IPv4 and/or IPv6: yes, but then what?

- IP serves only for sending packets with well-known addresses. Some crucial questions however remain open:
  - how to map an IP address to a link layer address?
    ➔ Address Resolution Protocol (ARP)
  - how to map a link layer address to an IP address?
    ➔ Reverse Address Resolution Protocol (RARP)
  - how to signal errors, send control messages at the network layer?
    ➔ Internet Control Message Protocol (ICMP)
  - how to acquire and assign an IP address for a device interface?
    ➔DHCP, NDP/SLAAC
  - how to establish/maintain paths through the Internet?
    ➔ Routing, with OSPF, BGP…
  - how to communicate efficiently with sets of devices/interface, at once?
    ➔ Multicasting, Internet Group Management Protocol (IGMP)

CONTENT of this CHAPTER

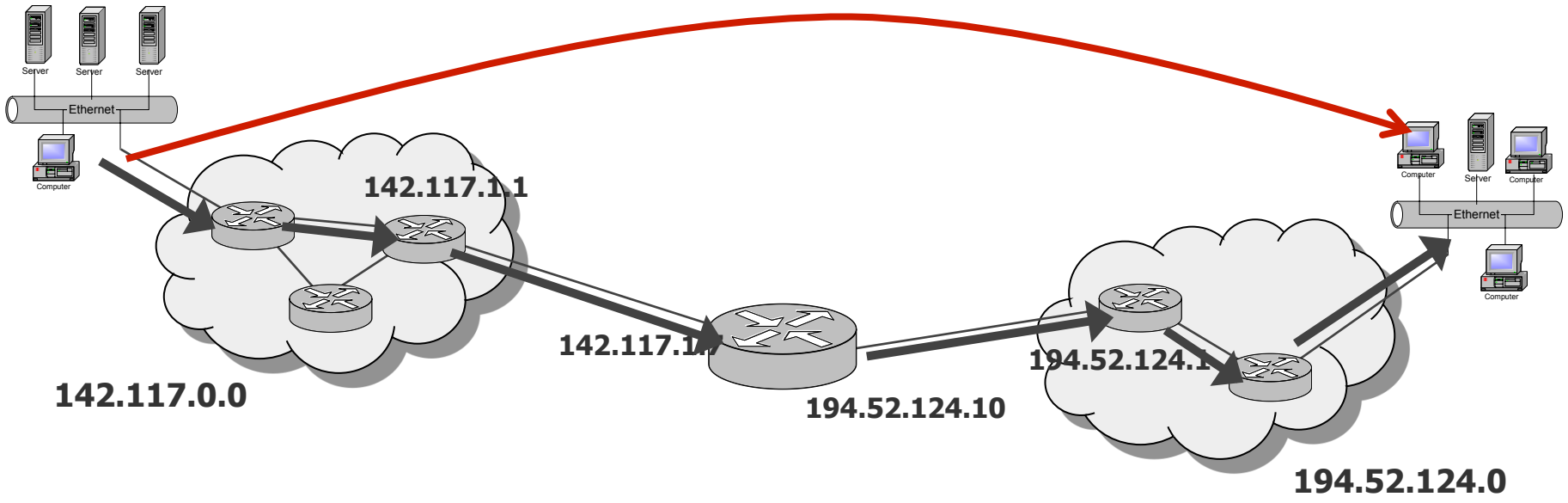
- Fundamental Goals of the Network Layer
- Network Layer Addressing
  - IPv4 Addressing
  - IPv6 Addressing
- Internet Protocol Design
  - Internet Protocol version 4
  - Internet Protocol version 6
  - IPv6 Migration: Transition and Coexistence
- IP Address Mapping & Assignment (ARP, RARP, DHCP, NDP/SLAAC)
- Control, Error Management and Diagnostic (ICMP)
- Network Address Translation (NAT)
- Routing (Concepts, OSPF, BGP…)
- Multicast (Concepts, IGMP…)

Delivery of Packets End-to-end

Application Layer
Transport Layer
Network Layer
Link Layer
Physical Layer

IP Addresses

Corresponding MAC Addresses ?



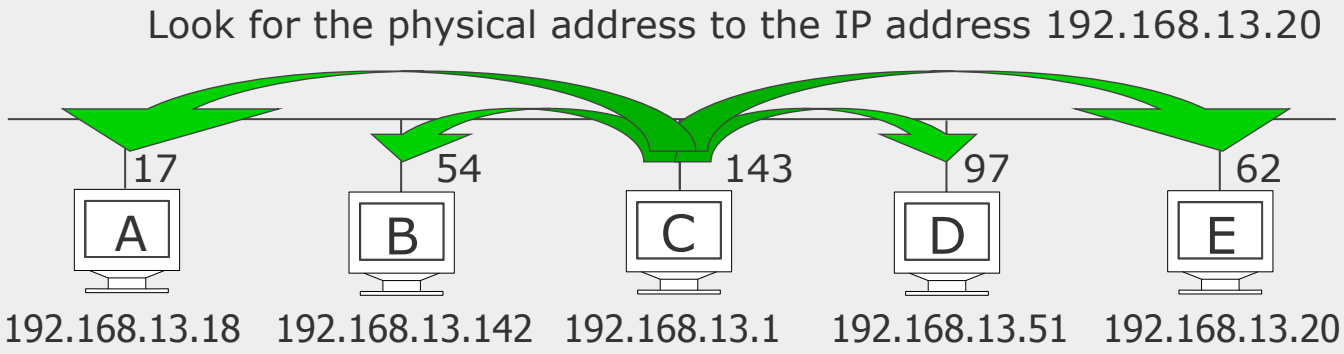
Address Resolution Protocol (ARP)

- The Internet is a virtual network, which is build upon physical networks.
  - **IP addresses** only offer a **logical address space**
  - The hardware on the lower layers do not understand IP addresses
  - Within a **local area network**, the sender must know the **hardware address** (MAC address) of the receiver before sending an IP packet to the destination host

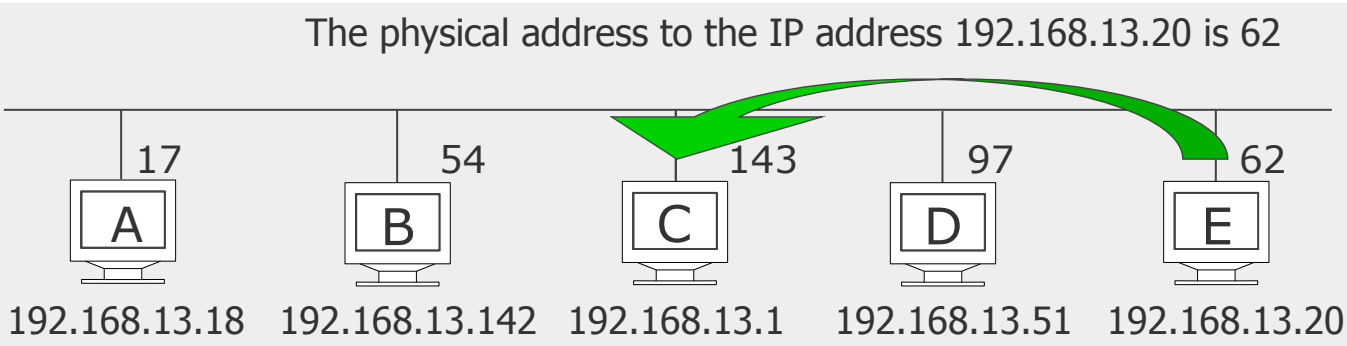
- With ARP, an IP address is mapped to the hardware address and vice versa
  - ARP uses the **local broadcast address** to dynamically inquire for the hardware address by indication of the searched IP address

Address Resolution Protocol (ARP)

ARP Request



ARP Response



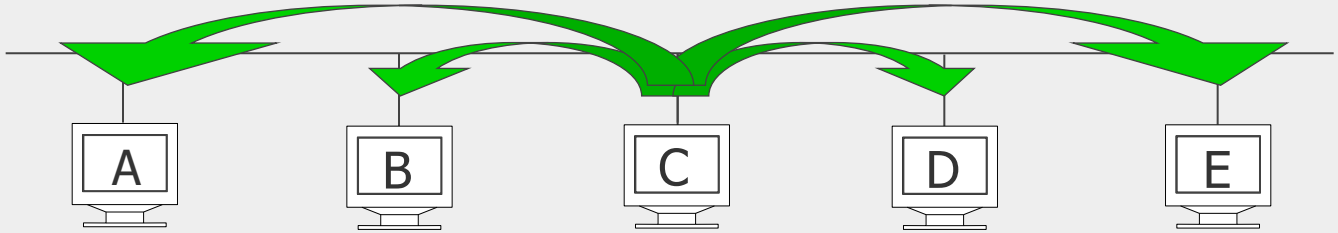
- The host with the inquired IP address sends a response
- Each host stores known IP and MAC addresses in a table: the ARP cache
- ARP cache entries become invalid after a certain time (e.g. 20mn)
  - to avoid mistakes e.g. with the exchange of a network interface card

Address Resolution Protocol (ARP)

Optimization of the procedure:

Each computer occasionally sends an ARP request (broadcast) to its own IP address.

ARP Request



Each receiving host stores the sender IP and sender hardware address in its ARP Cache.

On some systems (e.g. Linux) a host periodically even sends ARP Requests for all addresses listed in the ARP cache.

Advantage: this refreshes the ARP cache content
Drawback: introduces additional overhead

ARP Packet Format

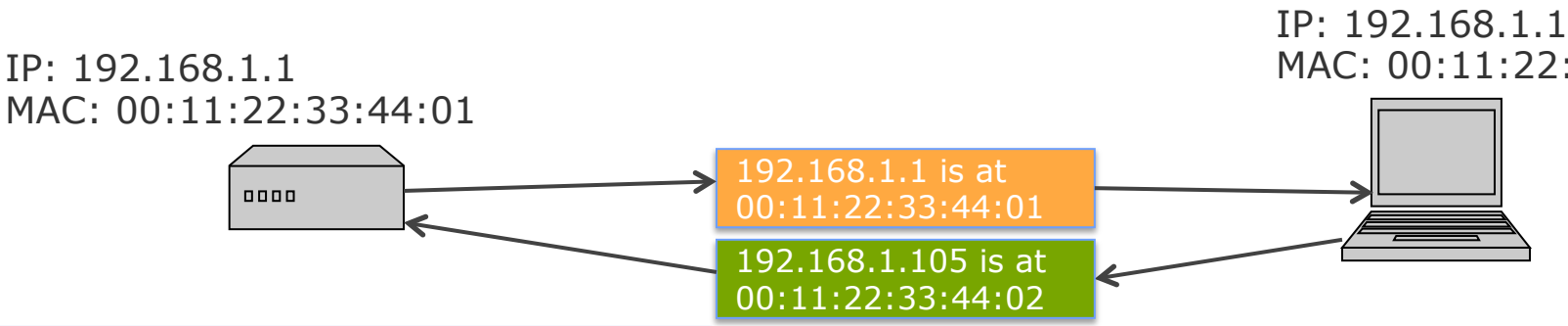
- Specified in RFC 826 (in 1982).
- Possible to use with various network layers and link layers
  - Flexibility in the length of link layer address and network layer address
  - Length of addresses specified in octets (e.g. Ethernet/Wifi size is 6)

Hardware type (2 bytes)		Protocol type (2 bytes)
Hardware address length (1 byte)	Protocol address length (1 byte)	Operation code (2 bytes)
Source hardware address		
Source protocol address		
Target hardware address		
Target protocol address		

ARP Spoofing

- According to the standard:
  - ARP implementations are stateless
  - ARP announcements are not authenticated (= machines trust each other)
- This means that:
  - The ARP cache is updated each time that it receives an arp reply
  - Even if it did not send any arp request
- A rogue machine can spoof other machines!
  - It is possible to "poison" an arp cache by sending forged gratuitous arp replies

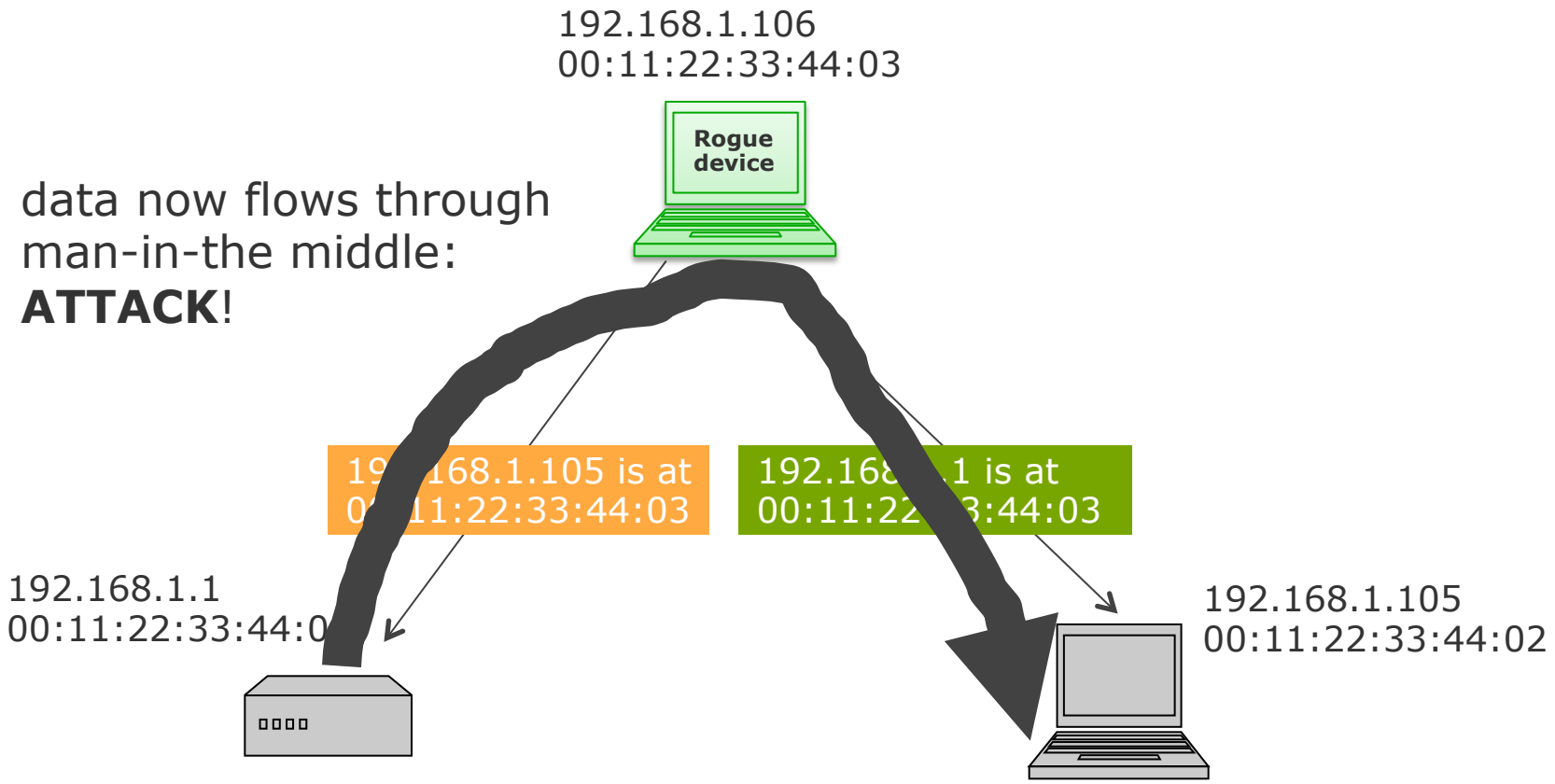
ARP Spoofing: Cache Poisoning



ARP Cache	
192.168.1.**105**	00:11:22:33:44:**02**

ARP Cache	
192.168.1.**1**	00:11:22:33:44:**01**

ARP Spoofing: Cache Poisoning



Poisoned ARP Cache	
192.168.1.**105**	00:11:22:33:44:**03**

Poisoned ARP Cache	
192.168.1.**1**	00:11:22:33:44:**03**

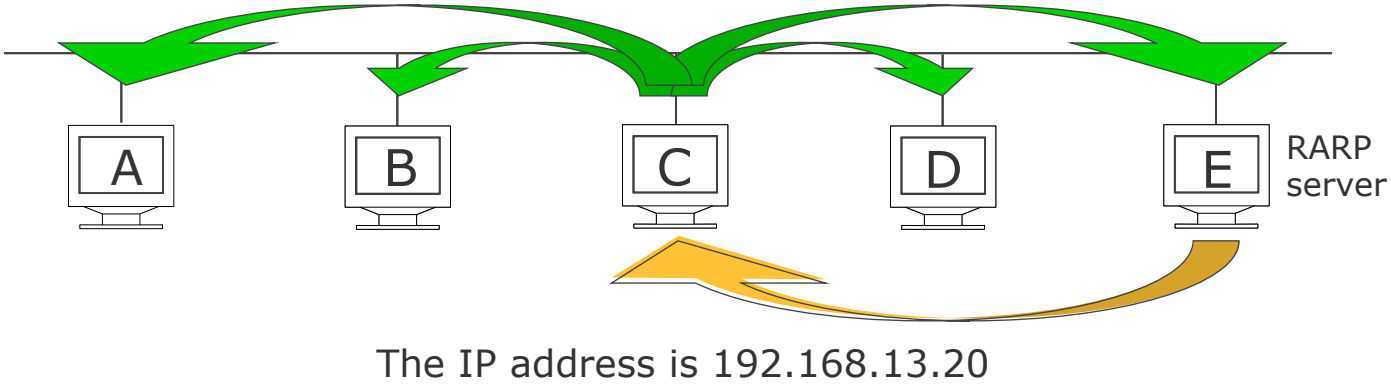
ARP on your machine

- The *arp* command
  - Display and insert entries into the arp cache

```
x:\>arp -a

Interface: 160.45.114.21 --- 0x2
  Internet Address      Physical Address      Type
  160.45.114.1          00-12-79-8d-68-00     dynamic
  160.45.114.28         00-14-32-49-c1-aa     dynamic
  160.45.114.29         00-04-15-d8-53-cb     dynamic
  160.45.114.30         00-04-85-8b-9a-ac     dynamic
  160.45.114.31         00-19-f9-18-82-3d     dynamic
  160.45.114.34         00-04-25-8b-a8-8e     dynamic
```

Reverse Address Resolution Protocol (RARP)

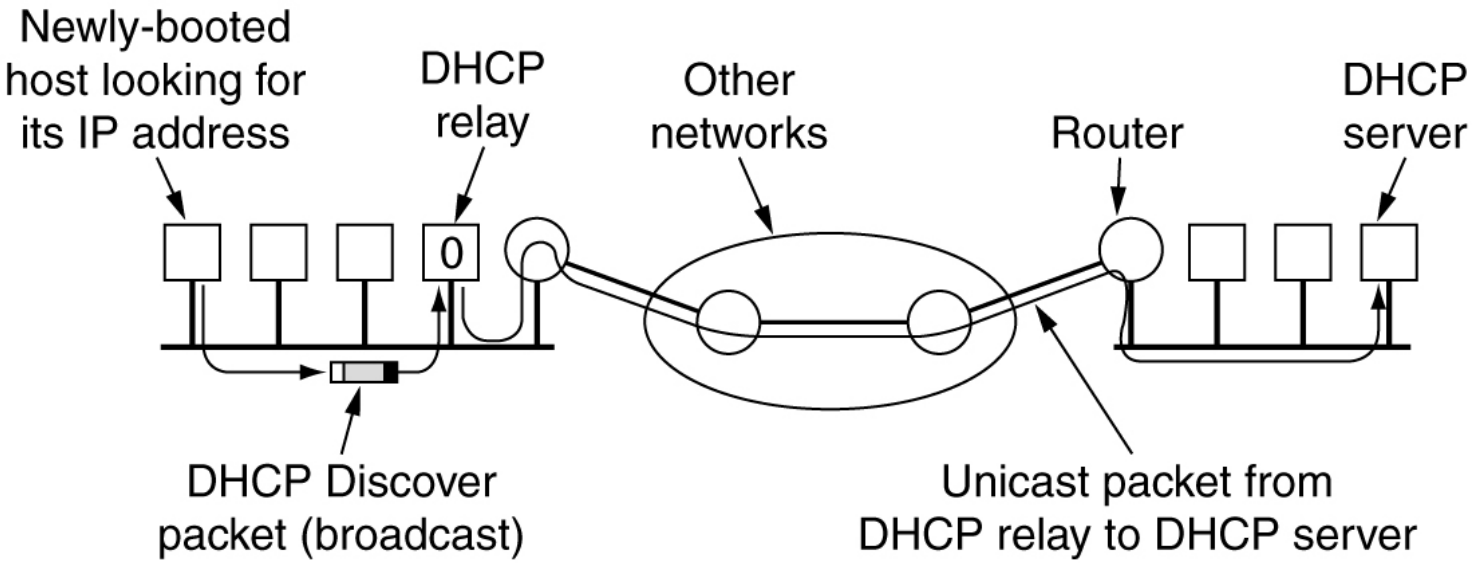
- Upon booting, an IP address is not always assigned.
  - How does such a computer receive its IP address after booting?
- With the help of Reverse ARP, well-known hardware addresses are assigned to IP addresses, and recorded on a RARP server.



- RARP is obsolete. Replaced by **DHCP** (more modern and feature-rich).

Dynamic Host Configuration Protocol (DHCP)

- Problem: RARP requests are not passed on by routers, therefore a RARP server must be set up in each local network.
- Solution: **DHCP**. A computer sends a DHCP DISCOVER packet. In each subnet a DHCP Relay Agent is placed, who passes such a message on to the DHCP server.

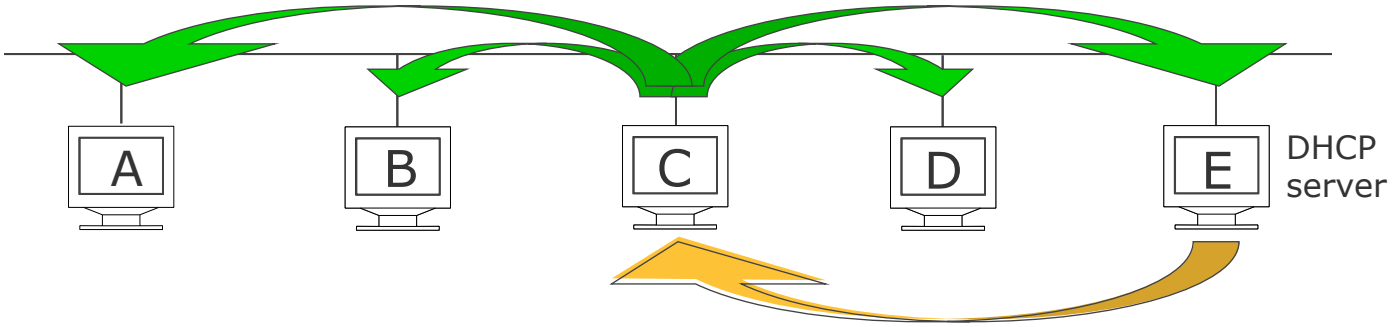


- Additionally to the IP address configuration, DHCP can also configure the subnet mask, default router, DNS server... DHCP can be used for full host configuration.
- DHCP also works if server and client are on the same physical network (no relay)

DHCP Discovery

- DHCP specified for IPv4 in RFC 2131 (published in 1997)
- Phase 1: Service discovery

DHCP Discover (broadcast) — my MAC address is X, I need an IP address etc.

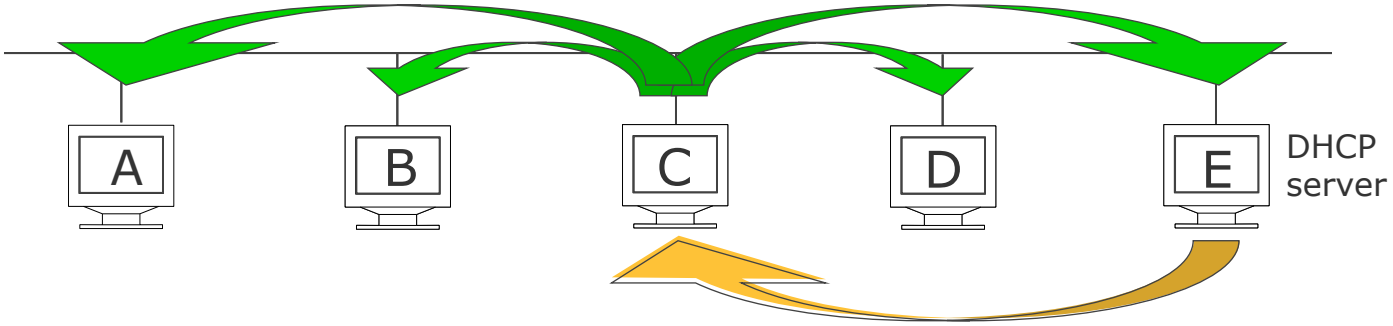


DHCP Offer (unicast) — Here's an IP address (+netmask + default router…)

DHCP: IP address assignment

- Phase 2: IP address assignment

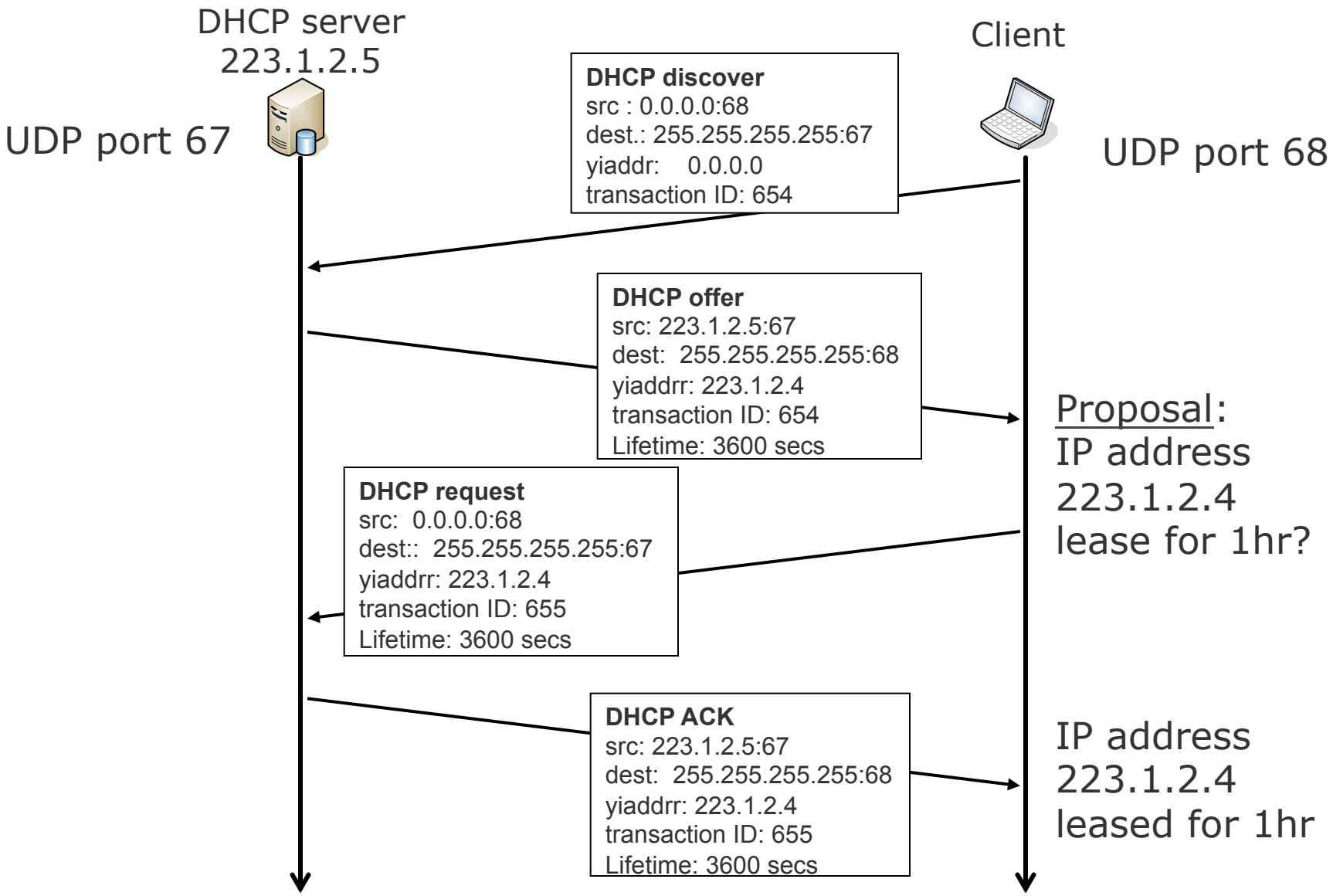
DHCP Request (broadcast) — my MAC address is X, I want to use IP address Y



DHCP Ack (unicast) — Yes you can use IP address Y

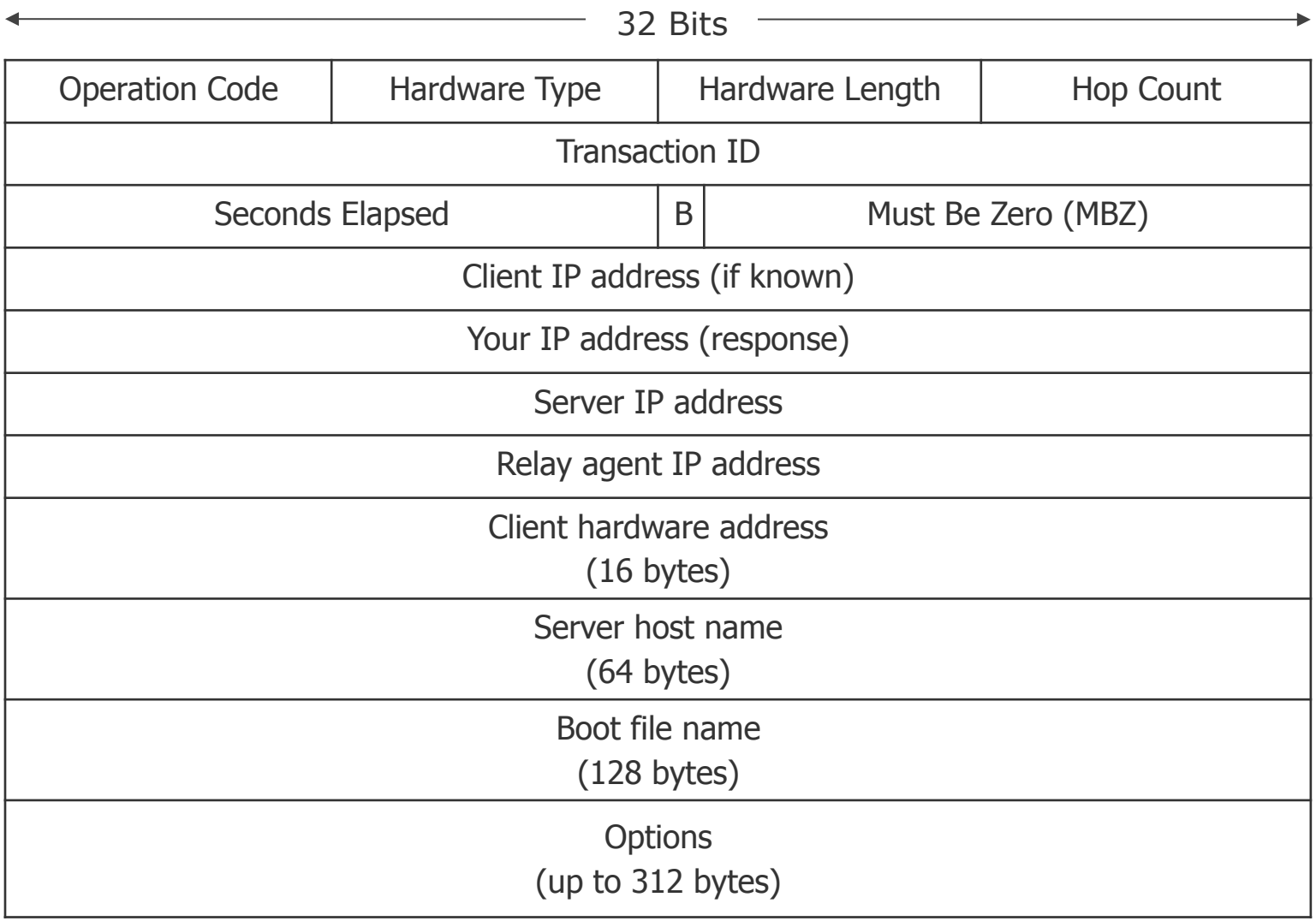
- Phase 3: IP address release (graceful shutdown)
- IP address may be permanently assigned or leased for limited period of time
  - After lease expiration, need to explicitly renew the lease, or relinquish address
    - Lease time: any value between 15 minutes to 1 year
  - Short lease useful for mobile devices
    - reduces the total number of addresses used in an organization

Example DHCP Client-Server Transaction



DHCP Packet Format

- Mostly inherited from a previous protocol (BOOTP)



DHCP Message Types

- 100+ options defined! e.g. subnet mask, name server, hostname, printer…
- Important option: Message Type (has to appear in every message)

DHCP Message	Use
DHCPDISCOVER	Client broadcast to locate available servers
DHCPOFFER	Server to client response offering configuration parameters
DHCPREQUEST	Client broadcast requesting offered parameters
DHCPDECLINE	Client to server notification that IP address is in use
DHCPACK	Server to client response confirming a request
DHCPNAK	Server to client response denying a request
DHCPRELEASE	Client to server request to relinquish IP address
DHCPINFORM	Client to server request for configuration parameters

Basic DHCP Server Logic

- Server stores pool of available IP addresses, mapping between clients/leases
- Server stores local configuration parameters
- Basic server logic:

Event	Action Taken
DHCPDISCOVER	If current lease for client exists, send DHCPOFFER Else, if IP address available, send DHCPOFFER Else, do nothing
DHCPREQUEST	If IP address available, send DHCPACK Else, send DHCPNAK
DHCPDECLINE	Mark IP address unavailable, notify network administrator
DHCPRELEASE	Mark IP address available, delete lease
DHCPINFORM	Send DHCPACK with configuration parameters
Lease expiration	Mark IP address available, delete lease

More DHCP

- DHCP uses UDP
  - Unreliable!
  - All the retransmissions are client-driven
- DHCP for IPv6
  - Also called DHCPv6
  - Specified in RFC 3315
- DHCP's security aspects
  - Detect and deal with unauthorized clients
  - Malicious clients could exhaust DHCP server address pool
  - Malicious server could provide incorrect/malicious configuration parameters

IPv6: NDP and SLAAC

- Neighbor Discovery Protocol (NDP) used instead of ARP in IPv6
  - Specified in RFC 2461
- NDP is actually used for much more than address resolution:

◆ neighbor unreachability detection (NUD)	◆ on-link prefix discovery	◆ link parameter discovery (MTU etc.)
◆ IPv6 address autoconfiguration	◆ next hop determination	◆ redirect (indicate better next hop)
◆ router discovery		◆ duplicate address detection (DAD)

- State-Less Address Auto-Configuration (SLAAC)
  - Specified in RFC 2462
  - Stateless equivalent for DHCP
  - **IPv6 address configuration without server**, based on NDP

NDP Message Types and Format

- NDP Message types
  - Router Solicitation (**RS**) : trigger RA
  - Router Advertisement (**RA**) : router+link information
  - Neighbor Solicitation (**NS**) : equivalent of ARP request
  - Neighbor Advertisement (**NA**) : equivalent of ARP reply
  - Redirect : router informs host of better first hop towards a destination
- All these messages are accomplished via ICMP packets
  - Basic format:

IPv6 *vs* IPv4 for Neighbor Functions

IPv4	IPv6
ARP Request	ICMPv6 Neighbor Solicitation
ARP Reply	ICMPv6 Neighbor Advertisement
ARP cache	Neighbor cache
Gratuitous ARP	Duplicate address detection
ICMPv4 Router Solicitation	ICMPv6 Router Solicitation
ICMPv4 Router Advertisement	ICMPv6 Router Advertisement
ICMPv4 Redirect message	ICMPv6 Redirect message

- Remark: one advantage of using ND is that it is a pure network layer protocol (above IP) contrary to ARP.
  - This implies that one can also apply network layer security mechanisms with ND, thus solving problems equivalent to ARP spoofing

ND: Data structures Maintained by Host

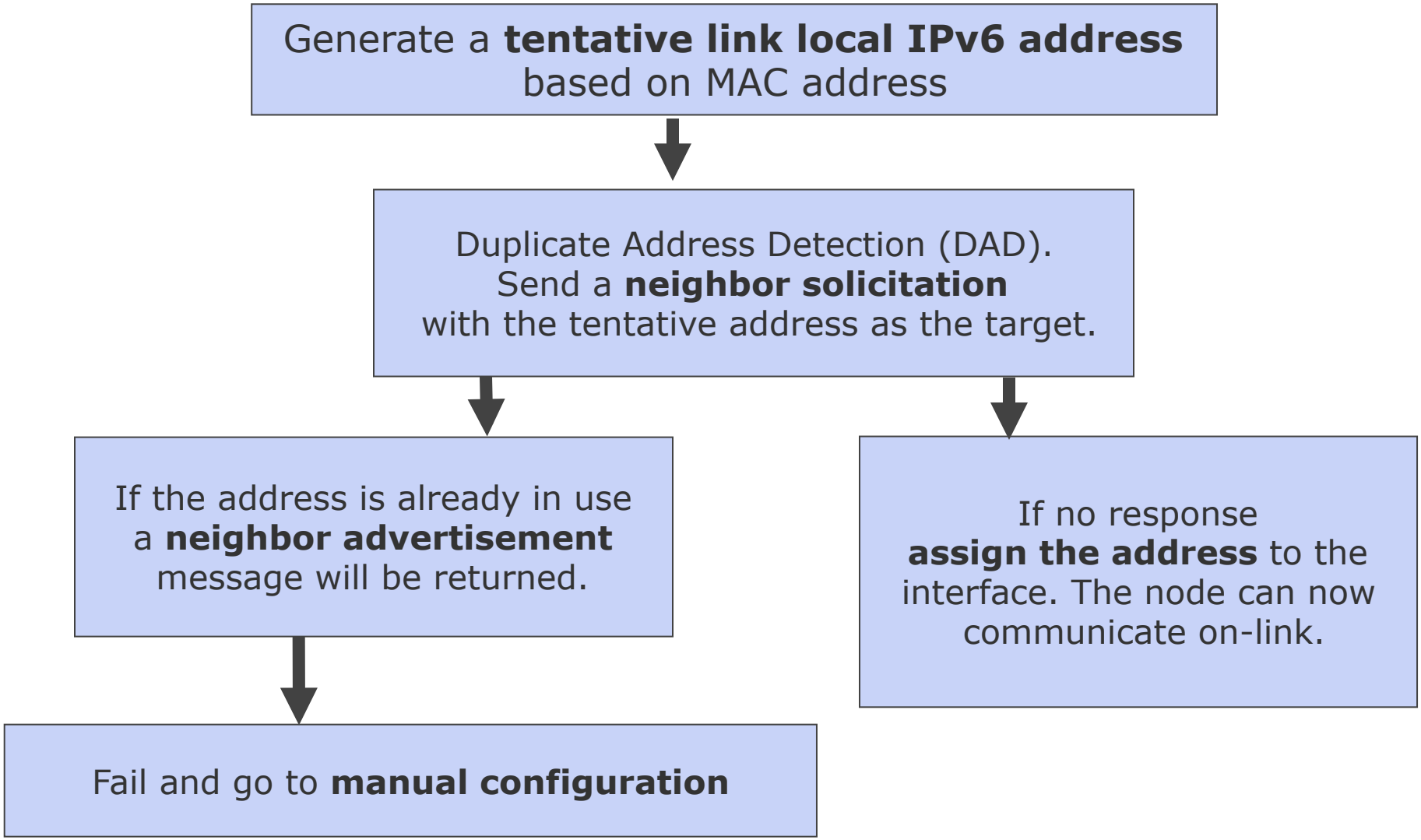
Destination Cache		
Destination	Next-Hop Address	PMTU

Neighbor Cache		
Next-Hop Address	Link Layer Address	State

Prefix List

Default Router List

SLAAC Phase 1: Acquire Link Local IPv6 Address



From MAC address to IP Host Identifier

- Converting the EUI 48 bit MAC address from the NIC, into a EUI 64 bit dynamically assigned host identifier.
  - EUI= Extended Unique Identifier

EUI-48 identifier ➡ EUI-64 identifier

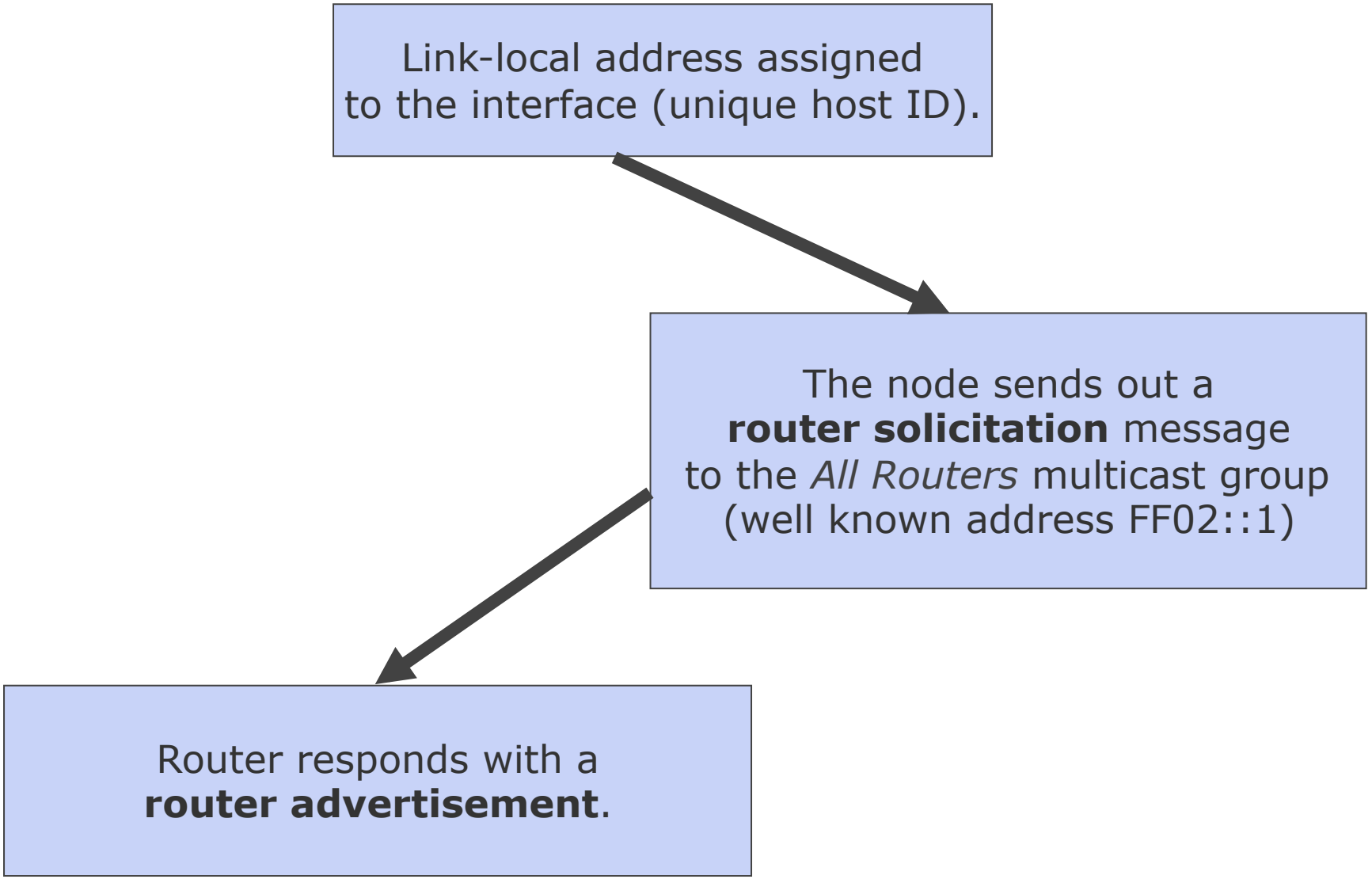
- leftmost 24 bits of the MAC form the leftmost 24 bits of EUI-64 host ID
- rightmost 24 bits of the MAC form the rightmost 24 bits of the EUI-64 host ID
- insert the constant FFFE (hex value) in the middle 16 bits
- tweak left-most bit 7 from zero to one (the "universal/local" bit in MAC address)

- Prepend the well known link local prefix FE80:: and there you go!
  - we have a tentative IPv6 link local address ➡ FE80:: EUI-64 identifier

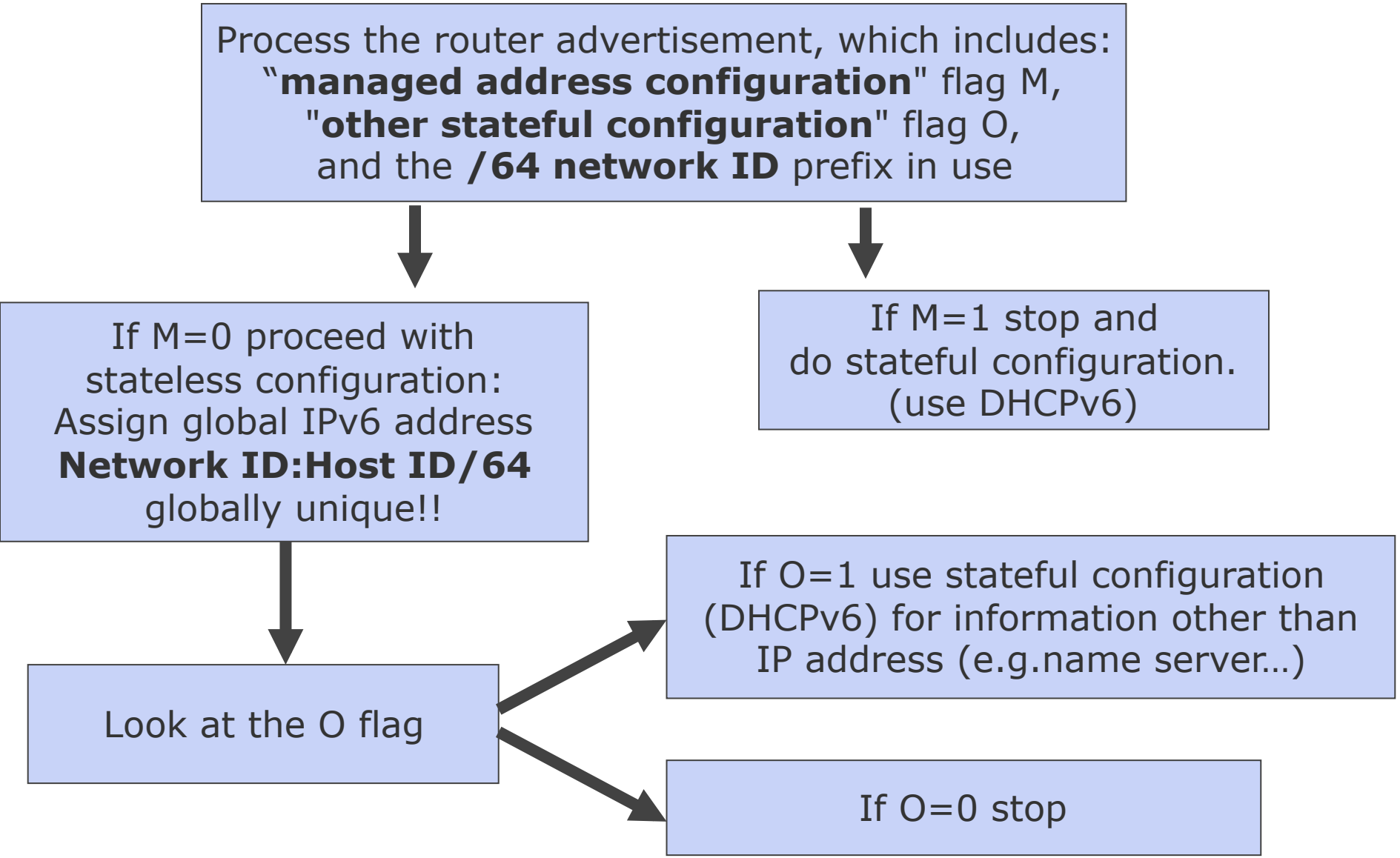
- With Duplicate Address Detection (DAD) host identifier is verified unique
  - Link Local address is unique locally, configured and usable

- We can move on to the next phase of SLAAC: contact the router on-link

SLAAC Phase 2: Contact Router



SLAAC Phase 3: Configure Global IPv6 Address

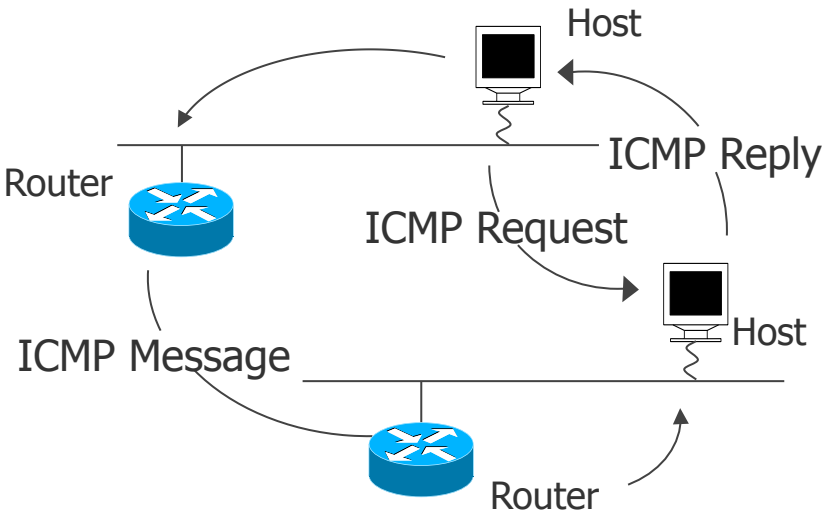


CONTENT of this CHAPTER

- Fundamental Goals of the Network Layer
- Network Layer Addressing
  - IPv4 Addressing
  - IPv6 Addressing
- Internet Protocol Design
  - Internet Protocol version 4
  - Internet Protocol version 6
  - IPv6 Migration: Transition and Coexistence
- IP Address Mapping & Assignment (ARP, RARP, DHCP, NDP/SLAAC)
- Control, Error Management and Diagnostic (ICMP)
- Network Address Translation (NAT)
- Routing (Concepts, OSPF, BGP…)
- Multicast (Concepts, IGMP…)

Internet Control Message Protocol (ICMP)

- ICMP is a control protocol of layer 3
  - Specified for IPv4 in RFC 792
- Use case 1: if a router cannot forward a packet, the source can be informed about it via an ICMP message.
- Use case 2: ping uses ICMP request/reply.



ICMP Message: Transmission of status information and control messages

ICMP Request: status request
ICMP Reply: status reply

ICMP Message Format

- ICMP message fields
  - **Type**: purpose of the ICMP message
    - 40+ types defined
    - Types 41 – 255 for future use
  - **Code**: additional information about the condition
    - Rarely used
  - **Checksum**: same type of checksum as used for IP header

- Message categories
  - **Error messages**: to report an event. These messages do not have a response.
    - Error messages include the IP header that generated the error!
    - Example: destination unreachable msg
  - **Queries**: To get some information from a node. These messages have a matching response.

1 byte	1 byte	2 byte
Type	Code	Checksum
ICMP Data (content and format depends on Type)		

General format of ICMP messages

1 byte	1 byte	2 byte
Type =3	Code	Checksum
Unused (all 0 bits)		
IP Header (20 bytes) and First 8 bytes of original packet data		

Format of Destination Unreachable ICMP message

ICMP Message Types

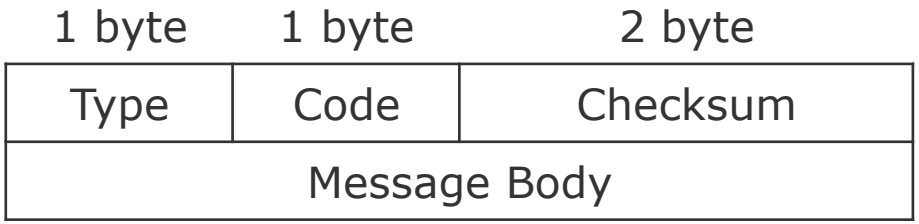
- ICMP transmits error and control messages at the network layer.
  - ICMP messages are encapsulated in IP packets

- Type/code indicates the type (and format) of the message, e.g.:

Type	Meaning
0	Destination unreachable (packet cannot be sent)
3	Echo request/reply (status request, e.g. for ping)
4	Source Quench (Choke packet, data rate reduction)
11	Time exceeded for datagram (TTL = 0, the packet is discarded)
12	Parameter problem on datagram (A header field is set wrongly)
15/16	Information Request/Reply
30	Trace route (Trace the network path)

ICMP with IPv6

- ICMPv6 specified in RFC 4443



General format of ICMPv6 messages

- ICMPv6 is used for much more purposes than ICMPv4
  - ICMPv6 messages for automatic address configuration (NDP/SLAAC)
  - ICMPv6 messages automatic path MTU discovery
  - New PacketTooBig message is sent to the source, since IPv6 routers do not fragment
  - There is no Source Quench in ICMPv6
  - IGMP for multicast is included in ICMPv6 (Multicast Listener Discovery, RFC4604)
  - ICMPv6 messages to help detect nonfunctioning routers and inactive partner hosts
  - Even routing! RPL uses ICMP (see RFC 6550)

Common Tools based on ICMP: Route

- Route
  - Display and set routing table entries

```
x:\>route PRINT
===========================================================================
Interface List
0x1 ........................... MS TCP Loopback interface
0x2 ...00 1a a0 b7 41 39 ...... Broadcom NetXtreme 57xx Gigabit Controller
===========================================================================
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0     160.45.114.1   160.45.114.21       10
        127.0.0.0        255.0.0.0        127.0.0.1       127.0.0.1        1
     160.45.114.0  255.255.255.192    160.45.114.21   160.45.114.21       10
    160.45.114.21  255.255.255.255        127.0.0.1       127.0.0.1       10
   160.45.255.255  255.255.255.255    160.45.114.21   160.45.114.21       10
        224.0.0.0        240.0.0.0    160.45.114.21   160.45.114.21       10
  255.255.255.255  255.255.255.255    160.45.114.21   160.45.114.21        1
Default Gateway:      160.45.114.1
===========================================================================
Persistent Routes:
  None
```

Common Tools based on ICMP: Ping

- Ping
  - Tool to test whether a host is reachable/alive

```
x:\>ping www.google.com

Pinging www.l.google.com [209.85.135.104] with 32 bytes of data:

Reply from 209.85.135.104: bytes=32 time=26ms TTL=243
Reply from 209.85.135.104: bytes=32 time=24ms TTL=243
Reply from 209.85.135.104: bytes=32 time=23ms TTL=243
Reply from 209.85.135.104: bytes=32 time=24ms TTL=243

Ping statistics for 209.85.135.104:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 23ms, Maximum = 26ms, Average = 24ms
```

Common Tools based on ICMP: Traceroute

- Traceroute/tracert
  - Tool to determine the used path in the Internet

```
x:\>tracert www.google.com

Tracing route to www.l.google.com [209.85.135.104]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  router-114.mi.fu-berlin.de [160.45.114.1]
  2    <1 ms    <1 ms    <1 ms  pollux.mi.fu-berlin.de [160.45.113.243]
  3    <1 ms    <1 ms    <1 ms  zedat.router.fu-berlin.de [160.45.252.181]
  4   307 ms    <1 ms    <1 ms  ice2.spine.fu-berlin.de [130.133.98.3]
  5    <1 ms    <1 ms    <1 ms  xr-zib1-ge8-3.x-win.dfn.de [188.1.33.46]
  6     *        *        *     Request timed out.
  7    18 ms    15 ms    15 ms  zr-fra1-te0-0-0-3.x-win.dfn.de [80.81.192.222]
  8    28 ms    15 ms    15 ms  de-cix10.net.google.com [80.81.192.108]
  9    16 ms    16 ms    15 ms  209.85.255.172
 10    24 ms    23 ms    24 ms  72.14.233.106
 11    24 ms    24 ms    23 ms  66.249.94.83
 12    26 ms    27 ms    26 ms  209.85.253.22
 13    24 ms    24 ms    24 ms  mu-in-f104.google.com [209.85.135.104]

Trace complete.
```

Common Tools based on ICMP: Pathping (1)

- Pathping
  - Combination of ping and traceroute (only for Windows)
  - mtr (my traceroute) for Linux

```
x:\>pathping www.google.com

Tracing route to www.l.google.com [209.85.135.147]
over a maximum of 30 hops:
  0  KAFPOT.pcpool.mi.fu-berlin.de [160.45.114.21]
  1  router-114.mi.fu-berlin.de [160.45.114.1]
  2  pollux.mi.fu-berlin.de [160.45.113.243]
  3  zedat.router.fu-berlin.de [160.45.252.181]
  4  ice2.spine.fu-berlin.de [130.133.98.3]
  5  xr-zib1-ge8-3.x-win.dfn.de [188.1.33.46]
  6  zr-pot1-te0-7-0-2.x-win.dfn.de [188.1.145.138]
  7  zr-fra1-te0-0-0-3.x-win.dfn.de [80.81.192.222]
  8  de-cix10.net.google.com [80.81.192.108]
  9  209.85.255.172
 10  72.14.233.106
 11  66.249.94.85
 12  209.85.253.26
 13  mu-in-f147.google.com [209.85.135.147]

Computing statistics for 325 seconds…
```

See next slide

Common Tools based on ICMP: Pathping (2)

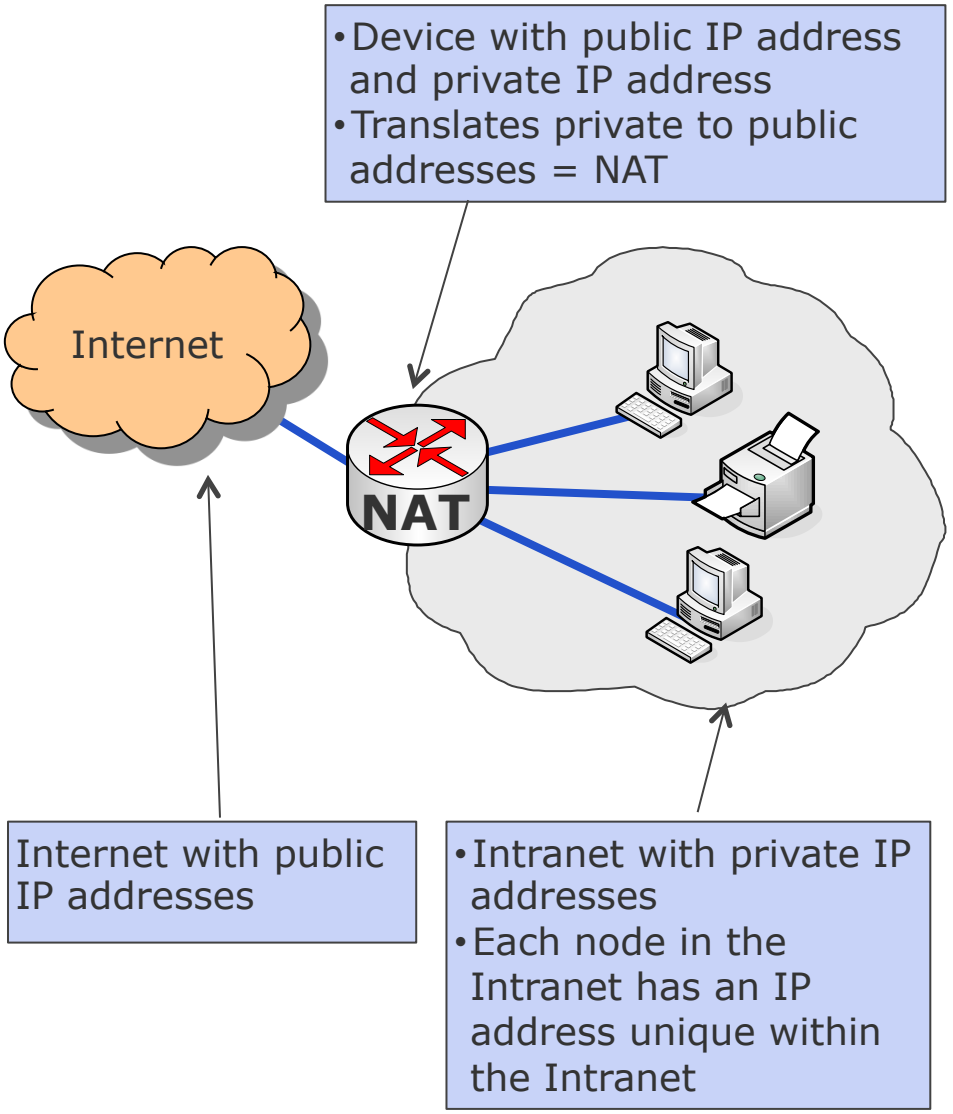
```
Computing statistics for 325 seconds...
            Source to Here   This Node/Link
Hop  RTT    Lost/Sent = Pct  Lost/Sent = Pct  Address
  0                                           KAFPOT.pcpool.mi.fu-berlin.de [160.45.114.21]
                                 0/ 100 =  0%   |
  1    0ms     0/ 100 =  0%     0/ 100 =  0%  router-114.mi.fu-berlin.de [160.45.114.1]
                                 0/ 100 =  0%   |
  2    0ms     0/ 100 =  0%     0/ 100 =  0%  pollux.mi.fu-berlin.de [160.45.113.243]
                                 0/ 100 =  0%   |
  3    3ms     0/ 100 =  0%     0/ 100 =  0%  zedat.router.fu-berlin.de [160.45.252.181]
                                 0/ 100 =  0%   |
  4    0ms     0/ 100 =  0%     0/ 100 =  0%  ice2.spine.fu-berlin.de [130.133.98.3]
                                 0/ 100 =  0%   |
  5    0ms     0/ 100 =  0%     0/ 100 =  0%  xr-zib1-ge8-3.x-win.dfn.de [188.1.33.46]
                                 0/ 100 =  0%   |
  6    1ms     0/ 100 =  0%     0/ 100 =  0%  zr-pot1-te0-7-0-2.x-win.dfn.de [188.1.145.138]
                                 0/ 100 =  0%   |
  7   14ms    93/ 100 = 93%    93/ 100 = 93%  zr-fra1-te0-0-0-3.x-win.dfn.de [80.81.192.222]
                                 0/ 100 =  0%   |
  8   17ms     0/ 100 =  0%     0/ 100 =  0%  de-cix10.net.google.com [80.81.192.108]
                                 0/ 100 =  0%   |
  9   20ms     0/ 100 =  0%     0/ 100 =  0%  209.85.255.172
                                 0/ 100 =  0%   |
 10   24ms     0/ 100 =  0%     0/ 100 =  0%  72.14.233.106
                                 0/ 100 =  0%   |
 11   29ms     0/ 100 =  0%     0/ 100 =  0%  66.249.94.85
                                 0/ 100 =  0%   |
 12   28ms     0/ 100 =  0%     0/ 100 =  0%  209.85.253.26
                                 1/ 100 =  1%   |
 13   24ms     1/ 100 =  1%     0/ 100 =  0%  mu-in-f147.google.com [209.85.135.147]

Trace complete.
```

CONTENT of this CHAPTER

- Fundamental Goals of the Network Layer
- Network Layer Addressing
  - IPv4 Addressing
  - IPv6 Addressing
- Internet Protocol Design
  - Internet Protocol version 4
  - Internet Protocol version 6
  - IPv6 Migration: Transition and Coexistence
- IP Address Mapping & Assignment (ARP, RARP, DHCP, NDP/SLAAC)
- Control, Error Management and Diagnostic (ICMP)
- Network Address Translation (NAT)
- Routing (Concepts, OSPF, BGP…)
- Multicast (Concepts, IGMP…)

Network Address Translation (NAT)

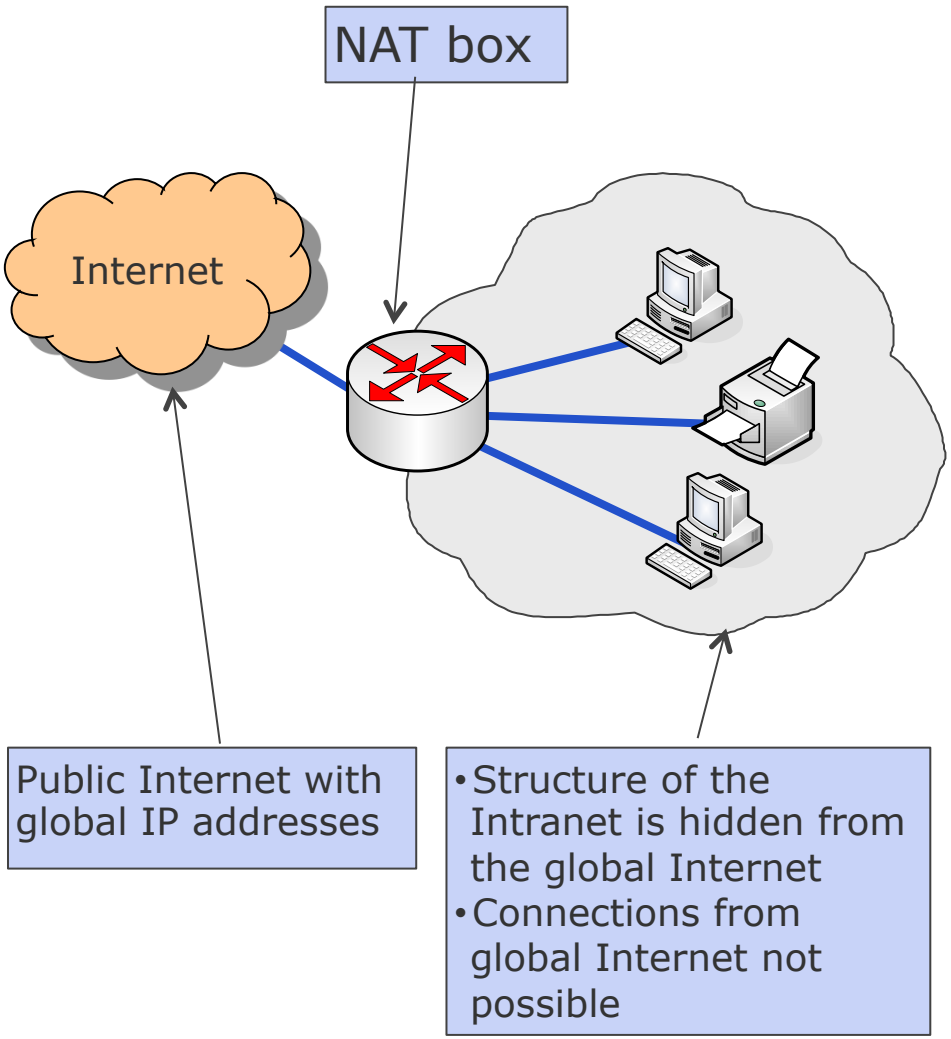
- Problem: each device needs an IP address, but IP addresses are scarce!
- Solution: reserve a well-known block of IP addresses for site-local use
  - Dedicated blocks of **private IP addresses**
    - 10.0.0.0 – 10.255.255.255
    - 172.16.0.0 – 172.31.255.255
    - 192.168.0.0 – 192.168.255.255
  - Other addresses are **public IP addresses**
- Advantage: Scalability! Reuse the same private IP addresses within another site!
- Drawback: To communicate with a computer from another site, still need a public address!
- **Network Address Translation** (NAT) is a technique to deal with this drawback
  - Enables several computers to share the same public IP address



Network Address Translation (NAT)

Basic idea of NAT:

- assign only a few public IP addresses to a company.
- These are known by a NAT box (often collocated with the router)
- when packets leave the network, address translation takes place: the NAT box exchanges the private address with a public address
- when packets enter the network, the reverse translation takes place



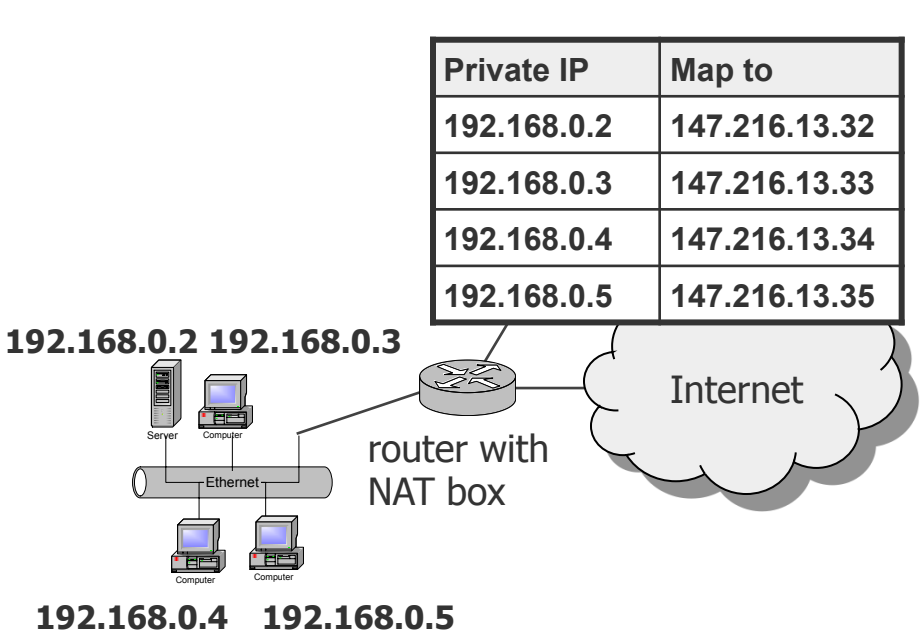
- Remark: packets with private IP addresses are not forwarded by routers
- Good side effect of NAT: internal network structure is hidden!
  ➡more security
- Bad side effect of NAT: bye-bye network transparency! (RFC4924)
  ➡communication with outside MUST be initiated from inside.

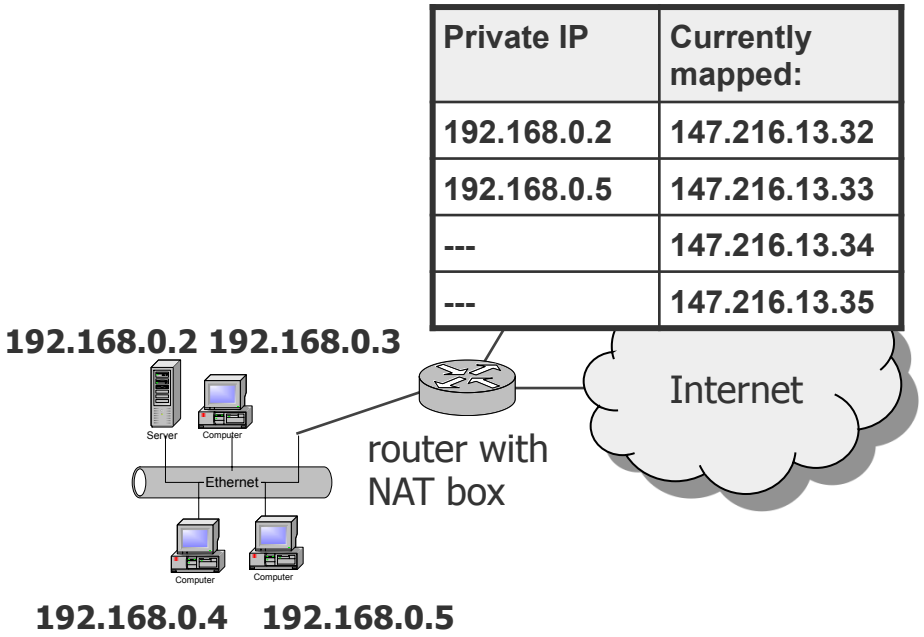
NAT Variants

- NAT is available in several “variants”, known under different names:
  - Basic NAT
  - Static NAT
  - One-to-many NAT
  - NAPT (Network Address Port Translation)
  - Masquerading
  - Carrier Grade NAT
  - NAT64

Static NAT (also known as Basic NAT)

- Principle: hide structure using one public IP address per private IP address
  - Permanent mapping: you need as much public addresses as private ones, otherwise no re-translation is possible when a reply arrives
  - Temporary mapping: you manage a pool of public IP addresses and assign one dynamically, when a request is sent out
    - Advantage: need less public addresses most of the time
    - Drawback: at peak host traffic, need almost as many public addresses as private ones

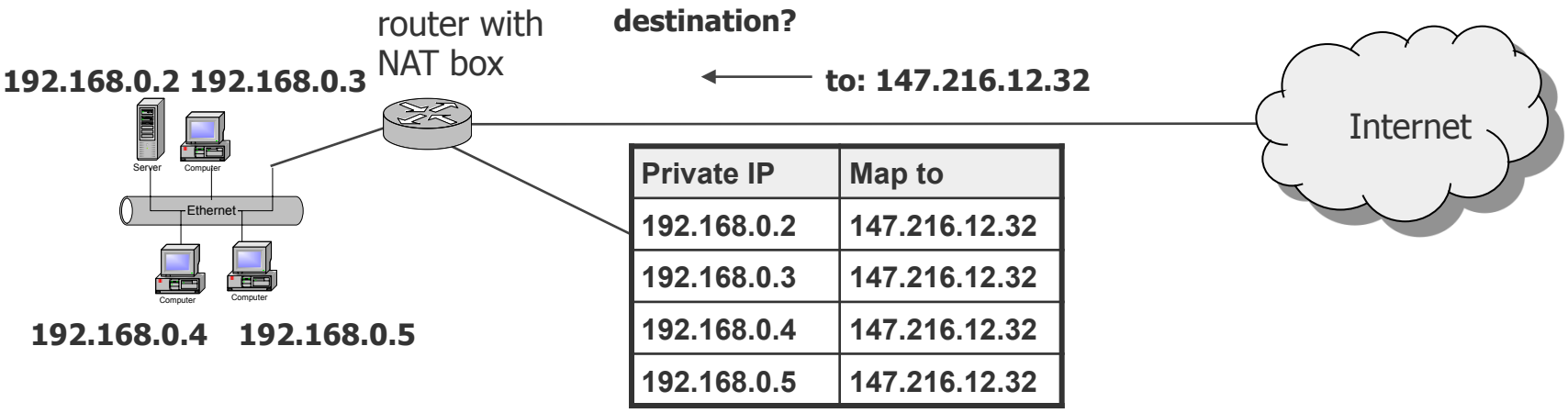




One-to-Many NAT (also known as Masquerading)

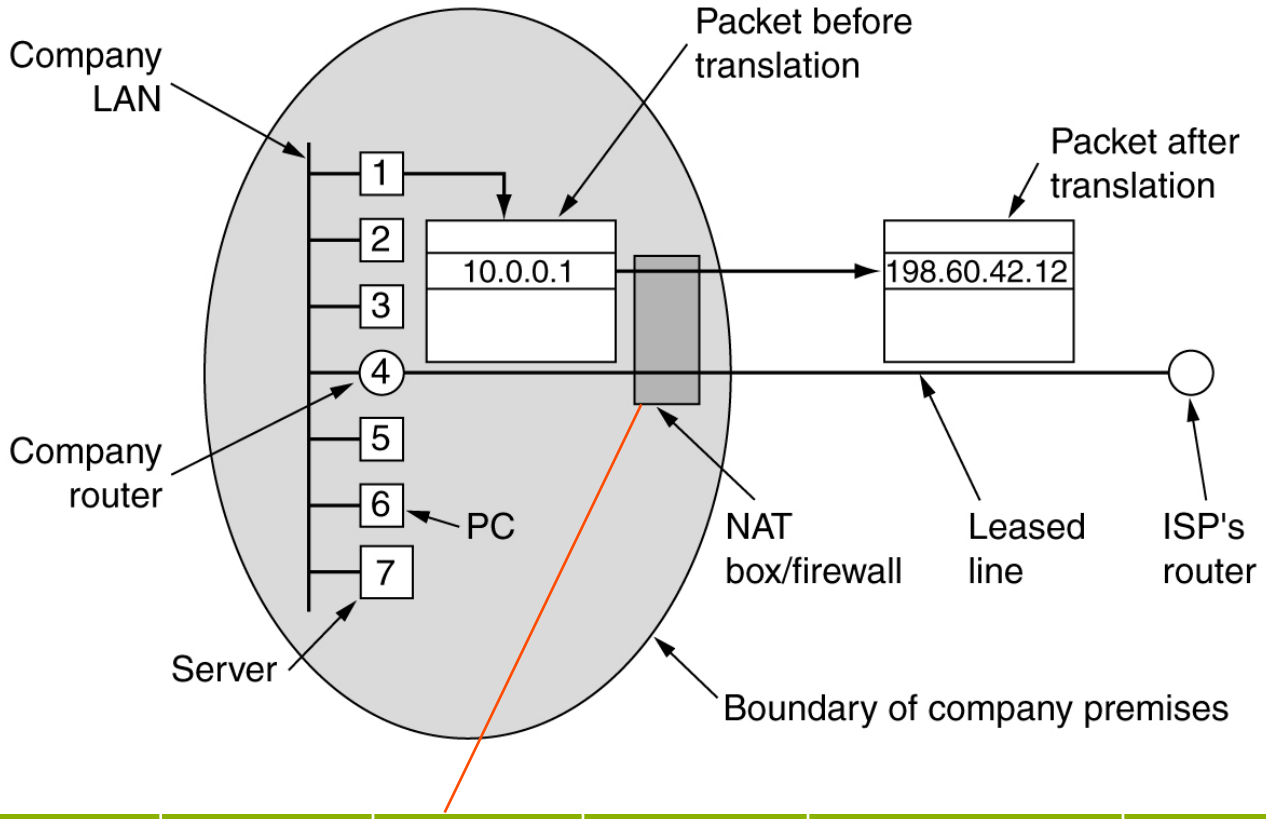
- Problem: Static NAT helps to hide the network structure, but not to save on the number of public addresses needed

- Solution: One-to-many NAT
  - Translates several local addresses into the same public address
  - But now some more details are necessary for "demultiplexing" to deliver a response
  - Most common technique is to vary ports: Network Address Port Translation (NAPT)



Private IP	Map to
192.168.0.2	147.216.12.32
192.168.0.3	147.216.12.32
192.168.0.4	147.216.12.32
192.168.0.5	147.216.12.32

Network Address Port Translation (NAPT)



Protocol	Port (local)	IP (local)	Port (global)	IP (global)	IP (destination)	Port (destination)
TCP	1066	10.0.0.1	1066	198.60.42.12	147.216.12.221	21
TCP	1500	10.0.0.7	1500	198.60.42.12	207.17.4.21	80

Problems with NAPT

- NAPT works well with communication initiated from within the Intranet.
- But: how could someone from outside give a request to the Intranet (e.g. to the web server)?
  - NAPT box needs to be a bit more "intelligent"
  - Some static information has to be stored, e.g., "each incoming request with port 80 has to be mapped to the private address of the web server"
  - Still problems with some applications (e.g. instant messaging)
  - Only more public IP addresses really helps to solve the problem

Even more NAT versions

- Carrier Grade NAT (CGN)
  - Also known as LSN (Large Scale NAT)
  - Use a NAT box for a larger number of customers, e.g., a street, a suburb to further mitigate IPv4 address exhaustion
  - Again – breaks end-to-end principle, difficult to use well-known ports etc.

- NAT64 (specified in RFC 6146)
  - Mechanism to connect IPv6 hosts to IPv4 servers
  - IPv6 client embeds IPv4 address (RFC 6052)
  - NAT maps IPv6 to IPv4

So… what do we REALLY think about NAT?

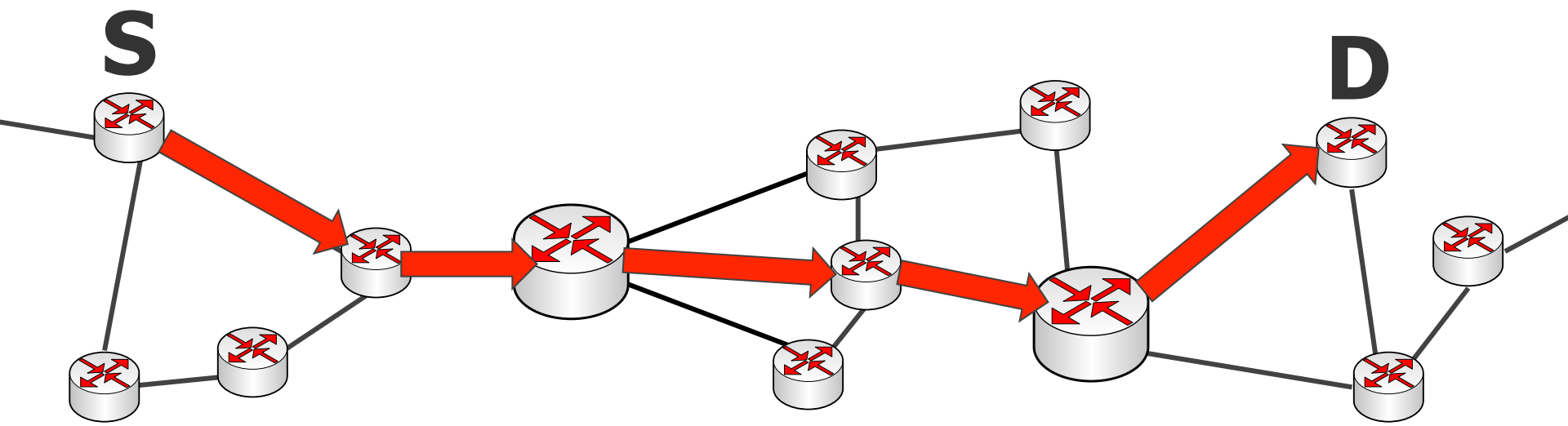
- There is a controversy about NATs
- On one hand, NAT saved the Internet by allowing it to scale
  - Allowing massive reuse of private addresses
  - "Doing more" with less public addresses
- On the other hand, NAT killed a fundamental principle of the Internet
  - End-to-end communication can only be initiated from one side
- Some hope that massive IPv6 adoption will eliminate the need for NAT and restore fundamental end-to-end communication as it should be
  - Access your connected things back at home, from anywhere, without headaches!
  - Only a dream for now…

CONTENT of this CHAPTER

- Fundamental Goals of the Network Layer
- Network Layer Addressing
  - IPv4 Addressing
  - IPv6 Addressing
- Internet Protocol Design
  - Internet Protocol version 4
  - Internet Protocol version 6
  - IPv6 Migration: Transition and Coexistence
- IP Address Mapping & Assignment (ARP, RARP, DHCP, NDP/SLAAC)
- Control, Error Management and Diagnostic (ICMP)
- Network Address Translation (NAT)
- Routing (Concepts, OSPF, BGP…)
- Multicast (Concepts, IGMP…)

Finding Your Path in the Maze

- The most important tasks of the network layer are:
  - Task 1: Providing identifiers that are valid throughout the internetwork
  - Task 2: Providing paths to arbitrary destinations in the internetwork



- In the Internet:
  - Task 1 is accomplished by **IP addresses**
  - Task 2 is accomplished by **routing**

Content of this Section

- Prerequisites to routing
- Routing *vs* forwarding
- Routing schemes: static/dynamic, source/hop-by-hop, flat/hierarchical…
- Basic concepts for routing: flooding, metrics, graphs
- Fundamental routing algorithms
- Standard routing protocols

Content of this Section

- Prerequisites to routing
- Routing *vs* forwarding
- Routing schemes: static/dynamic, source/hop-by-hop, flat/hierarchical…
- Basic concepts for routing: flooding, metrics, graphs
- Fundamental routing algorithms
- Standard routing protocols

Routing: Prerequisites

- Before you consider finding your way through the maze:
  - Make sure each place/crossing has an unambiguous, unique name
- Similarly, before routing, make sure each nodes has a unique IP address
  - Remark 1: It's OK if a node has several unique IP addresses
  - Remark 2: The scope of validity of the IP address should be appropriate
    - It should be routable, i.e. not link-local, or private etc…

Content of this Section

- Prerequisites to routing
- Routing *vs* forwarding
- Routing schemes: static/dynamic, source/hop-by-hop, flat/hierarchical…
- Basic concepts for routing: flooding, metrics, graphs
- Fundamental routing algorithms
- Standard routing protocols

Routing *vs* Forwarding

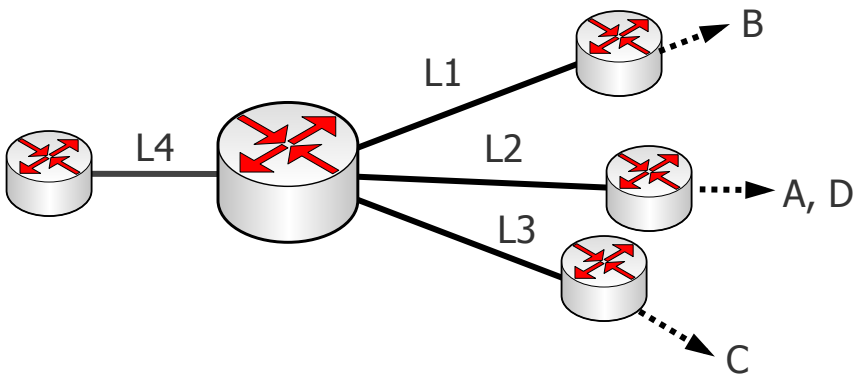
- DO NOT confuse the terms *routing* and *forwarding* (as most people do)

- **Forwarding is a local process**, which runs on a single machine
  - Moving incoming packets from input interface to appropriate output interface, towards their destination

- **Routing is a distributed process**, which requires the cooperation of several devices in the network (i.e. all the routers)
  - monitoring & exchanging enough network topology information between routers to allow correct forwarding decisions on each router

Routing Table *vs* Forwarding Table

- DO NOT confuse the terms *routing table* and *forwarding table*
- Routing tables and forwarding tables are data sets stored on each router
- The forwarding table is the final product of the routing process
  - It is derived from the raw topology information acquired by the routing scheme



Forwarding Table	
To destination	By connection
A	L2
B	L1
C	L3
D	L2

- The routing table is the raw topology information acquired by routing
  - Its shape/format depends on the routing scheme in use
  - With some routing schemes, there is no routing table, only a forwarding table

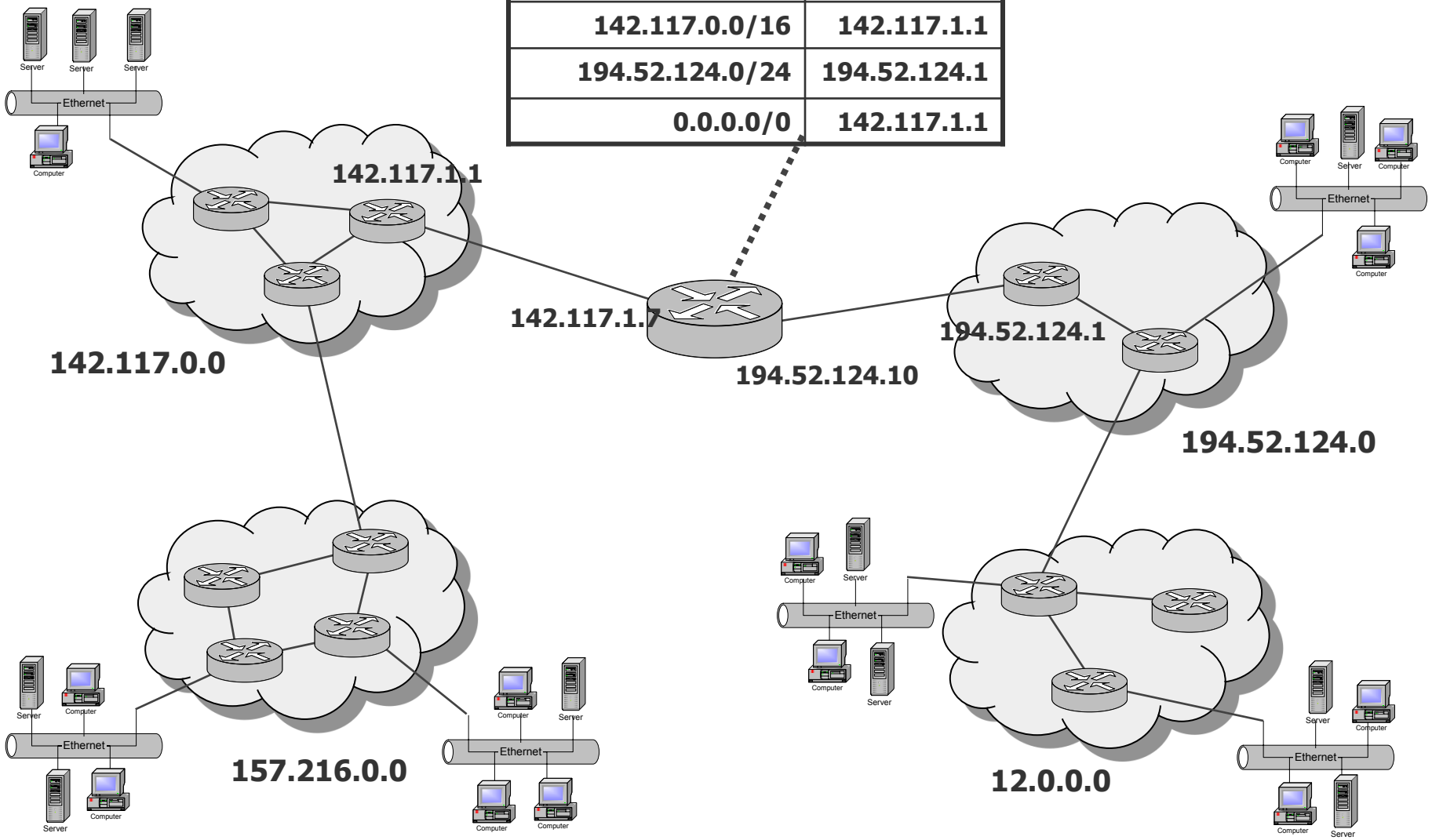
Forwarding Scheme Requirements

- A forwarding scheme moves packets from input interface to appropriate output interface
  - Does it according to what is currently in the forwarding table

- Requirements for a forwarding scheme:
  - Scalability in terms of speed of IO transfer
  - Fast table lookup
  - Efficient, sophisticated data structures

Forwarding Tables

Destination Address	Next Hop
12.0.0.0/8	194.52.124.1
157.216.0.0/16	142.117.1.1
142.117.0.0/16	142.117.1.1
194.52.124.0/24	194.52.124.1
0.0.0.0/0	142.117.1.1



Forwarding Table Lookup

- One-dimensional key: decision depends only on the destination D
- Two-dimensional key: decision depends on the source S and destination D

D	Output

S	D	Output

- Longest prefix match (**LPM**)
- Exact label match (**ELM**)

LPM Forwarding Table	
To destination	By connection
A*	I2
AB	I1
B*	I3
BC	I2

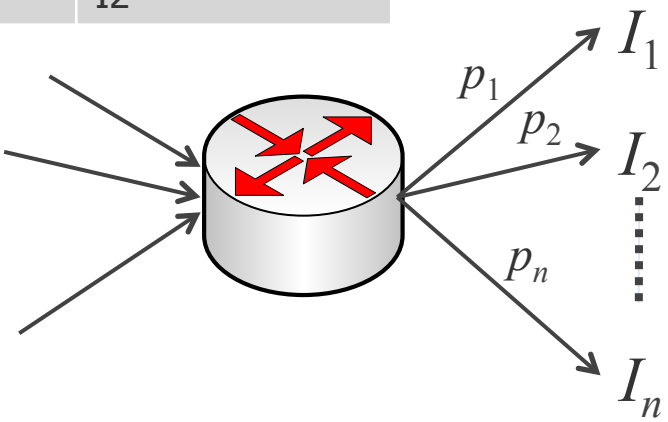
ELM Forwarding Table	
To destination	By connection
Label1	I2
Label2	I1
Label3	I3
Label4	I2

Deterministic Forwarding *vs* Probabilistic Forwarding

- Deterministic forwarding: decision based only on key lookup
- Probabilistic forwarding: decision based on key lookup and/or random

Deterministic	
To destination	By connection
A	I2
B	I1
C	I3
D	I2

Probabilistic	
To destination	By connection
A	50% I2, 50% I1
B	30% I1, 70% I3
C	I3
D	I2



- Remark 1: forwarding schemes are not really our focus here, because they are not distributed processes. We will now focus more on routing from now on.
- Remark 2: but we do care about the size of the forwarding table!! (memory constraints)

Content of this Section

- Prerequisites to routing
- Routing *vs* forwarding
- Routing schemes: static/dynamic, source/hop-by-hop, flat/hierarchical…
- Basic concepts for routing: flooding, metrics, graphs
- Fundamental routing algorithms
- Standard routing protocols

Routing Scheme Requirements

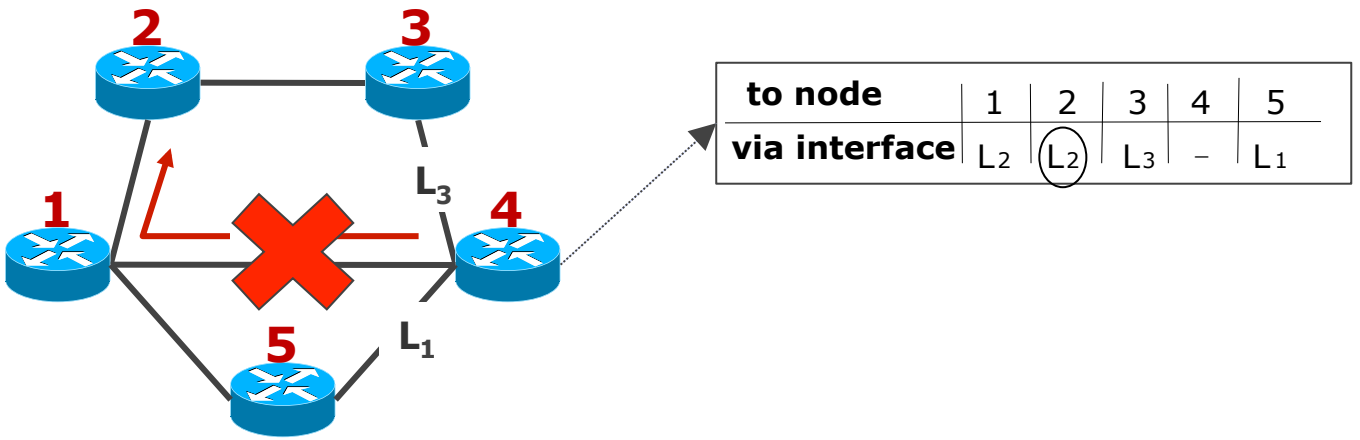
- A routing scheme monitors & exchanges network topology information between routers to allow correct forwarding decisions on each router

- Requirements for a routing scheme:
  - scalability in terms of network-wide convergence speed
  - small control overhead (memory, throughput, CPU)
  - fault-tolerance (maximum availability)
  - loop-freedom
  - (flexibility to accomodate different path requirements for different applications)

Static Routing *vs* Dynamic Routing

- **Static routing**: static forwarding tables are used within the routers, no reaction to changes in the network
  - Advantages: stable, simple
  - Drawbacks: No reaction to changing network conditions

  ➡ Link or router breakdown has catastrophic results

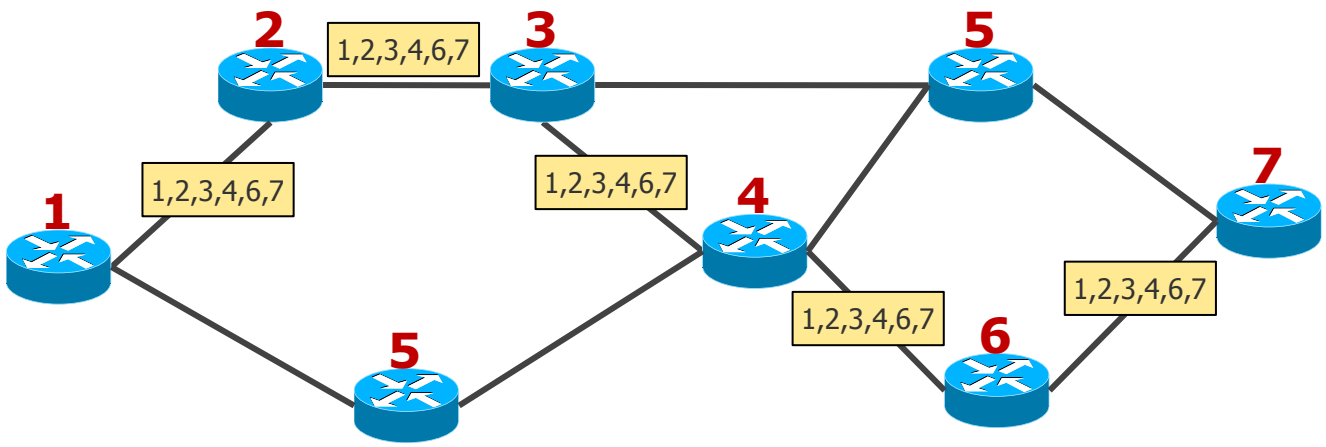


- **Dynamic routing**: routing tracks the network continuously, forwarding table are updated upon topology changes
  - Advantages: fault-tolerant, more reliability can be achieved
  - Drawbacks: more complex and more overhead

Source-Routing *vs* Hop-by-Hop Routing

- **Source routing**: the path is determined by the source node
  - Whole path is included in each packet, example:

Header	3,4,5,1	Data



  - Advantage of source routing: intermediate routers do not need state
  - Drawback of source routing: no fault tolerance if a downstream router/link breaks
  - Problem: How does the source node know the whole path?

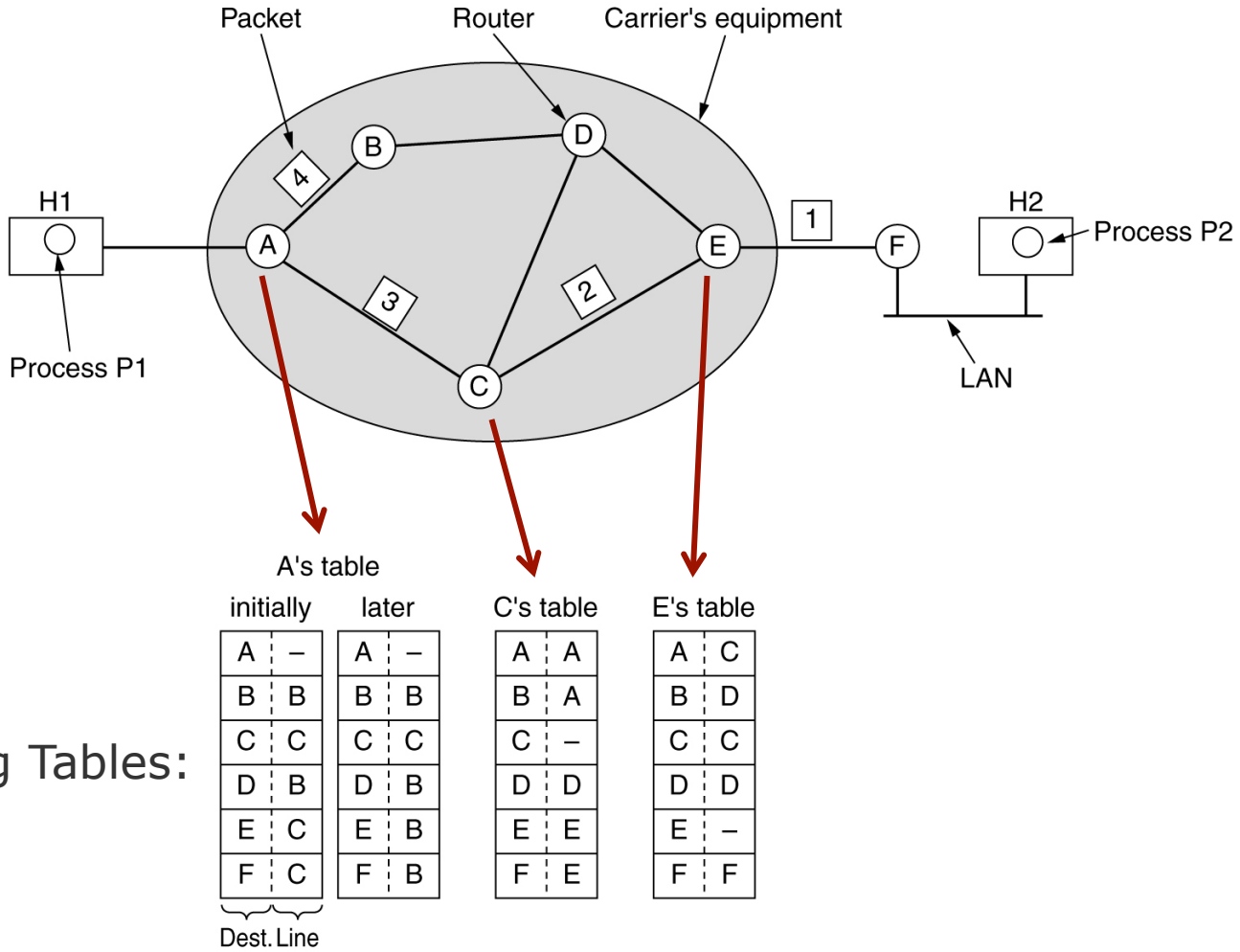
- **Hop-by-hop routing**: the path is determined by each intermediate router
  - Advantage: fault tolerance if a downstream router/link breaks
  - Drawback: intermediate routers must store state (e.g. forwarding table)

Packet Routing *vs* Virtual Circuits (VC)

- Packet routing:
  - connectionless, forwarding with longest prefix match at each hop
- Circuit switching:
  - connection-oriented, forwarding with exact match at each hop

	Packet Routing (Connectionless)	Circuit Switching (Connection-oriented)
Circuit setup	Not needed	Required
Addressing	Each packet contains the full source and destination address	Each packet contains a short VC label
Forwarding	Longest prefix match	Exact label match
Routing	Dynamic routing: routing info can be updated per packet	Static routing: routing happens only once, at VC setup
Router failure impact	None, except for packets lost during the crash	All VCs that passed through the failed router are terminated
Quality of service	Difficult	Easy if enough resources can be allocated in advance for each VC
Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC

Packet Routing: Per-Packet Behavior



Forwarding Tables:

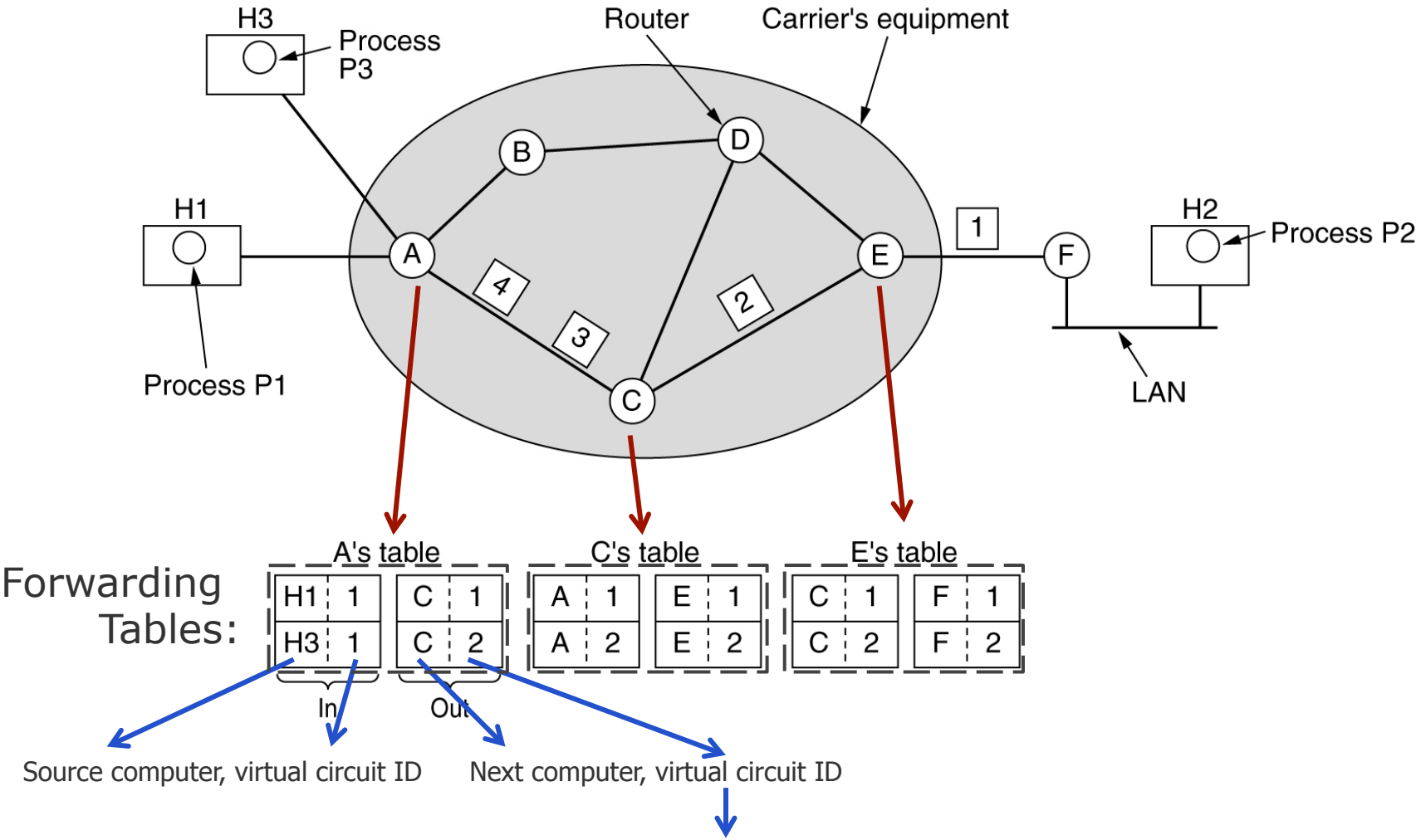
A	–
B	B
C	C
D	B
E	C
F	C

A	–
B	B
C	C
D	B
E	B
F	B

A	A
B	A
C	–
D	D
E	E
F	E

A	C
B	D
C	C
D	D
E	–
F	F

Circuit Switching: Per-flow Behavior



Also known as **Label ➡ Label Switching (MPLS)**

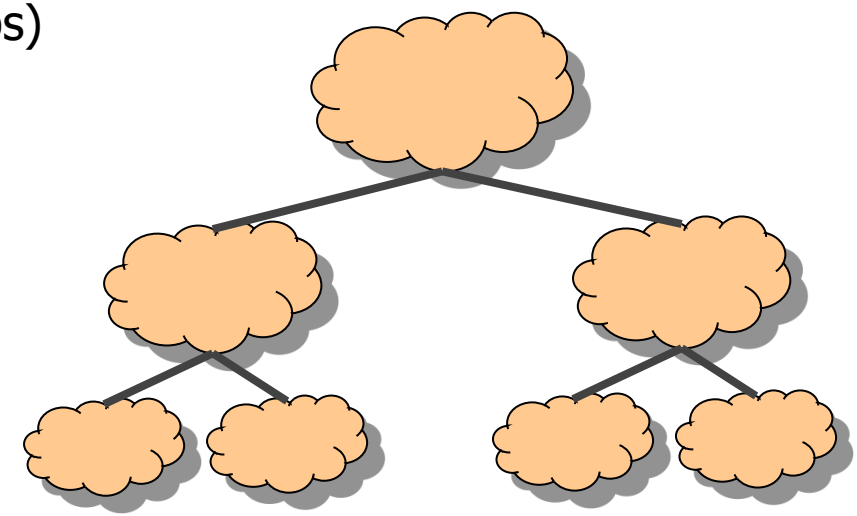
Reactive *vs* Proactive Routing

- **Reactive routing**: a path is discovered and maintained only when an application asks for it
  - Advantage: less control overhead (depending user data traffic patterns)
  - Drawback: path acquisition delay, user data buffering needs

- **Proactive routing**: all possible paths are discovered and maintained a priori
  - Advantage: no path acquisition delay, no user data buffering
  - Drawback: deterministic control overhead, whether there is user data or not

Flat Routing *vs* Hierarchical Routing

- **Hierarchical routing**: definition of separate virtual regions in the network
  - Regions are hierarchically organized, similar to a tree, with a central region as root (to avoid loops)



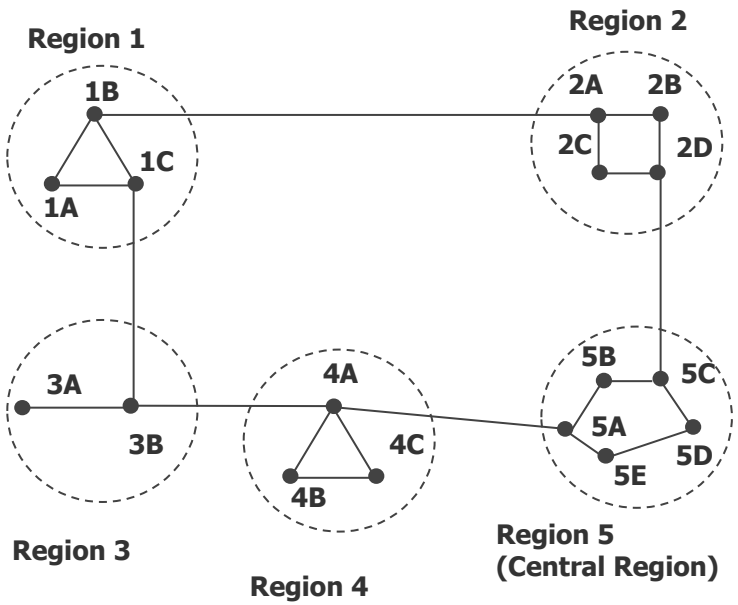
  - Advantage: less state, as routers only need to know their own region in detail
  - Drawback: may lead to worse paths (due to partial topology knowledge)
- **Flat routing**: all routers are part of the same region
  - Advantage: the best paths are available because the full topology is known
  - Drawback: more state, more control traffic overhead

Flat Routing *vs* Hierarchical Routing: Example

Forwarding table for router 1A

Flat routing

Dest.	Link to	Hops
1A	-	-
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5



Much smaller forwarding table!

Hierarchical routing

Dest.	Link to	Hops
1A	-	-
1B	1B	1
1C	1C	1
2	1B	2
3	1B	9
4	1B	8
5	1C	4

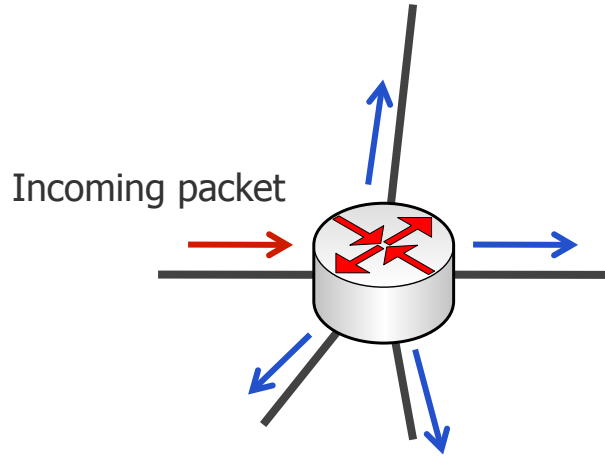
But some paths are longer than necessary!

Content of this Section

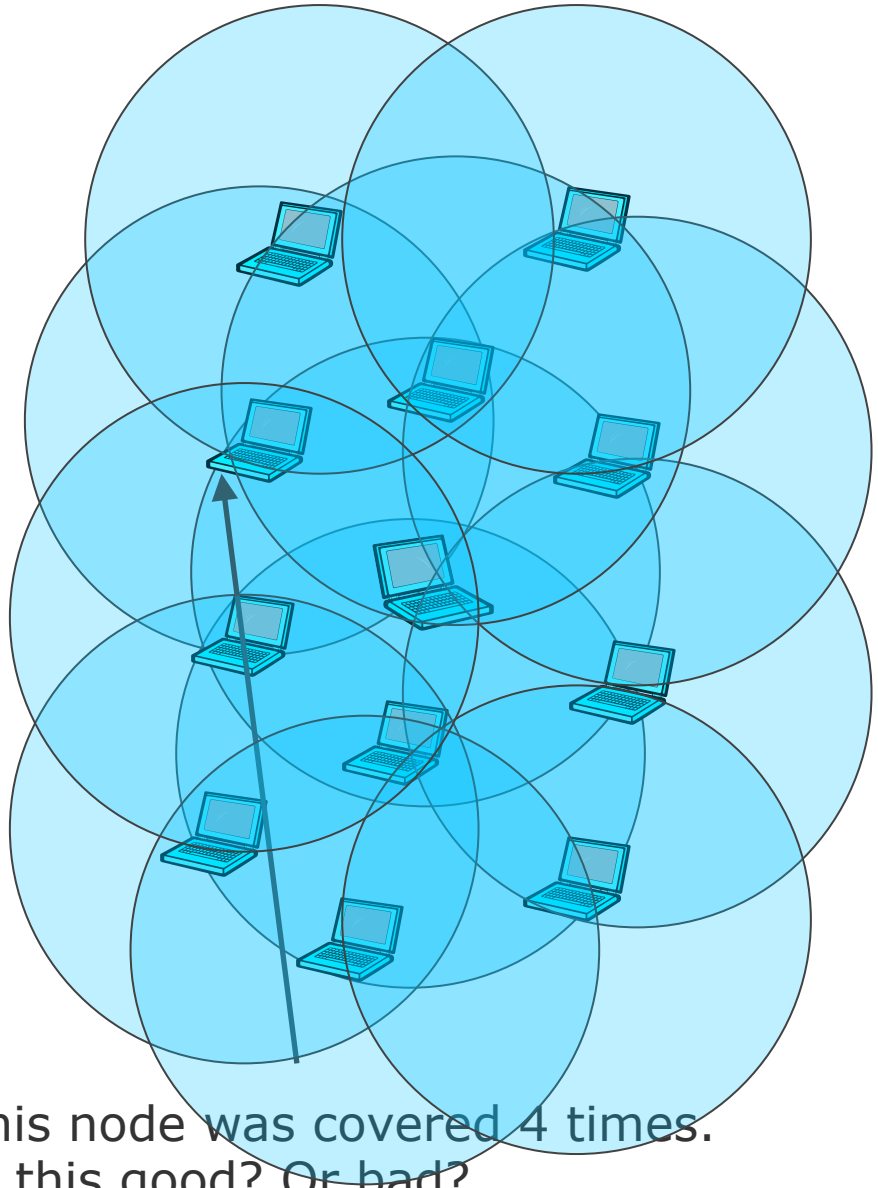
- Prerequisites to routing
- Routing *vs* forwarding
- Routing schemes: static/dynamic, source/hop-by-hop, flat/hierarchical…
- Basic concepts for routing: flooding, metrics, graphs
- Fundamental routing algorithms
- Standard routing protocols

Basic Concepts for Routing: Flooding

- Why care about specific paths? Why not just **flooding**?
  - Epidemic principle: each router repeats a received packet over all its interfaces
    - except over the interface where the packet came from (in some cases)
- Advantage: each packet reaches all routers
  - Robustness in face of single router failure
  - Simplicity
- Drawback: each packet reaches all routers
  - Privacy!
  - (Potentially dramatic) goodput decrease
  - Loop detection to avoid "broadcast storm"



Flooding: Example in Wireless Networks



This node was covered 4 times.
Is this good? Or bad?

Remark: with wireless, flooding may mean repeating packets on the interface they were received from…

Conclusion on Flooding

Flooding does not work at large scale but…

- Scoped flooding is used by some routing algorithms (e.g. OSPF)
- Basic flooding can be used as reference for performance evaluation of routing algorithms (e.g. in terms of delay)

Basic Concepts for Routing: Metrics

- Flooding does not scale, so we need to care about specific paths.
- Two questions arise at this point:
  - Q1: What paths are available towards the destination?
  - Q2: How do I compare these available paths to find the most favorable one?
- **Routing algorithms** take care of Q1
- **Metrics** take care of Q2
- The most favorable path depends on the criteria (= metric) we use:
  - Response time?
  - Throughput capacity?
  - Least number of intermediate devices?
  - Avoidance of local overload situations?
  - Security requirements?
  - … (so many more potential metrics)
  - … and even more: how about a mixture of all of the above?

Basic Concepts for Routing: Metrics

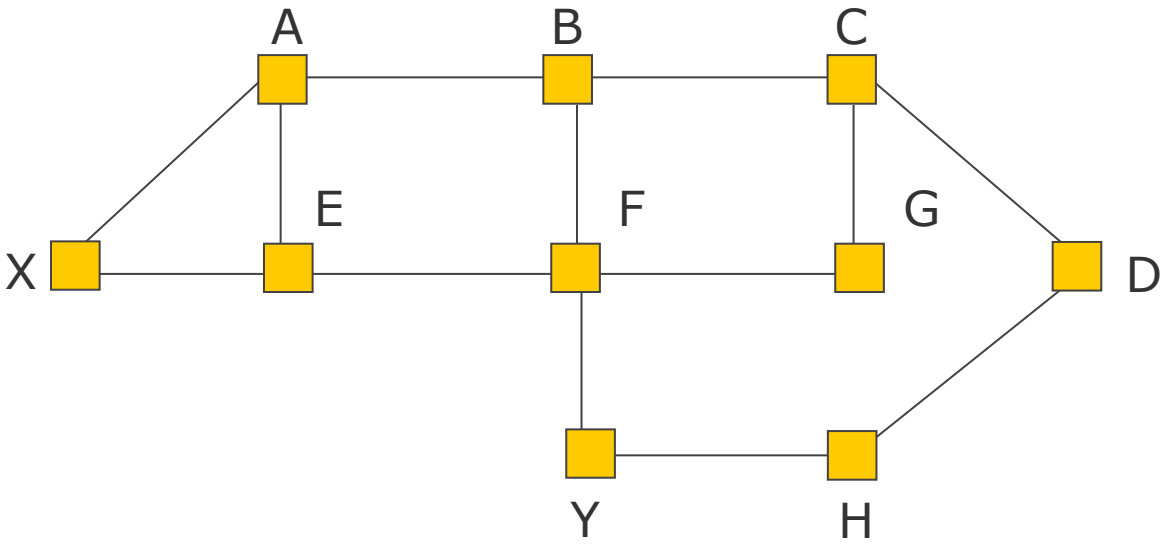
- Routing algorithms generally take as input a "cost" per link
  - Unless specified otherwise, all links have equal costs
  - According to the metric in use, costs may be configured (and updated) by external mechanisms (e.g. manual configuration, or automatic link-layer mechanisms)

- Routing algorithms find the available path(s) with smallest end-to-end cost
  - End-to-end cost depends on metric characteristics. Three main types:
    - additive metric (e.g. delay): add costs of each link on the path
    - concave metric (e.g. throughput): min cost of all links on the path
    - multiplicative metric (packet delivery ratio): multiply costs of each link on the path

- Default metric: equal costs, additive metric ➡ **hop-count metric**

Basic Concepts for Routing: Metrics

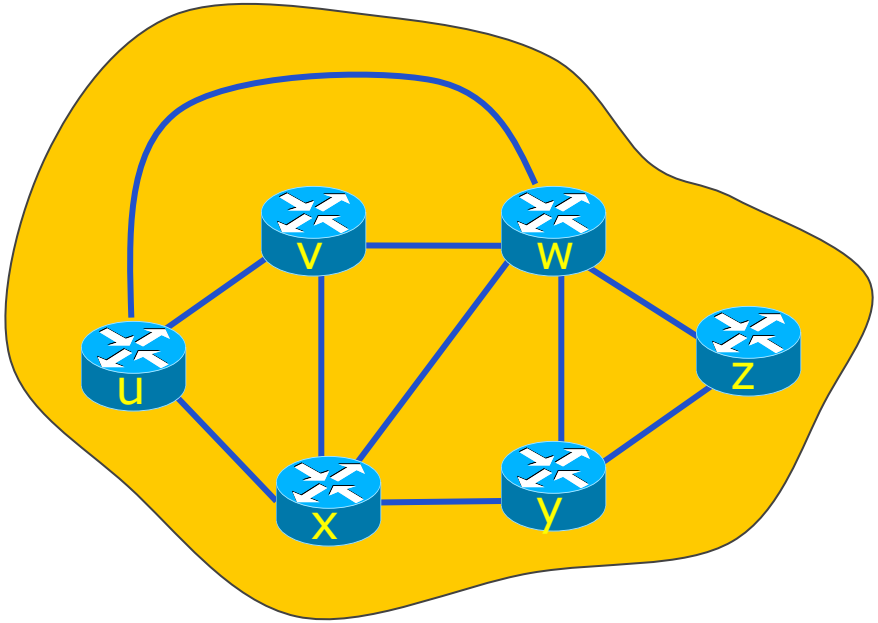
- Determining the optimal path may be difficult because:
  - Topology and metric information are never instantaneously available
  - Some metrics lead to **NP-complete computations** to find the optimum
  - Conflicts between **fairness** and optimum can arise



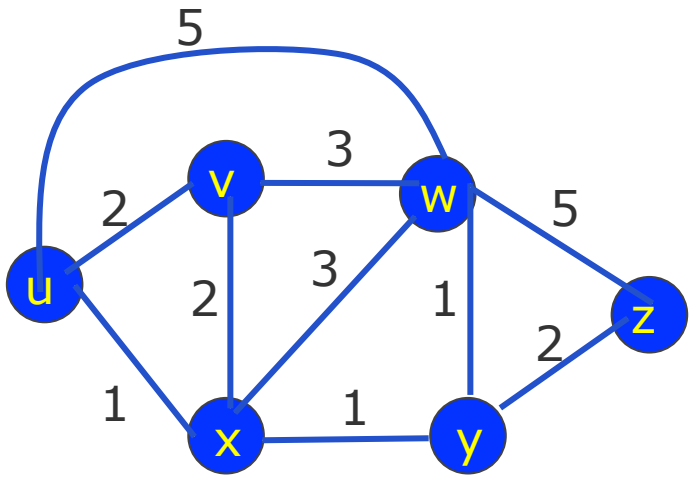
- **Highest throughput between E and G?**
- **The best throughput between E and G would require X and Y to either**
  - **Not communicate with one another**
  - **Use a huge detour**
- **Fairness! X and Y would certainly not be happy**

Basic Concepts for Routing: Graphs

- Abstraction of a computer network as a graph
  - Vertex = router, edge = link, link cost = weight of corresponding edge
- Graph: $G = (N, E)$
  - N = set of vertices = $\{ u, v, w, x, y, z \}$
  - E = set of edges = $\{ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) \}$



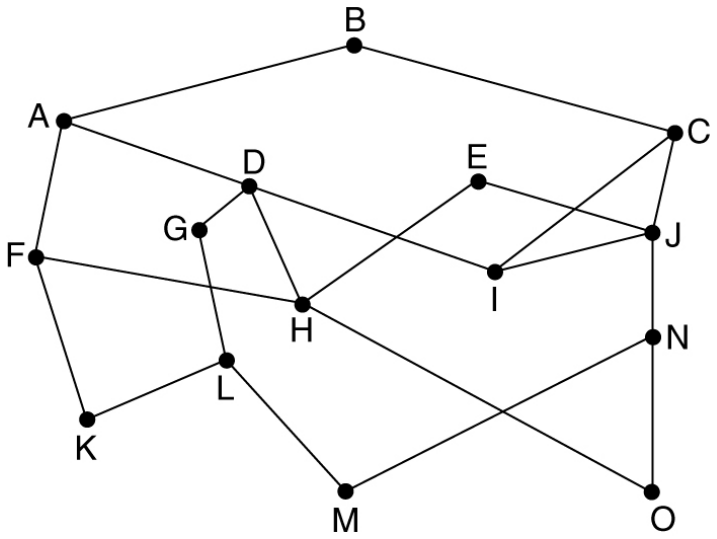
Physical network topology
(links between routers)



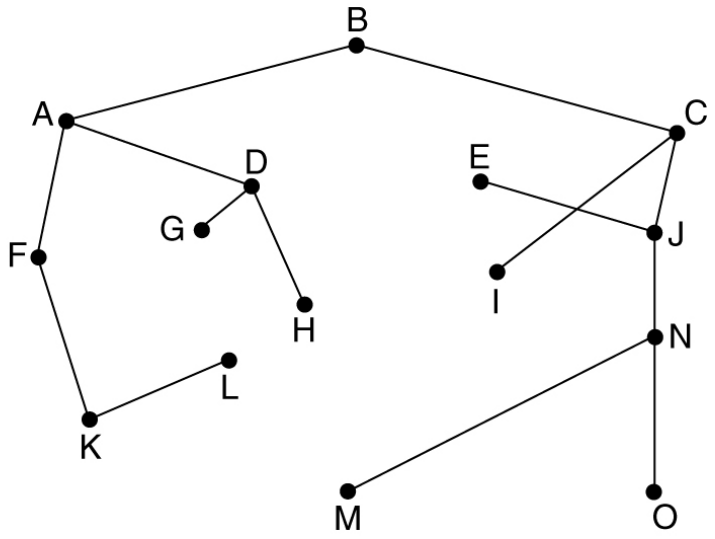
Abstract graph representation
Vertices connected by (weighted) edges

Basic Concepts for Routing: Graphs

- routing algorithms must discover & maintain a partial, but sufficient view of the topology to make paths available, from anywhere to everywhere



Full Topology

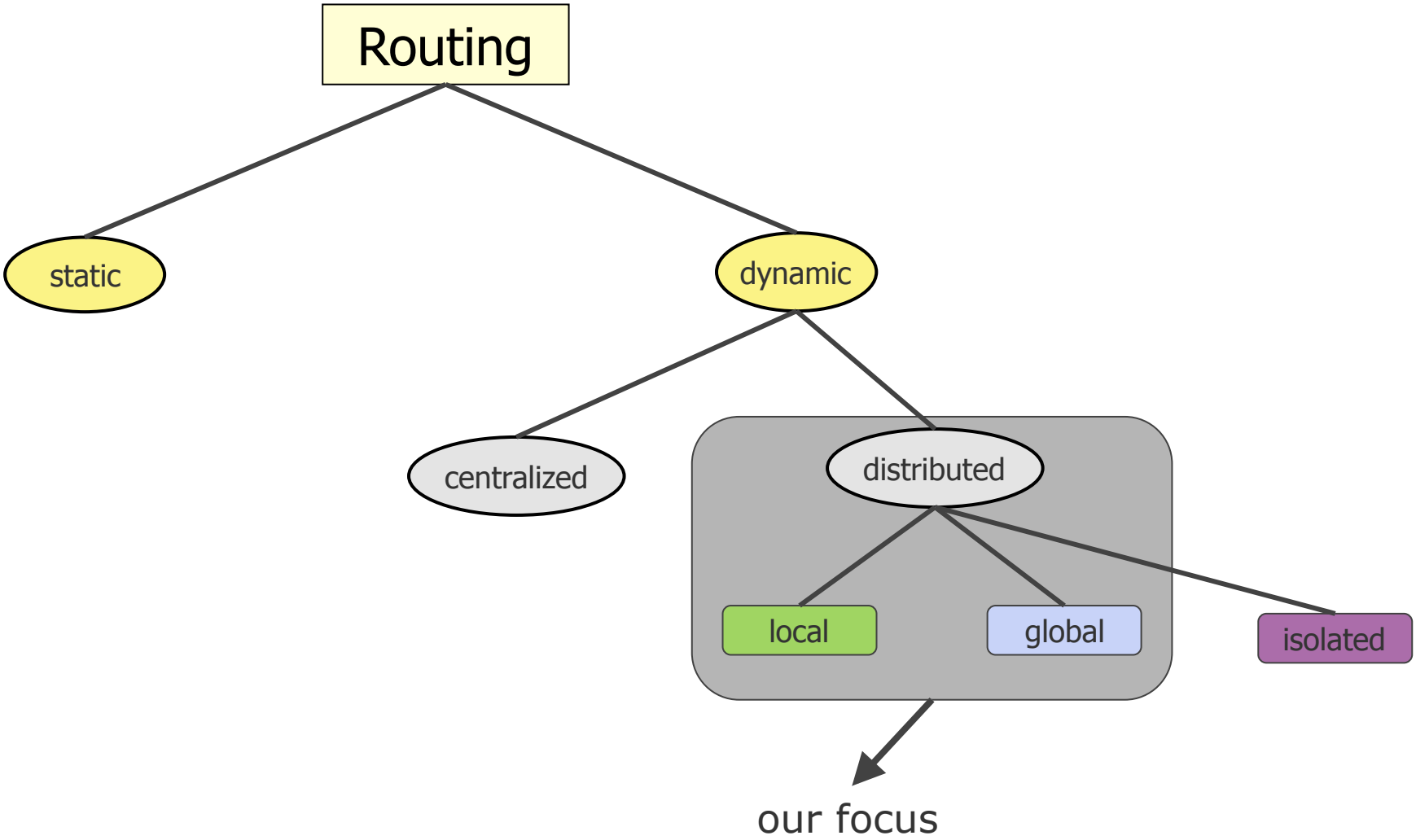


Partial, but sufficient topology

Content of this Section

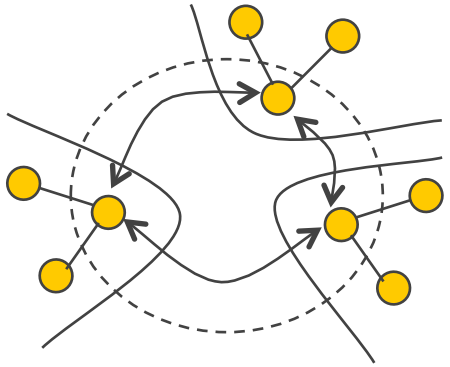
- Prerequisites to routing
- Routing *vs* forwarding
- Routing schemes: static/dynamic, source/hop-by-hop, flat/hierarchical…
- Basic concepts for routing: flooding, metrics, graphs
- Fundamental routing algorithms
- Standard routing protocols

Routing Algorithms: Taxonomy

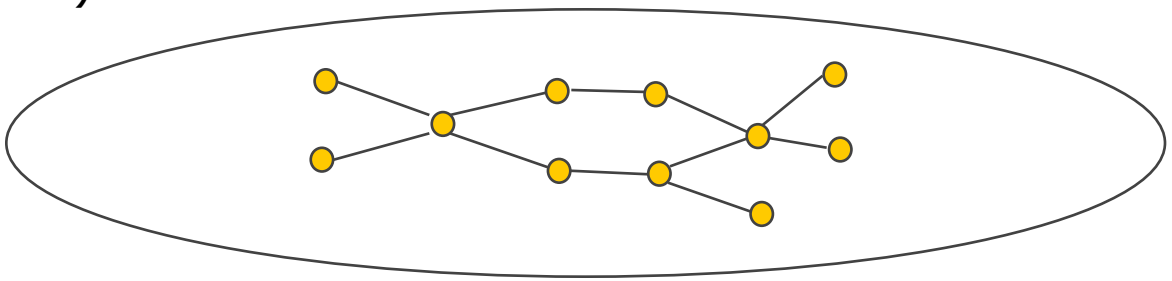


Fundamental Routing Algorithms

- Fundamental distributed local approach: Distance Vector Routing
  - Bellman-Ford algorithm (1957)

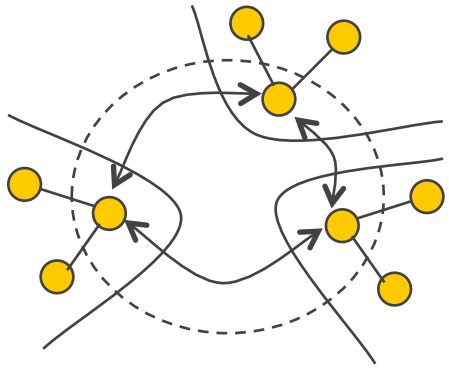


- Fundamental distributed global approah: Link State Routing
  - Djikstra algorithm (1959)



Distance Vector Routing

- Principle: global information is exchanged locally



  - each router periodically sends its full forwarding table to its direct neighbors
    - including the end-to-end cost towards each destination
  - if a router received the forwarding table of a neighbor, indicating a path to a new destination, or a better path to a known destination (=smaller end-to-end cost), the router updates its own forwarding table

- This distributed algorithm converges towards a state where the forwarding table of each router indicates the least cost path to each possible destination

Distance Vector Routing

Bellman-Ford Algorithm pseudo-code

```
1   Initialization:
2     for all destinations y in N:
3       D_x(y) = c(x,y)                  // if y is not a neighbor then c(x,y) = ∞
4     for each neighbor w
5       D_w(y) = ∞ for all destinations y in N
6     for each neighbor w
7       send distance vector DV_x = [D_x(y): y in N] to w
8
9   Loop:
10    wait (until I see a link cost change to some neighbor w or
11          until I receive a distance vector from some neighbor w)
12
13    for each y in N:
14      D_x(y) = min_v{c(x,v)+D_v(y)}
15
16    if D_x(y) changed for any destination y
17      send distance vector DV_x = [D_x(y): y in N] to all neighbors
```

Terms

- $c(x,y)$ Cost of link (x,y)
- $D_x(y)$ Distance from node x to y
- DV_x Distance vector of node x

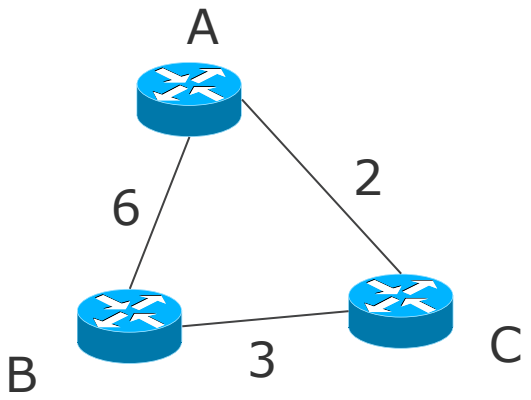
Distance Vector Example

- Initialisation: each router just knows itself + cost of each its interface(s)
- Each router sends its forwarding table on all its interfaces

From A to	NextHop	Costs
A	Locally	0

From B to	NextHop	Costs
B	Locally	0

From C to	NextHop	Costs
C	Locally	0



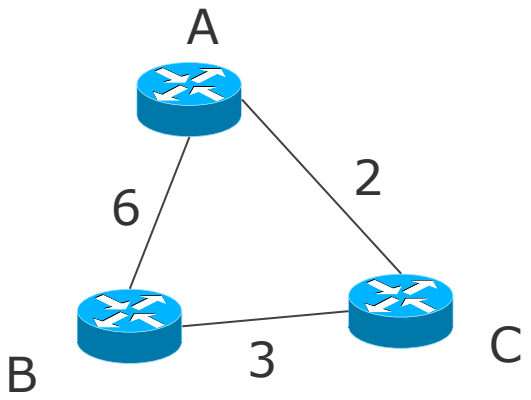
Distance Vector Example

- After iteration 1: each router got to know its direct neighbors
- Each router sends its new forwarding table on all its interfaces

From A to	NextHop	Costs
A	Locally	0
B	B	6
C	C	2

From B to	NextHop	Costs
B	Locally	0
A	A	6
C	C	3

From C to	NextHop	Costs
C	Locally	0
A	A	2
B	B	3



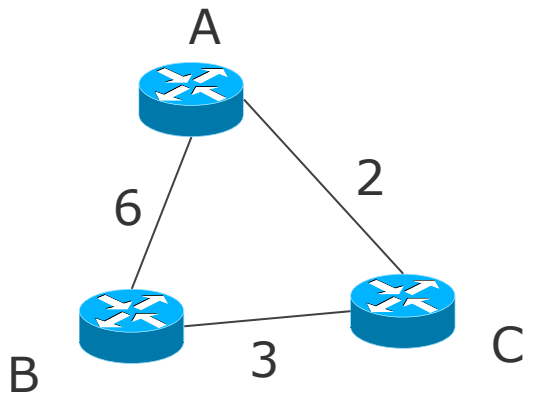
Distance Vector Example

- After iteration 2: each router heard about indirect neighbors
- A and B discovered that the indirect path through C is better

From A to	NextHop	Costs
A	Locally	0
B	C	5
C	C	2

From B to	NextHop	Costs
B	Locally	0
A	C	5
C	C	3

From C to	NextHop	Costs
C	Locally	0
A	A	2
B	B	3



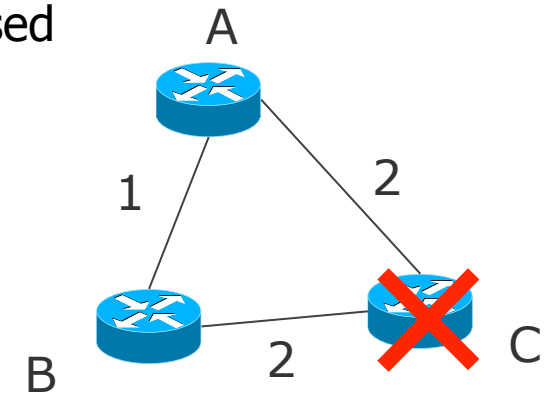
The algorithm terminates since forwarding tables converged (i.e. do not change in the next iterations)

Distance Vector Routing

- Advantage: very simple
- Drawback: not robust

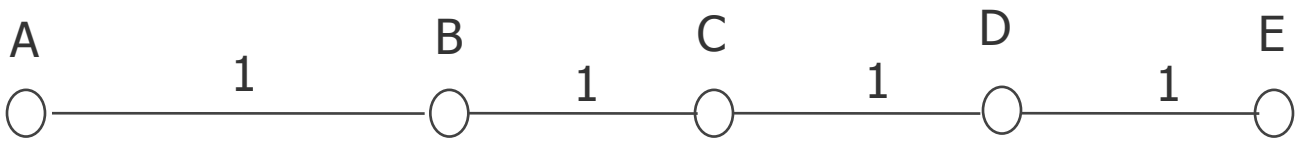
- Impact of wrong information: the deadlock example
  - A buggy router *j* announces a forwarding table with cost 0 to all destinations
  - As a consequence, nearly all traffic is led over j
  - Result: Collapse

- Impact of single router/link failure
  - In unlucky cases, router/link failure is too slowly diagnosed
  - Leads to routing loops (during divergence time)
  - Divergence time can be infinite
    - the count-to-infinity incident (ARPANET 1983)



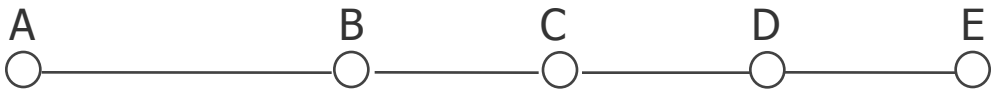
Distance Vector Routing: Count-to-Infinity

- Example:
  - Five routers A, B, C, D, and E are connected in a linear fashion, the cost between neighbor routers in each case is 1.



Distance Vector Routing: Count to Infinity

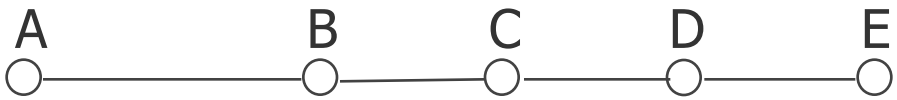
Step 1: Node A is switched off and will be switched on now!



		A	B	C	D	E
	Cost to A (A is off)		∞	∞	∞	∞
A switched on ➡	Iteration 1		1	∞	∞	∞
	Iteration 2		1	2	∞	∞
	Iteration 3		1	2	3	∞
	Iteration 4		1	2	3	4

Distance Vector Routing: Count to Infinity

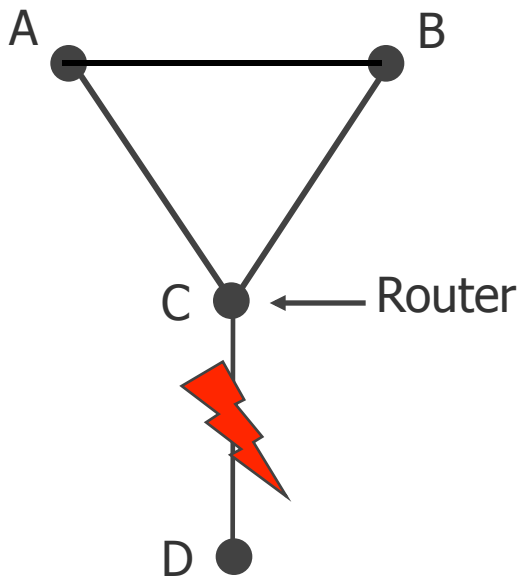
Step 2: A is switched off



A switched off	1	2	3	4
Iteration 1	3	2	3	4
Iteration 2	3	4	3	4
Iteration 3	5	4	5	4
Iteration 4	5	6	5	6
⋮	⋮	⋮	⋮	⋮
Iteration 6	7	8	7	8
⋮	⋮	⋮	⋮	⋮
Iteration ∞	∞	∞	∞	∞

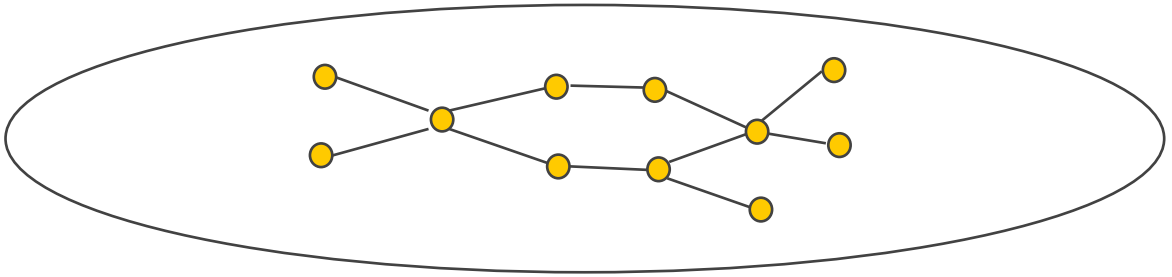
Distance Vector Routing: Split Horizon Algorithm

- Solution: Split-Horizon
- Principle: Do not send the distance to X over that link used to transfer packets for X (resp. path is reported as infinity).
- However, it does not work always:
  - Link from C to D breaks down
  - C announces A and B that D is unreachable
  - C hears from A and B that they do not reach D (because of split horizon)
  - But B announces however to A: D has distance 2
  - A updates and has D with distance 3
  - B updates for D with distance 4 .....
  - Count to Infinity
- Spit-Horizon with Poison Reverse variant



Link State Routing

- Principle: local information is exchanged globally



  - each router periodically sends the state of its links with its direct neighbors to all routers in the network (cost of links to its current neighbors)
  - routers use **flooding** to announce their link state to the whole network
  - each router receives information about the links of every other routers
    - router maintains its own **link state database** aggregating all received topology info
  - each router can derive its forwarding table from its link state database
    - If the link state database is updated, the router recomputes the forwarding table

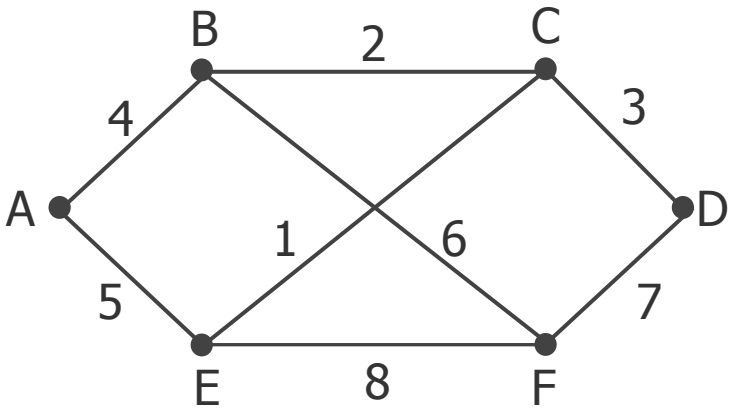
Link State Routing

- Each router:
  - determines its neighbors and their addresses via periodical, local **HELLO** packets
  - **floods** this information **periodically** via link state advertisement (**LSA**) packets
  - continuously updates its link state database (**LSDB**) based on received LSAs
  - recomputes paths based on LSDB, each time it is updated

- Djikstra algorithm: path computation based on the LSDB

Link State Routing: Example

- LSAs contain:
  - Identification of the sender
  - The list of neighbors with associated link cost
  - Sequence number (to identify newer LSAs)
  - Age (to indicate validity length of info)



LSAs from each router:

LSDB

A		B		C		D		E		F	
Seq.No.		Seq.No.		Seq.No.		Seq.No.		Seq.No.		Seq.No.	
Age		Age		Age		Age		Age		Age	
B	4	A	4	B	2	C	3	A	5	B	6
E	5	C	2	D	3	F	7	C	1	D	7
		F	6	E	1			F	8	E	8

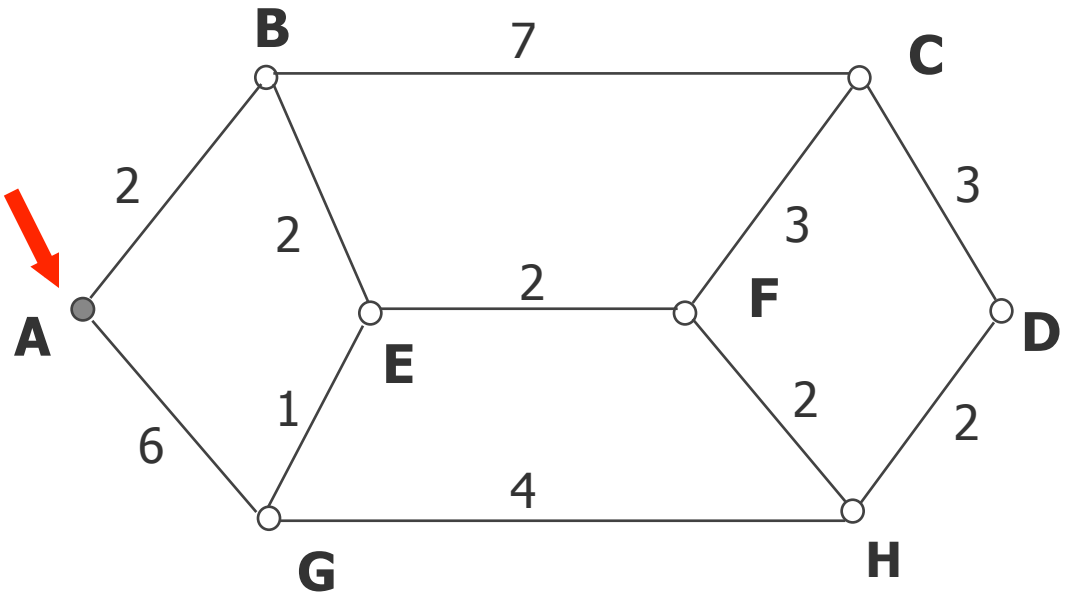
Djikstra Algorithm

- Solves the single-source shortest path problem
  - Finding the shortest paths from a source vertex to all other vertices in the graph
  - Works on both directed and undirected graphs, but all edges must have nonnegative weights
- Approach: Greedy
- Input: Weighted graph G={E,V} and source vertex v∈V, such that all edge weights are nonnegative
  - This initialisation information is derived from the LSDB
- Output: Lengths of shortest paths (or the shortest paths themselves) from a given source vertex v∈V to all other vertices

Dijkstra Algorithm: Outline

- Algorithm of Dijkstra (1959) for static Routing
  1. Mark source node (the initial **work node**) as **permanent,** i.e., distance and line do not change any more
  2. Consider **neighbor nodes** of the **work node,** and compute the distance from the source to them based on current knowledge and **link costs**
  3. Choose the node with the **smallest distance** to the source from the nodes not marked yet as permanent, mark it as **permanent**, and set it to be the new **work node**
  4. If there are still nodes which are not marked as permanent, go to step 2

Dijkstra Algorithm: Example (1)



As metric, exemplarily the distance in kilometers is chosen.

The shortest path from A to D is searched.

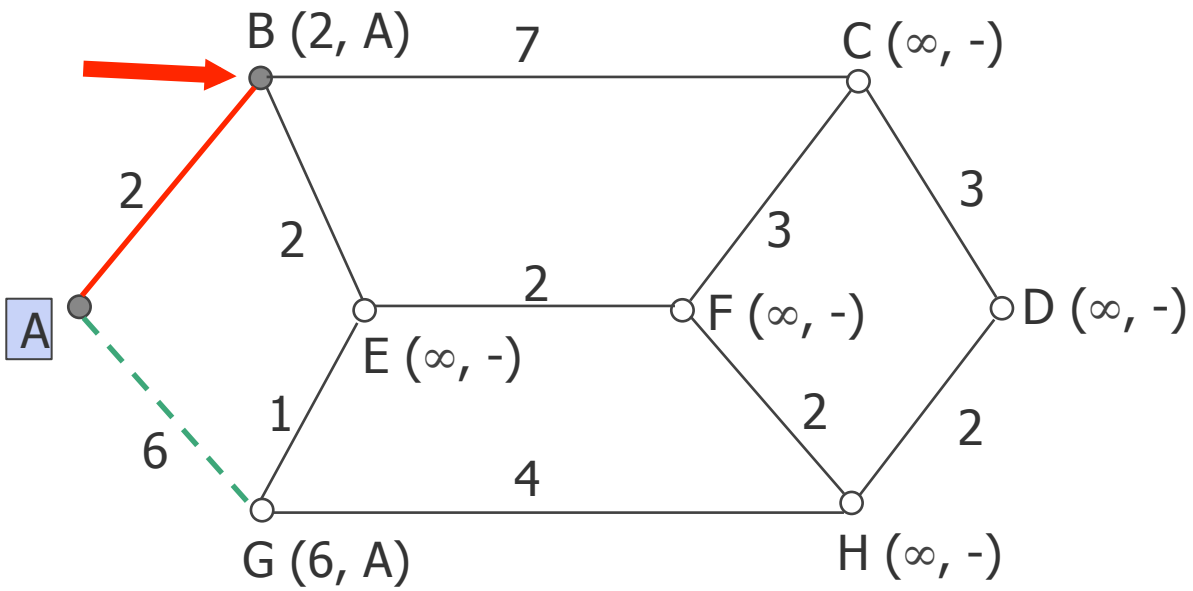
1. Step: Marking of node A as permanent.

2. Step: Marking of the neighbor nodes of A.

Current working node

Already marked permanent

Dijkstra Algorithm: Example (2)



In order to be able to trace back the path later on, the predecessor node is stored.

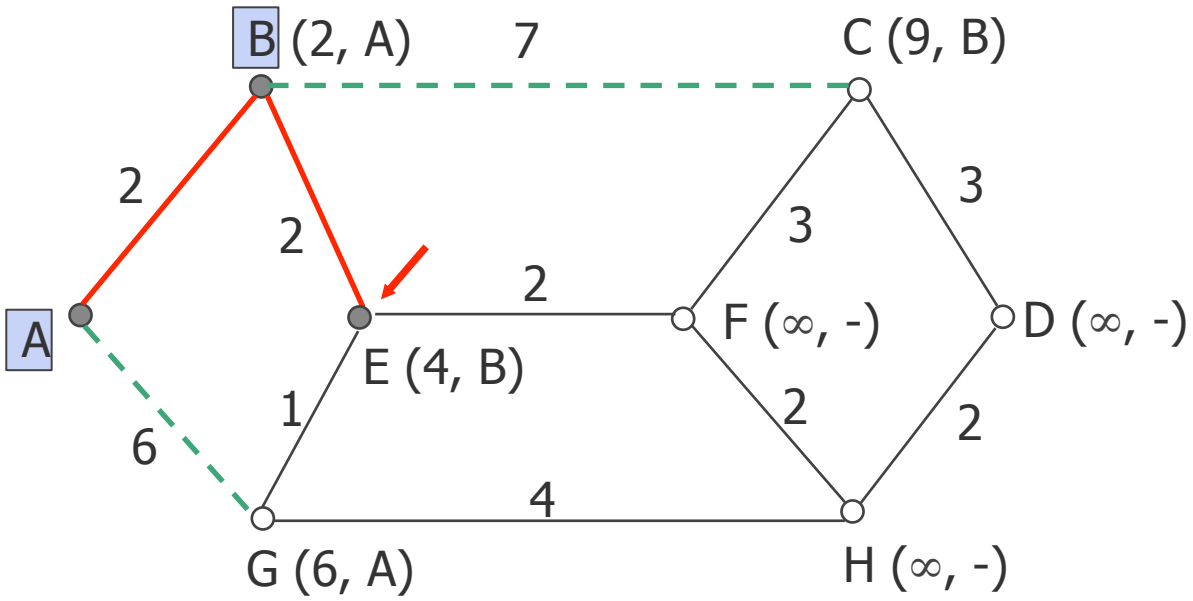
3. Step: B becomes permanent, because the distance to A is 2 (< 6).

4. Step: Tentative labeling of the neighbor nodes of B.

Current working node

Already marked permanent

Dijkstra Algorithm: Example (3)

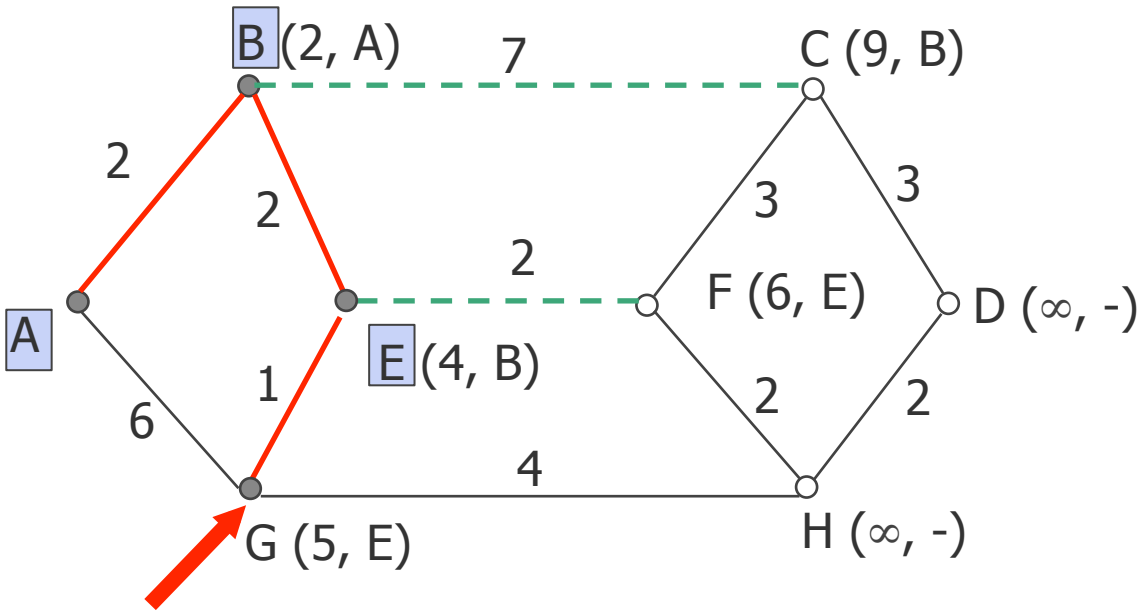


5. Step: E becomes permanent, since the distance to A is 4 (< 6 < 9).

6. Step: Tentative labeling of the neighbor nodes of E.

E (4, B): Distance of A to E sums up to 4 using the path BE.

Dijkstra Algorithm: Example (4)

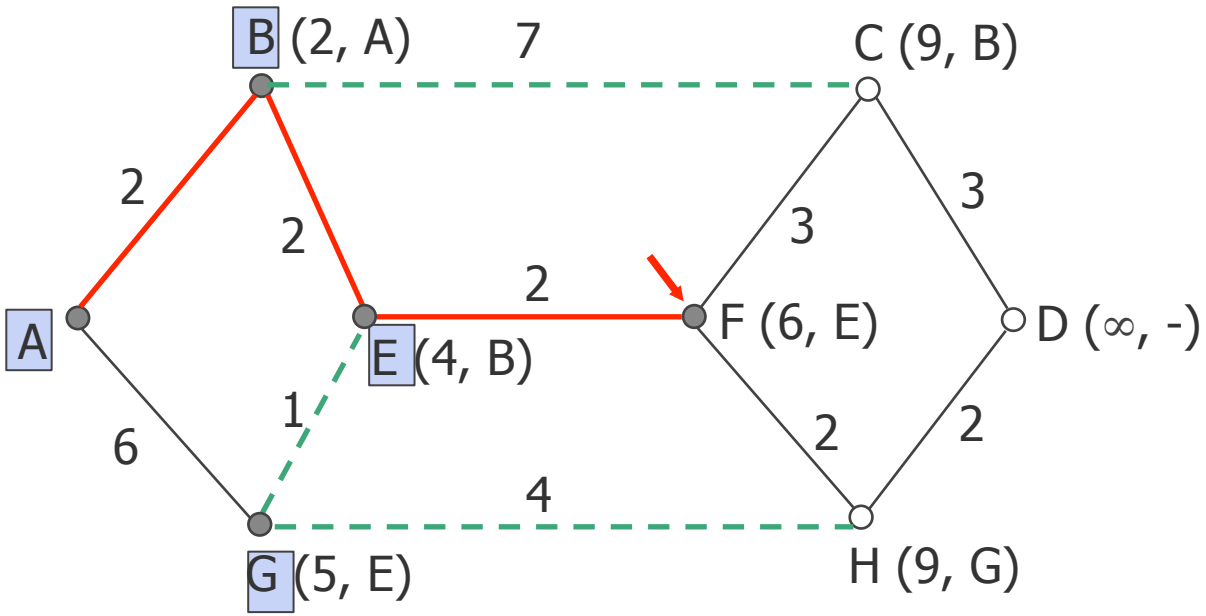


5. Step: Preliminary label of G is over-written.

6. Step: G becomes permanent, since the distance to A is 5 (< 6 < 9).

7. Step: Tentative labeling of the neighbor nodes of G.

Dijkstra Algorithm: Example (5)

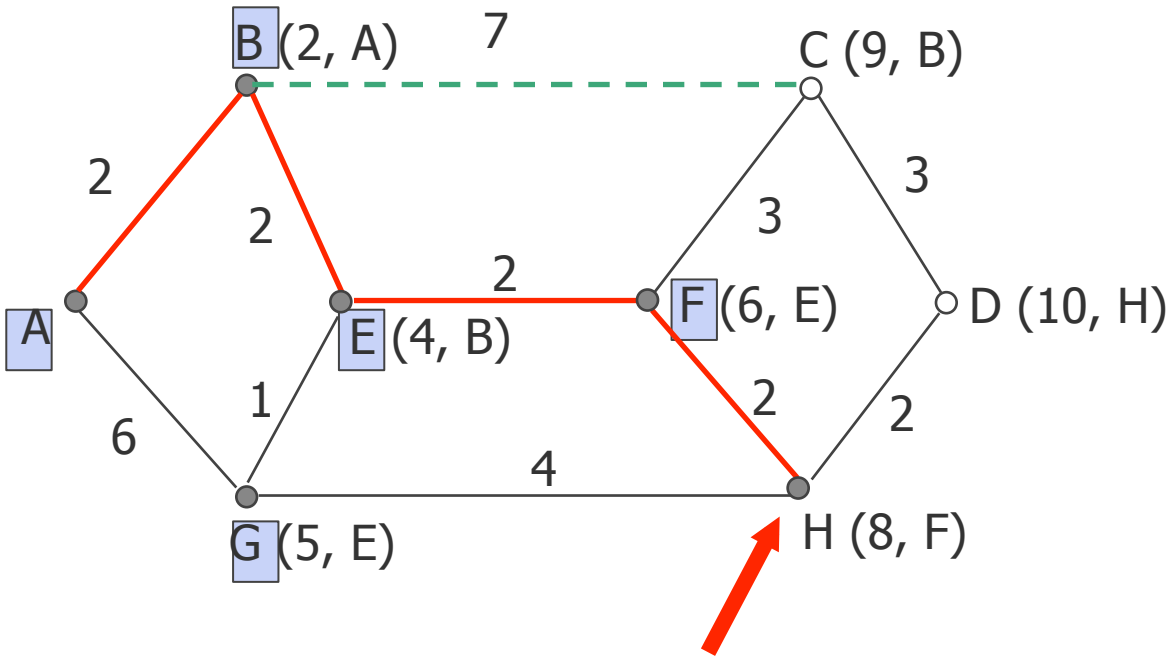


8. Step: Tentative label of G is overwritten.

9. Step: F becomes permanent, since the distance to A is 6 (< 9).

10. Step: Tentative labeling of the neighbor nodes of F.

Dijkstra Algorithm: Example (6)

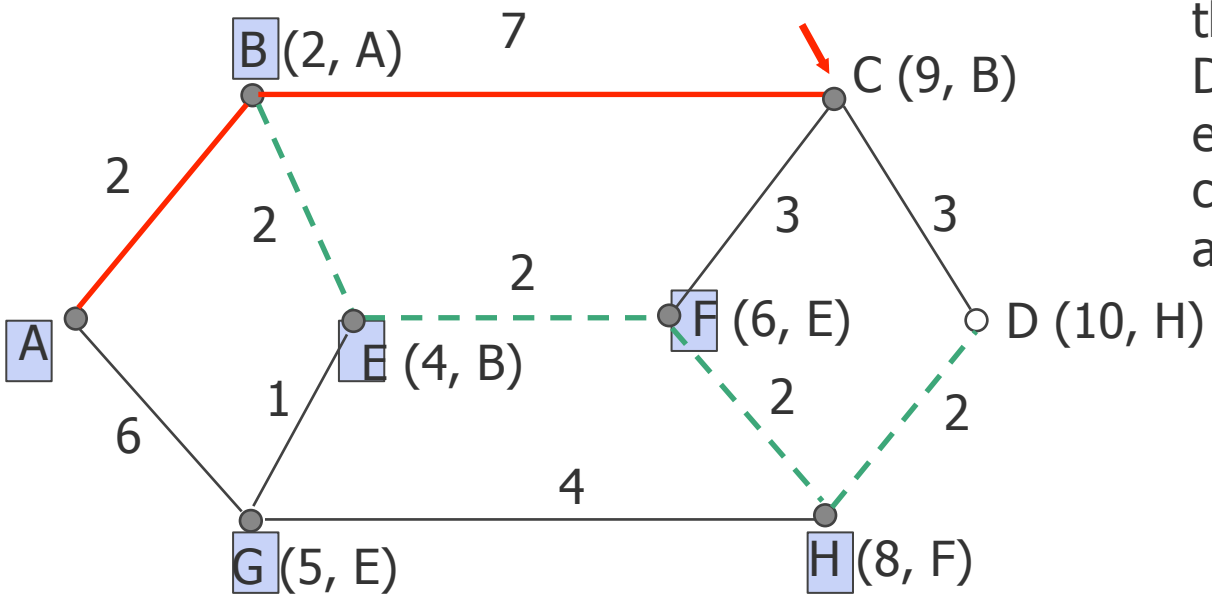


11. Step: H becomes permanent, since the distance to A is 8 (< 9).

12. Step: Tentative labeling of the neighbor nodes of H.

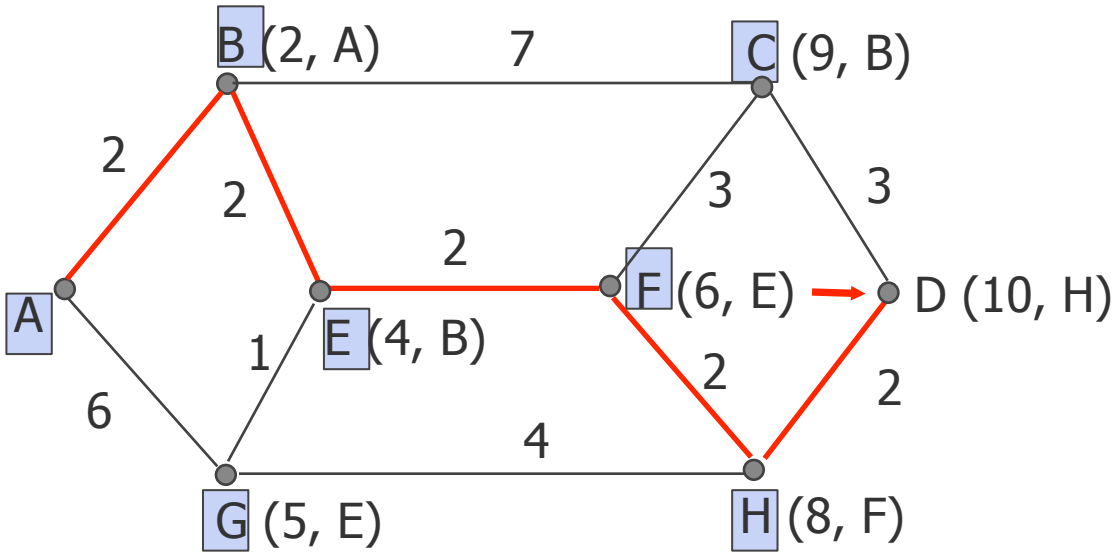
13. Step: C becomes permanent, since the distance to A is 9 (< 10).

Dijkstra Algorithm: Example (7)



The distance to D using C is larger than the tentative label of D. No more paths exist, no states are changed - the algorithm terminates.

Dijkstra Algorithm: Example (8)



14. Step: D is reached on the shortest path and algorithm terminates

Implementation of Dijkstra Algorithm (1)

```
#define MAX_NODES 1024                          /* maximum number of nodes                */
#define INFINITY 1000000000                     /* a number larger than every maximum path */
int n;
int dist[MAX_NODES][MAX_NODES];                 /* dist[i][j] is the distance from i to j  */

void shortest_path(int s, int t, int path[])
{                                               /* s = source node, t = terminal node */
    struct state {                              /* the path being worked on           */
        int predecessor;                        /* previous node                      */
        int length;                             /* length from source to this node    */
        enum {permanent, tentative} label;      /* label state                        */
    } state[MAX_NODES];

    int i, k, min;
    struct state *p;

    for (p = &state[0]; p < &state[n]; p++) {   /* initialize state                   */
        p->predecessor = -1;
        p->length = INFINITY;
        p->label = tentative;
    }

    state[t].length = 0;
    state[t].label = permanent;
    k = t;                                      /* k is the initial working node */
                                                /* The algorithm starts with the terminal node */
// Next Slide
```

Implementation of Dijkstra Algorithm (2)

```
    do {                                    /* Is there a better path from k? */
        for (i = 0; i < n; i++)             /* this graph has n nodes         */
            if (dist[k][i] != 0 && state[i].label == tentative) {
                if (state[k].length + dist[k][i] < state[i].length) {
                    state[i].predecessor = k;
                    state[i].length = state[k].length + dist[k][i];
                }
            }
        /* Find the tentatively labeled node with the smallest label. */
        k = 0;
        min = INFINITY;
        for (i = 0; i < n; i++)
            if (state[i].label == tentative && state[i].length < min) {
                min = state[i].length;
                k = i;
            }
        state[k].label = permanent;
    } while (k != s);

    /* Copy the path into the output array. */
    i = 0;
    k = s;
    do {
        path[i++] = k;
        k = state[k].predecessor;
    } while (k >= 0);

}// shortest_path
```

Link State Routing

- Drawbacks of link state routing:
  - More complex than distance vector
  - More state in routers

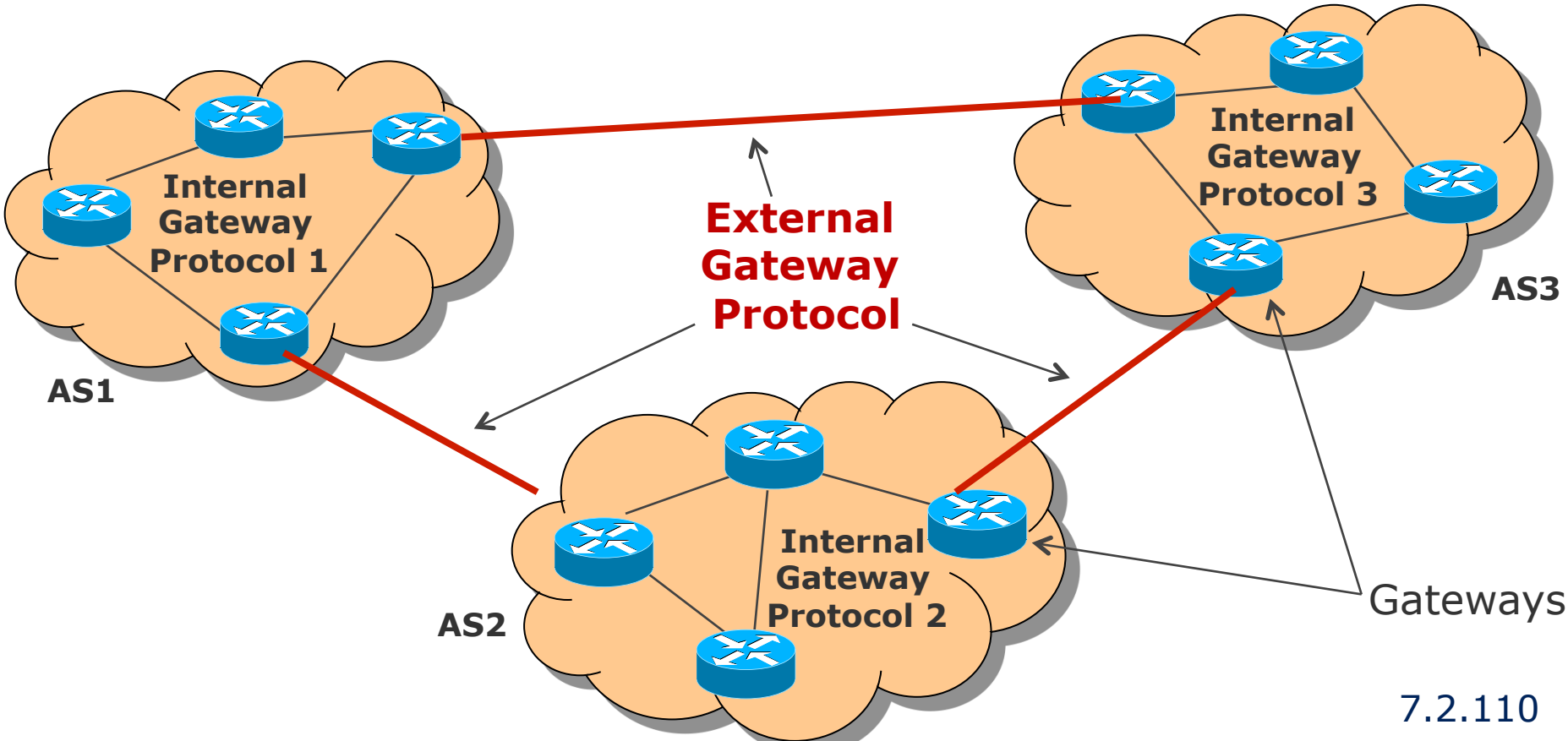
- Advantages of link state routing:
  - More stable in face of router/link failure (no count to infinity problem)
  - In general, wrong information from a single router does not crash the network

Content of this Section

- Prerequisites to routing
- Routing *vs* forwarding
- Routing schemes: static/dynamic, source/hop-by-hop, flat/hierarchical…
- Basic concepts for routing: flooding, metrics, graphs
- Fundamental routing algorithms
- Standard routing protocols

Internet Physical Topology

- The Internet consists of a large number of **Autonomous System**s (**AS**)
- An AS is connected to other Autonomous Systems via routers called **gateways**
- Each AS is operated independently, using an Interior Gateway routing protocol (**IGP**)
- All gateways use the same Exterior Gateway routing Protocol (**EGP**)



Standard Routing Protocols

- Example of Internal Gateway routing protocols (**IGP**)
  - RIP
  - OSPF
  - RPL
  - OLSR

- Example of External Gateway routing protocol (**EGP**)
  - BGP

Routing Information Protocol (RIP)

- Routing Information Protocol (RIP)
  - Specified in RFC 1058
  - The first IGP used in the Internet
- Based on distance vector, packet routing, flat routing, proactive
  - RIP messages are sent every 30 seconds as UDP datagrams
  - Used metric for the evaluation of the paths is the number of hops
    - The maximum number of hops is limited to 15 ➡ 16 = ∞
  - In a message (only) up to 25 entries of the routing table can be sent
  - Best fit for small systems
- RIPv2
  - Specified in RFC 2453
  - Support for subnets, CIDR, authentication, multicast, etc.
  - However: max. number of hops is still limited to 15
- RIPng extension of RIPv2 to support IPv6
  - Specified in RFC 2080

Issues with RIP and other Distance Vector Protocols

- Internet Gateway Routing Protocol (IGRP)
  - As reaction to the restrictions of RIP Cisco introduced IGRP
  - Also based on distance vector
  - Extension of the metric, load sharing, more efficient packet format
  - The protocol was not massively adopted (because too Cisco specific)
  - Replaced by EIGRP (still Cisco, new design, also based on distance vector)

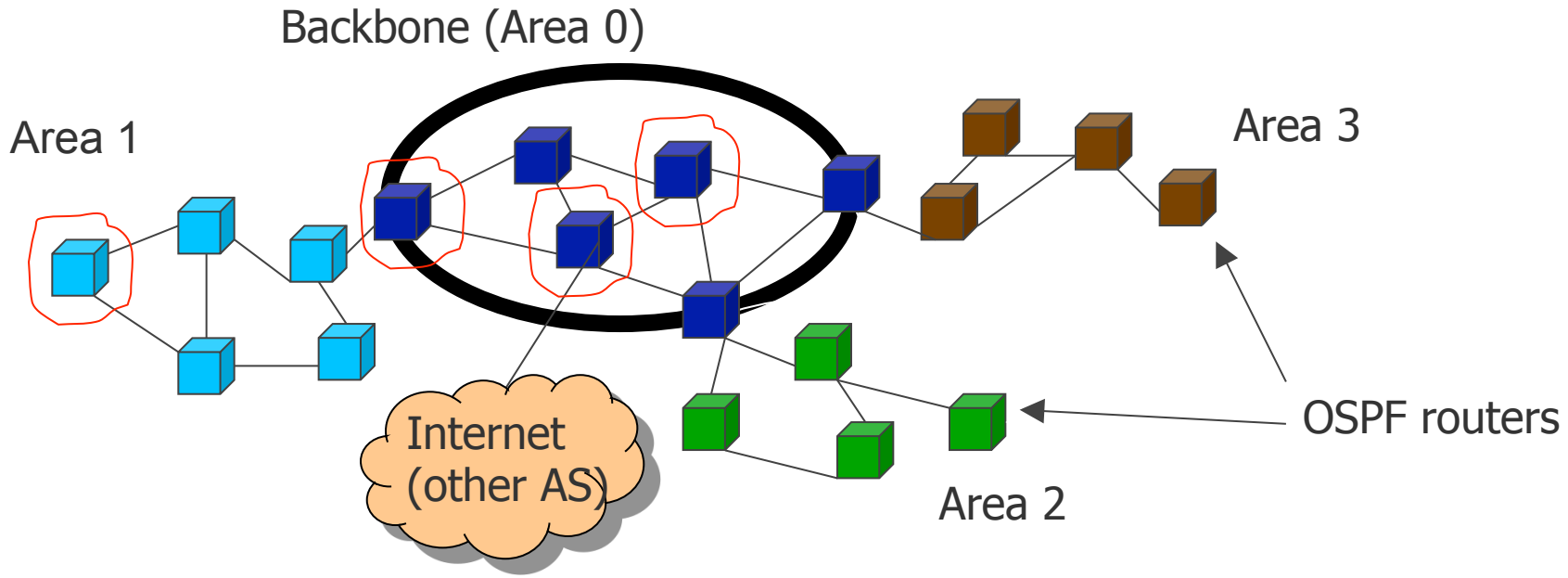
- Distance vector approaches suffer from known issues:
  - Slow convergence (duration of minutes)
  - Count-to-infinity (RIP failure in 1983: Count-to-infinity led to network partitioning)
  - Led to two types of efforts:
    - Designing more complex distance vector protocols (IGRP, RIPv2, EIGRP)
    - Designing entirely new protocols => OSPF

Open Shortest Path First (OSPF)

- Open Shortest Path First (OSPF)
  - Initially standardized in RFC 1247 (back in 1990)
  - OSPF version 2 (RFC 2328) for IPv4
  - OSPF version 3 (RFC 5340) with IPv6 support (actually address agnostic)
- Based on link-state, hierarchical routing, packet routing, proactive
  - Supports many different metrics (distance, delay, etc.)
  - Dynamic algorithm for fast adjustment to changing conditions in the network
  - Supports load sharing between redundant links
  - Supports hierarchical systems
  - Security mechanisms to protect routers from wrong routing info or attacks
- Modular, tailored support for three categories of link layers:
  - Point-to-point links between routers
  - Broadcast networks (mostly LANs)
  - Multi-access networks without broadcasting (e.g. packet switching WANs)

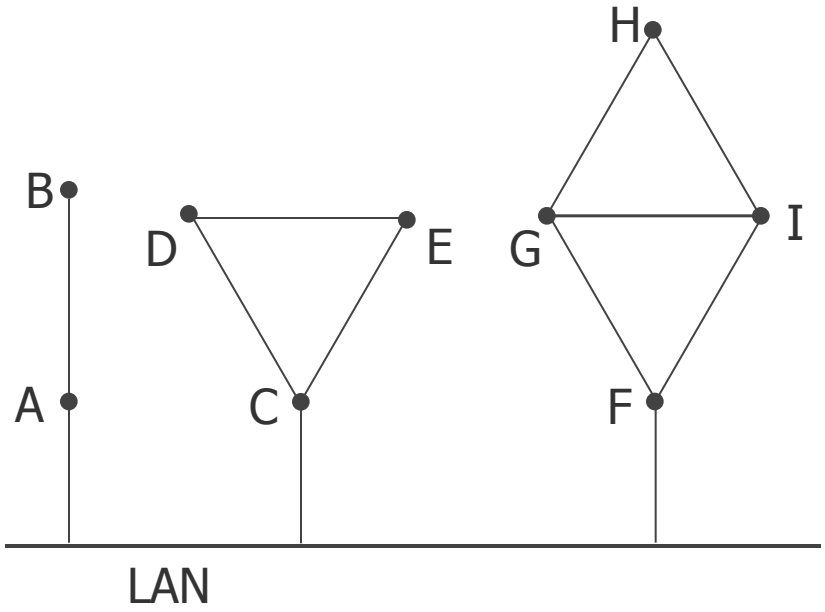
Open Shortest Path First (OSPF)

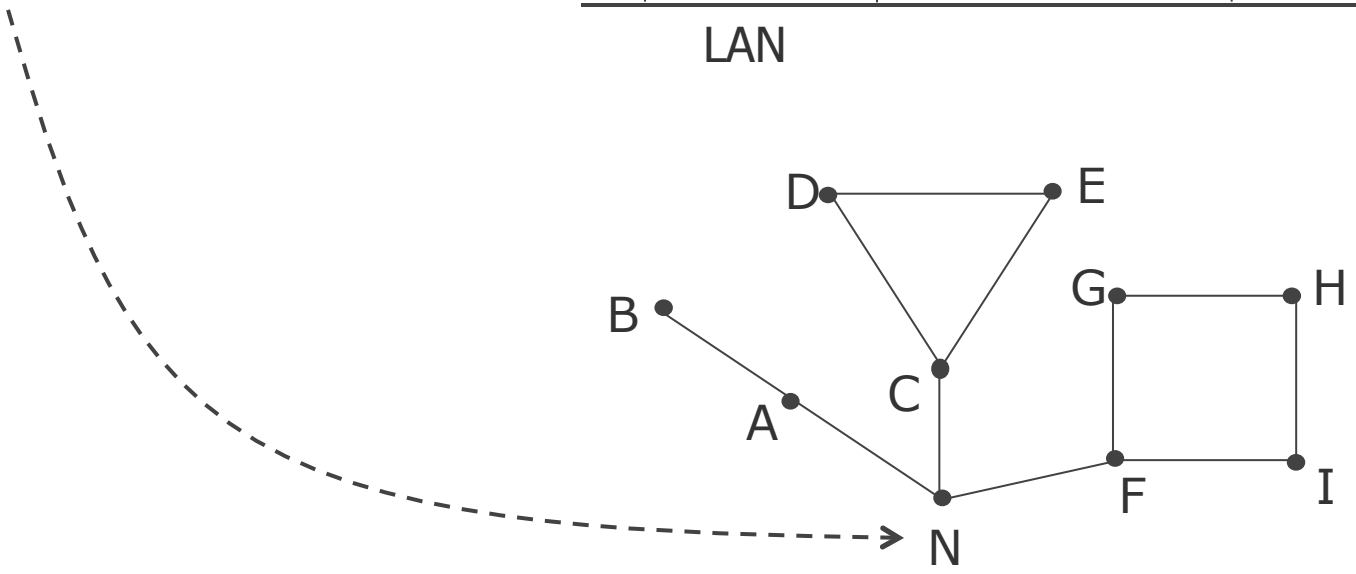
- Four types of OSPF routers
  - Internal routers, which only belong to one area
  - Area border routers, which connect two or more areas (thus part of the backbone)
  - Backbone routers, which are in the backbone
  - AS border routers, which connect to other autonomous systems
- Within an area every router has the **same link state database**
- A router connecting two areas, maintains link state databases from both areas



OSPF Broadcast Network Optimisation

- If several routers are connected in a (broadcast) network, a new "artificial" node is introduced for simplification



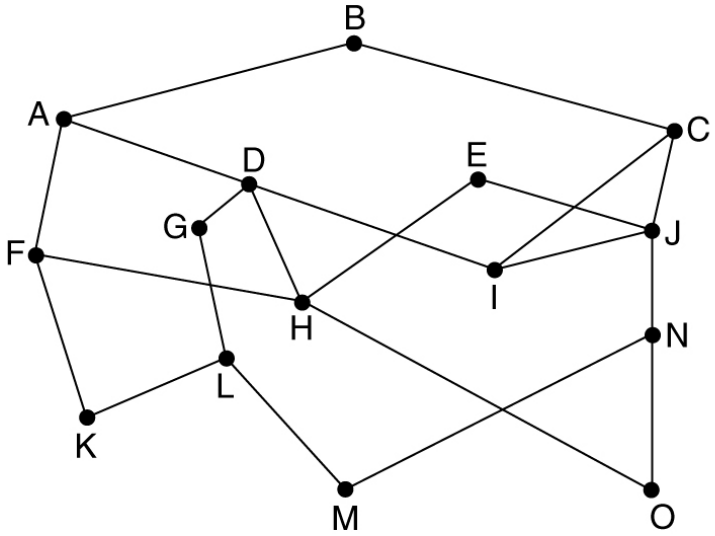


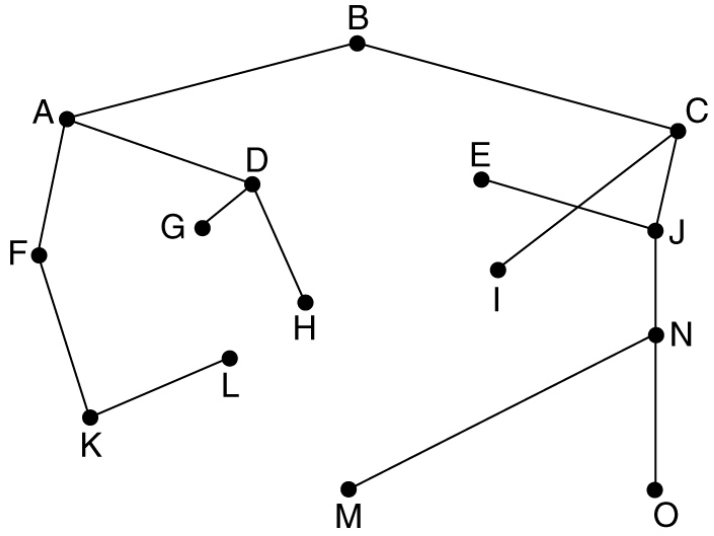
Intermediate System to Intermediate System (IS-IS)

- Standardized in the context of OSI for the connectionless layer 3 protocol
  - Republished by IETF as RFC 1142 (however sometimes also cited as IS-IS=0 …)
- very similar to OSPF, based on link-state, hierarchical routing, packet routing, proactive
- Neutral to layer 3 protocol (OSPF routes IP, originally v4)
  - Thus IPv6 got faster support by IS-IS
- Routers also build a map of the network, calculate shortest path
- Somewhat lower overhead compared to OSPF
  - Often used by network operators with large networks in terms number of routers

Routing in Wireless Sensor Networks: RPL

- Routing Protocol for Low power, Lossy Networks (RPL)
  - Specified in RFC 6550
- Target: Routing for sensors, small memory/CPU, lossy/low througput links
- Based on distance vector, proactive, flat routing, integrated support for hop-by-hop and source routing to accomodate lack of memory
- Assumption on data traffic ➡ mainly convergecast towards a sink
- Basic principle: tree structure with routers storing only state for parent



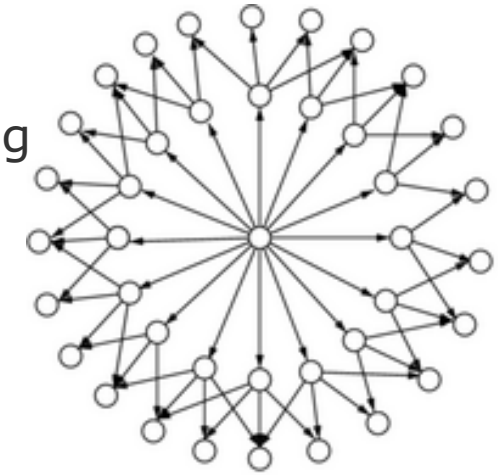


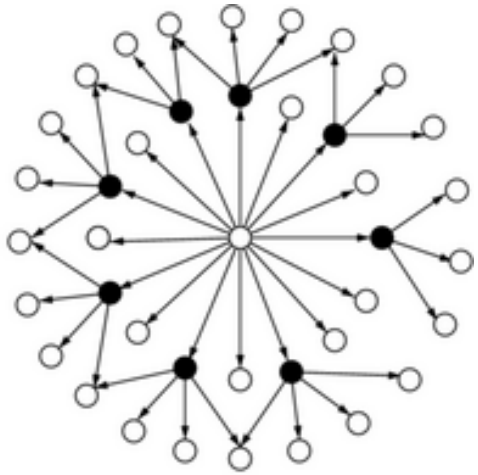
- P2P-RPL: extension to allow paths across the tree (see RFC 6997)

Routing in Mobile Ad How Networks: OLSR

- Optimized Link State Routing Protocol (OLSR)
  - RFC 3626 (published in 2003).
- Target: Routing for mobile ad hoc networks
- OLSR is similar to flat OSPF with overhead optimization
  - OLSR based on link state, packet routing, proactive, flat routing
  - MPR optimization (reduces both price of flooding andsize of LSAs & LSDB)

Basic flooding (every node relays)





Multi-Point Relay Flooding (only black nodes relay)

- Version 2 of the protocol is currently being standardized
- RFC 5449 MPR-OSPF : OPSF extension using MPR optimization

Border Gateway Protocol (BGP)

Motivation

- ISPs operate their networks independently of each other
  - Abstraction: one domain = Autonomous System (AS)
- ISP needs to reach also external IP networks
- Route selection between ISPs may not only follow efficiency but also economic, political etc. arguments

⇒ Exchange of routing information between ASes

⇒ Route selection based on complex policies

- BGP is the inter-domain protocol used in today's Internet
  - BGP4 specified in RFC 4271

- **BGP overview**
- BGP is a path vector protocol (different from distance vector or link-state)
  - BGP routers exchange path vectors with BGP neighbors (**peers**)
    - Typically, neighbors are connected directly or via a switch
    - Multihop BGP peering: Neighbors need not to be physically adjacent
- BGP peers accept or discard paths based on **policies** (e.g., shortest path, preferred neighbors, hot potato or cold potato)
- BGP router decides on outgoing advertisements based on policies

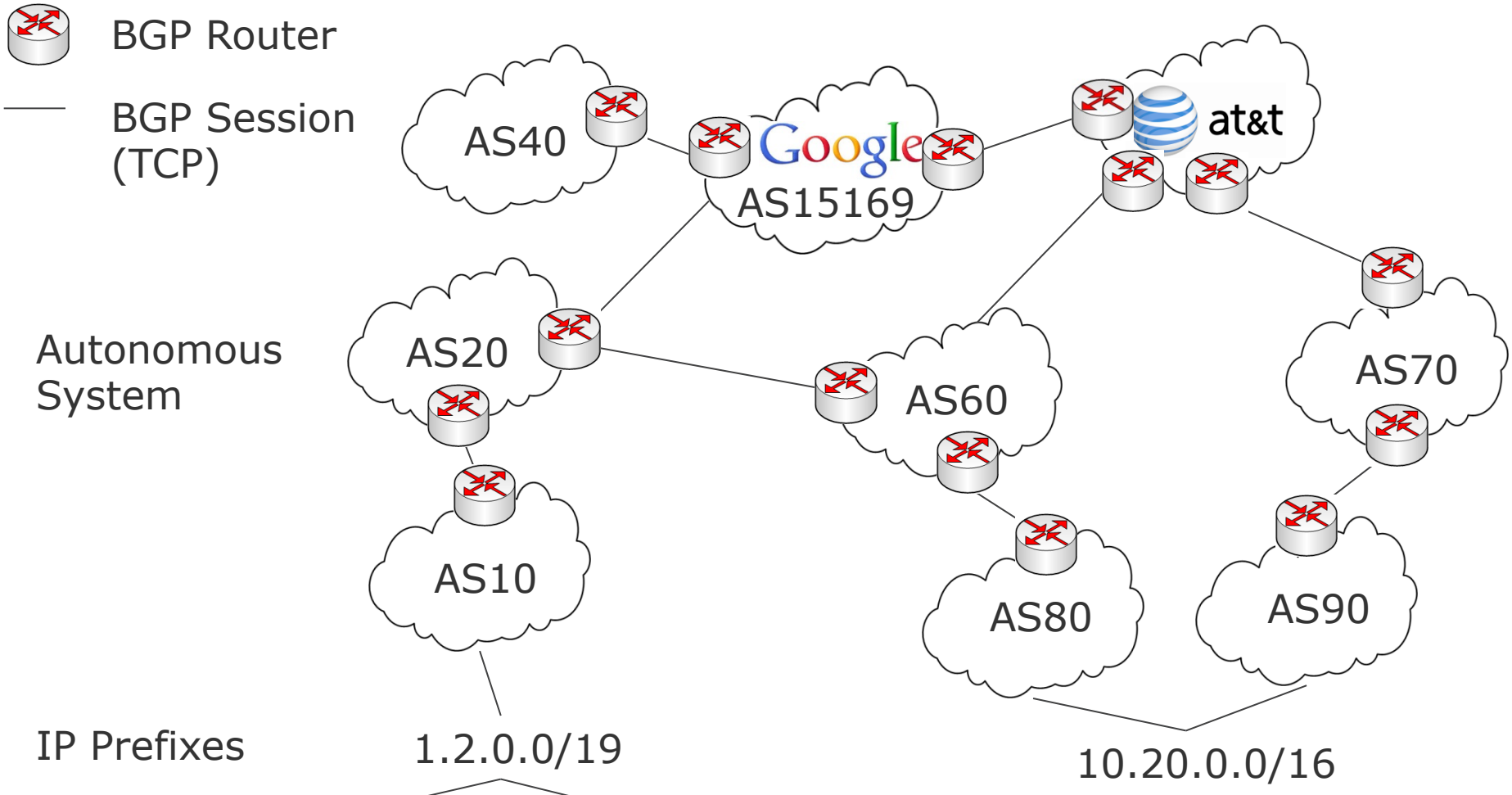
Learning of Routing Paths in BGP

- Autonomous systems are identified based on AS numbers (ASN)
  - ASN of DFN: 680, main ASN of Deutsche Telekom AG: 3320, etc.
- BGP peers establish a TCP connection for exchange of network layer reachability information and paths (default port 179)
  - Goal: Learn IP prefixes and the paths to these prefixes
- BGP router collects all (potentially redundant) paths and stores the data in the BGP RIB (**Routing Information Base**)
- BGP defines the default-free zone: All BGP router should maintain a location-dependent but complete RIB
  - At least one path to reach any IP address
  - Note, IP addresses are covered by IP prefixes

Next steps

- Create FIB (Forwarding Information Base)
- Decide which RIB entries propagate to BGP neighbors

Example BGP Peering



Example: BGP RIB Entry

TIME: 2008-7-1 02:36:49
TYPE: MSG_TABLE_DUMP/AFI_IP6
VIEW: 0 SEQUENCE: 2702
PREFIX: 1.2.0.0/19
ORIGINATED: Mon Jun 30 10:29:18 2008
FROM: 2001:0418:0000:1000:0000:0000:0000:f000 **AS15169**
AS_PATH: 15169 20 10
MULTI_EXIT_DISC: 1
COMMUNITIES: 15196:420 15169:2000 15169:3000

Inter-AS Link Metric: Prioritization of redundant inter-provider peering

Routing Policy Groups: Describe how to propagate routes

Route Decision in BGP

Objective: Path Selection in case of redundant routes to a destination

Phase 1: Calculation of Degree of Preference

- Based on local policies and attributes, a preference will be assigned to all RIB entries

Phase 2: Route Selection

- For each IP prefix, router selects
  1. Routes with highest preference
  2. Routes with shortest paths
  3. … additional tie breaking rules which lead to a unique decision

Phase 3: Route Dissemination

- Before RIB entries will be announced to neighbors, they will be filtered again based on policies

Debug BGP: BGP Looking Glasses

- BGP Looking Glasses / Routing Information Services give you access to routing tables of real BGP routers
  - Important for ISPs to debug

Example

- How does Sprint, a large ISP in the US, reach www.heise.de?
- www.heise.de → 193.99.144.85
- 193.99.144.85 belongs to 193.99.144.0/24 block
- 193.99.144.0/24 is originated by AS 12306 (Plusline), a German ISP
- RIB dumps → next slide

BGP Looking Glasses – Sprint Routing Table Digest

Sprint BGP Router in Chicago, IL, USA

BGP routing table entry for 193.99.144.0/24, version 466993885. Paths: (2 available, best #2)

3356 12306
144.228.241.247 (metric 8) from 144.228.243.246
Community: 1239:666 1239:667 1239:1000 1239:1004

3356 12306
144.228.241.247 (metric 8) from 144.228.243.241
Community: 1239:666 1239:667 1239:1000 1239:1004

Sprint BGP Router Frankfurt, Germany

BGP routing table entry for 193.99.144.0/24, version 483026157. Paths: (5 available, best #4)

2914 12306
213.206.131.94 (metric 122) from 213.206.128.26 (213.206.128.26)

3320 12306
213.206.131.1 (metric 122) from 213.206.128.25 (213.206.128.25)

3320 12306
80.150.168.149 (metric 1) from 217.147.96.25 (217.147.96.25)

3356 12306
4.68.111.141 from 4.68.111.141 (4.68.186.78)

3356 12306
217.149.32.5 (metric 121) from 217.149.32.5 (217.149.32.5)

Remark: AS3356 = Level 3, AS2914 = NTT America, AS3320 = Deutsche Telekom

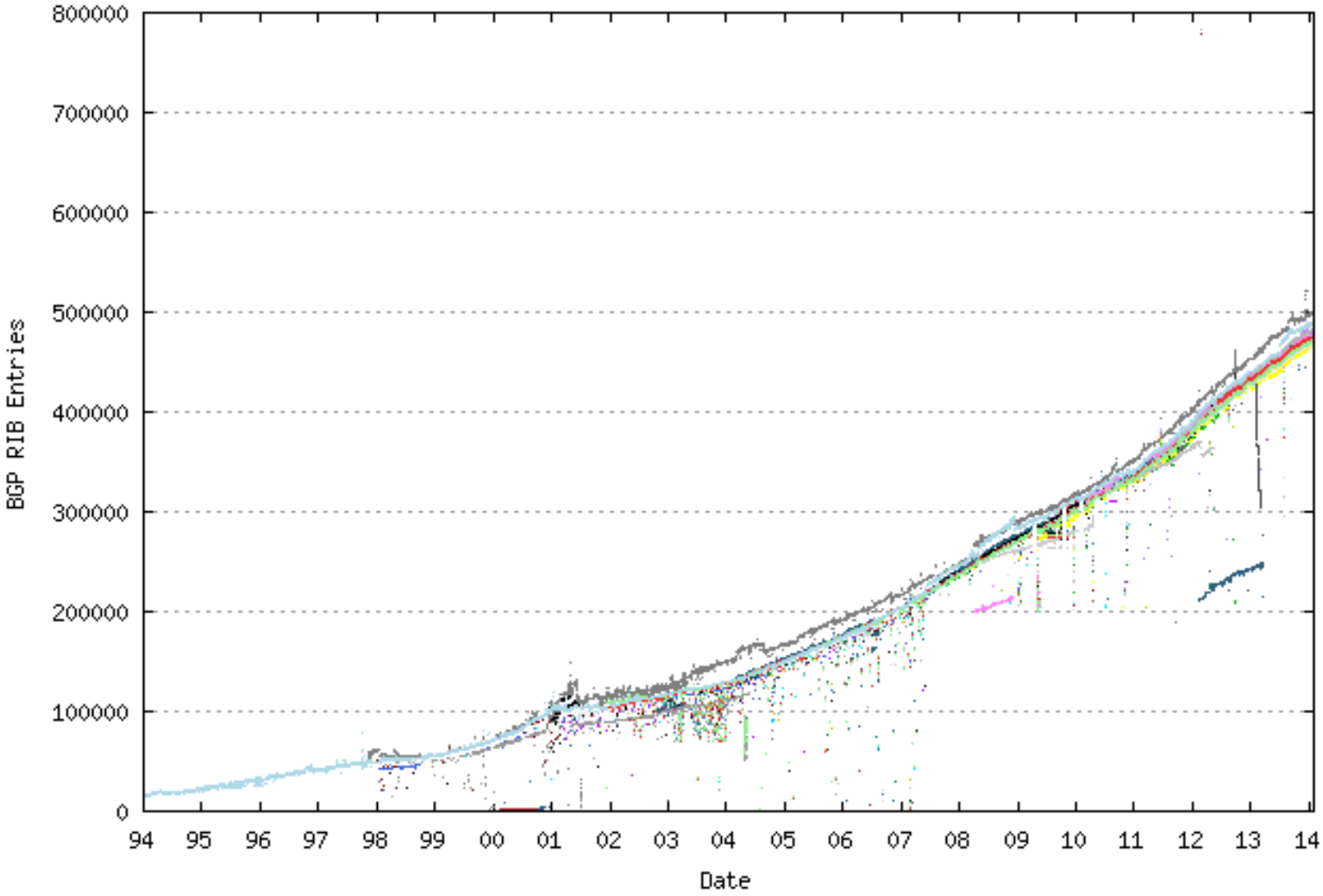
Intermediate Summary

- BGP routing information depends on the peering
- A single company can operate multiple ASes
- Multiple BGP routers of a single AS may have different paths for the same IP prefix
- Routing in the Internet might be asymmetric

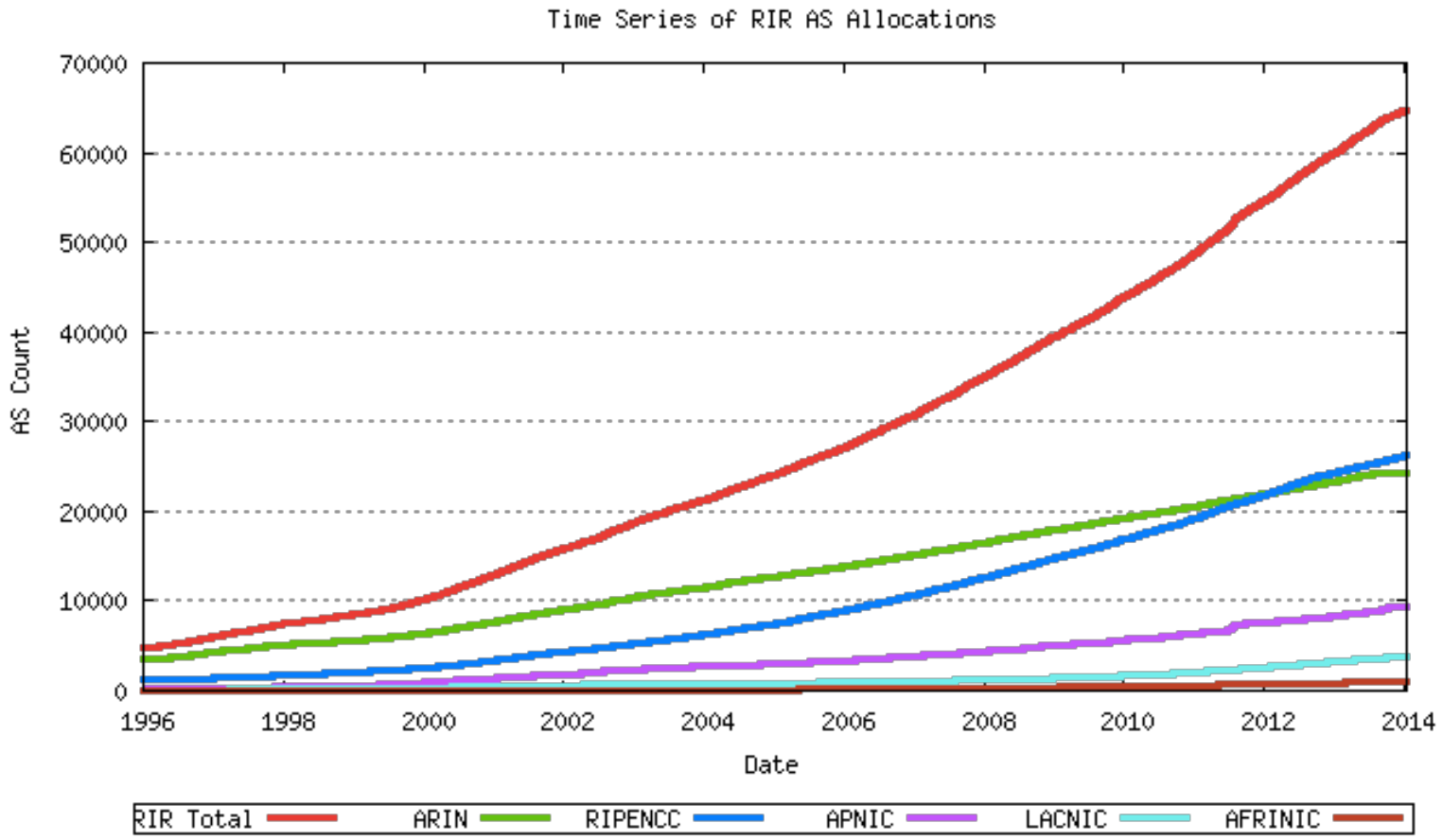
Quoting Geoff Huston

- There is no synchronized overview of the entire connectivity and policy state
- Every BGP viewing point contains a filtered view of the network
- Just because you can’t see it does not mean that it does not exist

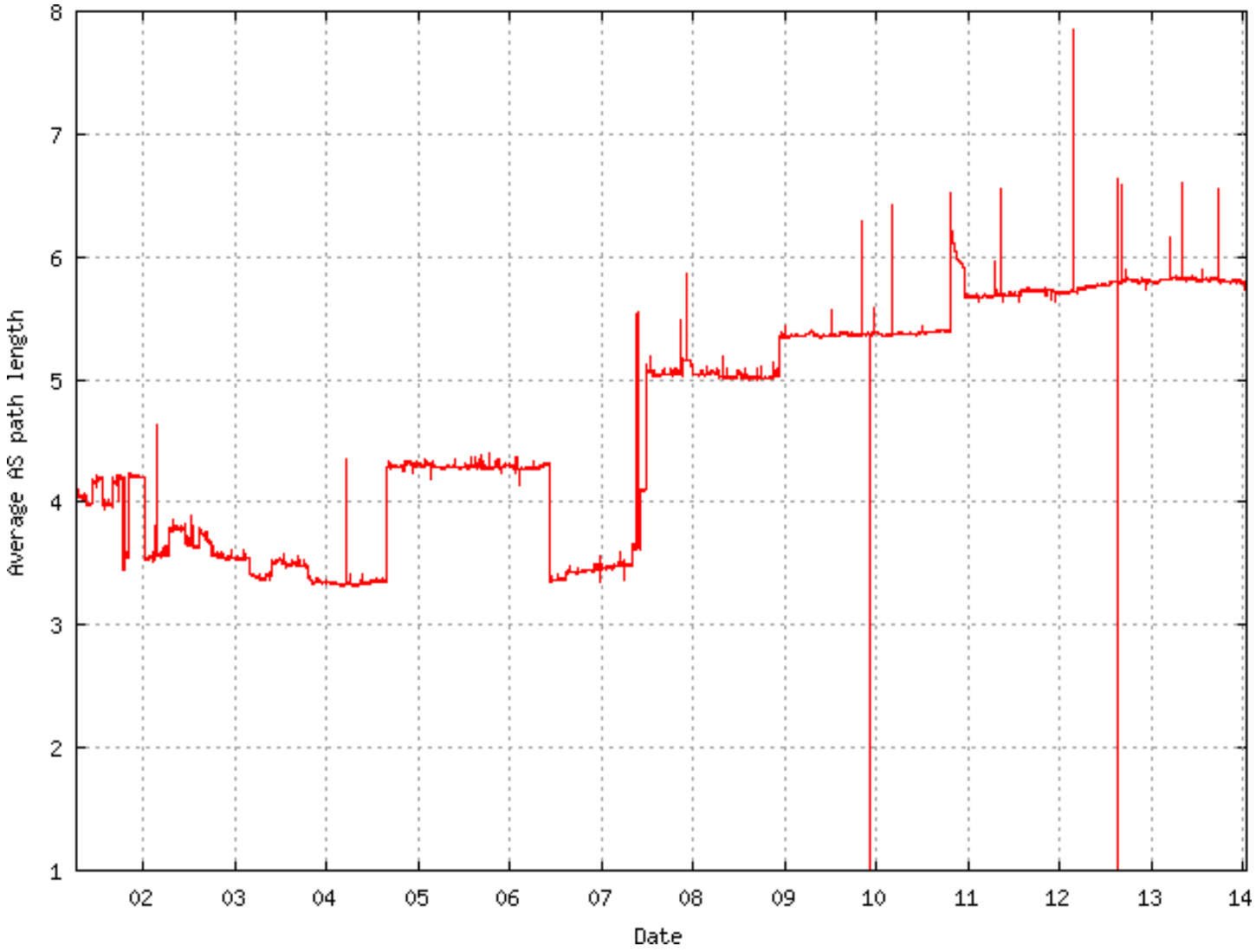
Size of BGP Routing Tables (bgp.potaroo.net)



Number of Assigned ASes (bgp.potaroo.net)



Average AS Path Length (bgp.potaroo.net)



Plot Range: 09-Apr-2001 0309 to 14-Jan-2014
Vantage Point: AS131072 (APNIC R&D)

BGP Problems and Challenges (1)

- Many issues / problems
  - Stability, routing table growth, load balancing in multi-homed networks, hijacking …
- Suggested reading
  - Delayed Internet Routing Convergence
    - 2-year measurements of BGP convergence
    - It takes between 3 and 15 minutes for paths to converge
    - http://conferences.sigcomm.org/sigcomm/2000/conf/paper/sigcomm2000-5-2.pdf
  - Analysis of BGP Update Burst during Slammer Attack
    - Consequences of the slammer worm on BGP in 2003
    - Shows nicely the effect of small ASs on the global update traffic
      - Two small ASs that contribute only to 0.25% of routing table entries caused 6% of all BGP update messages
    - http://irl.cs.ucla.edu/papers/mohit_iwdc03.pdf
  - Measurement of Highly Active Prefixes in BGP
    - Examines the stability of routes and shows that very few prefixes contribute to the majority of BGP update traffic
    - http://www.cs.ucla.edu/~lixia/papers/05Globcom.pdf

BGP Problems and Challenges (2)

Security

- BGP peering (TCP) sessions can be encrypted
- But: Advertised data might be incorrect
- Problems
  - BGP is based on trust between peering partners
  - Original version of BGP does not allow for verification of routing data

We will discuss details and solutions later in this course

CONTENT of this CHAPTER

- Fundamental Goals of the Network Layer
- Network Layer Addressing
- Internet Protocol (Version 4)
- Internet Protocol (Version 6)
- IP Address Assignment and Mapping
- NAT
- Routing
- Multicast

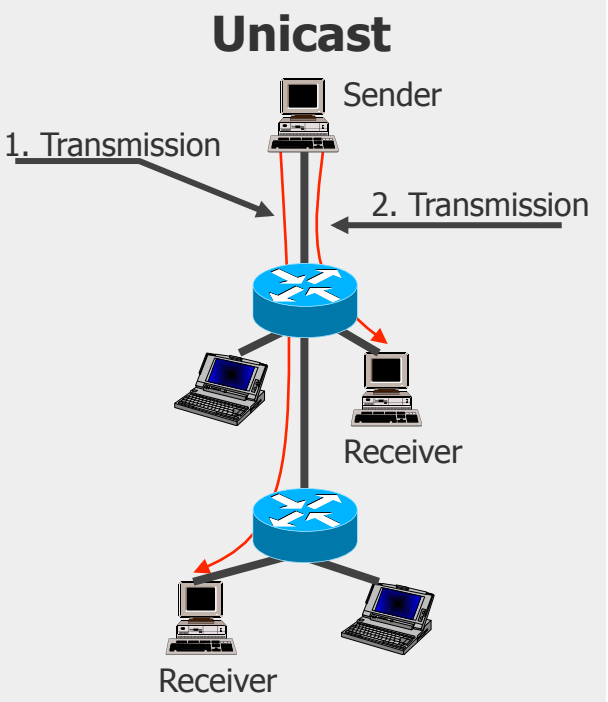
Motivation

Up until this point we have mainly focused on unicast, but…

group communication is used in many application scenarios!

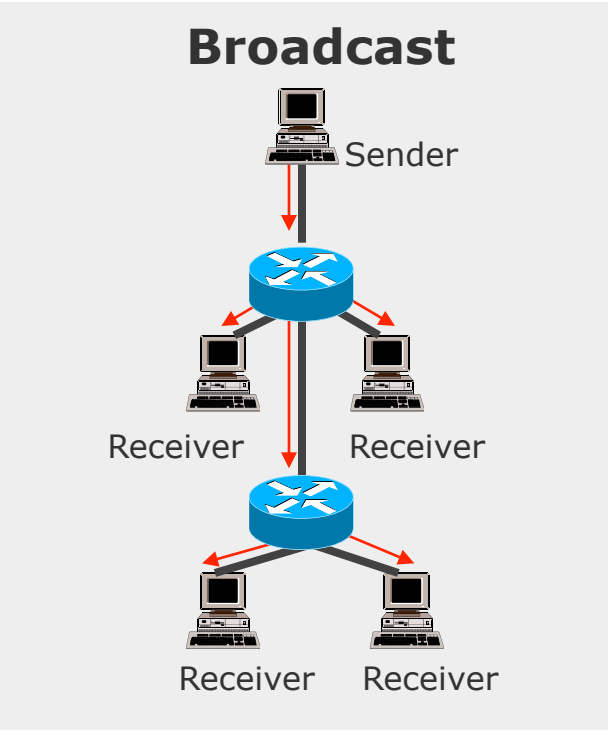
- Content broadcasting (IPTV …)
  - Multicast groups of ARD and ZDF http://www.zdf.de/ZDF/zdfportal/blob/26516094/1/data.pdf
- Voice and video conferencing
- Collaborative work
- Games (Massive multiplayer online games …)
- Distributed self-organizing systems
- Etc.

Inefficient Group Communication



Unicast is an end-to-end transmission between two hosts.

- Multiple transmissions have to be executed sequentially
- Inefficient use of time and capacities



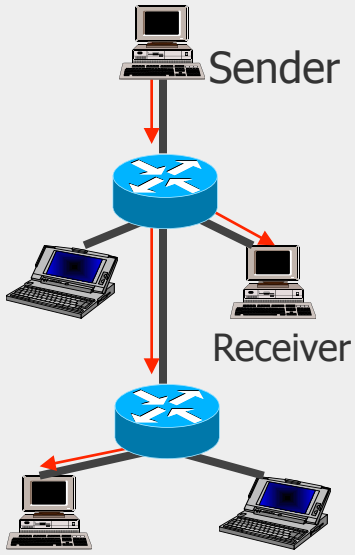
Broadcast is a one-to-all transmission

- A packet is sent to all possible receivers, many of them may not need it
- Network load by use of transmission paths, which are not needed actually

Efficient Group Communication: Multicast

Objective: Sender transmits data once, the network replicates the data to receivers

Multicast



Transmission to n > 1 selected stations: **Multicast**

Challenges:

- Support of multicast is not compulsory required to be supported by all devices in IPv4, mandatory in IPv6
- Subscription: How to indicate group interest?
- Routing: How to build forwarding paths for location-independent addresses?

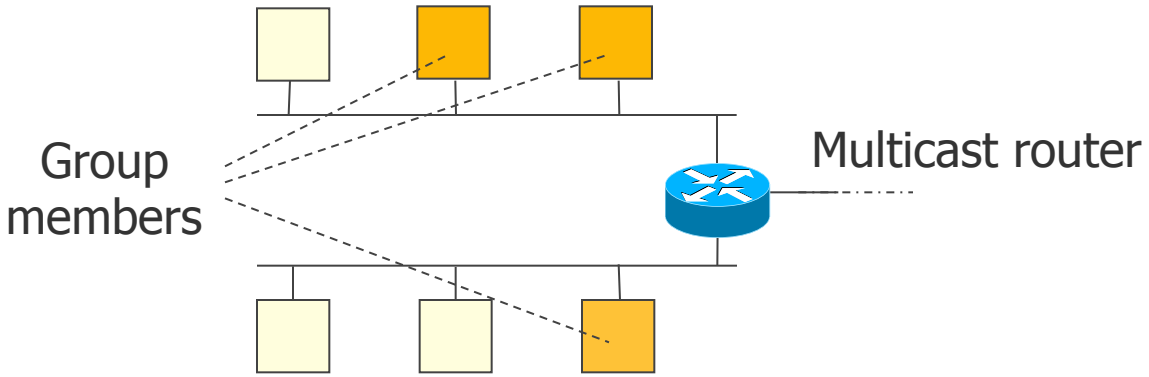
IP Multicasting (1)

- Approach to transmit IP datagrams to a group of hosts
  - Original approach specified in RFC 1112 (S. Deering et al., 1989)
  - Multicast addresses a group of hosts based on a single group address
- Two types of multicast
  - Any Source Multicast (ASM): receive what anybody sends to the group
  - Source Specific Multicast (SSM): receive only what a specified source sends
- Protocols for group subscription:
  - IGMP (IPv4)
  - MLD (IPv6)
- Protocols for multicast routing:
  - Goal: establish forwarding paths (in parallel with unicast)
  - IP layer multicast addresses will be mapped to layer 2 MAC addresses

IP Multicasting (2)

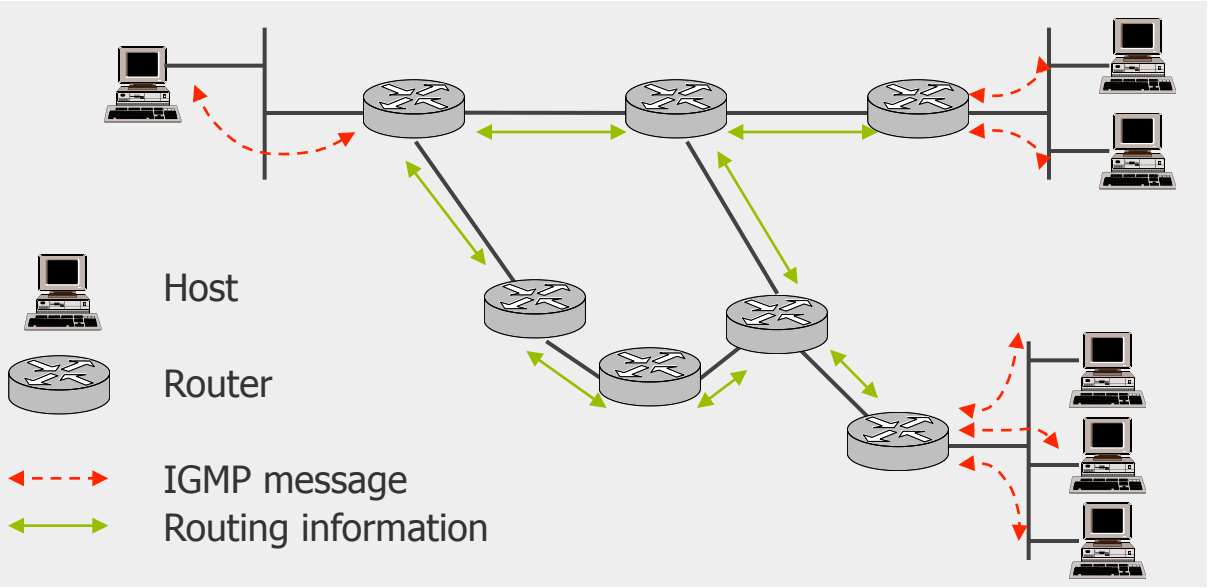
- Uses UDP as transport protocol, best effort transmission
  - Question: Why does TCP not easily work?
  - Error detection and flow control on application layer if required
- Applications identify data streams usually based on unicast source address
- No closed groups / authentication mechanisms
- ASM
  - No limitation regarding multicast sources
  - Receivers get data from unknown multicast sources
- SSM
  - Explicit selection of multicast sources
  - Source filters at routers and receivers

Internet Group Management Protocol (IGMP)

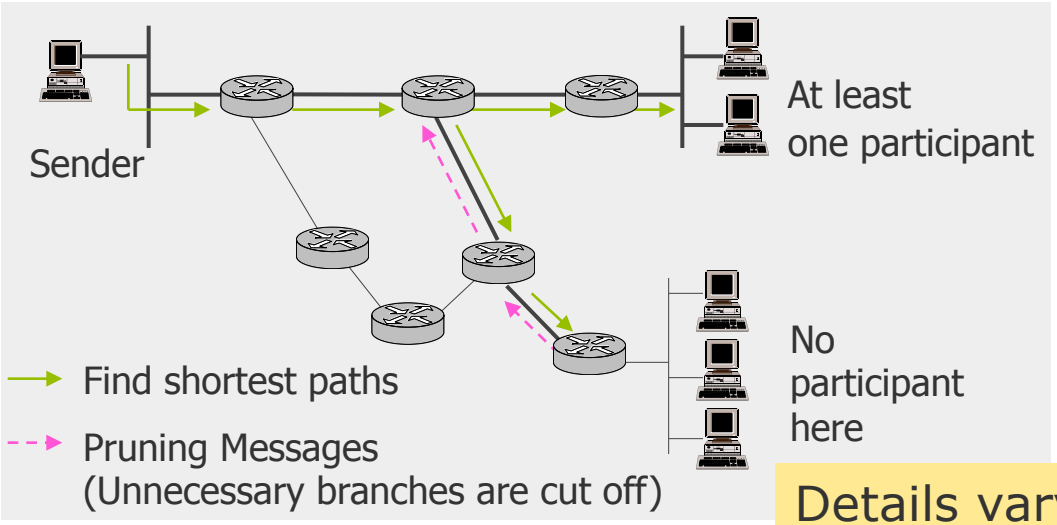


- For delivery of multicast messages to all group members that are located in different physical networks, routers need information about group associations.
- If groups are only temporary, routers have to acquire information about associated hosts by themselves.
- By means of IGMP messages (encapsulated into IP packets), a multicast receiver subscribes to a group – and thus informs routers about group interest
- Periodically, the routers ask (Polling), which groups of multicast are still present
- Routers exchange information to build multicast routing trees (routing protocol)
- Note: MLD works similarly (with IPv6)

Multicast Control Path – an example



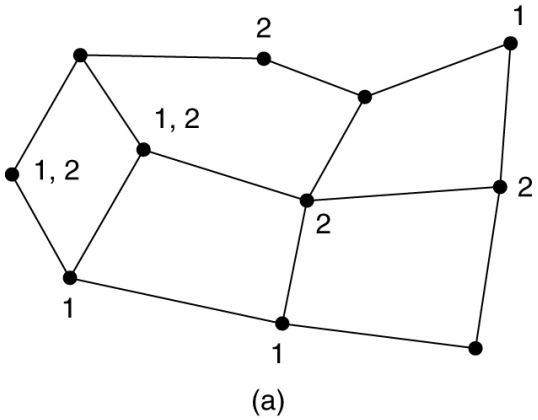
- The routers exchange their routing information
- By means of IGMP messages, group associations are being passed on
- For each multicast address the routers manage routing information



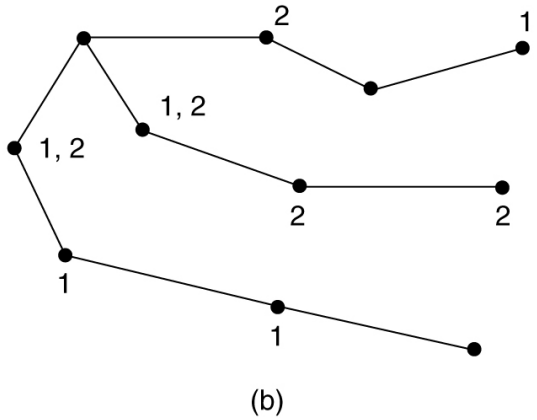
- The routing protocol computes the shortest paths to all computers in the network
- Routers, which do not have participants in their network, can send back Pruning Messages; next time no more multicast packets are sent to this routers

Details vary by the specific protocol used!

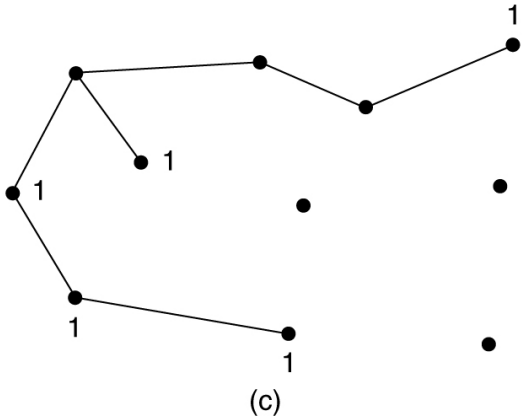
Multicast Groups: Multicast Tree Examples



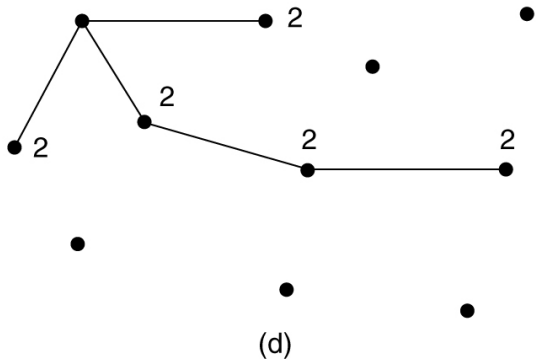
Network with interest into two multicast groups



Sink Tree for the left router



Multicast routing tree for group 1



Multicast routing tree for group 2

Multicast Routing – PIM-SM as example

- Protocol Independent Multicast – Sparse Mode (RFC 2362)
  - Can support Internet-wide MC groups, does not require specific unicast protocol
  - No flooding plus pruning but explicit construction of a tree
- Assumptions
  - Only a very low percentage of nodes will subscribe to a Multicast group
- Multicast receiver: joining a multicast group
  - IGMP join in the local subnet
  - PIM join to a so-called rendezvous point (RP)
  - PIM join between RP and sender
  - (Routers forward join messages to RP)
- Multicast sender
  - Send register messages to RP
  - RP distributes data along MC tree

Multicast API

Berkeley Sockets set/getsockopt():

- **IP_ADD_MEMBERSHIP** to join a multicast group on a specific interface
- **IP_DROP_MEMBERSHIP** to leave a multicast group (no protocol action initiated with IGMP v1, but there is with IGMP v2)
- **IP_MULTICAST_IF** to set or get default interface for use withmulticast sends
- **IP_MULTICAST_LOOP** to disable loopback of outgoing multicast datagrams
- **IP_MULTICAST_TTL** to set the IP time-to-live of outgoing multicast datagrams.

- Attention: The multicast sender does not need to register for a multicast group!

Java Multicast API

Package: java.net

Class MulticastSocket

With the methods:

*public void **joinGroup**(InetAddress mcastaddr)*

*public void **leaveGroup**(InetAddress mcastaddr)*

Example multicast receiver:

```
// join a Multicast group and send the group join ...
String msg = "Hello";
InetAddress group = InetAddress.getByName("228.5.6.7");
MulticastSocket s = new MulticastSocket(6789);
s.joinGroup(group);
DatagramPacket hi = new DatagramPacket(msg.getBytes(), msg.length(), group, 6789);
s.send(hi);
// get their responses!
byte[] buf = new byte[1000];
DatagramPacket recv = new DatagramPacket(buf, buf.length);
s.receive(recv);
...
// OK, I'm done talking - leave the group...
s.leaveGroup(group);
```

Multicast in the Real World

- Global deployment of multicast is limited
- Intra-domain multicast (i.e., multicast within an ISP)
  - Largely deployed due to IPTV, gaming etc.
- Inter-domain multicast (i.e., multicast between ISPs)
  - Very limited deployment, almost not available
  - Problems (see also C. Diot et al., IEEE Network Magazine, 14(1), pp. 78-88, 2000)
    - Globally scalable routing is hard to achieve
    - Economic incentives for ISPs are not obvious (ISP of source saves money)
- Mobile multicast: Routing gets much more complex due to the publish/ subscribe approach of multicast

Summary

- TCP/IP reference model defines only one protocol for layer 3
  - The Internet Protocol (IP)
  - Connectionless transmission, data packets are forwarded hop-by-hop
  - Supported by routing protocols to determine the best way to a destination
  - ICMP for exchange of control messages
  - ARP for mapping of IP addresses to MAC addresses
- But, there are several problems with the "current" IPv4
  - address space, security, mobility, quality of service, …
  - Large number of additional protocols to deal with these problems: network address translation, IPsec, Mobile IP, IntServ, DiffServ, MPLS, …
- Successor IPv6 could deal with (some of) the problems, but when/where/ to what extend will it come?
  - See e.g. http://resources.potaroo.net/iso3166/v6cc.html for current usage