Networks and Communications

"The Network/Internet Layer"

Konstantinos Gkoutzis
Imperial College

Outline

- The Network Layer
- IPv4
- Routing
- Internetworking
- IPv6

Introduction

- The *Network* Layer
  - also called The *Internet* Layer
- Here lives the *Internet Protocol* (IP)
- Data travel in *Datagrams* here
  - IP Datagrams
    - (*we also call them Packets*)
- Data are *Fragmented* into Packets
  - and a *Layer 3 Header* is added in front of each
- *What device makes sure that Packets reach their destination?*

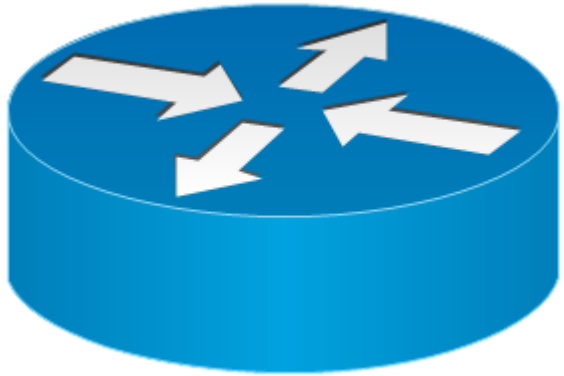
Router


LINKSYS
Wireless-G Broadband Router
Secure Easy Setup
Wireless-G
Ethernet
Power
DMZ
WLAN
1 2 3 4 Internet
Model WRT54GS

Router

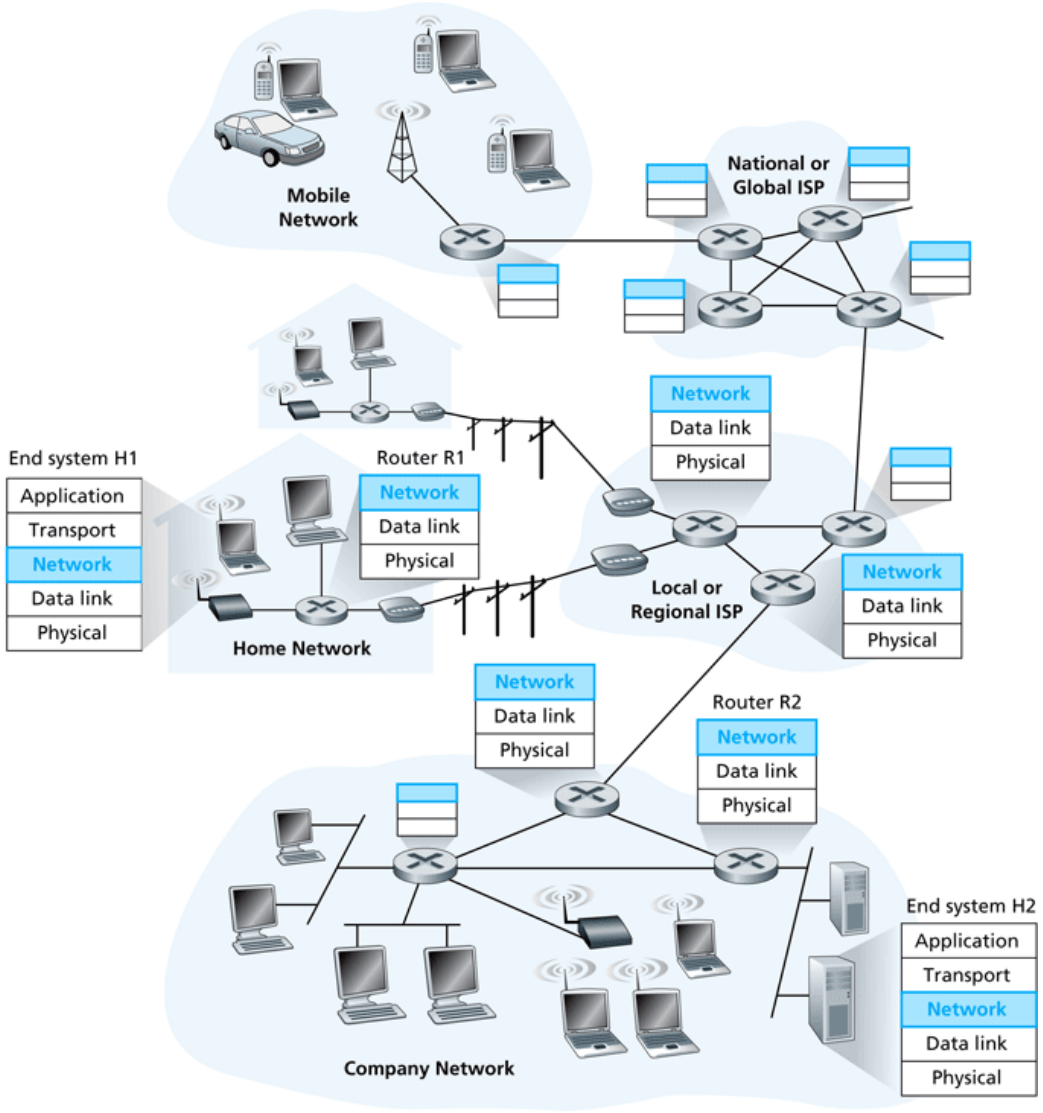
Router (cont'd)

- Fundamental component of the network layer
- (*A node in a graph*)
- Finite set of input/output (physical) connections
- Has *Interfaces* or *(physical) Ports*
  - *(not to be confused with the Transport Layer ports)*



The Network Layer

- Provides facilities for getting data from a **source** to a **destination**
- This may require making many **hops** at intermediate routers along the way (**routing**)
  - *Note: the Data Link layer moves Frames only within the same (sub)network*
- It must know the **topology** of the network to choose appropriate paths
- **Load Balancing** among routes is required
- It also has to deal with **network heterogeneity**



Remember these terms?

- **Connection-oriented**
  - modeled after the phone system
  - acts like a pipe: the sender puts in objects (*bits*) at one end, and then the receiver takes them out at the other end

Q: Which of these is IP?

- **Connection-less**
  - modeled after the postal system (*think of letters*)
  - each message is routed through the system independent of all the others

- **Circuit-switched**
  - Always uses the same path, which requires little processing after the original setup

- **Packet-switched**
  - Information is transmitted in packets (*formatted units of data*)
    - packets are forwarded from one router to the next based on forwarding decisions

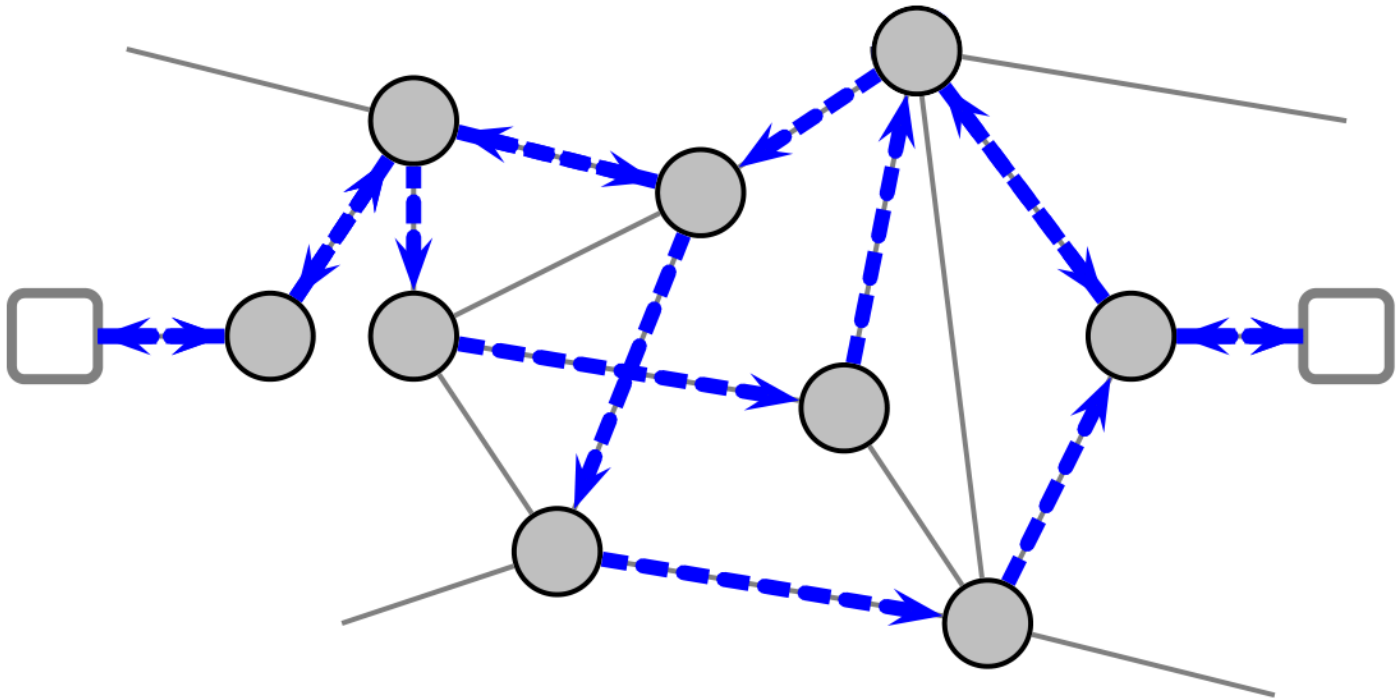
Datagram Networks

- **Packet-switched network**
  - information is transmitted in discrete units called Datagrams
    - (*a "telegram" of data*)

- **Connection-less service**
  - datagram is a *self-contained message*
    - treated independently by the network
      - no connection setup/tear-down phase

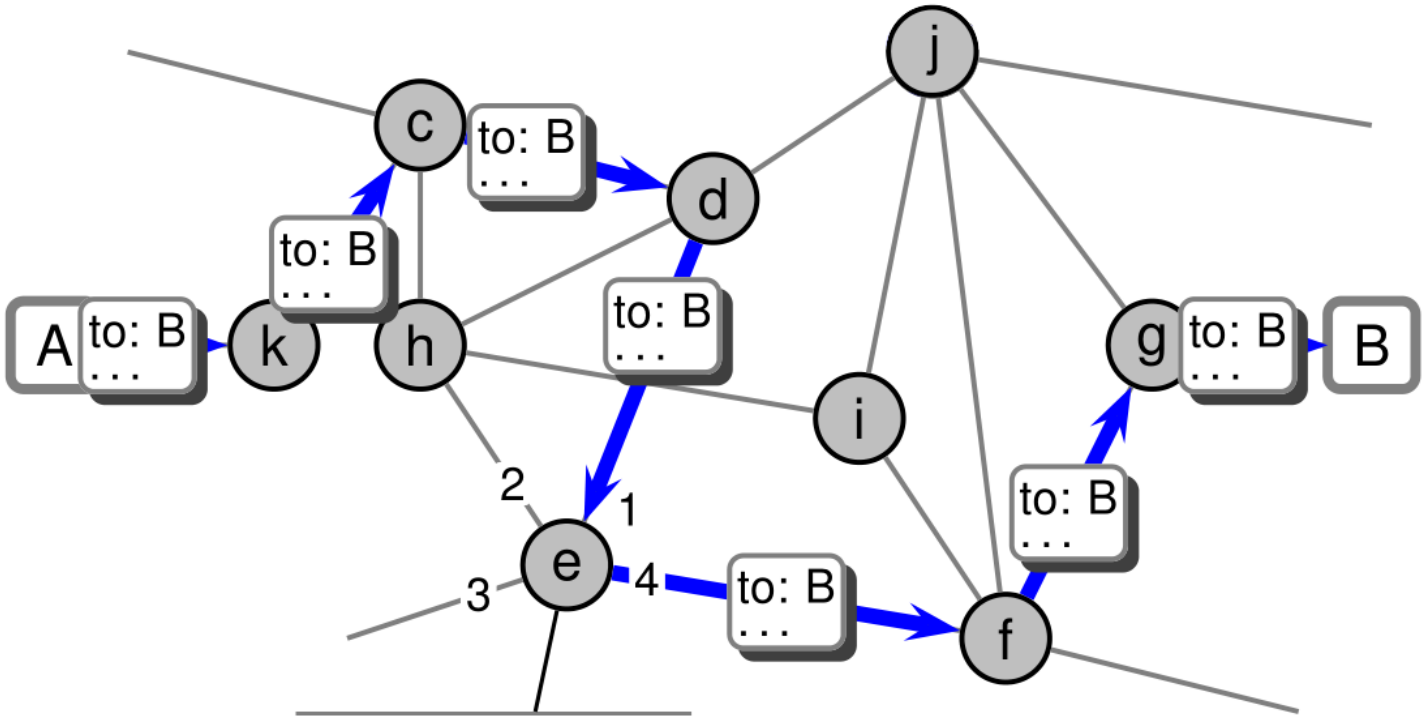
- **"Best-effort" service**
  - delivery guarantee: *none*
  - maximum latency guarantee: *none*
  - bandwidth guarantee: *none*
  - in-order delivery guarantee: *none*
  - congestion indication: *none*

Datagram Networks (cont'd)



- Potentially multiple paths for the same source/destination
- Potentially asymmetric paths

Forwarding



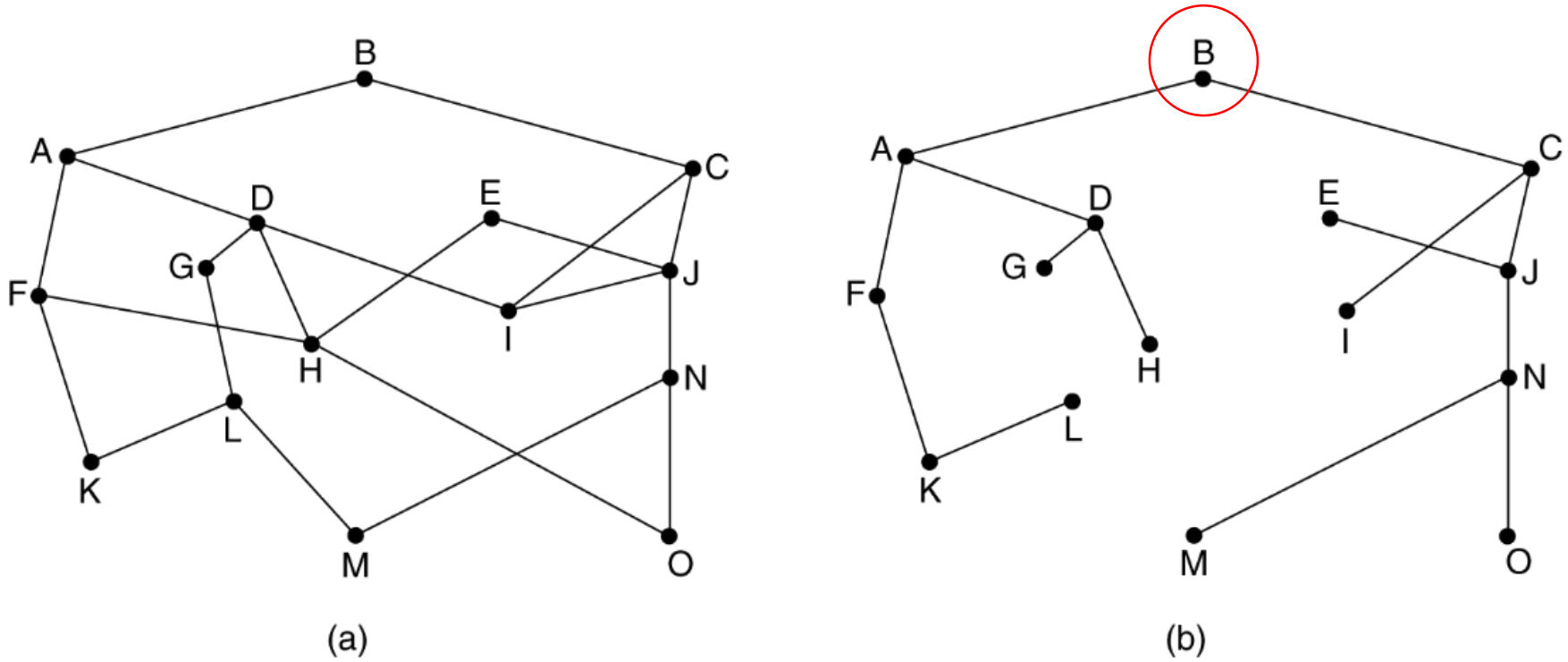
- **A** sends a datagram to **B**
- The datagram is forwarded towards **B**

Forwarding (cont'd)

- Input: datagram destination
- Output: output port
- Simple design: "Forwarding Table"
- *Issues*
  - *how big is the forwarding table?*
  - *how fast does the router have to forward datagrams?*
  - *how does the router build and maintain the forwarding table?*

Routing

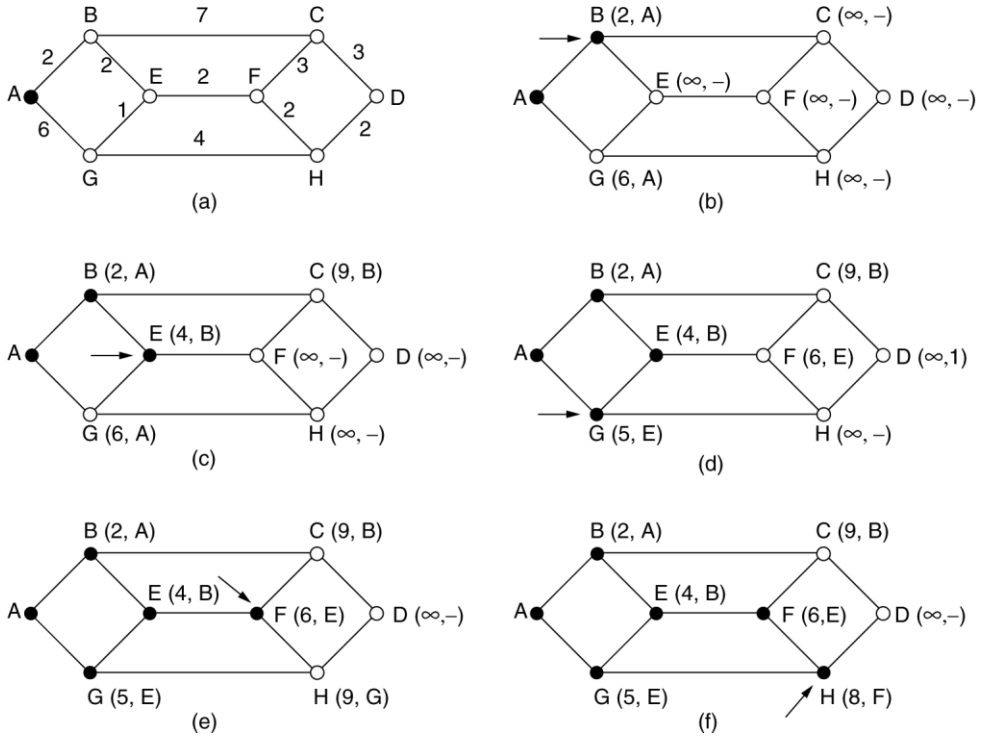
- Routers cooperate to find the **best routes** between all pairs of nodes
- Routers have to collaborate to build a sink tree (*or an approximation*) for each node



- The sink tree of each node *does not contain loops*
  - it only contains the **optimal routes** towards that specific host

Shortest Path Routing (SPR)

- Dijkstra's Algorithm
  - Labels on arcs represent **cost** (*e.g. delay, hops, etc.*)
  - *During each step, select newly reachable node with lowest cost, and then add edge to that node to the tree built so far*



Flood Routing

- Forward incoming packet across every outgoing link, *except* the original source

- **Problem**: how to avoid "drowning by packets"?
  - Use a **hop counter**:
    - after a packet was forwarded across N routers, discard it
    - *need to decide on the "correct" hop count however*
  - Forward packets only once (i.e. **avoid directed cycles**)
    - requires storing sequence numbers per source address
  - **Flood selectively**: only in a direction that "makes sense"

- Flooding always chooses **shortest path** because all paths are explored in parallel
  - However, **overhead** grows significantly

- Flooding makes sense only when **robustness** is needed (*i.e. "if it exists, get me there"*)
  - e.g. discovery of unknown destinations

Distance Vector Routing

- The previous protocols are **static**
  - *i.e. they do not take current* ***network load*** *into account*

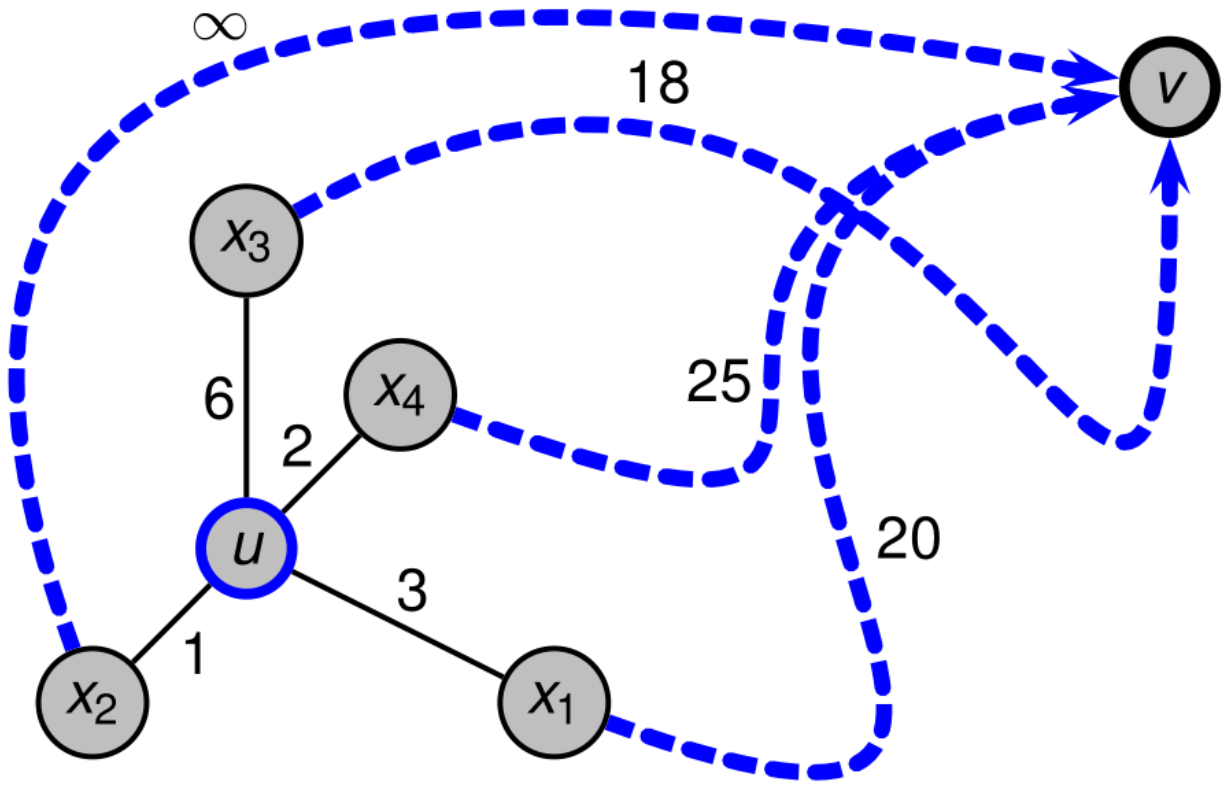
- We need to consider **dynamic protocols**

- **Distance Vector Routing** (DVR)
  - *a.k.a. Bellman-Ford*
  - consider costs that direct neighbours are advertising to get packet to destination
  - select neighbour whose advertised cost, added with own cost to get to that neighbour, is lowest
  - advertise new cost to other neighbours

Distance Vector Routing (cont'd)

- Main idea is expressed by the Bellman-Ford equation

$$D'_u[v] = \min_{x \in neighbors(u)} (c(u,x) + D_x[v])$$



Example 1: Distance Vector

a	a	b	c	d
D_a	0	2	∞	4
D_b	∞	∞	∞	∞
D_d	∞	∞	∞	∞

b	a	b	c	d
D_b	2	0	1	∞
D_a	∞	∞	∞	∞
D_c	∞	∞	∞	∞

c	a	b	c	d
D_c	∞	1	0	6
D_b	∞	∞	∞	∞
D_d	∞	∞	∞	∞

d	a	b	c	d
D_d	4	∞	6	0
D_a	∞	∞	∞	∞
D_c	∞	∞	∞	∞

a	a	b	c	d
D_a	0	2	**3**	4
D_b	2	0	1	∞
D_d	4	∞	6	0

b	a	b	c	d
D_b	2	0	1	**6**
D_a	0	2	∞	4
D_c	∞	1	0	6

c	a	b	c	d
D_c	**3**	1	0	6
D_b	2	0	1	∞
D_d	4	∞	6	0

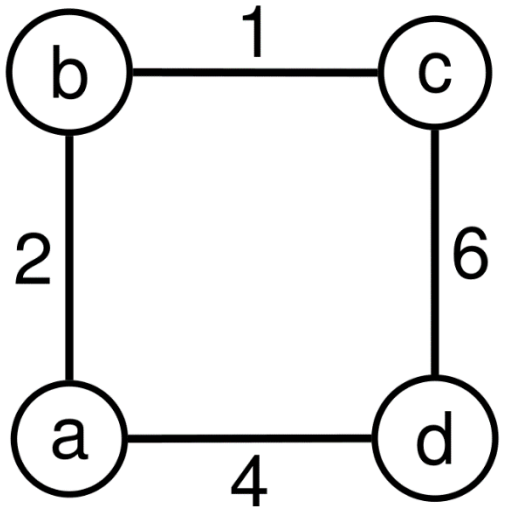
d	a	b	c	d
D_d	4	**6**	6	0
D_a	0	2	∞	4
D_c	∞	1	0	6

a	a	b	c	d
D_a	0	2	3	4
D_b	2	0	1	6
D_d	4	6	6	0

b	a	b	c	d
D_b	2	0	1	6
D_a	0	2	3	4
D_c	3	1	0	6

c	a	b	c	d
D_c	3	1	0	6
D_b	2	0	1	6
D_d	4	6	6	0

d	a	b	c	d
D_d	4	6	6	0
D_a	0	2	3	4
D_c	3	1	0	6



Example 2: Distance Vector

Imperial College London

a	a	b	c	d
D_a	0	2	∞	4
D_b	∞	∞	∞	∞
D_d	∞	∞	∞	∞

b	a	b	c	d
D_b	2	0	1	∞
D_a	∞	∞	∞	∞
D_c	∞	∞	∞	∞

c	a	b	c	d
D_c	∞	1	0	9
D_b	∞	∞	∞	∞
D_d	∞	∞	∞	∞

d	a	b	c	d
D_d	4	∞	9	0
D_a	∞	∞	∞	∞
D_c	∞	∞	∞	∞

a	a	b	c	d
D_a	0	2	**3**	4
D_b	2	0	1	∞
D_d	4	∞	9	0

b	a	b	c	d
D_b	2	0	1	**6**
D_a	0	2	∞	4
D_c	∞	1	0	9

c	a	b	c	d
D_c	**3**	1	0	9
D_b	2	0	1	∞
D_d	4	∞	9	0

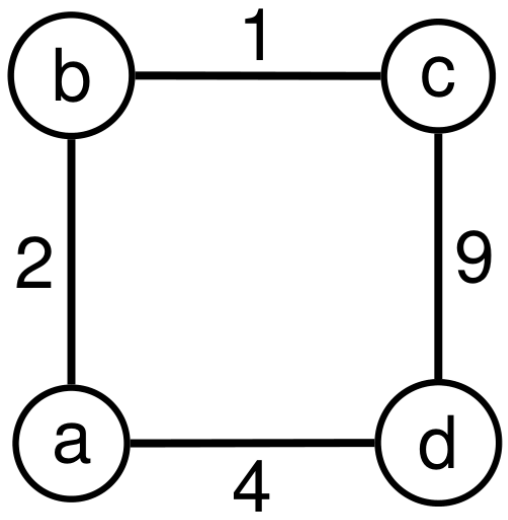
d	a	b	c	d
D_d	4	**6**	9	0
D_a	0	2	∞	4
D_c	∞	1	0	9

a	a	b	c	d
D_a	0	2	3	4
D_b	2	0	1	6
D_d	4	6	9	0

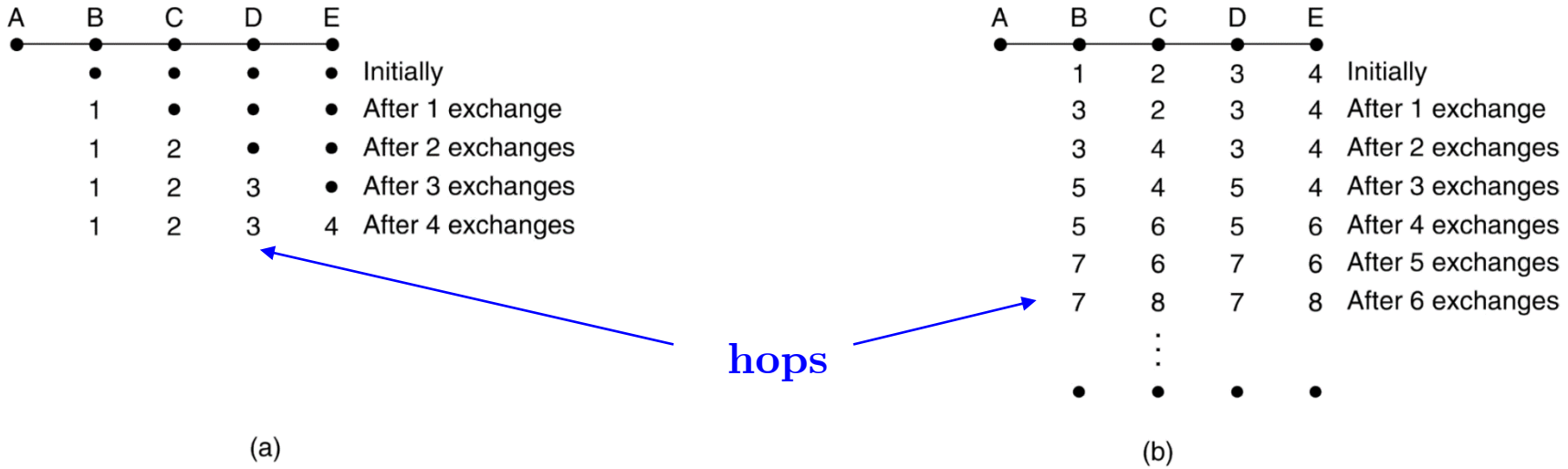
b	a	b	c	d
D_b	2	0	1	6
D_a	0	2	3	4
D_c	3	1	0	9

c	a	b	c	d
D_c	3	1	0	**7**
D_b	2	0	1	6
D_d	4	6	9	0

d	a	b	c	d
D_d	4	6	**7**	0
D_a	0	2	3	4
D_c	3	1	0	9



The problem with Distance Vector Routing



(a) (b)

- Example (a): Good news propagates quickly
  - e.g. **A** was down for a while, but then it comes back up
- Example (b): Bad news propagates slowly
  - **A** suddenly goes down again
  - **Count-to-infinity problem**
  - all routers increase their route up to infinity thinking they know where **A** is
    - *we can set infinity to => longest_ acceptable_ path + 1 (protocol dependent)*

Link State Routing

- In 1979, **Link State Routing** was published, aiming to replace DVR

- Goals:
  - Broadcast information about network topology to all routers
  - Each router calculates **sink tree** to other routers

- To achieve this, each router:
  1. Discovers direct neighbours and gets their network addresses
  2. Calculates cost for sending packet to each neighbour
  3. Constructs **link state advertisement** (**LSA**) packet with all information
  4. Sends/Collects that packet to/from ***all*** other routers (*not just neighbours*)
  5. Runs Dijkstra's algorithm locally (*to find shortest paths*)

- You are storing information about:
  a) your neighbours, b) the entire network, c) and the routing table you generated

Link State Routing (cont'd)

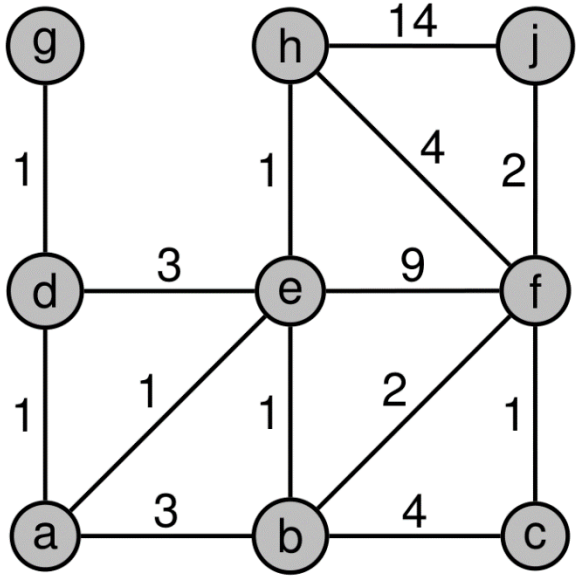
- How to identify neighbours?
  - Routers know their local network interfaces;
    - they send "**HELLO**" packet through each interface;
      - router on other end responds with its address.

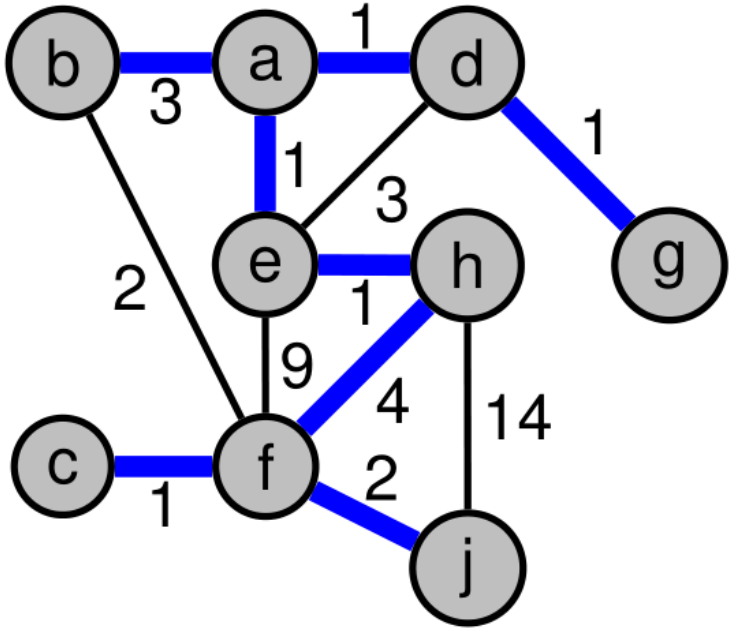
- How to measure Link Cost?
  - Send "**ECHO**" packet through each interface, and measure round-trip delay
    - *gives reasonable estimate of actual delay*

- **Problem**: do we take local load into account (*i.e. do we measure from the moment the packet is queued, or from when we actually send it on the network*)?
- **Pros**:
  - we can choose better routes that take current network conditions into account
- **Cons**:
  - may redirect traffic in such a way that the alternative route becomes overloaded

Link State Routing (cont'd)

- Every router uses flooding to sends its LSA to all other routers
- Once we have all LSAs from every router, we have complete knowledge of network
  - and we can run Dijkstra locally





$$LSA_a = \{(a,b,3),(a,e,1),(a,d,1)\}$$
$$LSA_h = \{(h,e,1),(h,f,4),(h,j,14)\}$$
$$LSA_d = \{(d,a,1),(d,g,1),(d,e,3)\}$$
$$LSA_f = \{(f,c,1),(f,b,1),(f,e,3),(f,h,4),(f,j,2)\}$$

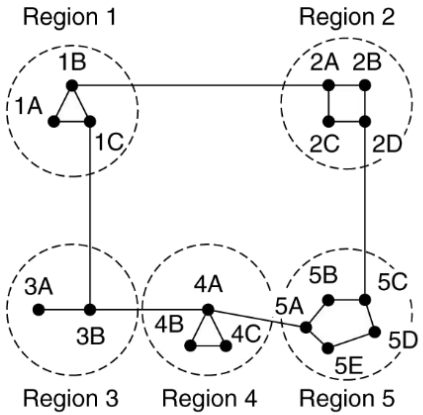
Comparison: DVR vs LSR

	Distance Vector Routing	Link State Routing
Network Knowledge	Local	Global
Computation	Global	Local
Synchronisation	Gradual	"Instant" (*after SPR calculation*)

Hierarchical Routing

- **Problem**: No routing algorithm discussed so far can scale..!
  - all of them require each router to know about all others
  - *too demanding with respect to memory and processing power*

- **Solution**: Go for suboptimal routes by introducing **regions**, and separate algorithms for **intra-region** and **inter-region** routing
  - *two or three levels generally sufficient*



(a)

Full table for 1A

Dest.	Line	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

(b)

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(c)

- **No optimal** routes any more (*you just know where to go*)

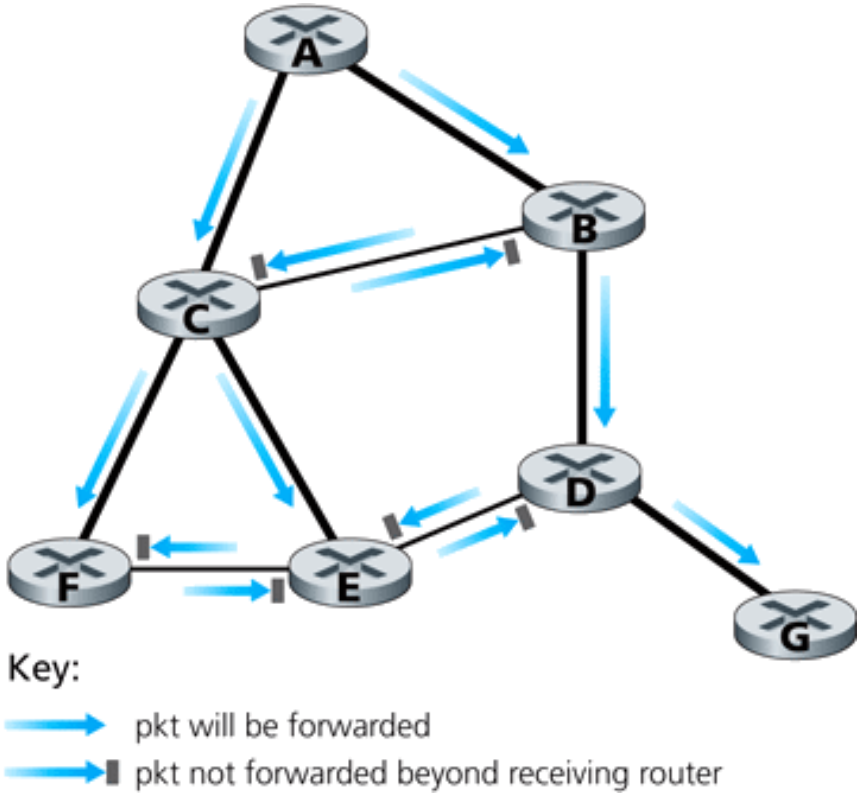
Broadcast Routing

- **Problem**: Send message to (almost) every host on the network
  - (*feasible only on LANs or a small WAN*)
- Send message to each host *individually?*
  - **not efficient**
- Use **flood** routing?
  - acceptable, provided that we can *limit* the flood
- A multi-destination routing approach, by using a list that is sent *with* the packet?
  - router checks destinations and splits list when forwarding it to different interfaces
  - but message must contain all destinations somehow
- Build sink tree at source and use that for multicast route?
  - sink tree must be spanning tree & routers need to agree on trees somehow

Broadcast Routing (cont'd)

- **Problem**: Suppose we don't know the spanning tree. How can we construct one at low cost?
- **Solution**: Exploit information available at each router

- **Reverse-path Forwarding (RPF) broadcast**
  - every router forwards/broadcasts a packet to every adjacent router, *except* where it came from
  - router X accepts broadcast packet p originating at router A **only if** p arrives on a link that is on direct (**unicast**) path from X to A

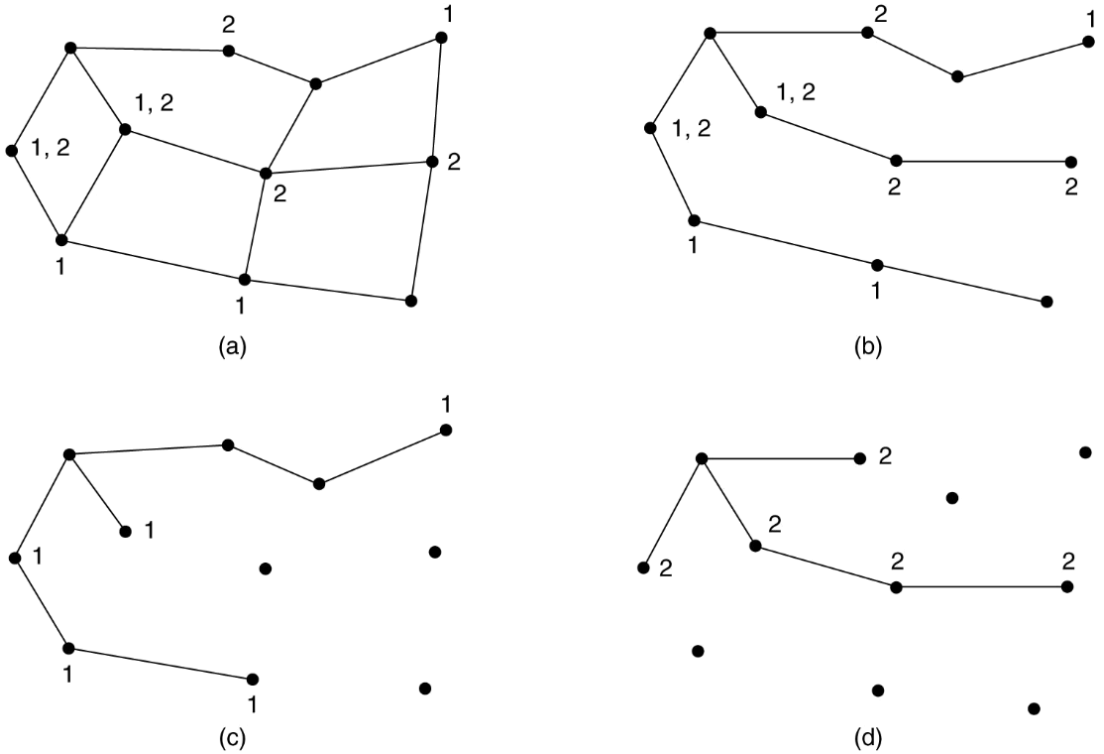


- **Question**
  - Could this be used to disseminate LSAs (Link-State Advertisements)?
    - No, because this is a unicast approach to build the spanning tree
      - LSR relies on broadcasting methods (e.g. flooding)

Multicast Routing

- **Problem**: Message should go only to a subset of nodes
  - *need to know when host enters or leaves multicast group*

- **Solution**: Construct spanning tree at each router
  - use a group-id to prune paths to nodes that do not contain members of group

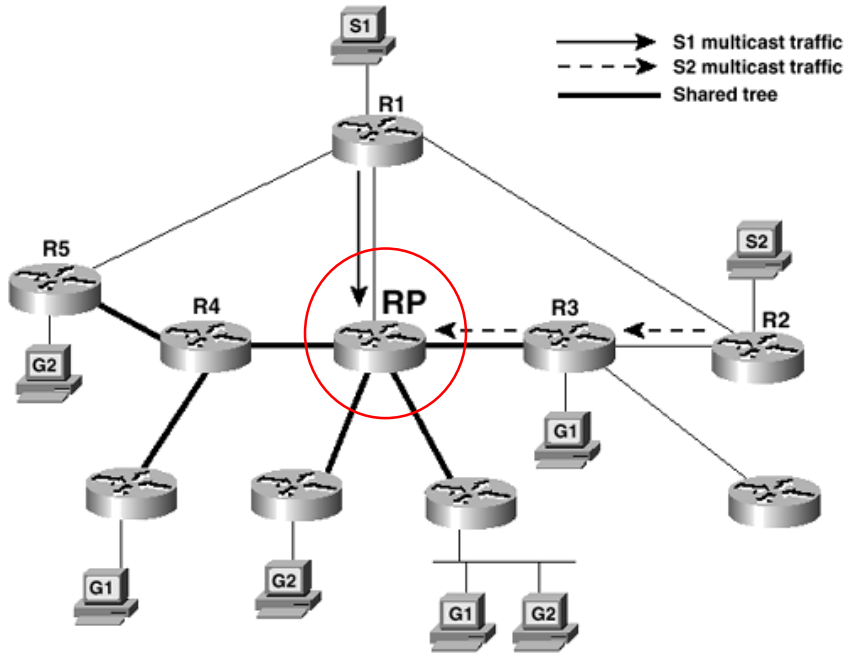


(a) entire network
(b) spanning tree for *leftmost router*
(c) multicast tree for *group 1*
(d) multicast tree for *group 2*

Multicast Routing (cont'd)

- **Question:**
  - What is the main issue with the solution which was just presented?
    - Scalability! We need to maintain a *spanning tree per source*

- **Another solution**: Use ***Core-based trees***
  - single spanning tree *per group*, with root (core) near the *middle* of the group
  - to send multicast message, host sends it to core, which performs multicast along tree
  - Note: tree may not optimal for *all* sources but it drastically reduces *storage* and *maintenance* costs

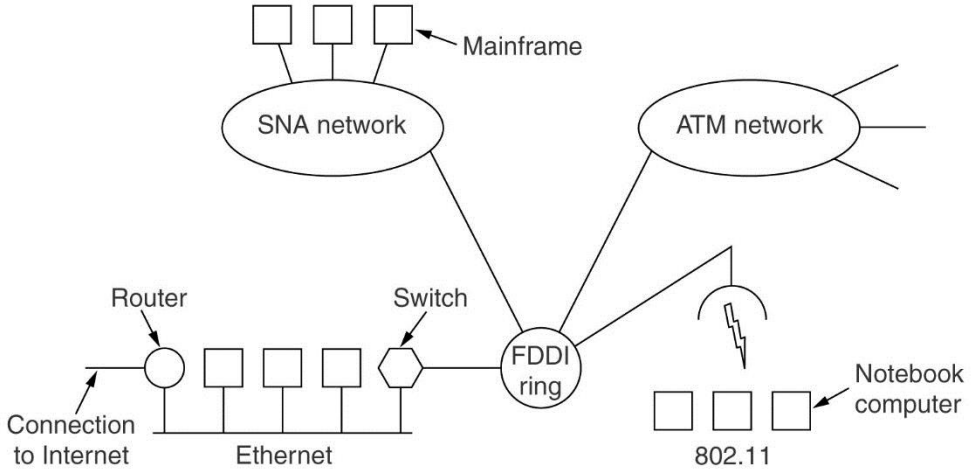


During your break:

Find a lab machine and run: **traceroute www.google.co.uk**

How many **hops** did you get?

Internetworking



- **Problem**: We have all these networks, with **different protocol stacks**, and somehow they need to talk to each other..!
- **Not the solution**: Force all networks to run the same protocol stack
- **Solution**: Construct ***gateways*** that interconnect different kinds of networks

Network Terminology

- **PAN**
  - Personal Area Network
  - e.g. your phone connected to your PC via Bluetooth
- **LAN**
  - Local Area Network
  - e.g. your laptop connected to your desktop using a WiFi Access Point
- **MAN**
  - Metropolitan Area Network
  - e.g. the bus stop signs that display live bus schedules (*not by using the Internet*)
- **WAN**
  - Wide Area Network
  - e.g. the Internet

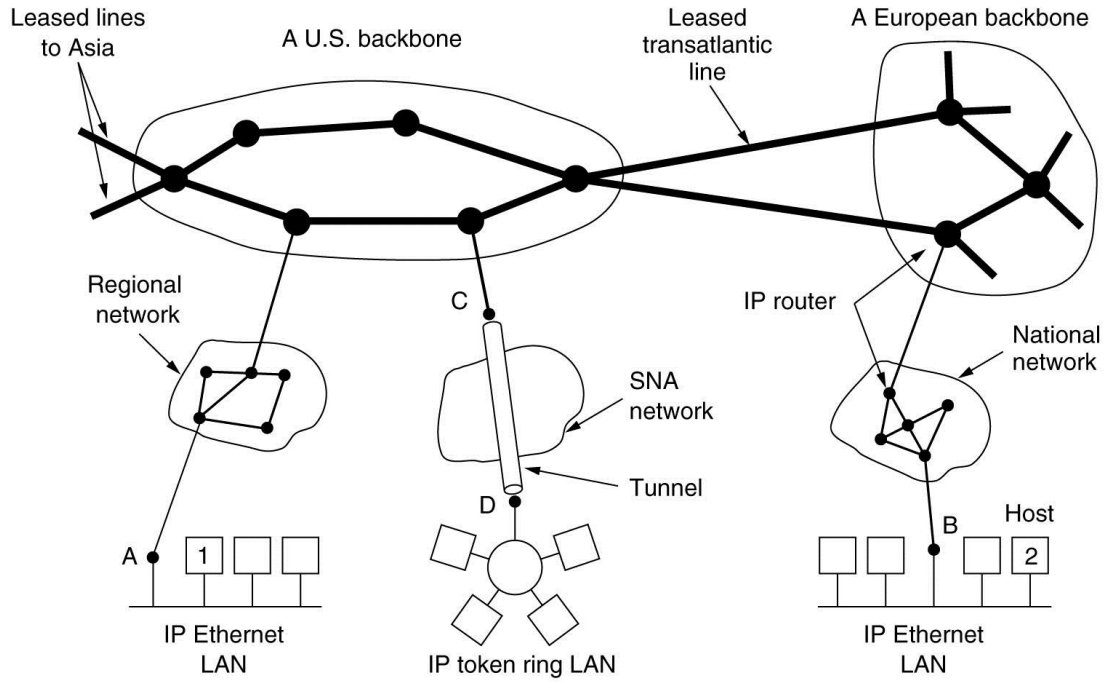
Device Terminology

- **Repeaters** (a.k.a. **Hubs**) live at the **Physical Layer** for *boosting signals*
  - *they just repeat everything they hear to everyone*
- **Switches** and **Bridges** live at the **Data Link Layer** to make *interconnections*
  - decisions are based on MAC addresses
- Multi-protocol **Routers** (a.k.a. **Gateways**) live at the **Network Layer** for *forwarding* (and possibly splitting up) packets
  - Decisions are based on IP addresses
  - Transport Layer Gateways and Application Layer Gateways also exist
- In order to connect IP-based networks to each other, you will need a Router acting as the Gateway in-between them
- *Do you know of any popular, perhaps even global, IP-based networks?*

The Internet

- The Internet:
  - a collection of **autonomous systems** (ASes) connected together by **backbones**

- Designed according to principles in RFC 1958:
  - keep it simple
  - exploit modularity
  - look for good design (*but not perfect*)
  - think about scalability
  - (*see Section 3 of the RFC for more info*)



Internet Model

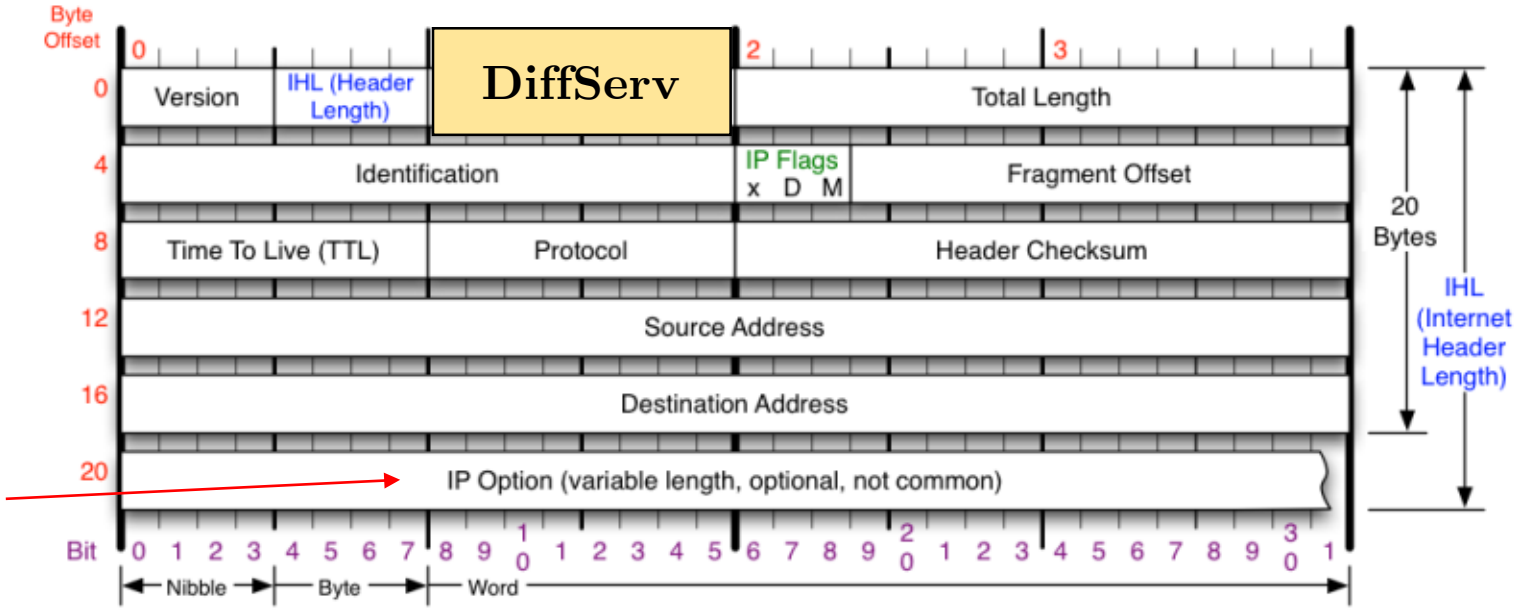
- An application offers a data stream to the Transport Layer, using either connection-oriented or connection-less services
- The Transport Layer data stream is converted into an IP Datagram for the Network Layer
- IP Datagrams are routed through the Internet
  - Routers pass datagrams to the underlying Data Link layer
    - (*e.g. to LANs and WANs*)
  - Then the cables take over..! (Physical Layer)

Internet Network Layer

- Internet Protocol (**IP**)
  - datagram format
  - fragmentation
  - addressing
  - packet handling
- Internet Control Message Protocol (**ICMP**)
  - error reporting
  - signaling
- Dynamic Host Configuration Protocol (**DHCP**)
  - address configuration
- **Routing**: defines paths and creates forwarding tables
  - RIP, OSPF, BGP

Layer 3 (IP) Header

Note:
Many of the available **IP Options** are not used because of security reasons



Version

Version of IP Protocol. 4 and 6 are valid. This diagram represents version 4 structure only.

Header Length

Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

Protocol

IP Protocol ID. Including (but not limited to):

1 ICMP	17 UDP	57 SKIP
2 IGMP	47 GRE	88 EIGRP
6 TCP	50 ESP	89 OSPF
9 IGRP	51 AH	115 L2TP

Total Length

Total length of IP datagram, or IP fragment if fragmented. Measured in Bytes.

Fragment Offset

Fragment offset from start of IP datagram. Measured in 8 byte (2 words, 64 bits) increments. If IP datagram is fragmented, fragment size (Total Length) must be a multiple of 8 bytes.

Header Checksum

Checksum of entire IP header

IP Flags

x D M

x 0x80 reserved (evil bit)
D 0x40 Do Not Fragment
M 0x20 More Fragments follow

RFC 791

Please refer to RFC 791 for the complete Internet Protocol (IP) Specification.

IPv4 Header

Fragmentation

- Routers must handle cases where size of input datagram exceeds the Maximum Transmission Unit (**MTU**) of the output link
  - In those cases, the IP Datagram needs to be **Fragmented**

- The destination reassembles fragmented datagrams
  - *not the intermediate routers*
  - mainly to push complexity *out* of network
  - datagram may have to be fragmented further along the path

- Fragmentation scheme must ensure that destination host:
  - can recognise two fragments of same original datagram
  - can figure out if and when all fragments were received

- Fragmentation scheme must ensure that intermediate routers can fragment datagram to whatever level necessary

Fragmentation (cont'd)

- Let's assume an initial (non-fragmented) datagram format of size X
  - the sender host assigns a 16 bit identifier to the datagram (Identification)
  - the fragment offset is set to 0, indicating that this packet contains data starting at position 0 *of the original datagram*
    - the fragment offset is actually the offset in units of 8 Bytes
    - (*all fragments should be multiples of 8B, except the last one*)
    - The Fragment Offset field is only 13 bits,
      i.e. 2^13=8192, i.e. between 0 and 8191 units of 8B (*possible maximum*)
    - However, it would be impossible to have 8191 units of 8B in one fragment
      - because the Total Length is 16 bits = 65536, i.e. from 0 to 65535B max
      - IP Header = 20B
      - 65535B - 20B = 65515B
      - 65515B / 8B = 8189 max possible units of 8B (*plus change*)

- If the "more fragments" flag is set to 1, more fragments are expected to follow

Fragmentation (cont'd)

- Example:
  - Let's assume an MTU of 512B
    - The packet is 1020B in total
      - Fragmentation is required

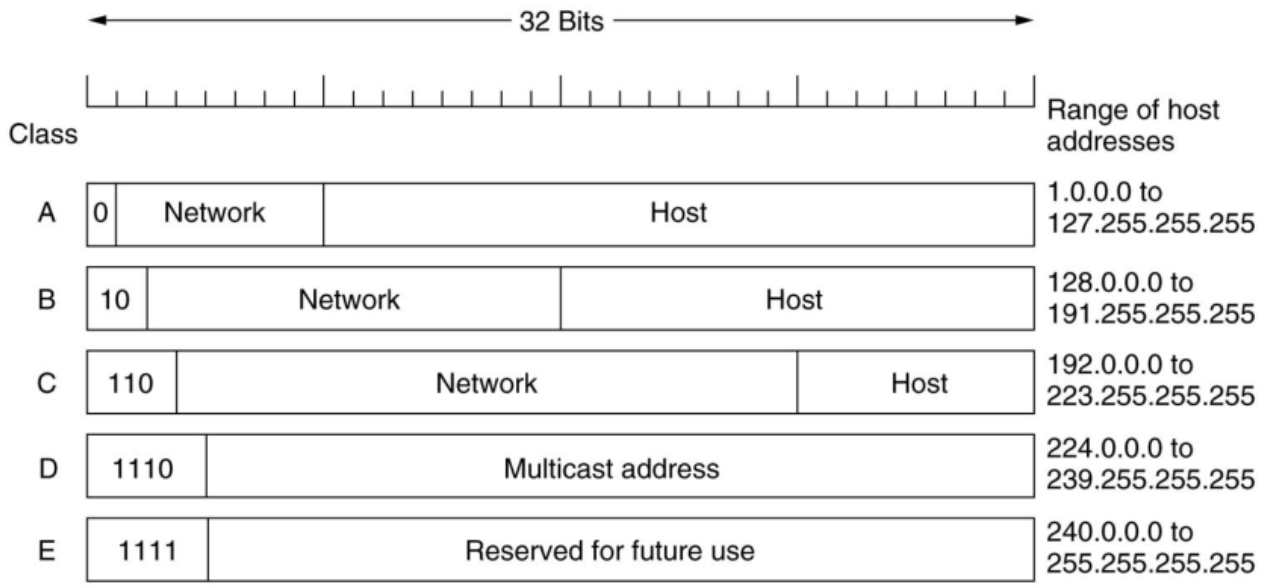
Identifier	Fragment Offset	More Fragments	Header Length	Total Length
789	0	0	20	1020
789	0	1	20	508
789	61	1	20	508
789	122	0	20	44

IPv4 Addressing

- 32-bit addresses
  - 4 billion possible IP addresses
- XXX.XXX.XXX.XXX, where XXX from 0 to 255 => 256 values (i.e. 1B * 4)
- Each IP address associated with **interface**, not host
  - host with more than one interfaces may have more than one IP addresses
- Assignment of addresses over Internet topology crucial to limit complexity of routing
- Key idea:
  - assign addresses with same **prefix** to interfaces that belong to same organisation
    - e.g. the network IP address *groups* all addresses on that network

IP Addresses

- For several decades, IP addresses were divided into 5 categories called **Classes**
- Allocation was called Classful Addressing (*no longer used*)



- Each class allows a maximum number of hosts

Class	Max. networks	Max. hosts per network
A	126	16,777,214
B	16,382	65,536
C	2,097,150	254

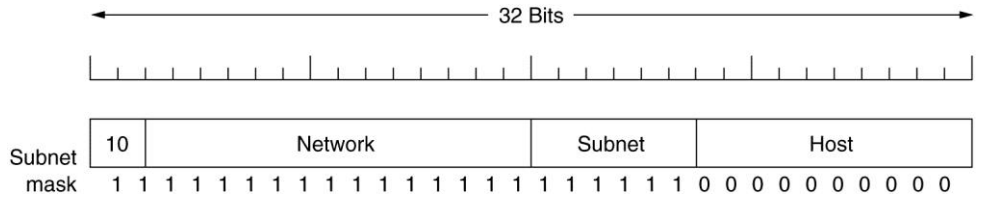
IP Addresses (cont'd)

- Network IP addresses managed by nonprofit Internet Corporation for Assigned Names and Numbers (**ICANN**) to avoid conflicts
- ICANN delegates parts of address space to various regional authorities, which give out IP addresses to ISPs and companies
- Network addresses, which are 32-bit numbers, usually written in **dotted decimal notation**
  - each of the 4 bytes is written in decimal, from 0 to 255
  - e.g. a 32-bit hexadecimal address C0290614 written as 192.41.6.20

Subnet Masking

- **Problem**: Hosts on the same network must share network address
  - may cause single organisation to acquire several classes of addresses, which would have to be announced worldwide

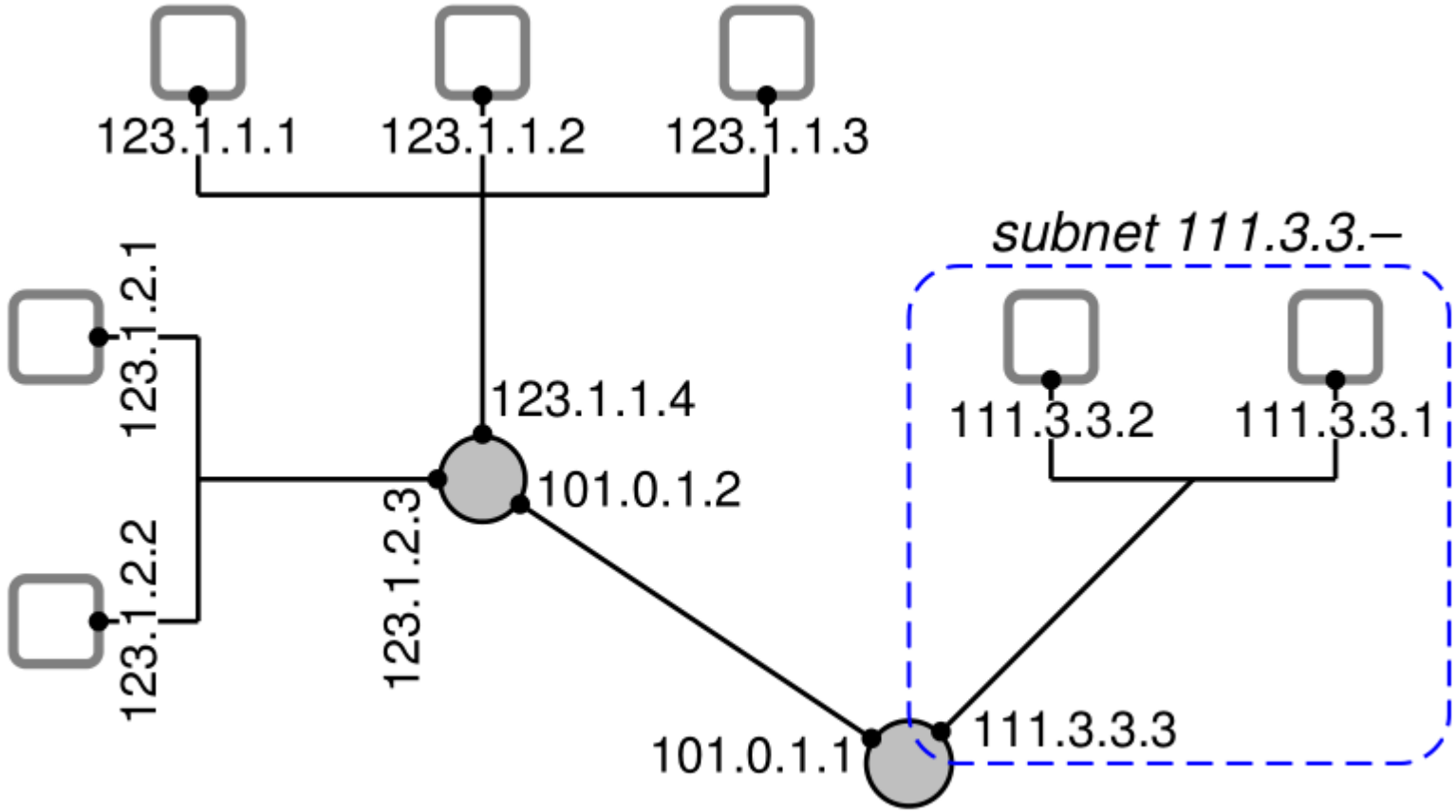
- **Solution**: Use a single network address for the entire organisation, and divide internally into subnet addresses and host ids:



- Introduces **3-level** routing hierarchy:

1. External routers only consider network address and forward the packet to one of the routers of the organisation
2. Subnet routers apply the subnet mask, and look up if the destination is on their subnet, or if they should forward it to another subnet router
3. After host was found, router knows which interface to use to forward packet

Interconnection of Networks



Subnet Addressing

- All interfaces in same subnet share same address prefix
  - e.g. previously, we had:
  - 123.1.1.—, 123.1.1.—, 101.0.1.—, and 111.3.3.—

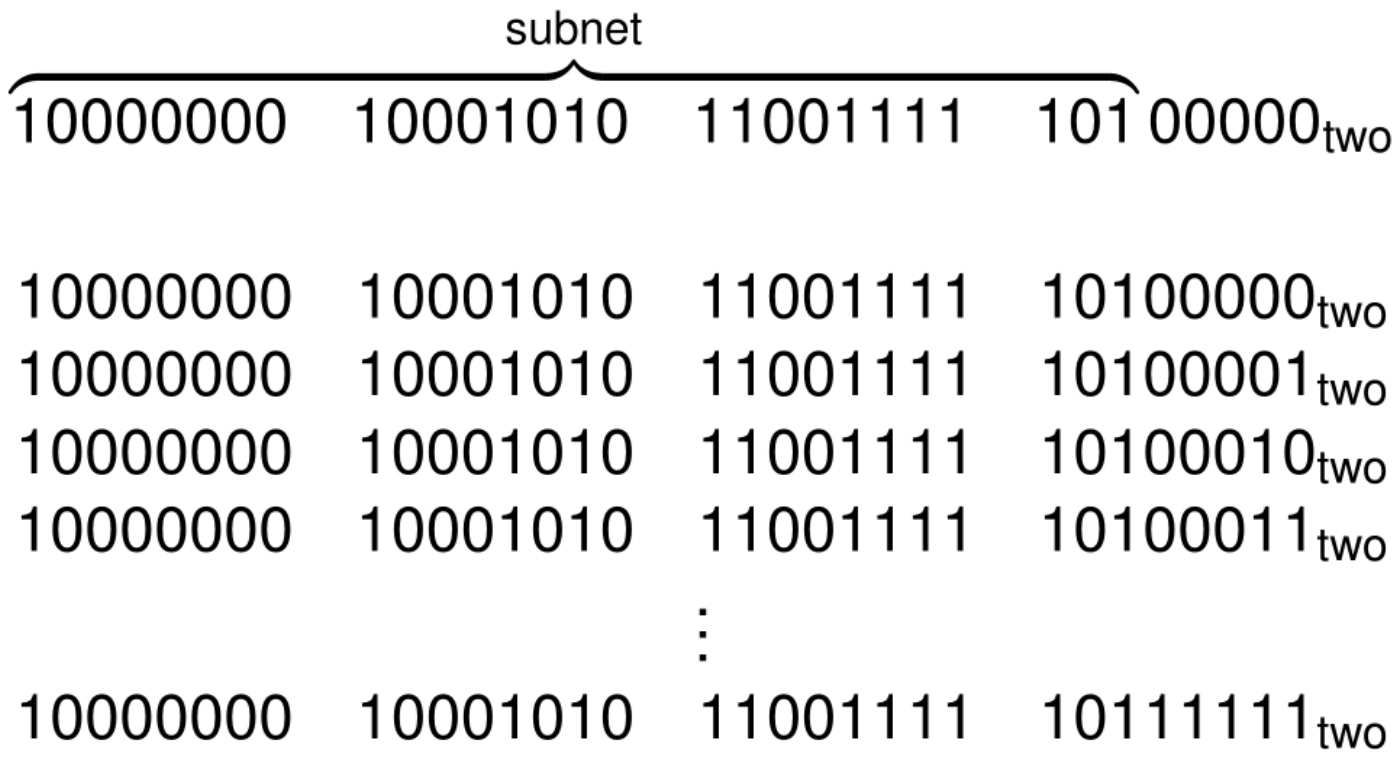
- Network addresses prefix-length notation: address/prefix-length
  - This notation is called Classless Inter-Domain Routing (**CIDR**)
  - e.g. 123.1.1.0/24, 123.1.1.0/24, 101.0.1.0/24 and 111.3.3.0/24
  - 123.1.1.0/24 means that all addresses share same leftmost 24 bits with address 123.1.1.0

- Addressing scheme not limited to entire bytes:
  - e.g. network address might be 128.138.207.160/27

- *More information on this from Cisco*

Subnet Ranges

- What is the range of addresses in 128.138.207.160/27?



Net Mask

- Network addresses, mask notation: **address/mask**

- Prefix of length p corresponds to mask

$$M = \overbrace{11\cdots1}^{p \text{ times}}\ \overbrace{00\cdots0}^{32-p \text{ times}}{}_{\text{two}}$$

- e.g. 128.138.207.160/27=128.138.207.160/255.255.255.224
- 127.0.0.1/8=127.0.0.1/255.0.0.0
- 192.168.0.3/24=192.168.0.3/255.255.255.0
- 195.176.181.11/32=195.176.181.11/255.255.255.255

- In Java:

```
boolean match(int address, int network, int mask) {
    return (address & mask) == (network & mask);
}
```

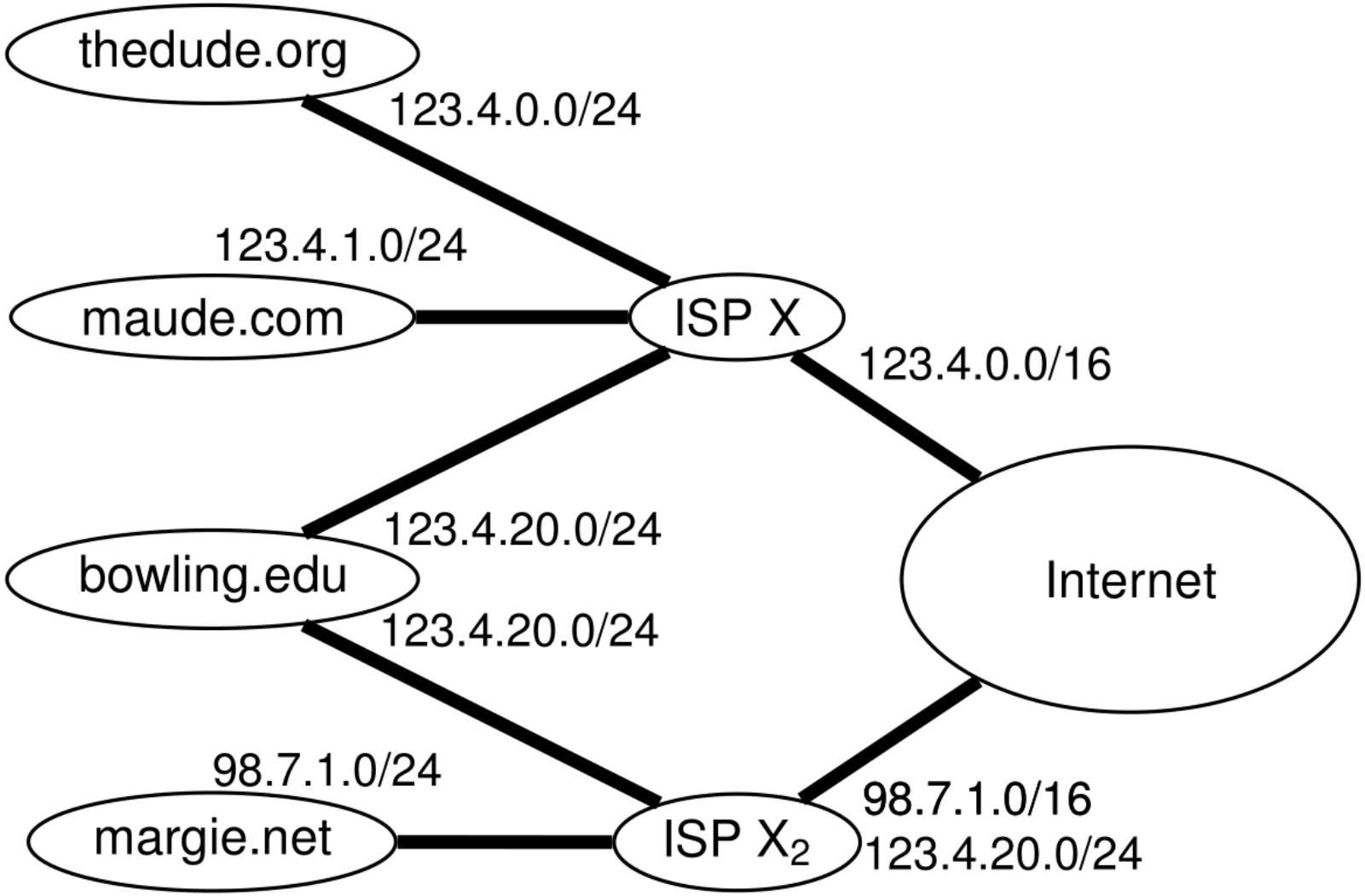
Classless InterDomain Routing

- This any-length prefix scheme also called Classless InterDomain Routing (**CIDR**)
  - as opposed to original scheme which divided address space in specific "**classes**"

- Use of classes makes addresses scarce resource
  - *A → 8, B → 16, C → 24*
  - *too many class B addresses*

- Why is the idea of the common prefix important?
  - Routers outside (sub)network can ignore specifics of each address within network
    - there may be 64k hosts in 128.138.0.0/16, but they all appear as one network address from outside

Allocation of Address Blocks



Longest-Prefix Matching

- When forwarding datagrams, routers choose entry that matches destination address with the **longest prefix**

- Where do they go?
  - 123.4.1.69 →
  - 98.7.2.71 →
  - 200.100.2.1 →
  - 123.4.20.11 →
  - 123.4.21.10 →
  - 128.138.207.167 →
  - 68.142.226.44 →

forwarding table	
network	port
123.4.0.0/16	1
98.7.1.0/16	2
123.4.20.0/24	2
128.0.0.0/1	3
66.249.0.0/16	3
0.0.0.0/0	4
128.138.0.0/16	4

Longest-Prefix Matching

- Answers:
  - 123.4.1.69 → 1
  - 98.7.2.71 → 2
  - 200.100.2.1 → 3
  - 123.4.20.11 → 2
  - 123.4.21.10 → 1
  - 128.138.207.167 → 4
  - 68.142.226.44 → 4

- 0.0.0.0/0 is called the "**default route**" – an IP that is used for packet forwarding when no other applicable IP matches (*i.e. in any other case*)

Internet Control Message Protocols (ICMP)

- We need to be able to inform hosts and routers when things go wrong, or to send queries about status information

- **ICMP** encapsulates *control messages* in IP packets
  - Several message types exist (*destination unreachable, time exceeded, and more*)
  - Each message type is further specified by a code
    - e.g. destination can be unreachable (message type = 3) due to unsupported protocol (code = 2)

- Example:
  - **ping**
    - querying node sends ICMP type 8 code 0 message to host, which responds with type 0 code 0 ICMP reply

- All ICMP types & codes (as well as links to their RFCs) can be found on IANA

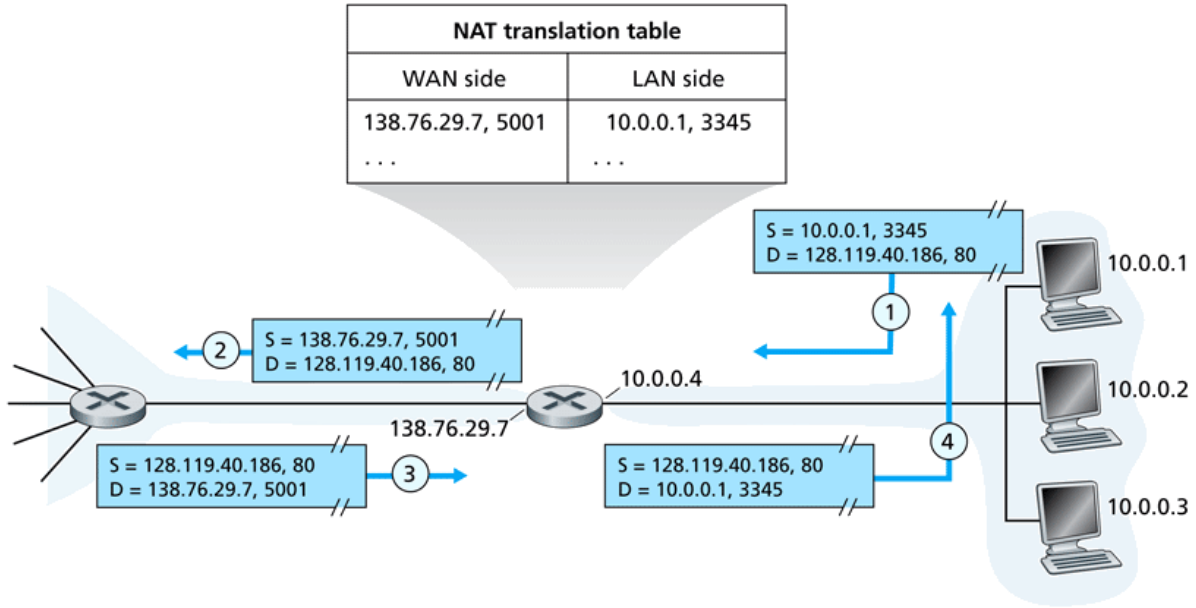
Network Address Translation (NAT)

- NAT partially solves IP address shortage by assigning each company a single IP
  - also used to share home ADSL connection among multiple devices
  - *within* a company, every computer gets a unique (local) IP address, used for routing *internal* traffic
  - when packet exits the local network, address translation takes place

- Most IP packets carry either TCP or UDP payloads
  - both have headers with source and destination port
  - ports identify different process running on host
  - e.g. web server running on port 80 and SSH on port 22

- The following address ranges are declared as "private" to be used in local networks:
- 10.0.0.0 – 10.255.255.255/8 (16,777,216 hosts)
- 172.16.0.0 – 172.31.255.255/12 (1,048,576 hosts)
- 192.168.0.0 – 192.168.255.255/16 (65,536 hosts)

Network Address Translation Example



- When a connection is set up from 10.0.0.1, port X, the router uses its public source address 138.76.29.7 on port Y, and registers mapping X ↔ Y
- A reply on port Y is sent to 10.0.0.1 on port X

- **Question:**
- How can you allow incoming connections?
- Create static mapping, e.g. redirect all traffic on port 80 to 10.0.0.3 on port 8080

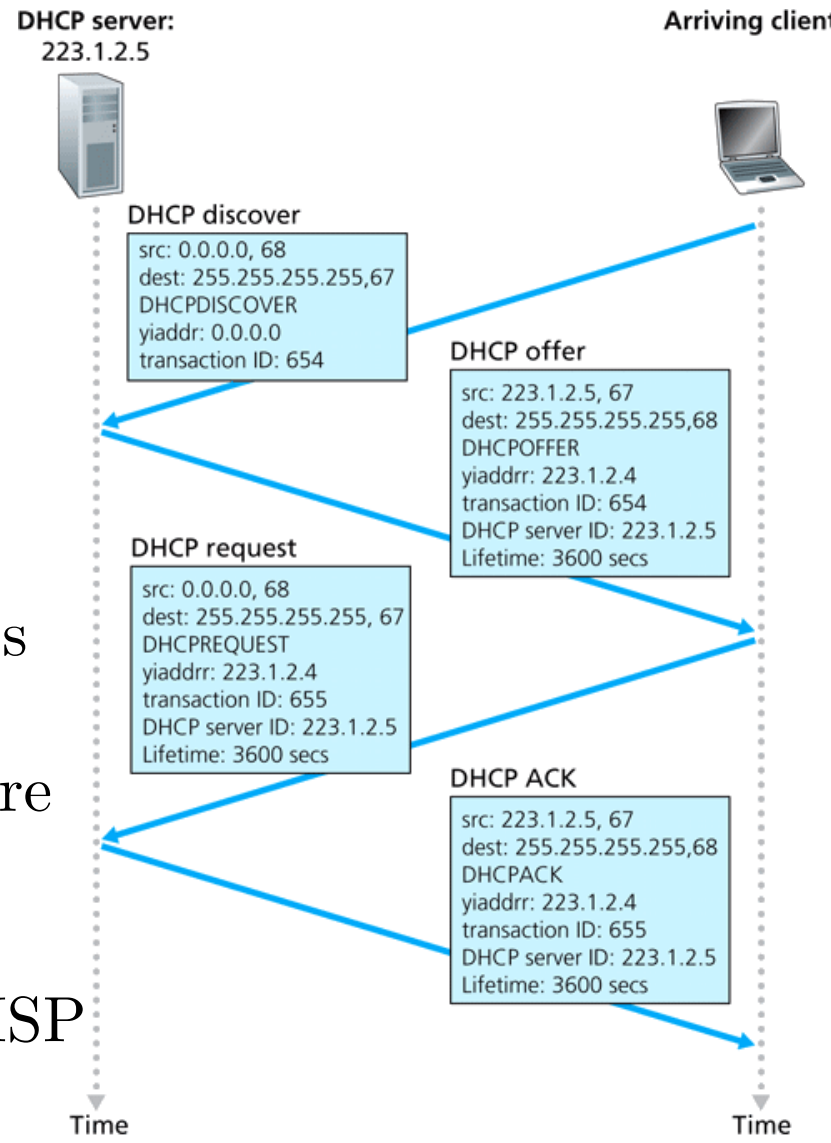
Network Address Translation Criticism

- Despite its widespread use, this has problems (see RFC 2993):
  - (a) NAT violates architectural IP model, which states that every IP address **uniquely** identifies single machine worldwide
  - (b) NAT changes Internet from **connectionless** to **connection-oriented** network
    - *NAT gateway must maintain mapping for each connection*
  - (c) NAT violates fundamental rule of protocol layering: layer k must not make assumptions about layer k + 1
    - *For example, if TCP is later upgraded to TCPv2, NAT will fail*
  - (d) NAT cannot easily support new transport protocols
  - (e) many P2P protocols (e.g. BitTorrent) require *full connectivity* among hosts
    - users behind NAT cannot be contacted directly by other peers
    - techniques exist to circumvent these issue to some extent (*e.g. port forwarding and TURN relays or NAT punching holes, using third-party servers*)

- IPv6 is the best solution to deal with the lack of IP addresses (*How?*)

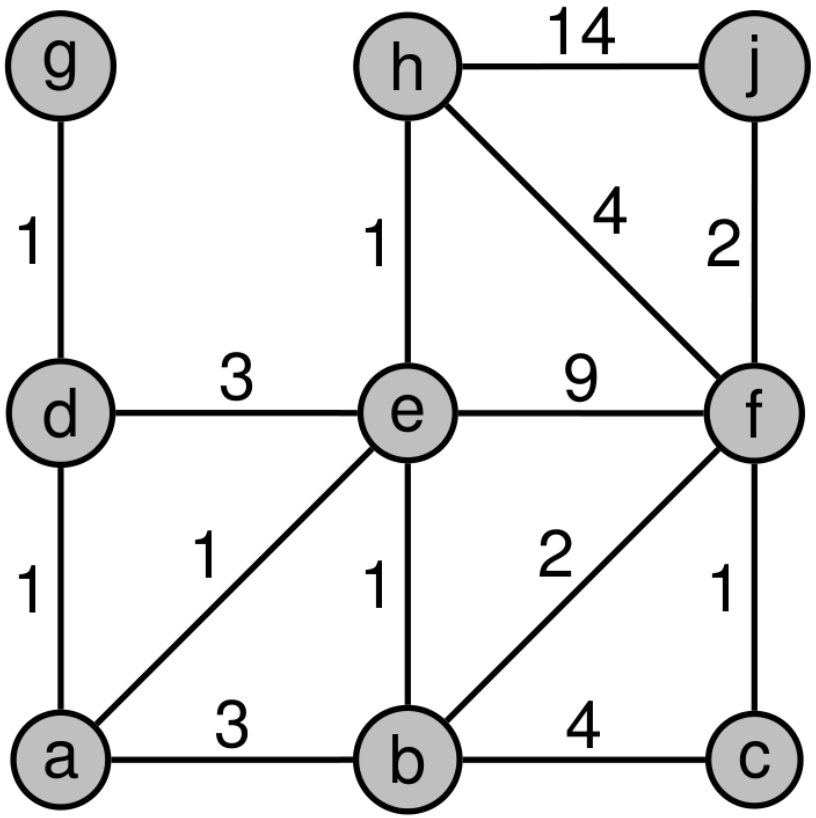
Dynamic Host Configuration Protocol

- **Problem**: How does a host decide on its own IP address?
- **Solution**: Dynamic Host Configuration Protocol (DHCP) (RFC 2131 and 2132)
  - Each newly-booted machine **broadcasts** an "**DHCP DISCOVER**" packet when it connects
  - DHCP server replies with the assigned IP address
  - DHCP server can either maintain **static** mappings or assign **different** addresses each time a host connects
- To prevent hosts from keeping addresses forever, we require periodic refresh (**leasing**)
- This is how you (= *your router*) receive an IP from your ISP



Network Model

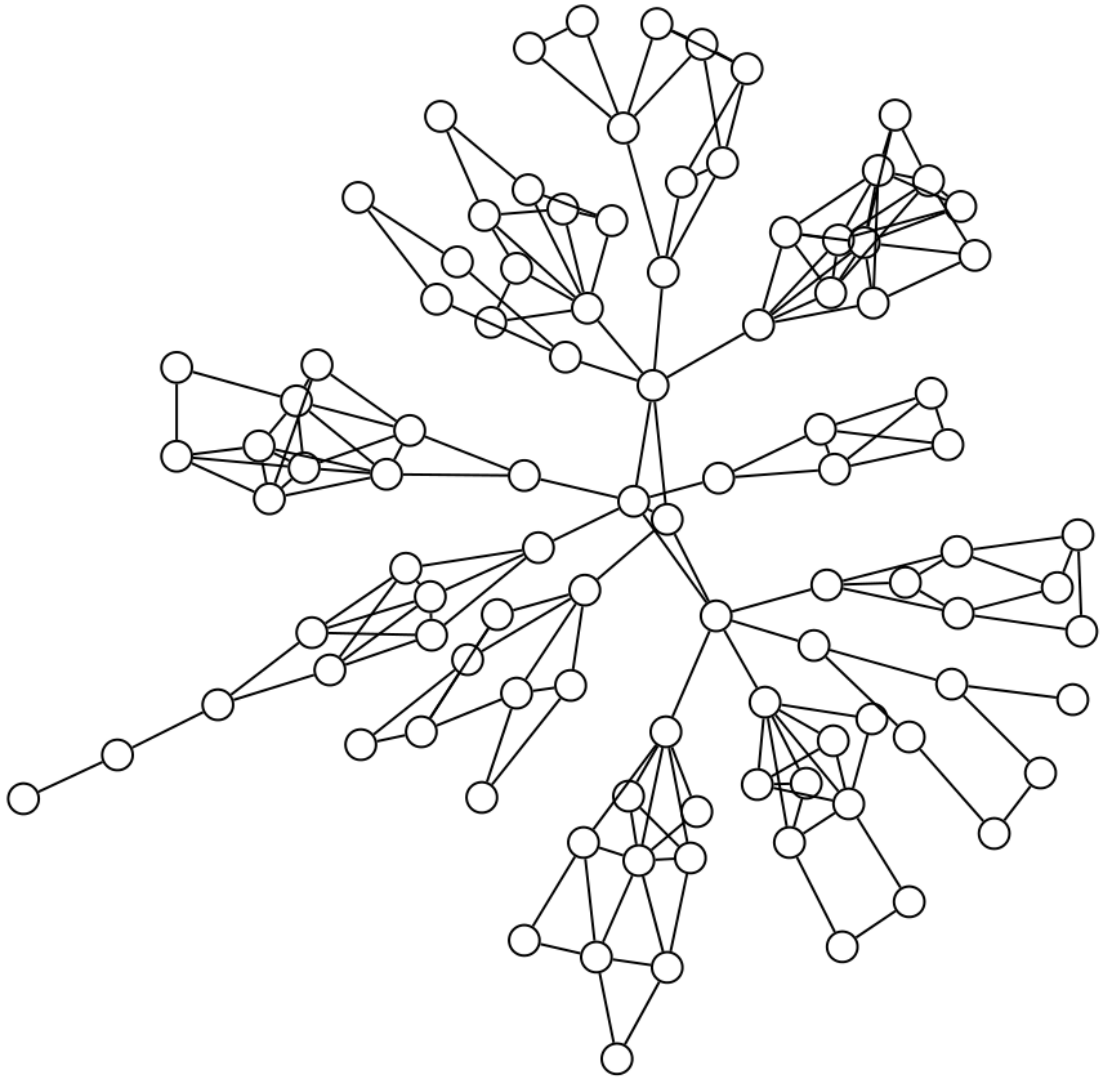
- So far we have studied routing over a "flat" network model



- The objective was to find least-cost paths between sources and destinations

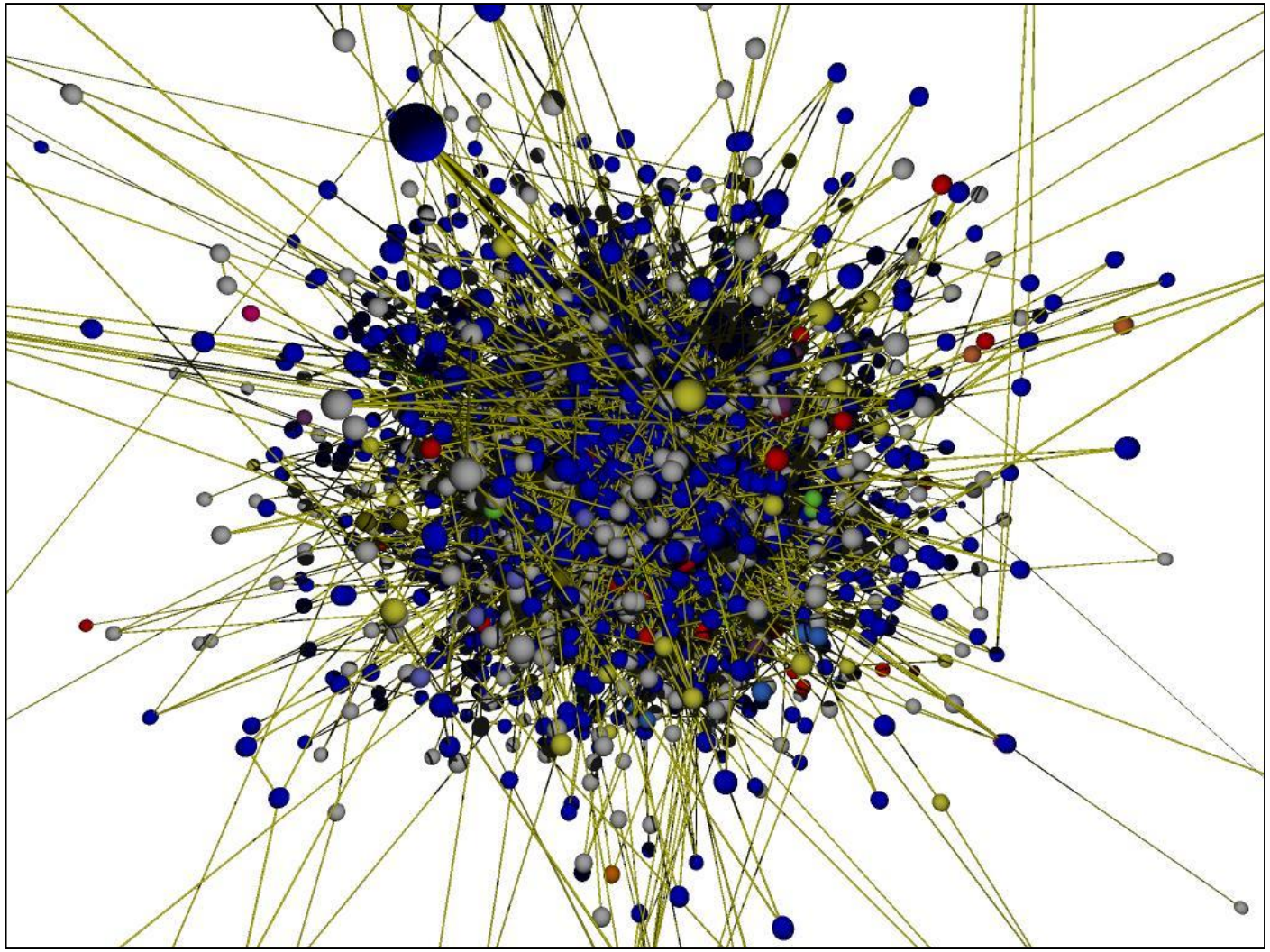
Network Model (cont'd)

- More Realistic Topologies



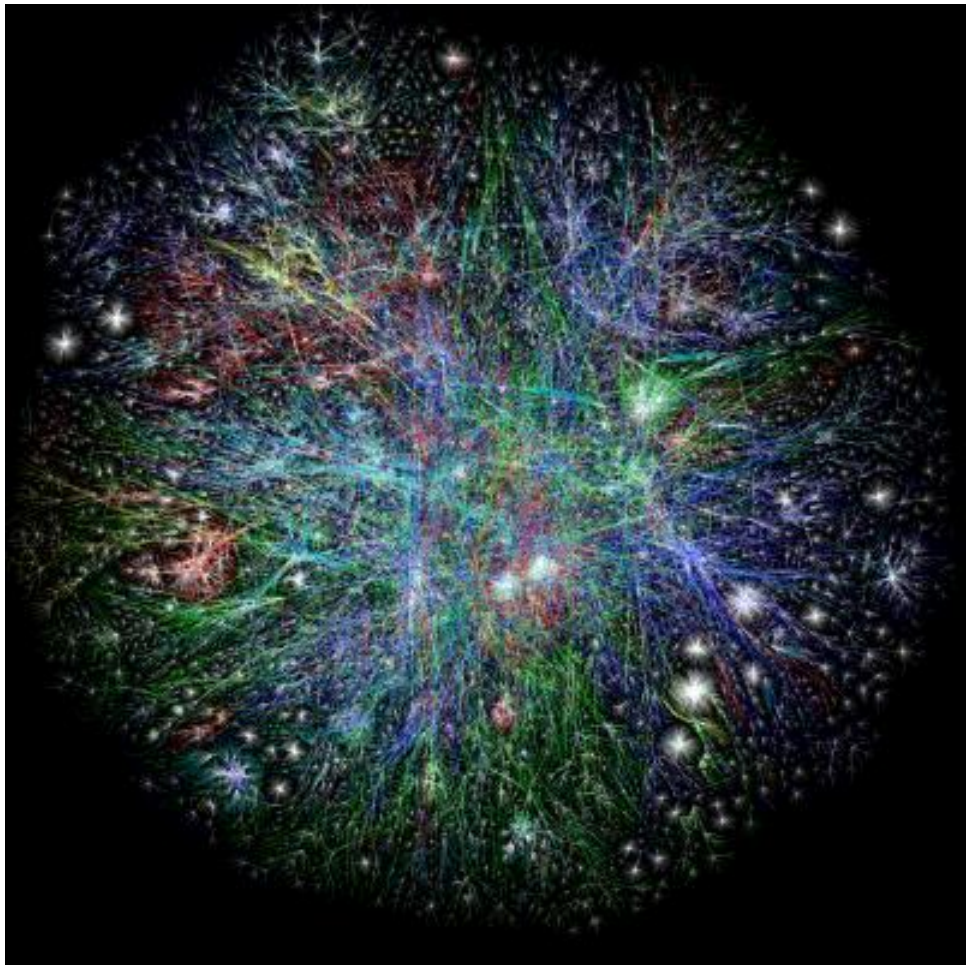
Network Model (cont'd)

- Even More Realistic



Network Model (cont'd)

- An Internet Map

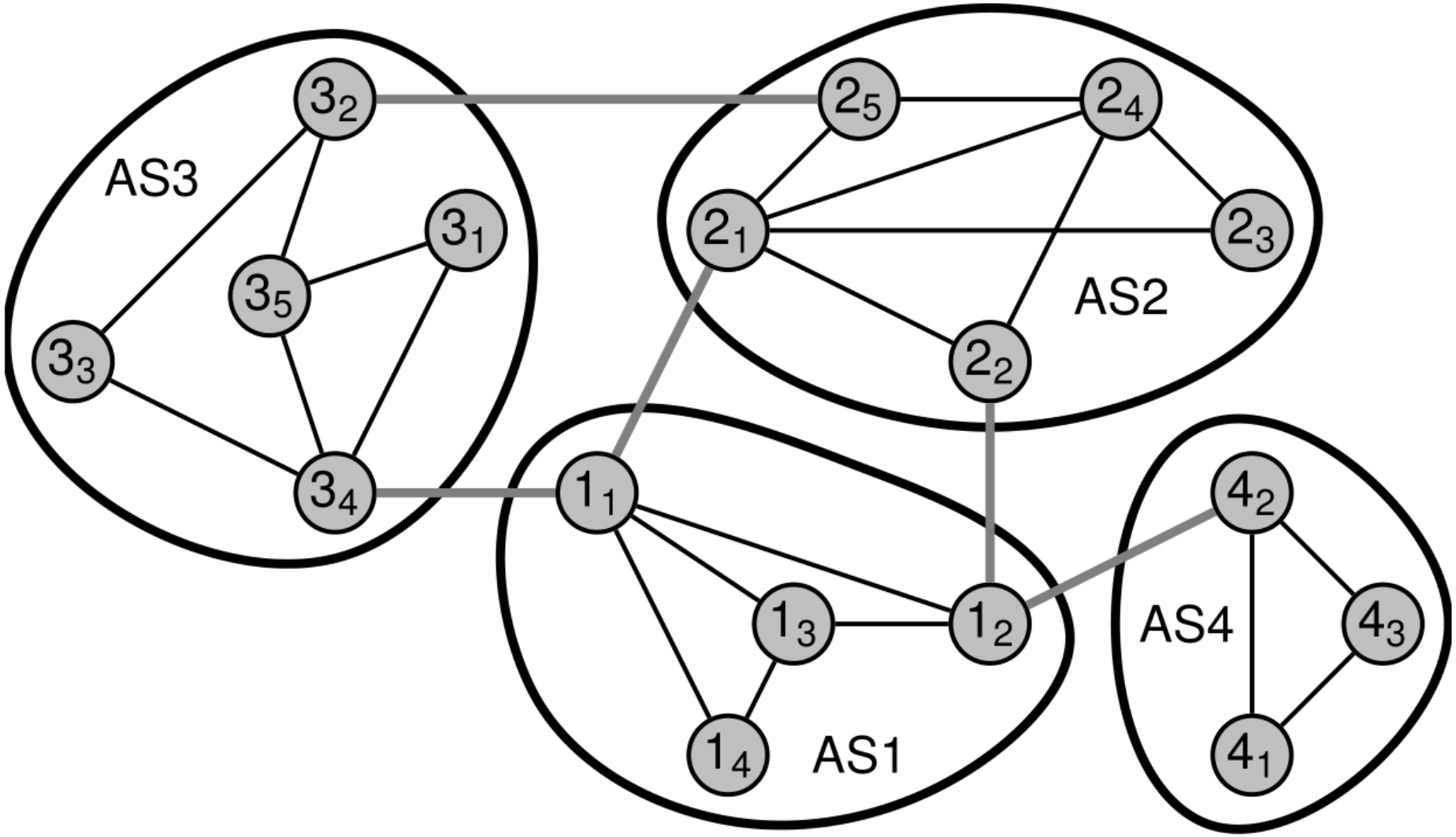


Hierarchical Routing

- Flat network model too simplistic for multiple reasons
  - **Scalability**
    - hundreds of millions of hosts in today's Internet
    - transmitting routing information too expensive
    - forwarding also too expensive
  - **Administrative autonomy**
    - one organisation may want to run distance-vector routing protocol, while another may want to run link-state protocol
    - organisations do not want to expose internal network structure

- Today's Internet organised in **Autonomous Systems** (**ASs**)
  - independent administrative domains
  - **Gateway routers** interconnect autonomous systems

Hierarchical Structure



Internet Routing

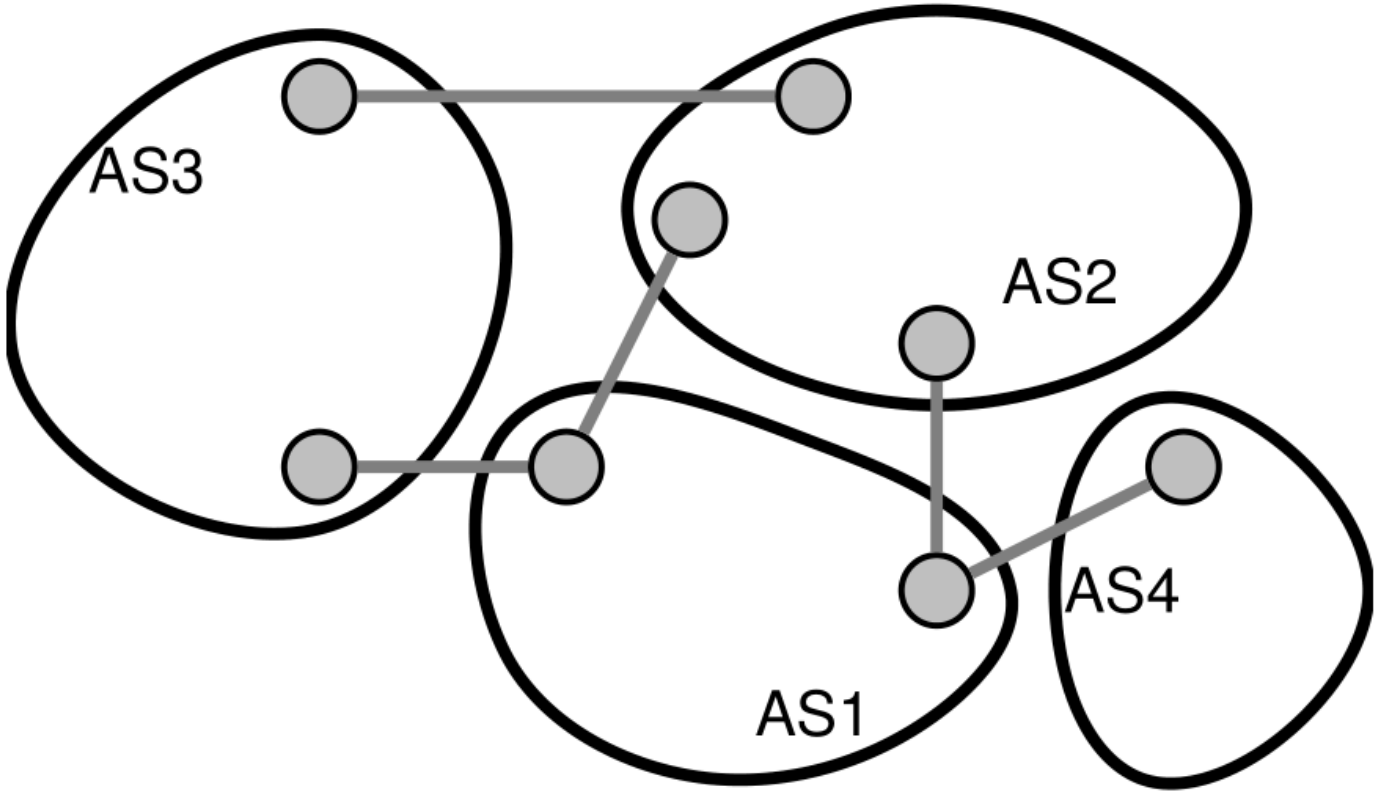
- Distinction between routing **within** an autonomous system and **between** autonomous systems:

- **Intra-AS routing protocol** runs within autonomous system
  - determines internal routes
    - internal router ↔ internal router
    - internal router ↔ gateway router
    - gateway router ↔ gateway router (*same AS*)
  - should provide best possible routes (**optimal routing**)
  - **RIP** (distance vector) mainly used in old ISPs and small organisations, and **OSPF** (link state) used by larger organisations

- **Inter-AS routing protocol** determines routing at autonomous-system level
  - Inter-AS routing has to deal with politics
  - *e.g. some ASes should not be traversed or are more expensive*
  - **BGP** is the Internet standard

Inter-AS Routing

- Inter-AS routing protocol determines routing at autonomous-system level



- At AS3:
  - AS1 → AS1
  - AS2 → AS2
  - AS4 → AS1

Hierarchical Routing

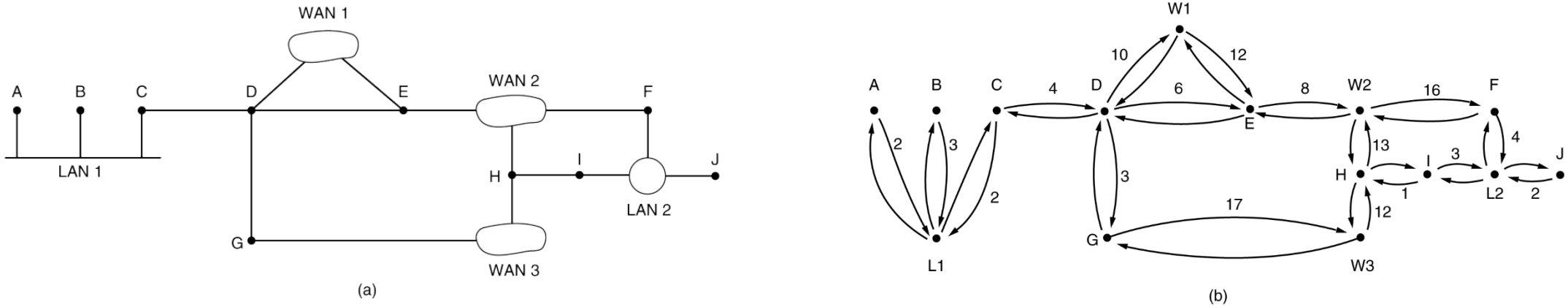
- Routers within AS exchange their **intra-AS** routing information
- Gateway routers discover **inter-AS** routing information
  - **Inter-AS** routing information propagated within AS
- Both **inter-AS** and **intra-AS** routing information used to populate forwarding tables
- Destinations within same AS reached as usual
- What about destination x outside AS?
  - **Inter-AS** information used to decide if x reachable through gateway G x
  - **Intra-AS** information used to decide how to reach Gx within AS
  - What if x reachable through multiple gateway routers Gx ,G'x ,...?
    - Use **intra-AS** information to determine costs of paths to Gx ,G'x ,...
    - "**Hot-potato**" routing: send it through closest gateway

Interior Gateway Routing: OSPF

- **Open Shortest Path First (OSPF)**
  - link state routing protocol, replacement for RIP

- Requirements:
  - Openness: algorithm publicly available so anyone can implement
  - Support for different distance metrics (hops, delays, etc.)
  - Dynamic and efficient adaptability to changing topologies
  - Support routing based on Type of Service (*e.g. real-time multimedia traffic*)
  - Support for load balancing
    - *when a route is heavily used, another one should be selected*
  - Support for hierarchical routing
  - Offers security (*even though not always perfect*)

Interior Gateway Routing: OSPF (cont'd)

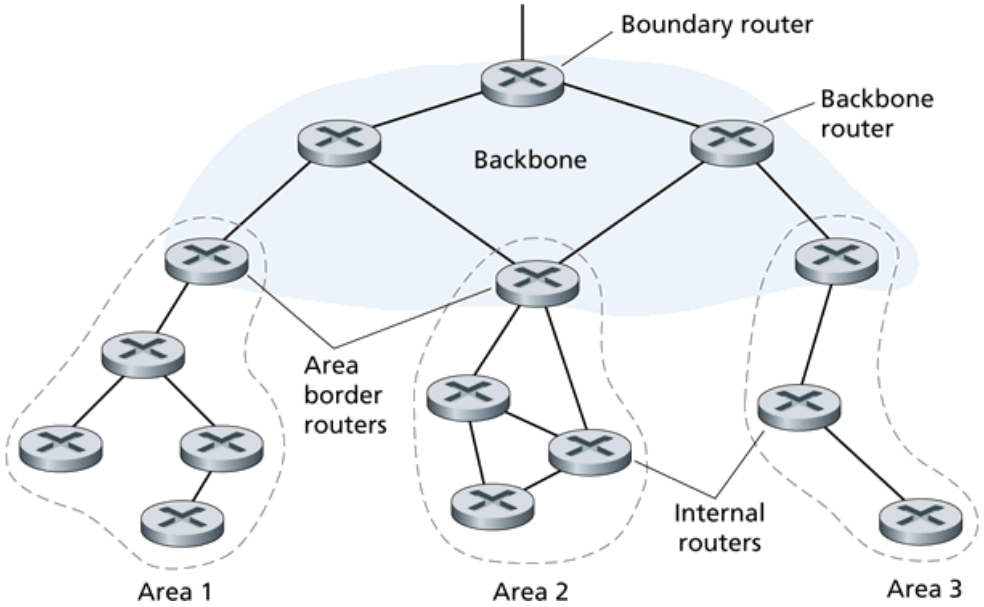


- Abstracts collection of networks, routers and links into directed graph; each edge with its cost (distance, delay, etc.)
  - Serial link between two routers represented by pair of arcs; one in each direction
  - Multi-access network represented by node for network itself, plus node for each router

- Computes shortest path from every router to every other router

Interior Gateway Routing: OSPF (cont'd)

- ASs on the Internet may themselves be large and non-trivial to manage
  - OSPF allows them to be divided into areas



- Use the same algorithm for the areas and the backbone
  - within each area, one or more **area border routers** responsible for routing packets outside area
  - **backbone area** routes traffic between other areas in AS
  - must contain all border routers

Inter-AS Routing: BGP

- **Border Gateway Protocol** (BGP) is inter-AS routing protocol in today's Internet

- Neighbouring routers maintain **connection** to simplify reliability
  - provides reachability information from neighbour ASs
  - transmits reachability information to all internal routers within AS

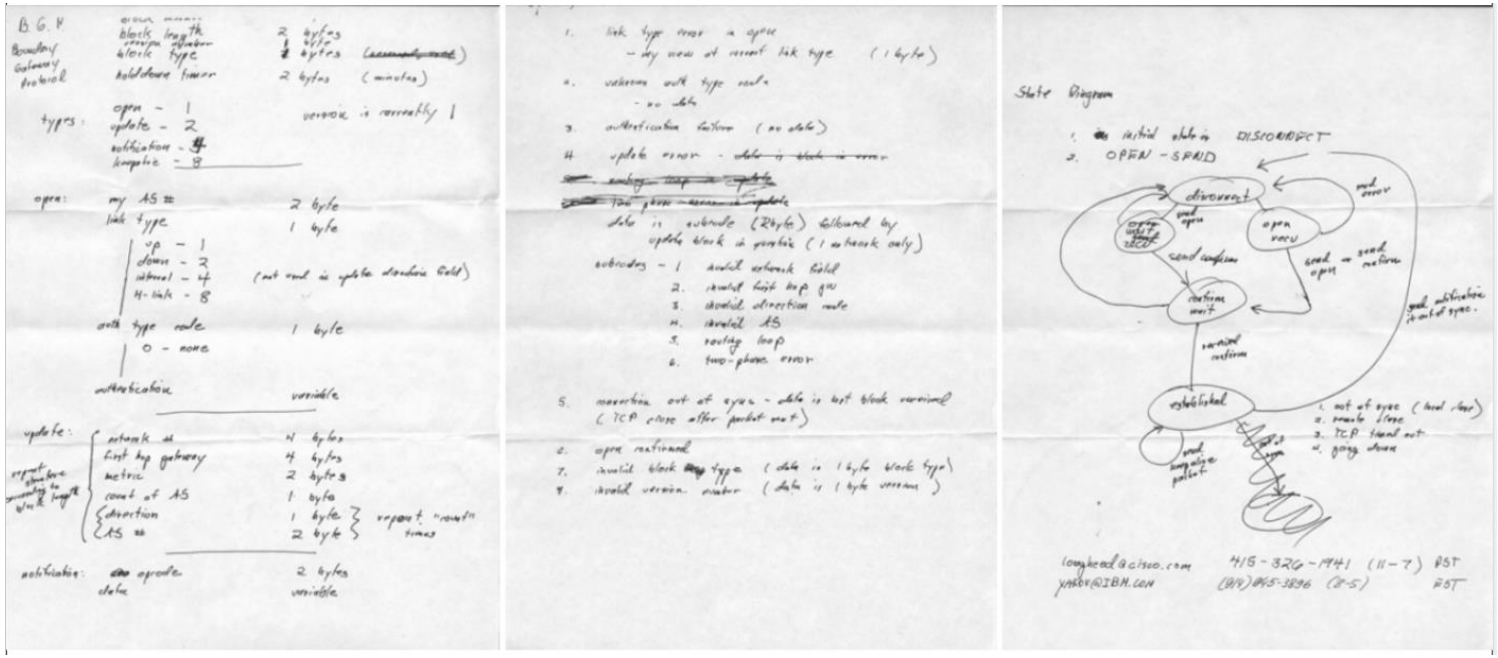
- Determines good routes to all outside subnets
  - based on reachability information
  - based on **policies**
  - routers do not automatically use discovered routes, but check if **allowed**

- BGP is **path-vector** protocol
  - based on distance vector routing, but paths (not distances) announced
  - does not suffer from count–to–infinity problem ⇒ router decides on entire path

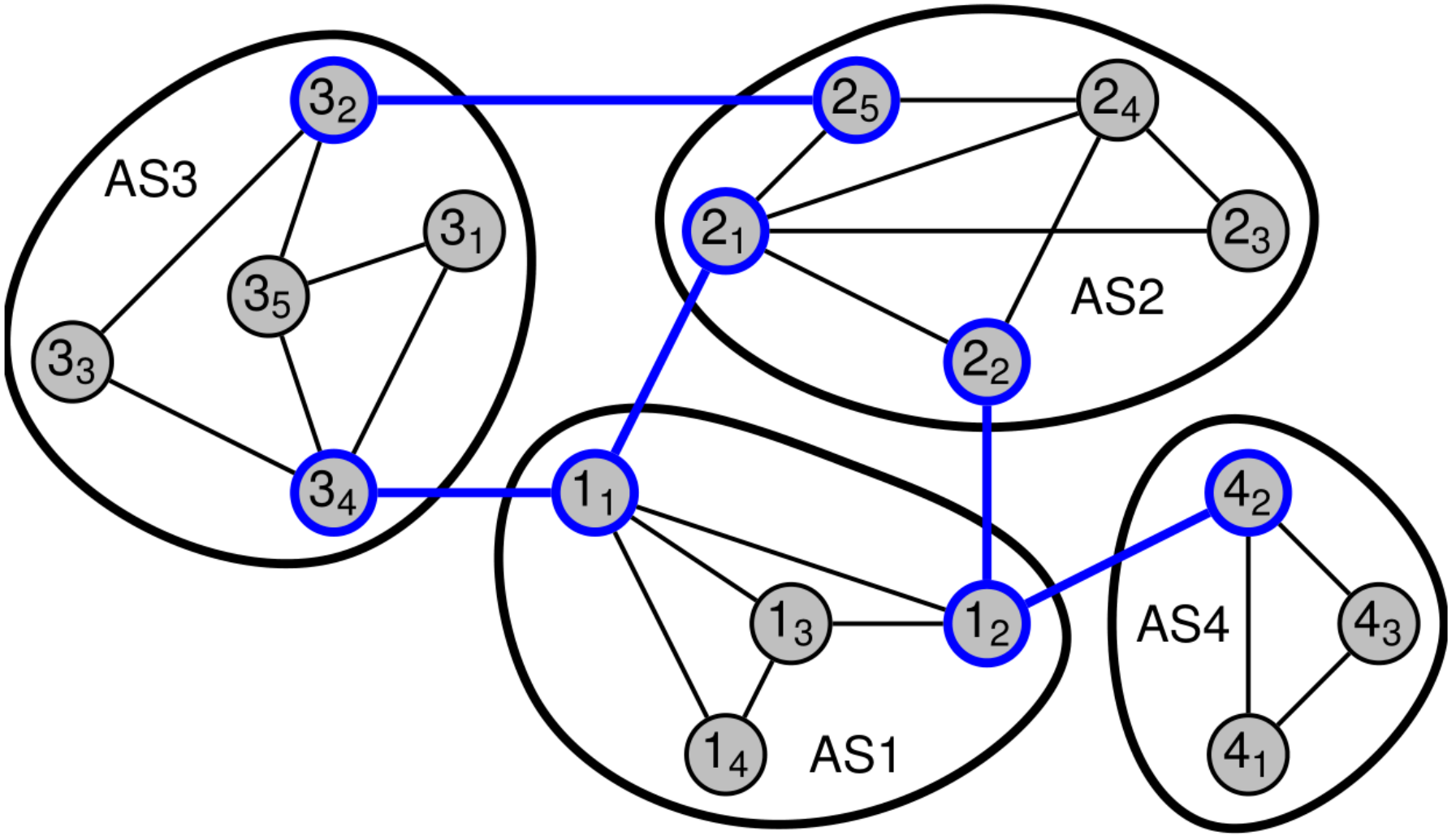
BGP: the “3 napkins" protocol

Imperial College London

- BGP was designed during a conference lunch-break on three napkins..!
- The figure below is a copy which was made from the originals:



Inter-AS Routing: BGP Example



Inter-AS Routing: BGP

- BGP advertisement: router advertises routes to networks (*much like in distance-vector*)
  - destinations denoted by address prefixes
  - AS may or may not forward advertisement for foreign network; doing so means willingness to carry traffic for network
  - routers may *aggregate prefixes* (*a.k.a.* ***supernetting***)

- For example:

$$\left.\begin{array}{l}128.138.242.0/24\\128.138.243.0/24\end{array}\right\} \rightarrow 128.138.242.0/23$$

$$\left.\begin{array}{l}191.224.128.0/22\\191.224.136.0/21\\191.224.132.0/22\end{array}\right\} \rightarrow 191.224.128.0/20$$

Message Structure

- Autonomous System Number (ASN)
  - unique identifier for each AS (*with more than one gateway*)

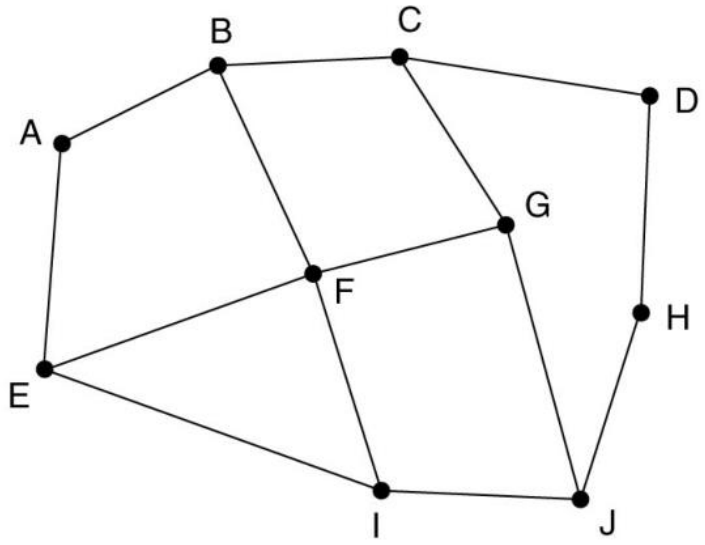
- BGP attributes: route advertisements include attributes
  - AS-PATH: sequence of ASNs through which advertisement was sent
  - NEXT-HOP: specifies interface (IP address) for forwarding packets towards advertised destination
    - resolves ambiguous cases where AS reachable through multiple interfaces

- BGP import policy: decides whether to accept or reject route advertisement
  - e.g. router may not want to send traffic through one AS listed in AS-PATH

Inter-AS Routing: BGP

- Count-to-infinity problem solved by inspecting path:
  - assume **G** crashes and then **F** receives routes from neighbours
  - **F** sees that two latter routes failed because they pass through **G**
  - therefore **F** chooses **FBCD** as new route

- Router preference: routes ranked according to:
  1. **preference** value
     - depends on policy (*may be* ∞)
  2. **shortest** AS-PATH
  3. **closest** NEXT-HOP router



(a)

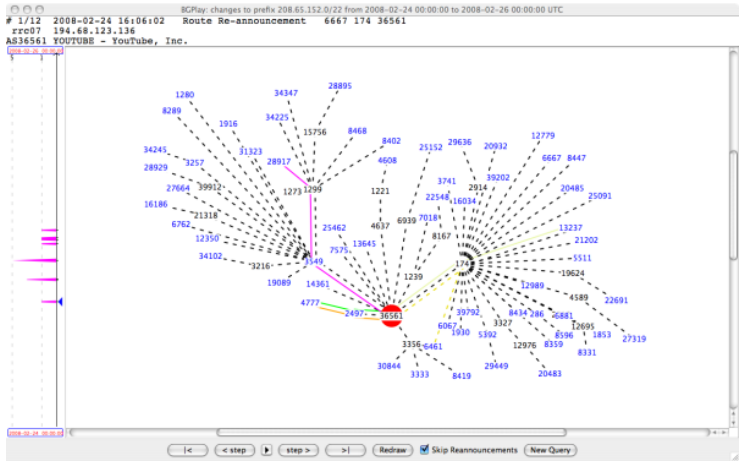
Information F receives from its neighbors about D

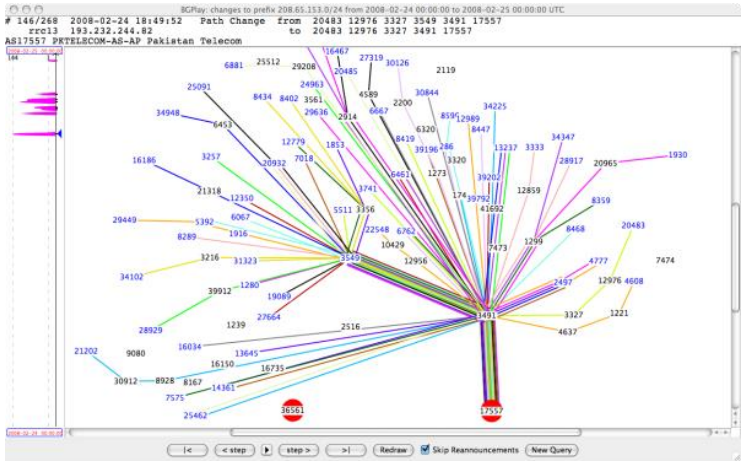
From B: "I use BCD"
From G: "I use GCD"
From I: "I use IFGCD"
From E: "I use EFGCD"

(b)

Inter-AS Routing: BGP (cont'd)

- Pakistani authorities wanted to prevent access to YouTube content
- On 24-02-2008, Pakistan Telecom started unauthorized announcement of prefix **208.65.153.0/24**
  - aimed at redirecting all YouTube traffic to their servers
- One of Pakistan Telecom's upstream providers, PCCW Global, forwarded this announcement to rest of Internet
  - thus hijacking YouTube traffic on a *global scale*





Inter-AS Routing: BGP (cont'd)

- Normally AS36561 (YouTube) announces **208.65.152.0/22**
- 24-02-2008, **18:47** (UTC): AS17557 (Pakistan Telecom) announces **208.65.153.0/24**; AS3491 (PCCW Global) propagates announcement
  - Routers worldwide receive announcement, redirect YouTube traffic to Pakistan
- 24-02-2008, **20:07** (UTC): AS36561 (YouTube) announces **208.65.153.0/24**
  - With two identical prefixes, BGP policy rules, preferring shortest AS path, determine chosen route
  - AS17557 (Pakistan Telecom) continues to attract some YouTube's traffic
- 24-02-2008, **20:18** (UTC): AS36561 (YouTube) announces **208.65.153.128/25** and **208.65.153.0/25**
  - due to **longest prefix match** rule, every router receiving announcements sends traffic to YouTube again
- 24-02-2008, **21:01** (UTC): AS3491 (PCCW Global) withdraws all prefixes originated by AS17557 (Pakistan Telecom), stopping hijack of 208.65.153.0/24

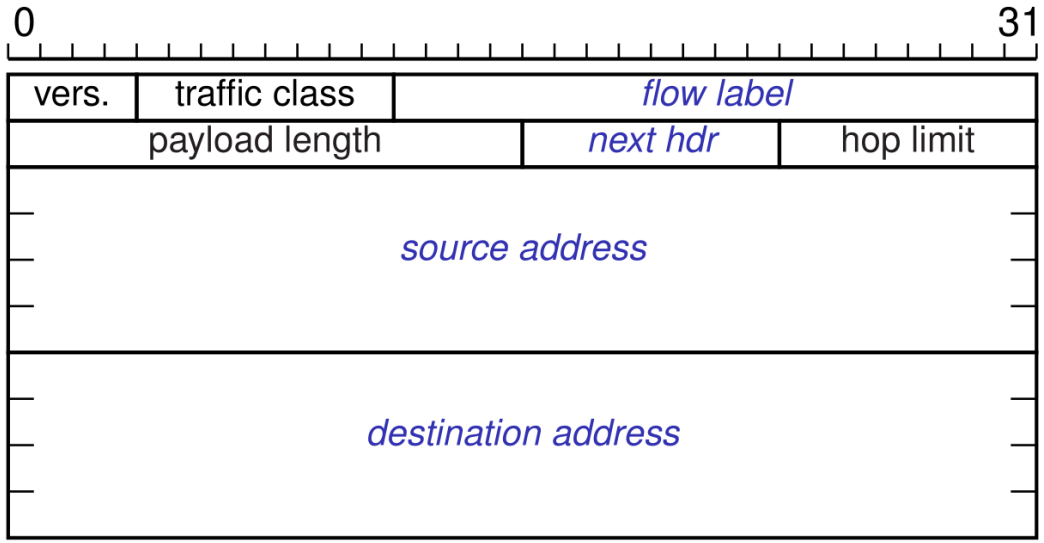
IPv6

- **Problem**: Current version of IP cannot support **enough addresses** and lacks **flexibility** to support many users

- Goals of IPv6:
  - Support billions of hosts
  - Reduce size of routing tables
  - Simplify protocol to higher router performance
  - Provide (improved) security
  - Better support for type of service
  - Support scopes with multicasting
  - Support roaming hosts without address changes
  - Coexistence of old and new protocols, and evolution of new one

- Somewhat backwards compatible but adoption is an issue...

IPv6



- Flow label is used to set up pseudo-connection between source and destination
  - identifies flow for which, e.g. bandwidth is reserved
- Expanded addressing
  - 128-bit addresses
  - anycast address (i.e. single address representing multiple interfaces)
- Simpler header structure
  - reduces processing cost and bandwidth usage
  - support for extensions and options provided by next headers

IPv6 (cont'd)

- What's been removed?

- **Fragmentation**
  - IPv6 pushes fragmentation onto end-systems ⇒ more efficient

- **Header checksum**
  - efficiency
  - avoid redundancy
    - Remember: both the Transport protocols and the lower layer protocols already provide error-detection features

- **Options**
  - efficiency: fixed-length header easier to process
  - better modularity for extensions and options

IPv6 (cont'd)

- IPv6 uses **16 Byte** (i.e. 128 bit) addresses
  - **A lot**: 3.4×10^{38} addresses in total
- Permits less efficient address allocation
  - 72% **unassigned**
- **Hexadecimal** notation:
  - eight groups of four hexadecimal digits with colons between groups
  - e.g. **2001:630:12:600:1:2:0:10b**
- IPv4 addresses are still supported however
  - **::146.179.40.24**

Prefix	Usage	Fraction
0000 0000	*Reserved (incl. IPv4)*	1/256
0000 0001	*Unassigned*	1/256
0000 001	OSI NSAP addresses	1/128
0000 011	*Unassigned*	1/128
0000 1	*Unassigned*	1/32
0001	*Unassigned*	1/16
001	Globally unique unicast addr.	1/8
010	Provider-based addresses	1/8
011	*Unassigned*	1/8
100	Geographic-based addresses	1/8
101	*Unassigned*	1/8
110	*Unassigned*	1/8
1110	*Unassigned*	1/16
1111 0	*Unassigned*	1/32
1111 10	*Unassigned*	1/64
1111 110	*Unassigned*	1/128
1111 1110 0	*Unassigned*	1/512
1111 1110 10	Link local unicast addr.	1/1024
1111 1110 11	Site local unicast addr.	1/1024
1111 1111	Multicast	1/256

IPv6 (cont'd)

Ext. header	Description
Hop–by–hop options	Information for routers (e.g. quality of service)
Routing	Full or partial route to follow
Fragmentation	Management of datagram fragments
Authentication	Verification of sender identity
Encrypted payload	Info on encrypted contents
Destination options	Add. info for destination (e.g. mobile IP)

- Basic idea: Keep main header as simple as possible; provide further information in (optional) extension headers
- Fragmentation still supported by source
  - routers don't fragment datagrams anymore
- More information on IPv6: from NTIA and Cisco

Q&A

- There is an on-going **assessed coursework** on CATe (*deadline: Monday 27/11/2017*)
  - Keep using Piazza to ask (and answer) questions
- **Suggested reading**: Tanenbaum#5; Peterson#4; Kurose#4&5.
- Please provide *anonymous* feedback on www.menti.com using the code **49 80 49**
  - *always active throughout the term*
- You can also provide *eponymous* feedback or ask questions via email (*username:* ***kgk***)
- Thank you for your attention
- **Movie of the week**: The Net
- **Next time**: What happens in the LAN, stays in the LAN (*or does it?*) – Data Link Layer