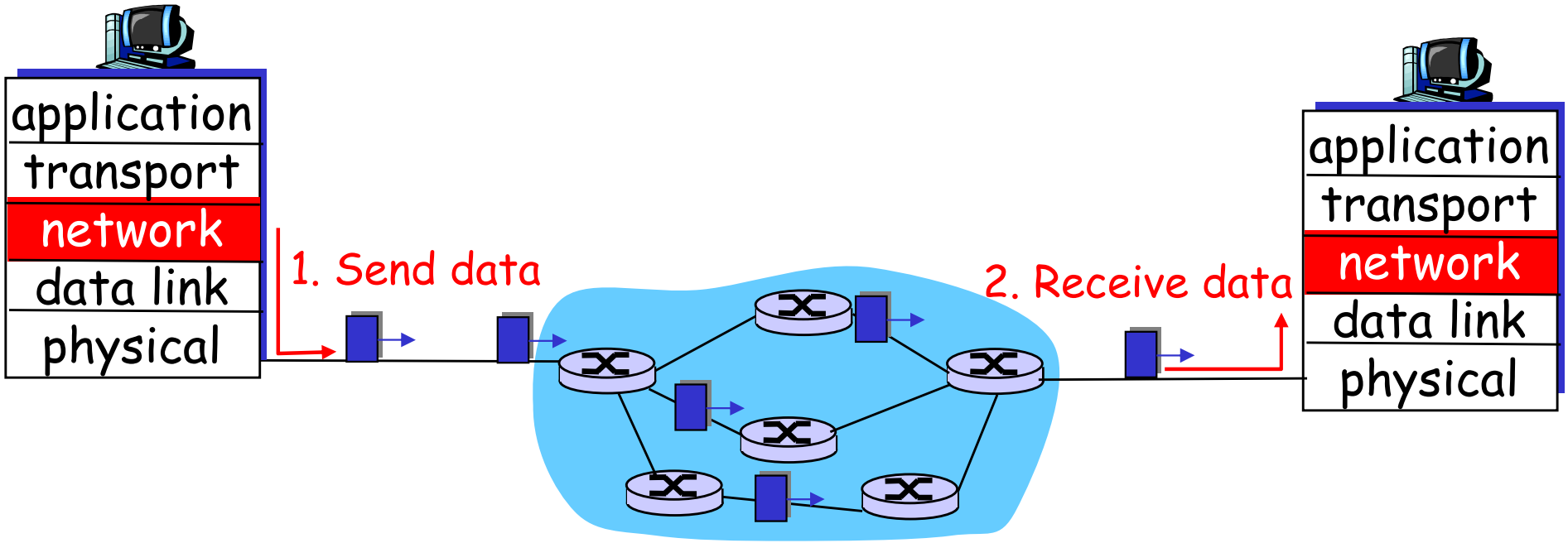
Virtual circuits: signaling protocols

- used to setup, maintain teardown VC
- used in ATM, frame-relay, X.25
- not used in today's Internet
- Pros and cons?



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

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

ATM

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

Outline

- IP addresses
- Virtual circuit vs. datagrams
- Routing algorithms
  - Link state
  - Distance vector
- Mobile IP
  - Architecture
- Encapsulation

IPv4

- **Definition:** Internet Protocol version 4, the primary protocol for communication on the Internet since 1983.
- **Structure:** Uses a **32-bit** address space.
- **Format:** Dotted-decimal (e.g., 192.168.1.1).
- **Capacity:** Approximately **4.3 billion** unique addresses.
- **The Bottleneck:** With the explosion of smartphones, IoT, and always-on devices, we officially "ran out" of unallocated IPv4 addresses in 2011.

IPv6

- **Definition:** Internet Protocol version 6, designed to replace IPv4.
- **Structure:** Uses a **128-bit** address space.
- **Format:** Hexadecimal colon-separated (e.g., 2001:0db8:85a3:0000:0000:8a2e:0370:7334).
- **Capacity:** Roughly **340 undecillion** addresses (3.4×10^{38}).
    - *Analogy:* Enough for every grain of sand on Earth to have trillions of IP addresses.

NAT

- **NAT (Network Address Translation):** A method of remapping one IP address space into another.
- **How it works:** Allows a single public IPv4 address to represent an entire private network (like your home Wi-Fi).
- **Core Types:**
  - **Static NAT:** One-to-one mapping (server to public IP).
  - **Dynamic NAT:** One-to-many from a pool of addresses.
- **PAT (Port Address Translation):** Maps multiple private IPs to one public IP using unique **port numbers**.

NAT: Pros and Cons

The Good

- **Address Conservation:** Significantly delayed the total collapse of IPv4.
- **Security:** Hides internal network IP schemes from the public internet.

The Bad

- **Breaks "End-to-End" Connectivity:** Direct peer-to-peer communication (like VoIP or gaming) becomes difficult without "hole punching" (STUN/TURN).
- **Latency:** Every packet must be modified by the router, adding processing overhead.
- **Layer Violation:** NAT must look at transport layer ports, which violates strict networking layering principles.

Co-existence and Transition

We are currently in a "Dual-Stack" era where both protocols live together.

Dual-Stack: Devices run both IPv4 and IPv6 simultaneously.

Tunneling: Wrapping IPv6 packets inside IPv4 packets to travel across older infrastructure.

NAT64: A translation mechanism that allows IPv6-only devices to communicate with IPv4-only resources.

Summary

- **IPv4** is the foundation but is hindered by its small address pool.
- **NAT** is a necessary workaround that keeps IPv4 alive but complicates connectivity.
- **IPv6** is the ultimate solution, offering infinite scale, better security, and simplified routing.

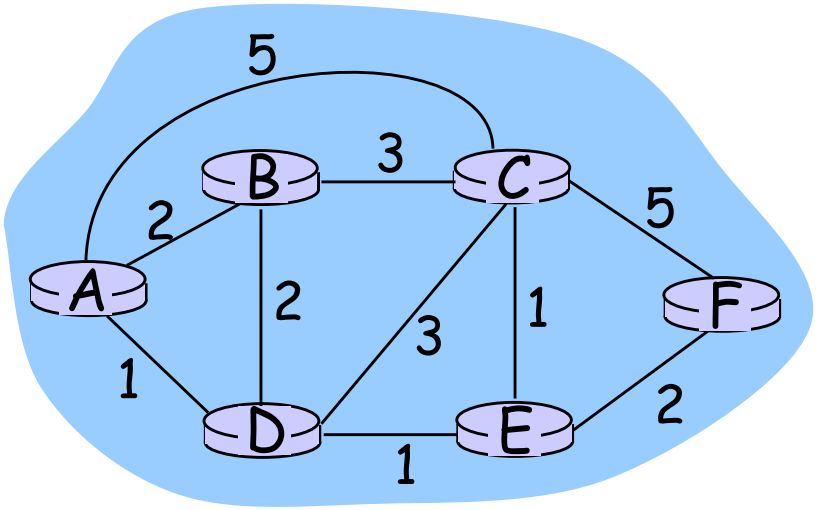
Outline

- IP addresses
- Virtual circuit vs. datagrams
- Routing algorithms
  - Link state
  - Distance vector
- Mobile IP
  - Architecture
- Encapsulation

Routing

Routing protocol

Goal: determine "good" path (sequence of routers) thru network from source to dest.



Graph abstraction for routing algorithms:

- graph nodes are routers
- graph edges are physical links
  - What are possible link cost metrics?

- "good" path:
  - typically means minimum cost path
  - other def's possible

Routing Algorithm classification

Global or decentralized information?

Global:

- all routers have complete topology, link cost info
- "link state" algorithms

Decentralized:

- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- "distance vector" algorithms

Static or dynamic?

Static:

- routes change slowly over time

Dynamic:

- routes change more quickly
  - periodic update
  - in response to link cost changes

A Link-State Routing Algorithm

Dijkstra's algorithm

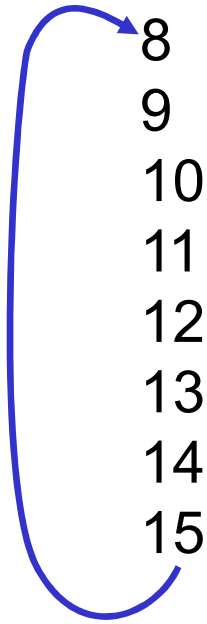
- net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ('source") to all other nodes
  - gives routing table for that node
- iterative: after k iterations, know least cost path to k dest.'s

Notation:

- c(i,j): link cost from node i to j. cost infinite if not direct neighbors
- D(v): current value of cost of path from source to dest. V
- p(v): predecessor node along path from source to v, that is next v
- N: set of nodes whose least cost path definitively known

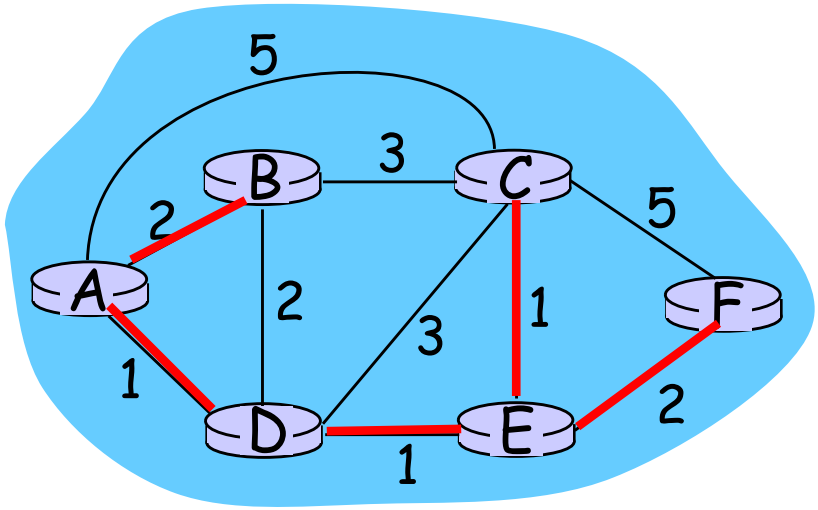
Dijsktra's Algorithm

```
1  Initialization:
2    N = {A}
3    for all nodes v
4      if v adjacent to A
5        then D(v) = c(A,v)
6        else D(v) = infinity
7
8   Loop
9     find w not in N such that D(w) is a minimum
10     add w to N
11     update D(v) for all v adjacent to w and not in N:
12        D(v) = min( D(v), D(w) + c(w,v) )
13     /* new cost to v is either old cost to v or known
14      shortest path cost to w plus cost from w to v */
15  until all nodes in N
```



Dijkstra's algorithm: example

Step	start N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	infinity	infinity
1	AD	2,A	4,D		2,D	infinity
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
5	ADEBCF					



Distance Vector Routing Algorithm

iterative:

- continues until no nodes exchange info.
- *self-terminating*: no "signal" to stop

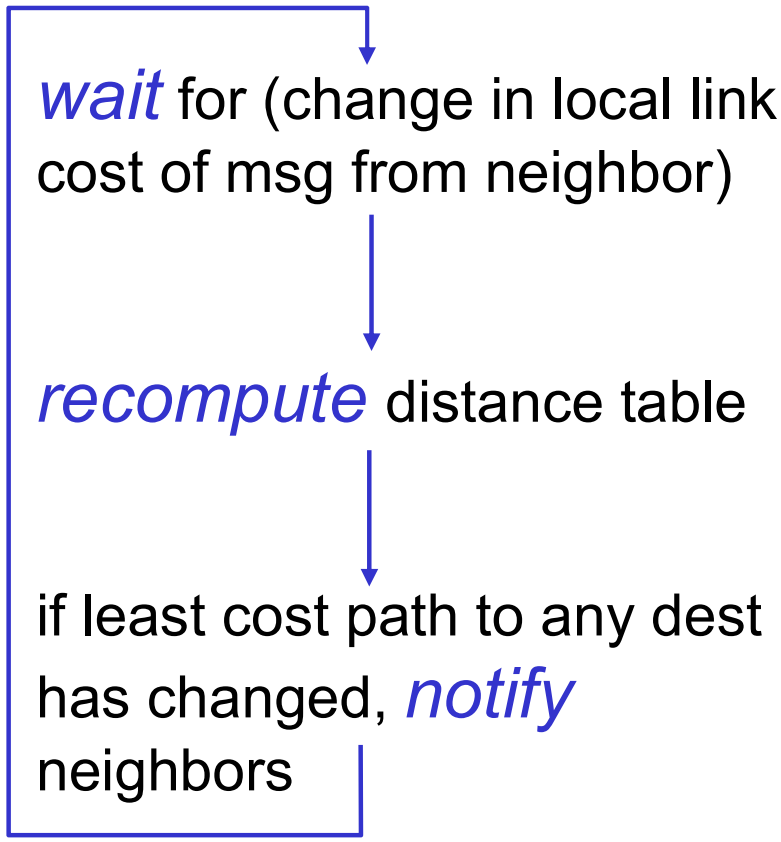
asynchronous:

- nodes need *not* exchange info/iterate in lock step!

distributed:

- each node communicates *only* with directly-attached neighbors

Each node:



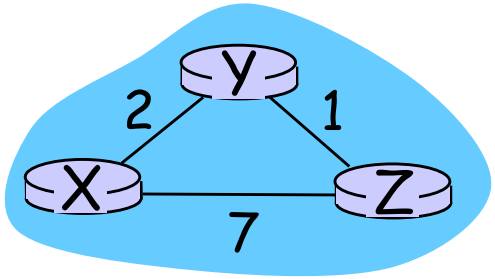
Distance Vector Algorithm: Data Structures

- Each node *x* maintains:
  - For each neighbor *v*, cost *c(x,v)*
  - Node *x*'s distance vector: $D_x = [D_x(y): y \in N]$ containing x's estimate of cost to all destinations
  - Distance vectors for each neighbor v: $D_v = [D_v(y): y \in N]$

- Basic operation: Bellman-Ford algorithm

$$D_x(y) = \min_v \{c(x,v) + D_v(y)\} \quad y \in N$$

Distance Vector Algorithm:



At all nodes, X:

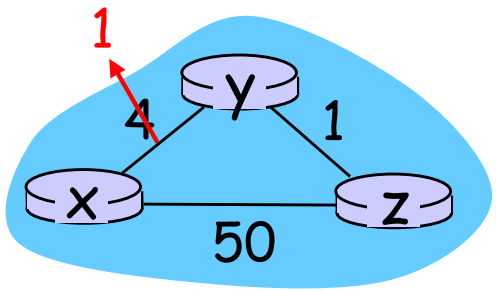
1 Initialization:
2 For all destinations y ∈ N:
3 $D_x(y) = c(x,y)$ /* if y is not a neighbor, then $c(x,y) = \infty$ */:
4 For each neighbor w
5 $D_w(y) = \infty$ for all destinations y ∈ N
6 For each neighbor w
7 Send distance vector $D_x = [D_x(y): y \in N]$ to w

8 Loop:
9 Wait (until communication from neighbor w)
10 For each y ∈ N:
11 $D_x(y) = \min_v \{c(x,v) + D_v(y)\}$
12 If $D_x(y)$ changes for any destination y
13 Send distance vector $D_x = [D_x(y): y \in N]$ to all neighbors

Distance Vector: link cost changes

Link cost changes:

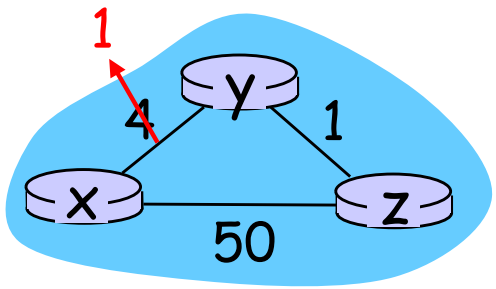
- node detects local link cost change
- updates routing info, recalculates distance vector
- if DV changes, notify neighbors



Distance Vector: link cost changes

Link cost changes:

- node detects local link cost change
- updates routing info, recalculates distance vector
- if DV changes, notify neighbors



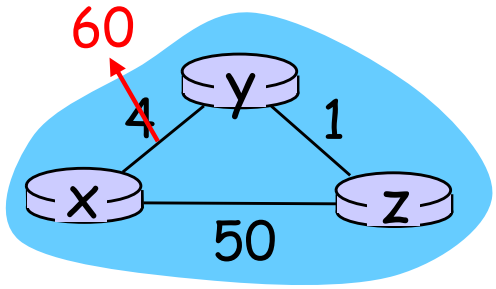
"good news travels fast"

At time t_0, y detects the link-cost change, updates its DV, and informs its neighbors.

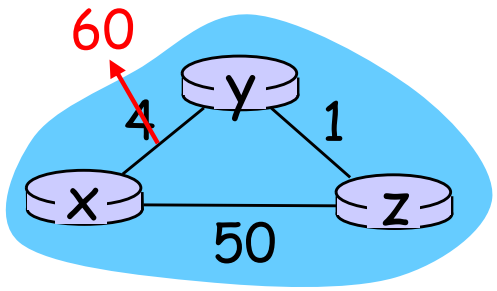
At time t_1, z receives the update from y and updates its table. It computes a new least cost to x and sends its neighbors its DV.

At time t_2, y receives z's update and updates its distance table. y's least costs do not change and hence y does *not* send any message to z.

Distance Vector: link cost changes



Distance Vector: link cost changes

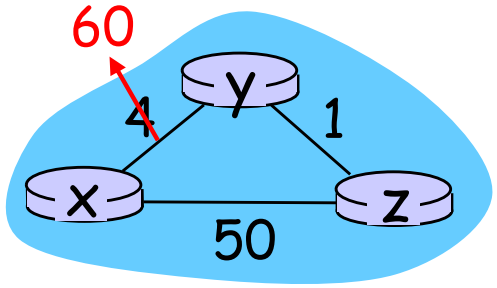


$$Dy(x) = \min\{c(y,x)+Dx(x),\ c(y,z)+Dz(x)\}$$
$$= \min\{60+0,\ 1+5\} = 6$$

Distance Vector: link cost changes

Link cost changes:

- good news travels fast
- bad news travels slow - "count to infinity" problem!
- 44 iterations before algorithm stabilizes



Poissoned reverse:

- If Z routes through Y to get to X :
  - Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
- will this completely solve count to infinity problem?

Comparison of LS and DV algorithms I

Message complexity

- with n nodes, E links
- LS:
- DV:

Comparison of LS and DV algorithms II

Message complexity

- LS: with n nodes, E links, O(nE) msgs sent each
- DV: exchange between neighbors only
  - convergence time varies

Speed of Convergence

- LS:
- DV:

Outline

- IP addresses
- Virtual circuit vs. datagrams
- Routing algorithms
  - Link state
  - Distance vector
- Mobile IP
  - Architecture
- Encapsulation

Comparison of LS and DV algorithms

Message complexity

- LS: with n nodes, E links, O(nE) msgs sent each
- DV: exchange between neighbors only
  - convergence time varies

Speed of Convergence

- LS: O(n^2) algorithm requires O(nE) msgs
  - may have oscillations
- DV: convergence time varies
  - may be routing loops
  - count-to-infinity problem

Robustness: what happens if router malfunctions?

Comparison of LS and DV algorithms

Message complexity

- LS: with n nodes, E links, O(nE) msgs sent each
- DV: exchange between neighbors only
  - convergence time varies

Speed of Convergence

- LS: O(n^2) algorithm requires O(nE) msgs
  - may have oscillations
- DV: convergence time varies
  - may be routing loops
  - count-to-infinity problem

Robustness: what happens if router malfunctions?

LS:

- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagate thru network

How does Internet routing work?

Hierarchical Routing

Our routing review thus far - idealization

- all routers identical
- network "flat"

… *not* true in practice

scale: with 200 million destinations:

- can't store all dest's in routing tables!
- routing table exchange would swamp links!

administrative autonomy

- internet = network of networks
- each network admin may want to control routing in its own network

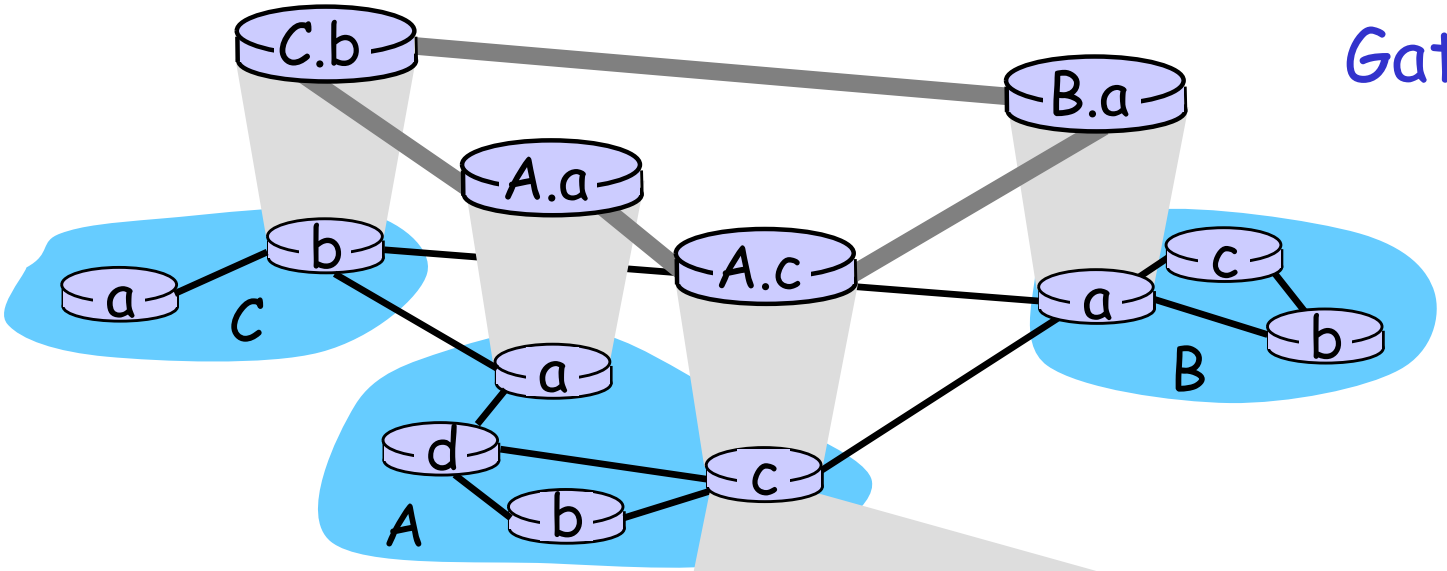
Hierarchical Routing

- aggregate routers into regions, "autonomous systems" (AS)
- routers in same AS run same routing protocol
  - "intra-AS" routing protocol
  - routers in different AS can run different intra-AS routing protocol

gateway routers

- special routers in AS
- run intra-AS routing protocol with all other routers in AS
- *also* responsible for routing to destinations outside AS
  - run *inter-AS routing* protocol with other gateway routers

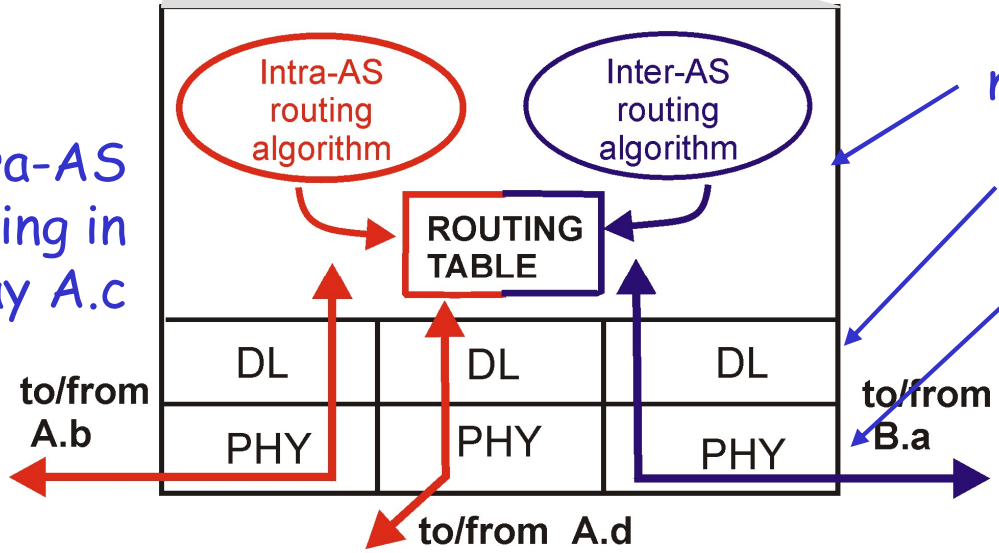
IntraAS and InterAS routing



Gateways:

- perform inter-AS routing amongst themselves
- perform intra-AS routers with other routers in their AS

inter-AS, intra-AS routing in gateway A.c

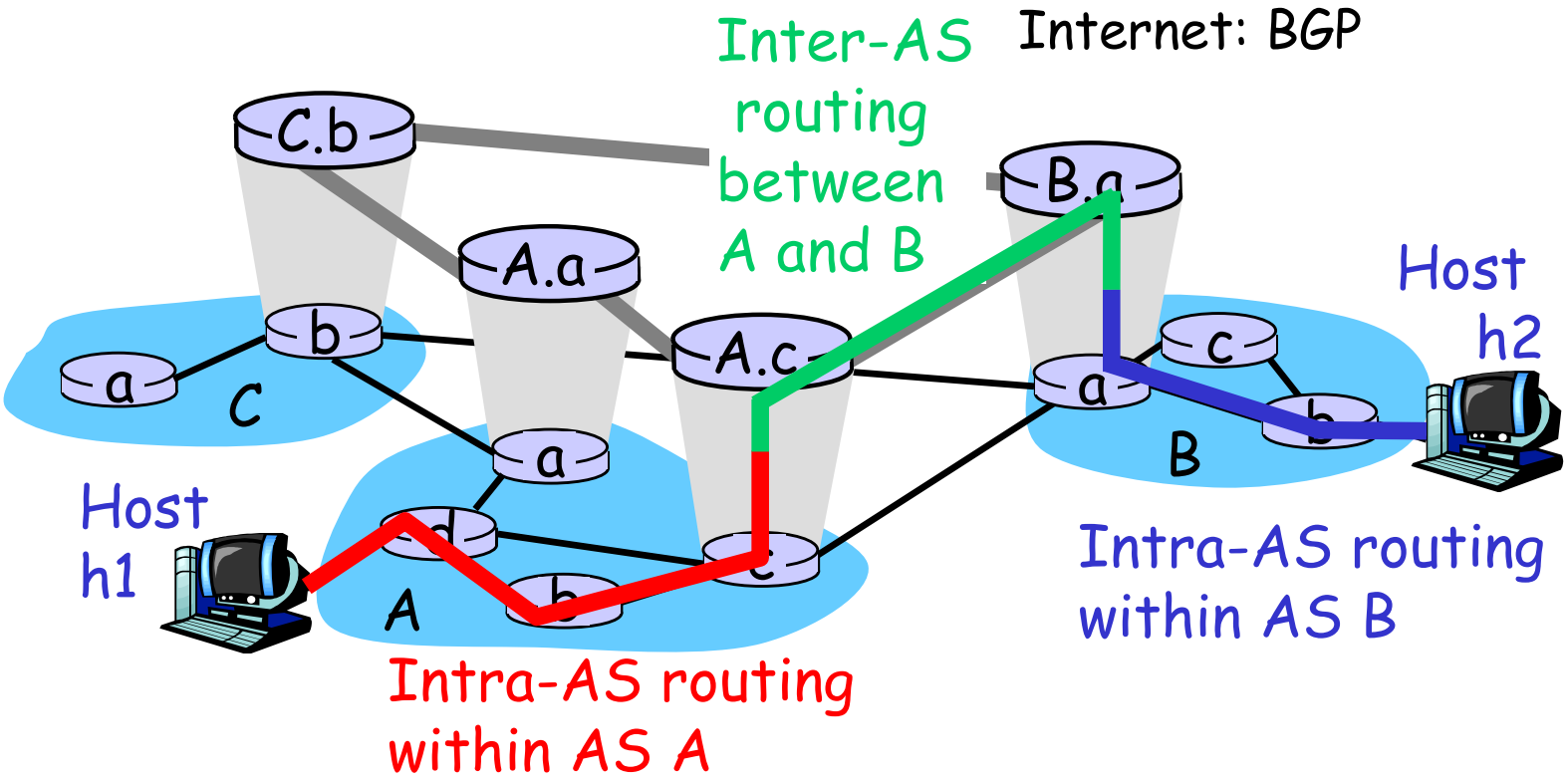


network layer

link layer

physical layer

IntraAS and InterAS routing



Intra-AS: OSPF, IS-IS, RIP
Inter-AS: BGP

Discussion

- IP works fine for the Internet
  - it has problems; but during vast majority of the time it gets its job done efficiently—moving a packet from a src. to a dest.

- What problem can mobility cause?

Motivation for Mobile IP

- Routing
  - based on IP destination address, network prefix (e.g. 129.13.42) determines physical subnet
  - change of physical subnet implies change of IP address to have a topological correct address (standard IP) or needs special entries in the routing tables

Motivation for Mobile IP

- Routing
  - based on IP destination address, network prefix (e.g. 129.13.42) determines physical subnet
  - change of physical subnet implies change of IP address to have a topological correct address (standard IP) or needs special entries in the routing tables
- Keeping the IP address while moving
  - Specific routes to end-systems
  - change of all routing table entries to forward packets to the right destination
  - does not scale with the number of mobile hosts and frequent changes in the location, security problems
- Changing the IP address
  - adjust the host IP address depending on the current location
  - almost impossible to find a mobile system, DNS updates take a long time
  - TCP connections break, security problems

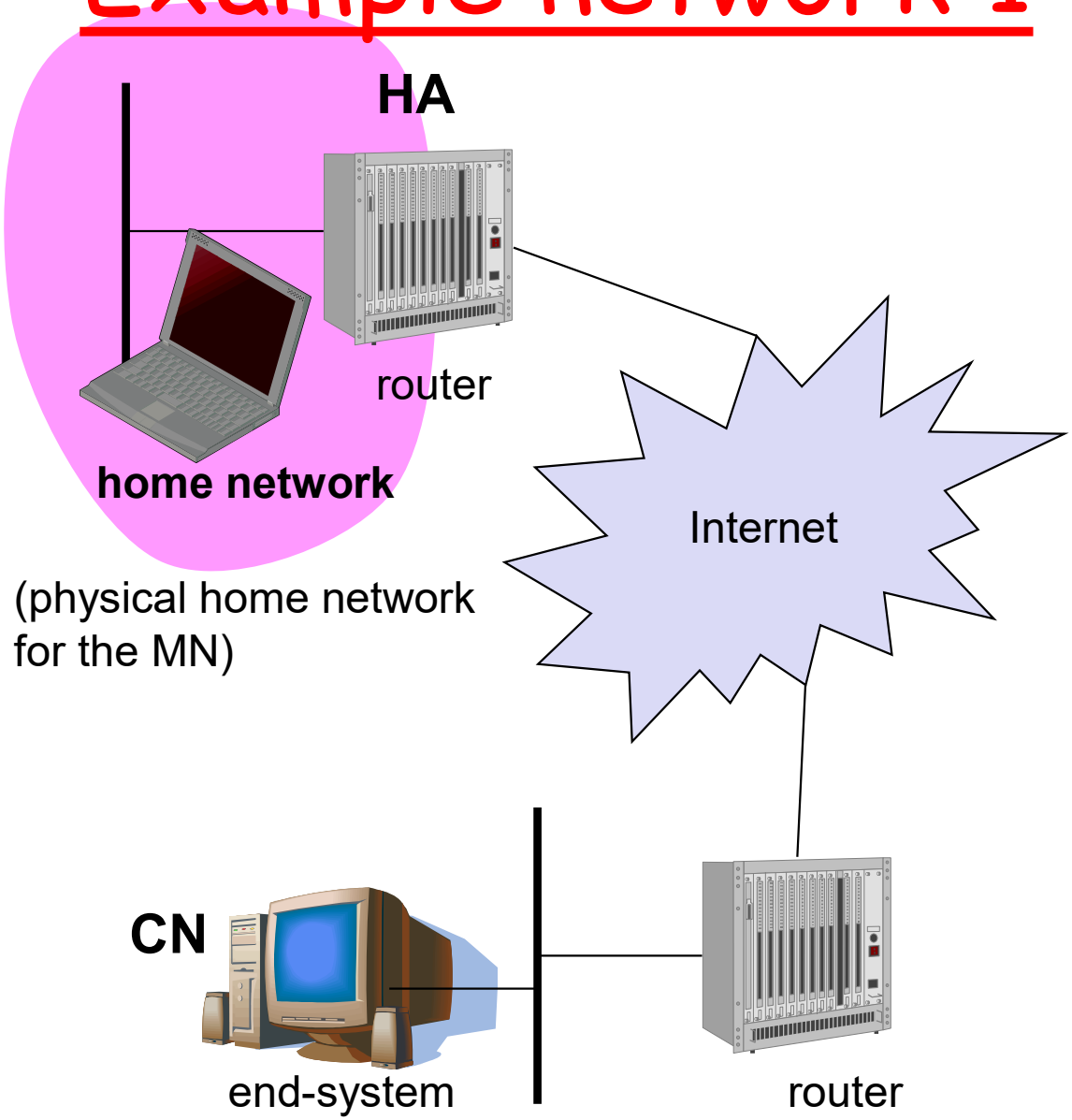
Requirements to Mobile IP (RFC 3344, was: 3220, was: 2002)

- Transparency
  - mobile end-systems keep their IP address
  - continuation of communication after interruption of link possible
  - point of connection to the fixed network can be changed
- Compatibility
  - support of the same layer 2 protocols as IP
  - no changes to current end-systems and routers required
  - mobile end-systems can communicate with fixed systems
- Security
  - authentication of all registration messages
- Efficiency and scalability
  - only little additional messages to the mobile system required (connection typically via a low bandwidth radio link)
  - world-wide support of a large number of mobile systems in the whole Internet

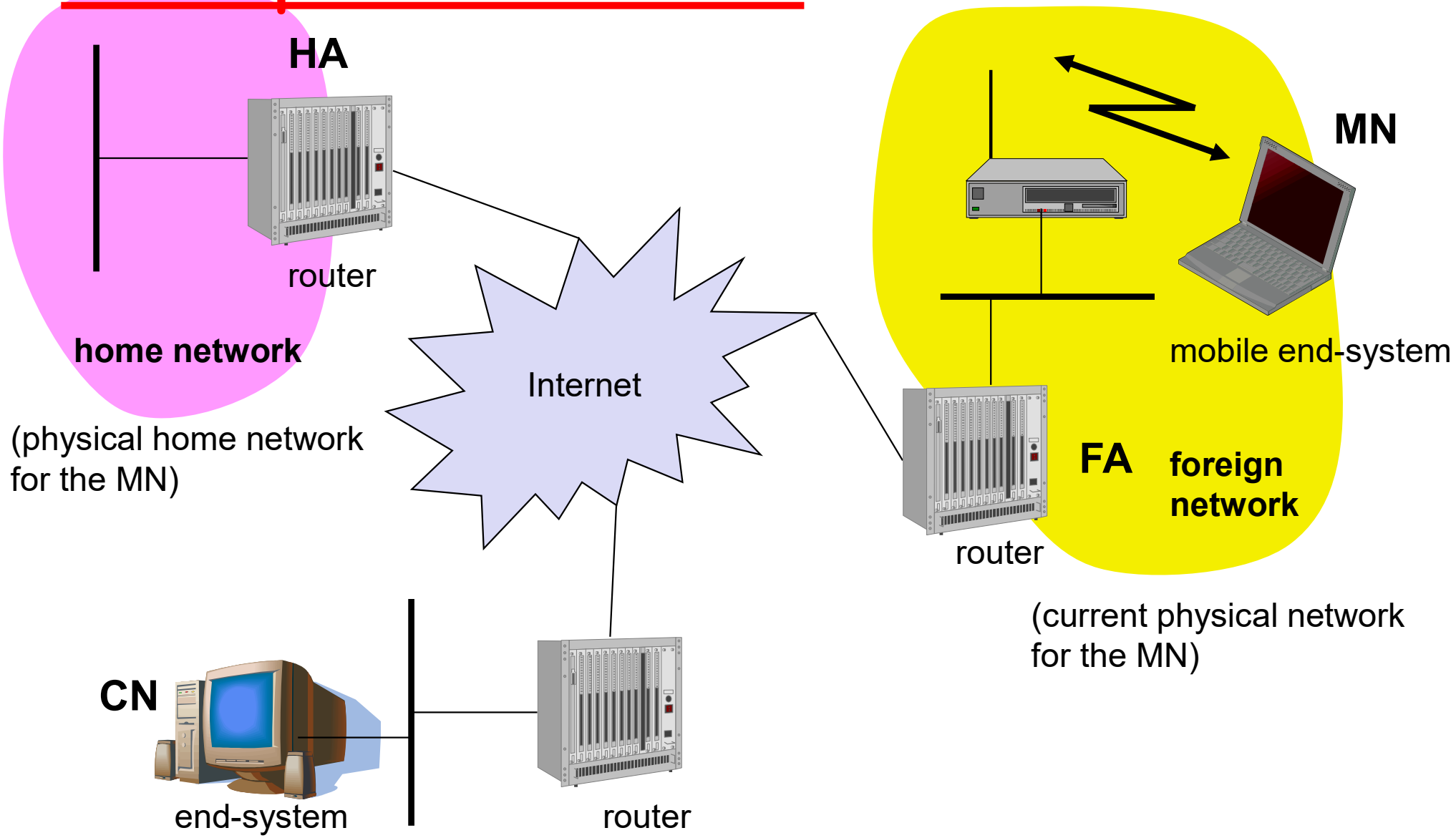
Terminology

- Mobile Node (MN)
  - system (node) that can change the point of connection to the network without changing its IP address
- Home Agent (HA)
  - system in the home network of the MN, typically a router
  - registers the location of the MN, tunnels IP datagrams to the COA
- Foreign Agent (FA)
  - system in the current foreign network of the MN, typically a router
  - forwards the tunneled datagrams to the MN, typically also the default router for the MN
- Care-of Address (COA)
  - address of the current tunnel end-point for the MN (at FA or MN)
  - actual location of the MN from an IP point of view
  - can be chosen, e.g., via DHCP
- Correspondent Node (CN)
  - communication partner

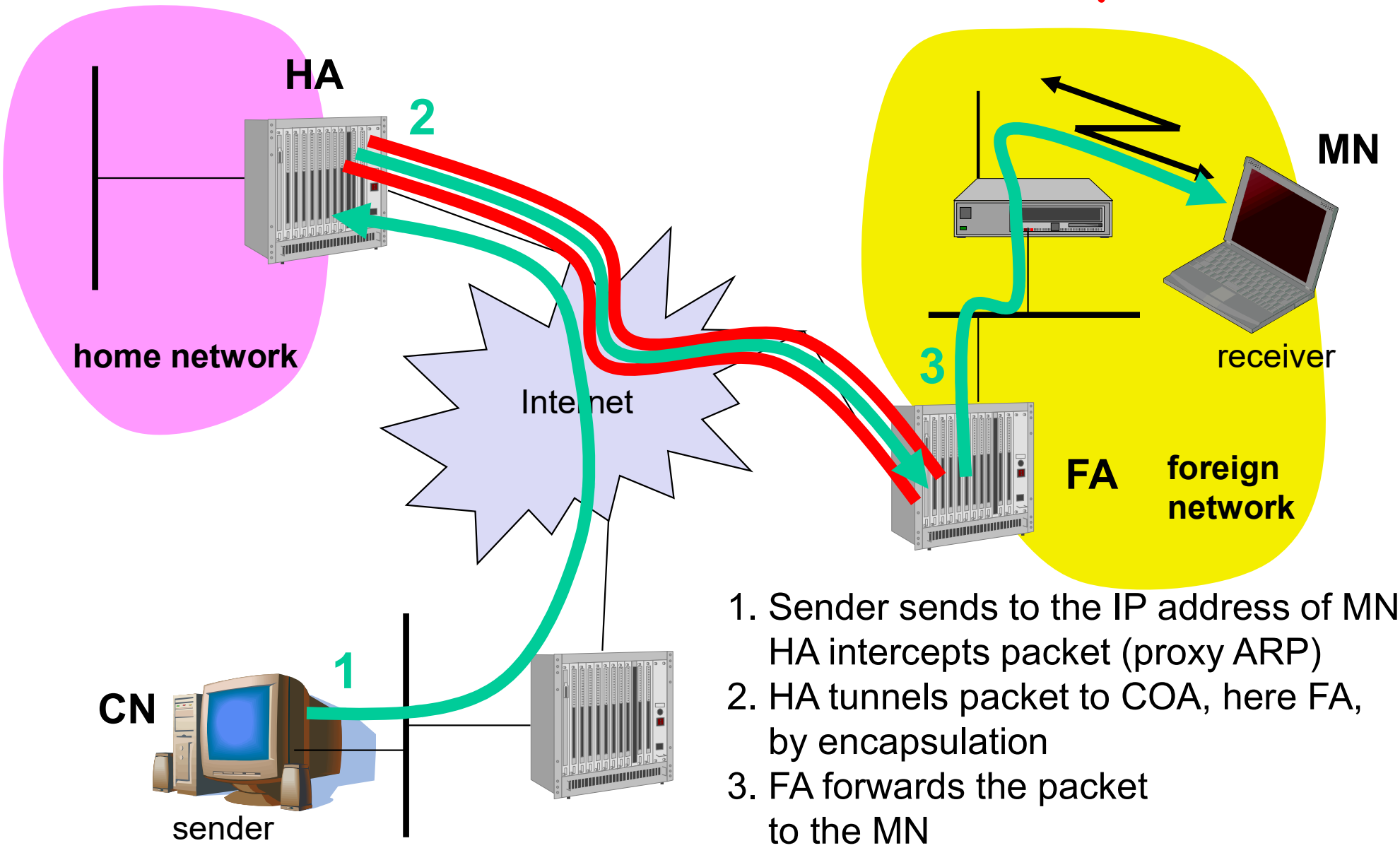
Example network 1



Example network 2

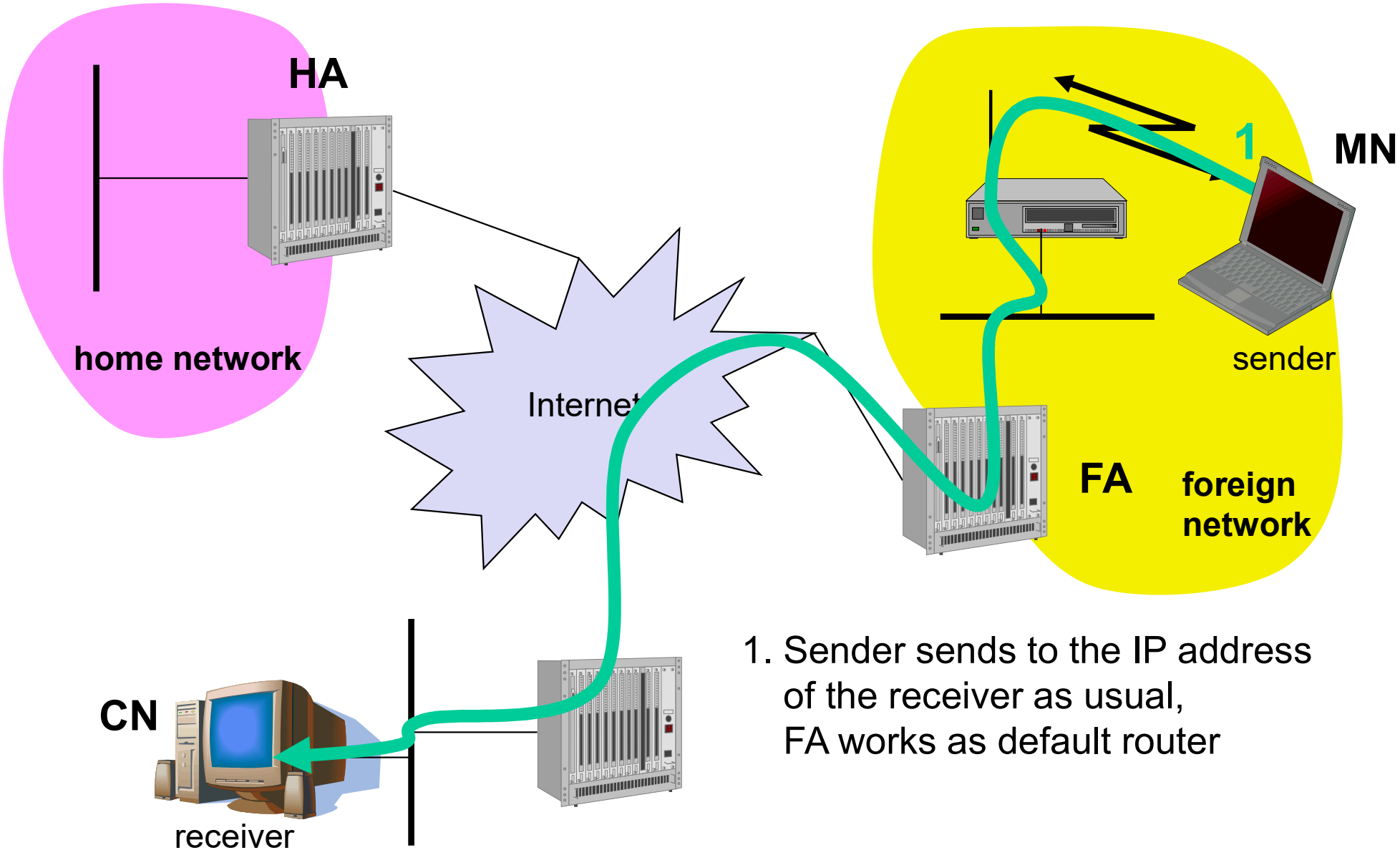


Data transfer to the mobile system



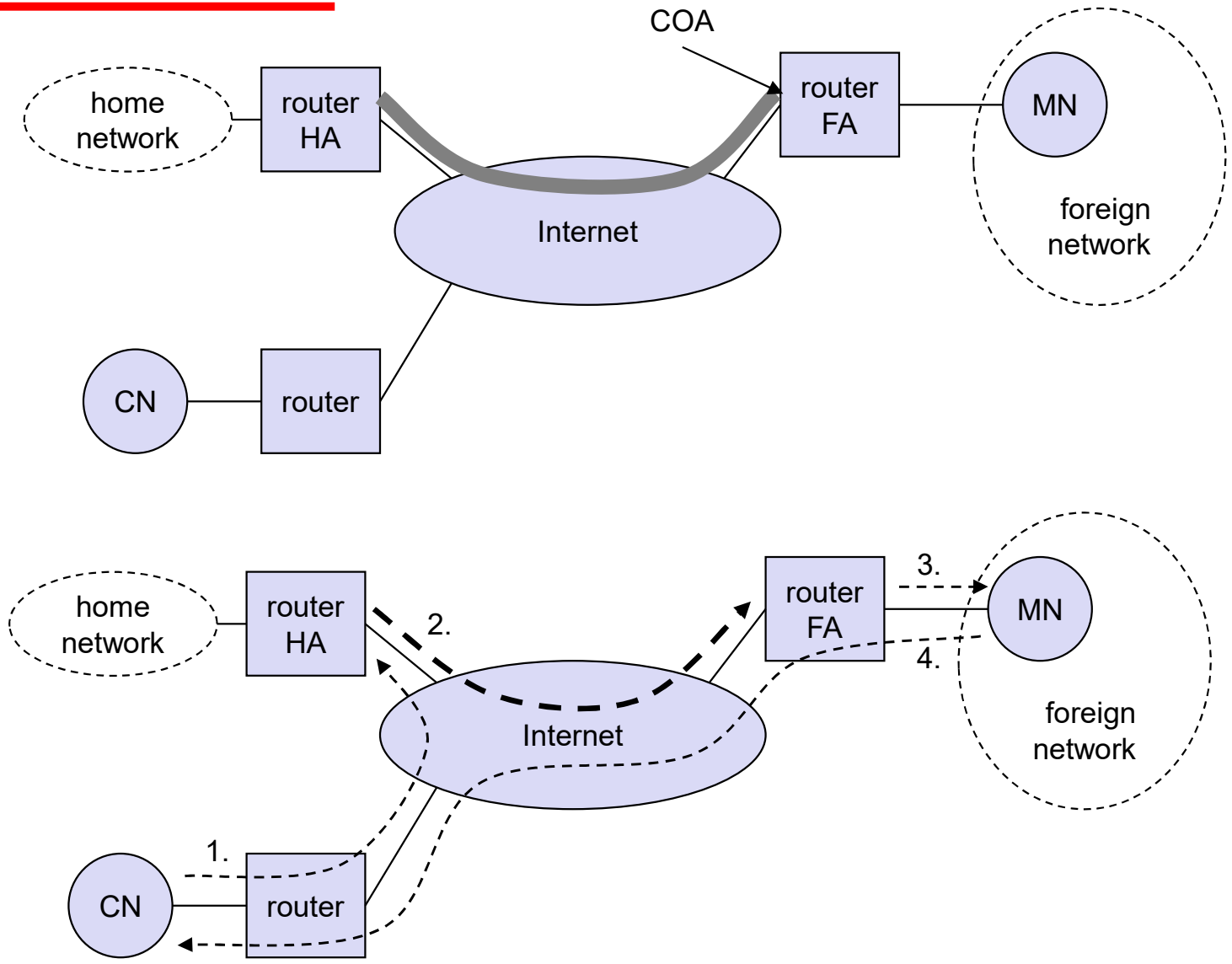
1. Sender sends to the IP address of MN, HA intercepts packet (proxy ARP)
2. HA tunnels packet to COA, here FA, by encapsulation
3. FA forwards the packet to the MN

Data transfer from the mobile system



1. Sender sends to the IP address of the receiver as usual, FA works as default router

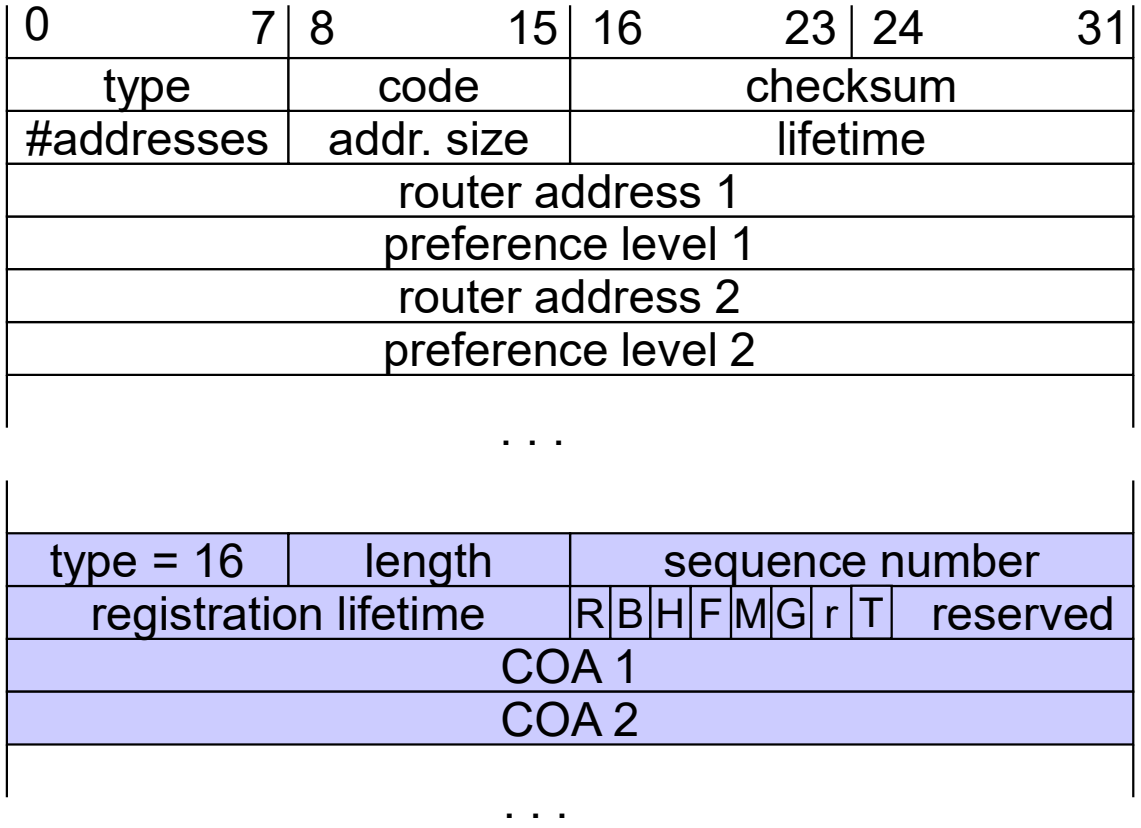
Overview


COA
home network
router HA
router FA
MN
Internet
foreign network
CN
router
home network
router HA
2.
router FA
3.
MN
4.
Internet
foreign network
1.
CN
router

Network integration

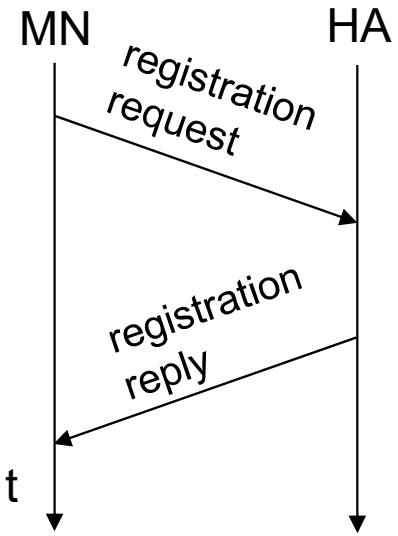
- Agent Advertisement
  - HA and FA periodically send advertisement messages into their physical subnets
  - MN listens to these messages and detects, if it is in the home or a foreign network (standard case for home network)
  - MN reads a COA from the FA advertisement messages
- Registration (always limited lifetime!)
  - MN signals COA to the HA via the FA, HA acknowledges via FA to MN
  - these actions have to be secured by authentication
- Advertisement
  - HA advertises the IP address of the MN (as for fixed systems), i.e. standard routing information
  - routers adjust their entries, these are stable for a longer time (HA responsible for a MN over a longer period of time)
  - packets to the MN are sent to the HA,
  - independent of changes in COA/FA

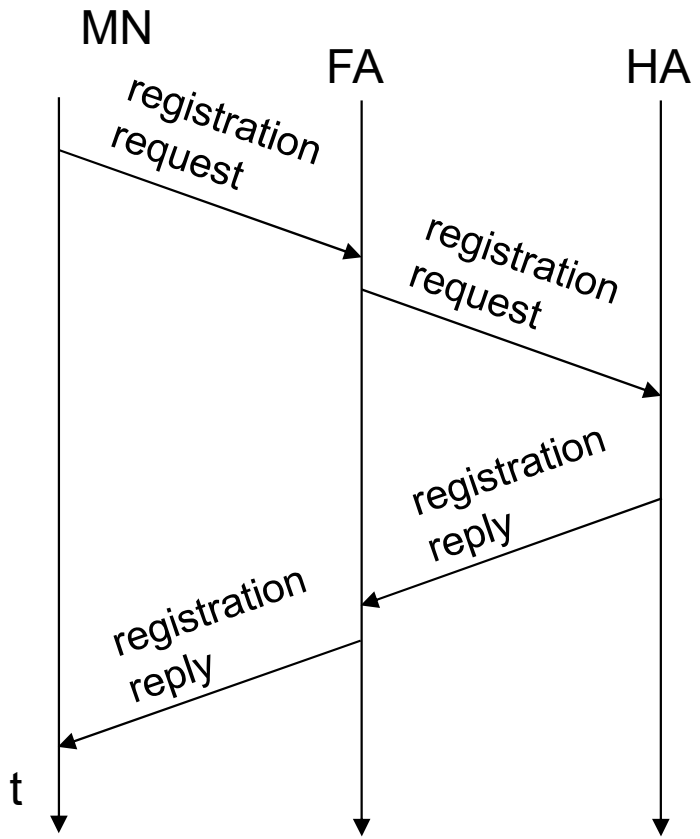
Agent advertisement



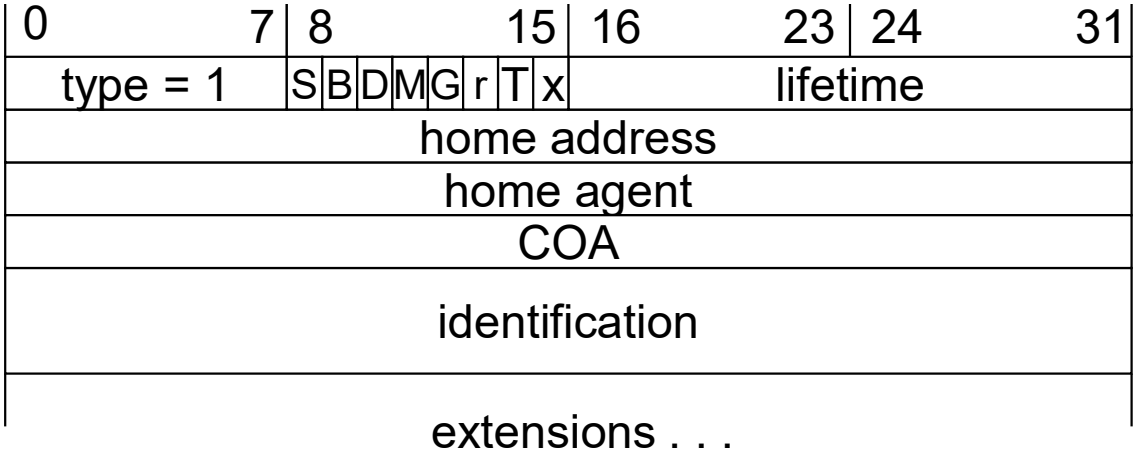
ICMP packets
type = 16
length = 6 + 4 * #COAs
R: registration required
B: busy, no more registrations
H: home agent
F: foreign agent
M: minimal encapsulation
G: GRE encapsulation
r: =0, ignored (former Van Jacobson compression)
T: FA supports reverse tunneling
reserved: =0, ignored

Registration



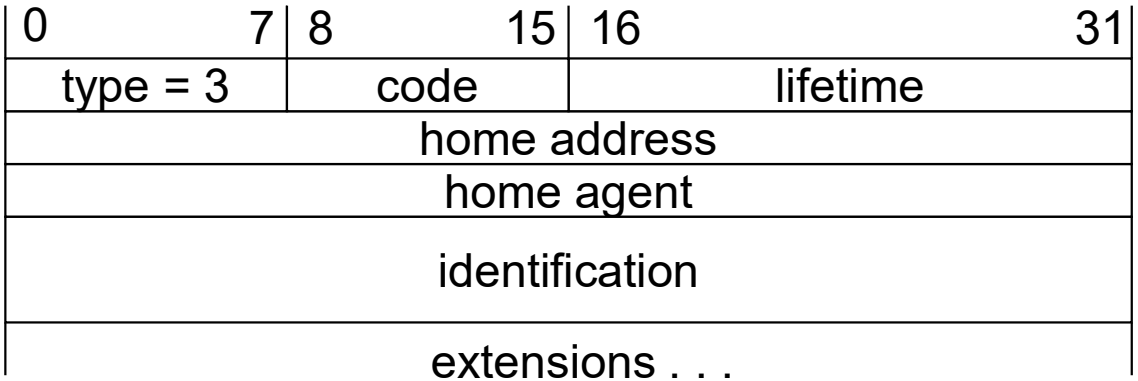


Mobile IP registration request



UDP packets
S: simultaneous bindings
B: broadcast datagrams
D: decapsulation by MN
M mininal encapsulation
G: GRE encapsulation
r: =0, ignored
T: reverse tunneling requested
x: =0, ignored

Mobile IP registration reply



Example codes:

registration successful

0 registration accepted

1 registration accepted, but simultaneous mobility bindings unsupported

registration denied by FA

65 administratively prohibited

66 insufficient resources

67 mobile node failed authentication

68 home agent failed authentication

69 requested Lifetime too long

registration denied by HA

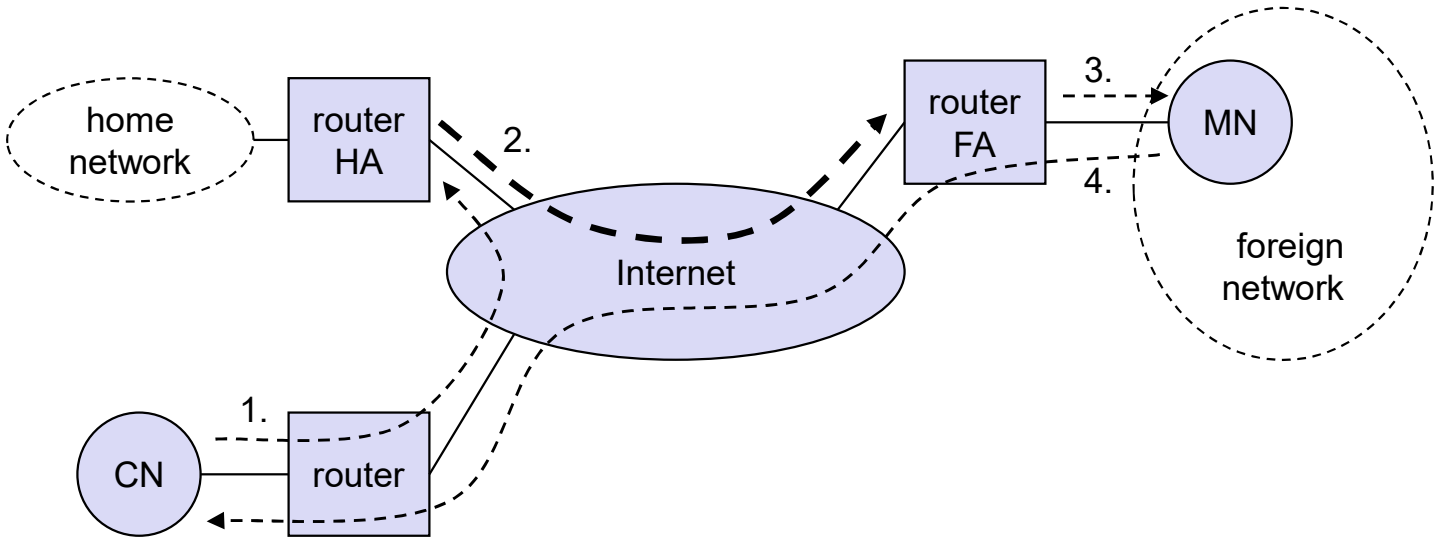
129 administratively prohibited

131 mobile node failed authentication

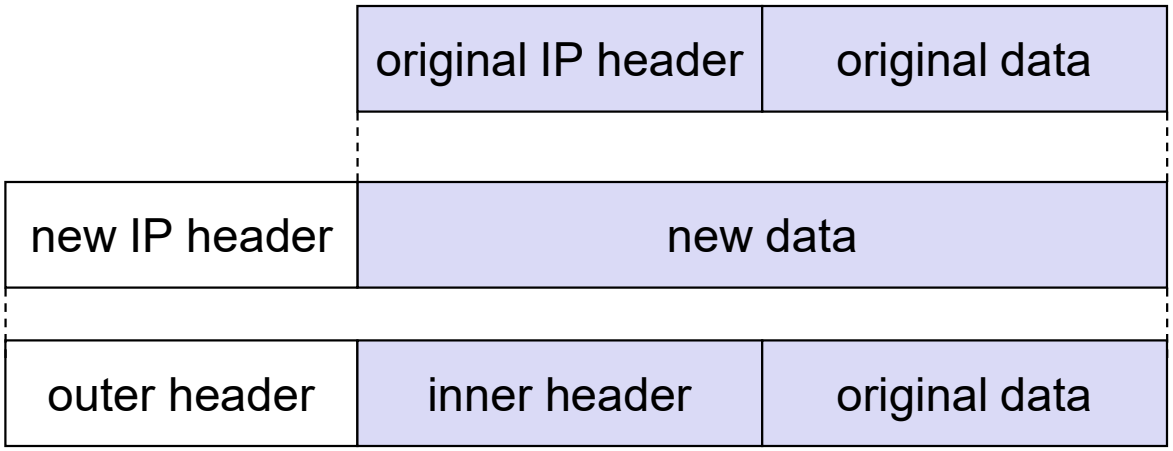
133 registration Identification mismatch

135 too many simultaneous mobility bindings

Encapsulation 1



Encapsulation II



Encapsulation I

- Encapsulation of one packet into another as payload
  - e.g. IPv6 in IPv4 (6Bone), Multicast in Unicast (Mbone)
  - here: e.g. IP-in-IP-encapsulation, minimal encapsulation or GRE (Generic Record Encapsulation)
- IP-in-IP-encapsulation (mandatory, RFC 2003)
  - tunnel between HA and COA

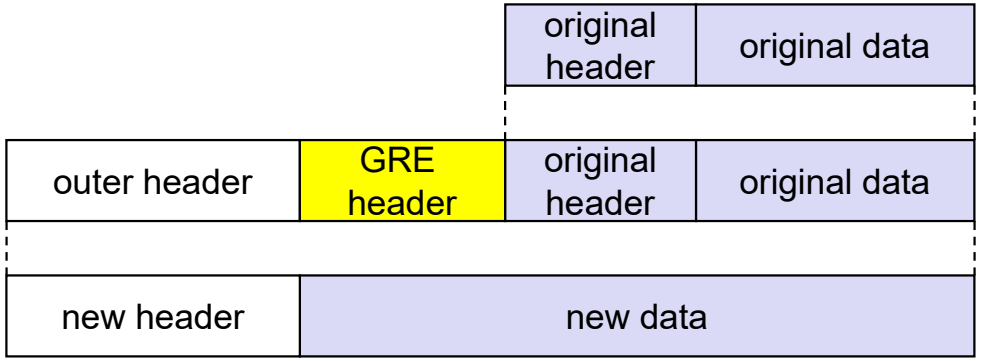
ver.	IHL	DS (TOS)	length	
IP identification			flags	fragment offset
TTL		*IP-in-IP*	IP checksum	
IP address of HA				
Care-of address COA				
ver.	IHL	DS (TOS)	length	
IP identification			flags	fragment offset
TTL		lay. 4 prot.	IP checksum	
IP address of CN				
IP address of MN				
TCP/UDP/ ... payload				

Encapsulation II

- Minimal encapsulation (optional)
  - avoids repetition of identical fields
  - e.g. TTL, IHL, version, DS (RFC 2474, old: TOS)
  - only applicable for unfragmented packets, no space left for fragment identification

ver.	IHL	DS (TOS)	length	
IP identification			flags	fragment offset
TTL		*min. encap.*	IP checksum	
IP address of HA				
care-of address COA				
lay. 4 protoc.	S	reserved	IP checksum	
IP address of MN				
original sender IP address (if S=1)				
TCP/UDP/ ... payload				

Generic Routing Encapsulation



RFC 1701

ver.	IHL	DS (TOS)	length				
IP identification			flags	fragment offset			
TTL		*GRE*	IP checksum				
IP address of HA							
Care-of address COA							
C	R	K	S	s	rec.	rsv.	ver.	protocol
checksum (optional)						offset (optional)	
key (optional)							
sequence number (optional)							
routing (optional)							
ver.	IHL	DS (TOS)	length				
IP identification			flags	fragment offset			
TTL		lay. 4 prot.	IP checksum				
IP address of CN							
IP address of MN							
TCP/UDP/ ... payload							

RFC 2784

C	reserved0	ver.	protocol
checksum (optional)			reserved1 (=0)