Redes de Computadores

LEIC-A, MEIC-A

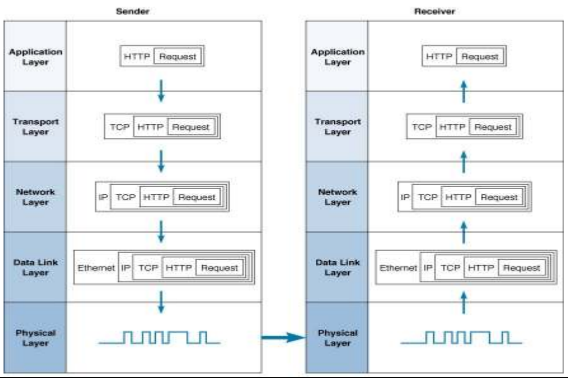
4 – Network Layer

Prof. Paulo Lobato Correia
IST, DEEC – Área Científica de Telecomunicações

1

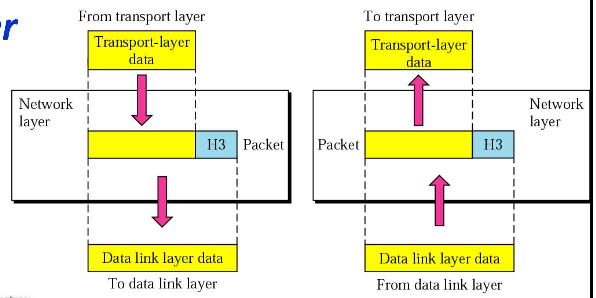
Network Layer

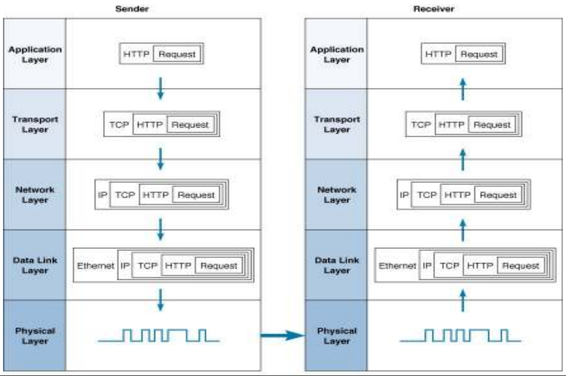
Application layer	GET /index.html HTTP/1.1 Host: www.tejo.tecnico.ulisboa.pt
Transport layer	TCP Header | GET /index.html H TCP Header | TTP/1.1 Host: www TCP Header | .tejo.tecnico.ulisboa.pt



2

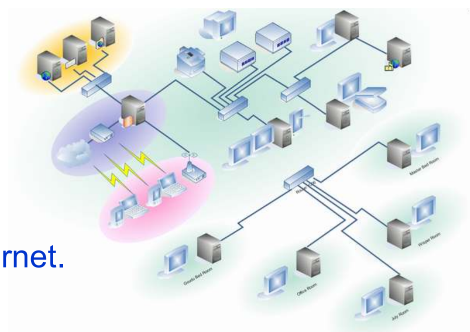
Network Layer





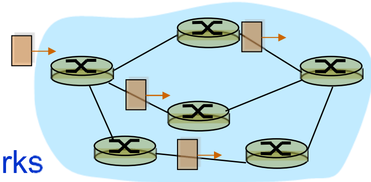
Objectives

- Understand principles behind network layer services:
  - Network layer service models;
  - Forwarding versus routing;
  - Routing (path selection);
  - Dealing with scale.
- The network layer of the Internet.



TÉCNICO LISBOA

Outline



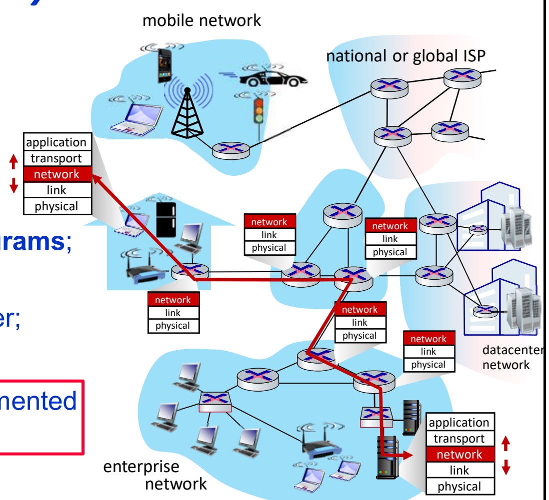
- Introduction
- Virtual circuit and datagram networks
- IPv4 addressing and forwarding tables
- Internet Protocol (IP) - Datagram format, Fragmentation
- ICMP, DHCP
- NAT, IPv6
- Routing algorithms
  - Link state, Distance Vector
- Routing in the Internet
  - Hierarchical routing, RIP, OSPF, BGP
- Broadcast and multicast routing

5

TÉCNICO LISBOA

Network Layer

- Delivers segments from sending to receiving host;
- Sending side: encapsulates segments into **datagrams**;
- Receiving side: delivers segments to transport layer;
- Network layer protocols are implemented in *every* host and router;
- Routers examine header fields in all IP datagrams passing by.



6

Network Layer: Key Functions

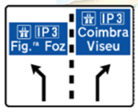
- *Routing:*
  Determine route taken by packets from source to destination:
  - *Routing algorithms.*

- *Forwarding:*
  Move packets from router's input to appropriate router output.

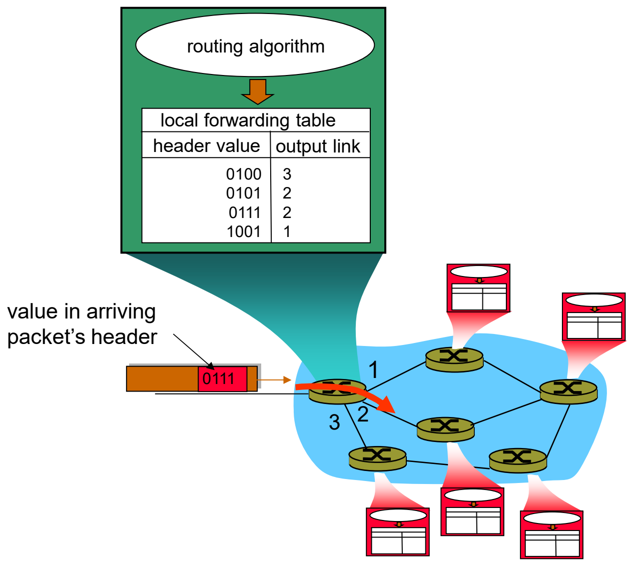
Analogy:

- Routing: process of planning trip from source to destination;

- Forwarding: process of getting through a single interchange.



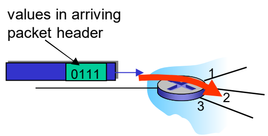
Interplay between Routing and Forwarding



Network Layer:
Data Plane + Control Plane

Data plane:
- ***Local***, per-router function
- ***Forwarding*** implementation

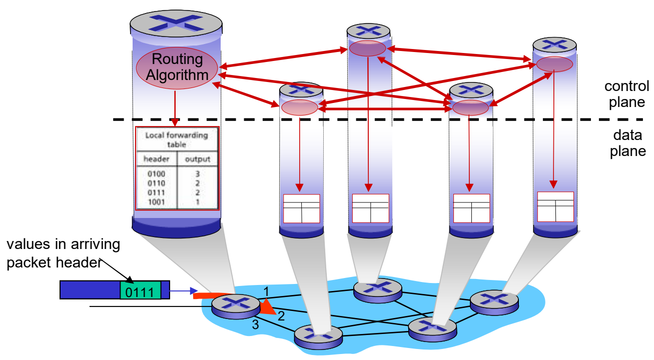
Control plane
- ***Network-wide*** *logic*
- ***Routing*** implementation



- Two control-plane approaches:
  - *Traditional routing algorithms:* implemented in routers
  - *Software-defined networking (SDN)*: implemented in (remote) servers

Traditional Routing Algorithms
(per-router Control Plane)

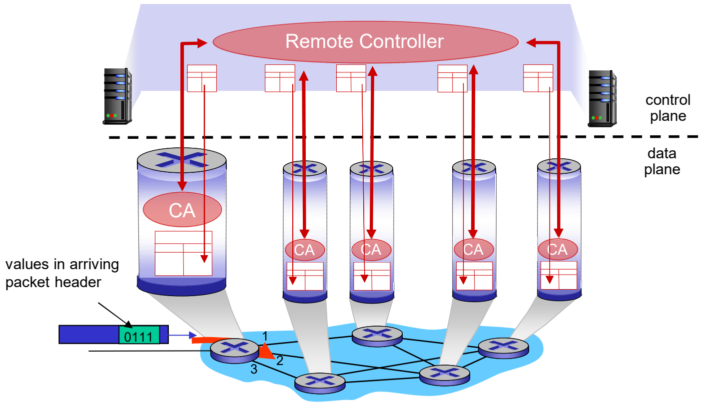
Individual routing algorithm components *in each and every router* interact in the control plane



Traditional router includes control and data planes.

Software-Defined Networking (SDN) Control Plane

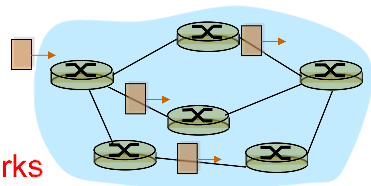
Remote controller computes and installs forwarding tables in routers



SDN (software defined network) router separates control plane (remote) from data plane (local).

11

Outline



- Introduction
- Virtual circuit and datagram networks
- IPv4 addressing and forwarding tables
- Internet Protocol (IP) - Datagram format, Fragmentation
- ICMP, DHCP
- NAT, IPv6
- Routing algorithms
  - Link state, Distance Vector
- Routing in the Internet
  - Hierarchical routing, RIP, OSPF, BGP
- Broadcast and multicast routing

12

Network Layer
Connection and Connectionless Services

- **Datagram** network provides network-layer *connectionless* service (e.g., Internet);
- **Virtual Circuit** (VC) network provides network-layer *connection oriented* service (e.g., X.25);
- Analogous to the transport-layer services, but:
  - Service: host-to-host (not end-to-end);
  - **No choice**: network provides **one or the other**;
  - Implementation: in network core.

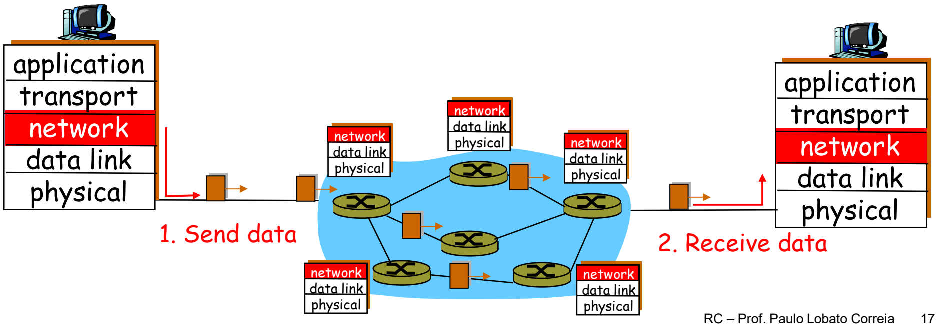
Virtual Circuits

"Source-to-destination path behaves much like a telephone circuit"
- Performance-wise;
- Network actions along source-to-destination path.

- Call setup for each call *before* data can flow;
- Each packet carries a VC identifier (and not the destination host address);
- *Every* **router** on source-destination path **maintains "state"** for each passing connection;
- Link and router resources (bandwidth, buffers) may be *allocated* to VC (**dedicated resources** = predictable service).

Datagram Networks

- No call setup at network layer;
- Routers: don't keep state information about end-to-end connections:
  - ***No network-level concept of "connection"***;
- Packets forwarded using destination host address:
  - ***Packets between same source-destination pair may take different paths.***



17

Datagram or VC Network: Why?

Internet (Datagram):

- Data exchange among computers: "elastic" service, no strict timing reqs;
- "Smart" end systems (computers) can adapt, perform error recovery, …
- Simple inside network, complexity at the "edge";
- Many different link types: difficult to offer uniform service.

ATM (Virtual Circuit):

- Evolved from telephony, which required strict timing, posed reliability requirements and needed a guaranteed service;
- Uses "dumb" end systems: the telephones;
- Complexity is inside the network.

18

Network Layer Service Models

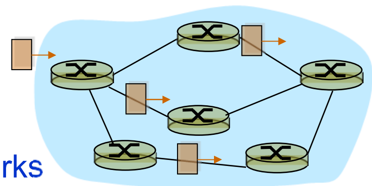
Network Architecture	Service Model	Guarantees ?				Congestion feedback
		Bandwidth	Loss	Order	Timing	
Internet	best effort	none	no	no	no	no (inferred via loss)

Internet "best effort" service model

No guarantees on*:*

i. successful datagram delivery to destination
ii. timing or order of delivery
iii. bandwidth available to end-end flow

Outline



- Introduction
- Virtual circuit and datagram networks
- IPv4 addressing and forwarding tables
- Internet Protocol (IP) - Datagram format, Fragmentation
- ICMP, DHCP
- NAT, IPv6
- Routing algorithms
  - Link state, Distance Vector
- Routing in the Internet
  - Hierarchical routing, RIP, OSPF, BGP
- Broadcast and multicast routing

TÉCNICO LISBOA

IP Addressing

At the network layer each station must be uniquely identified to allow global communication among any pair of stations connected to the Internet.

IP addresses should be unique and universal.

IP v4 address (RFC 760):

- Composed by 4 bytes (**32 bits**);
- It is usual to represent IP addresses using decimal notation, to ease human reading (e.g.: 193.136.128.1);
- Stations and routers manipulate binary addresses.

1001000100111… (32 bits)
(Used by hosts and routers)

193.136.128.1
(For human reading)

TÉCNICO LISBOA

IPv4 Addressing Space

The addressing space is the total number of available addresses. N bit addresses provide 2^N values;

With 32 bit addresses, the Internet IPv4 addressing space contains 2^{32} = *4 294 967 296* addresses.

Without other restrictions more than 4000 million devices could be connected to the Internet using IPv4.

Decimal numbering (base 10): $2022_{(10)}$

2022 = **2**x1000 + **0**x100 + **2**x10 + **2**x1

10^3=1000, 10^2=100, 10^1=10, 10^0=1

Binary numbering (base 2): $1011\ 0110_{(2)}$ -> $182_{(10)}$

182 = **1**x128 + **0**x64 + **1**x32 + **1**x16 + **0**x8 + **1**x4 + **1**x2 + **0**x1

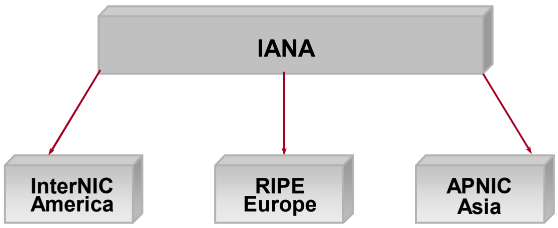
2^7=128, 2^6=64, 2^5=32, 2^4=16, 2^3=8, 2^2=4, 2^1=2, 2^0=1

IP Addressing

Q: How does an ISP get a block of addresses?

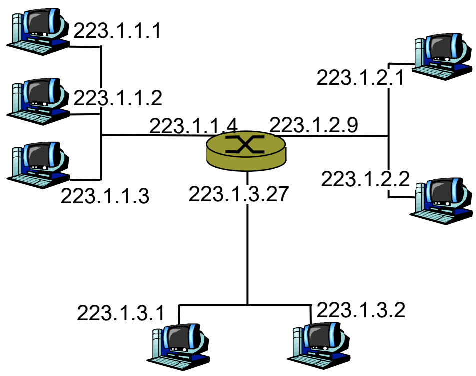
A: ICANN: Internet Corporation for Assigned Names and Numbers

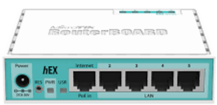
- Allocates addresses through **IANA**: Internet Assigned Numbers Authority;
- Manages DNS;
- Assigns domain names, resolves disputes.



IP Addressing

- IP address: 32-bit identifier for host and router *interfaces*.
- *Interface:* connection between host/router and a physical link:
  - Router's typically have multiple interfaces;
  - Host typically has one interface;
  - **IP addresses are associated with each interface.**





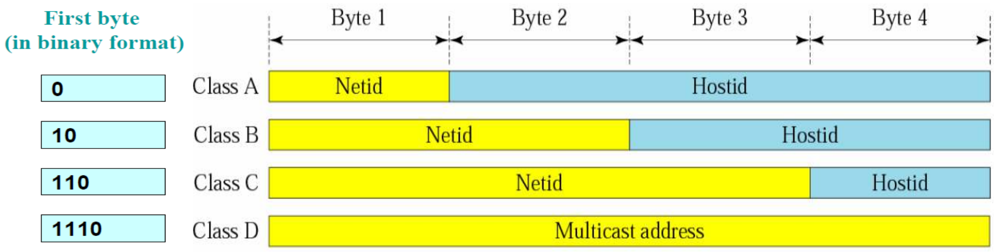
$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$

TÉCNICO LISBOA

IP Addressing

IP addresses have **two components**:
- Network identification (***NetID***);
- Station's interface identification (***HostID***);

Stations with the same network component communicate directly;
Stations with different network components communicate through routers;



TÉCNICO LISBOA

IP Addressing: CIDR

CIDR: Classless InterDomain Routing
- Subnet portion of address of arbitrary length;
- Address format:
  - a.b.c.d/**x**,
    where x is the number of bits in the subnet portion of the address.

200.23.16.0/**23**

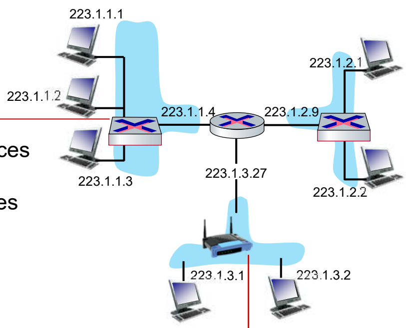
TÉCNICO LISBOA

Subnets

Q: How are interfaces actually connected?

A: We'll learn about that in chapters 6, 7

A: wired Ethernet interfaces connected by Ethernet switches



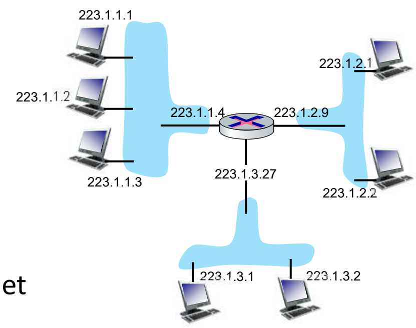
For now: don't need to worry about how one interface is connected to another (with no intervening router)

A: wireless WiFi interfaces connected by WiFi base station

TÉCNICO LISBOA

Subnets

- *What's a subnet ?*
  - device interfaces that can physically reach each other without passing through an intervening router
- IP addresses have structure:
  - subnet part: devices in same subnet have common high order bits
  - host part: remaining low order bits

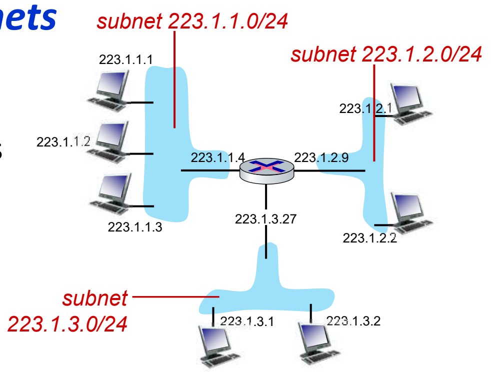


network consisting of 3 subnets

Subnets

Recipe for defining subnets:

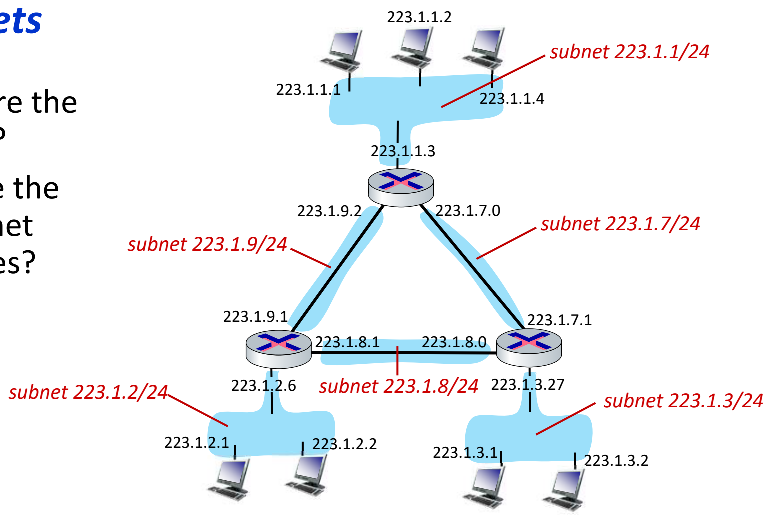
- detach each interface from its host or router, creating "islands" of isolated networks
- each isolated network is called a *subnet*



subnet mask: /24
(high-order 24 bits: subnet part of IP address)

Subnets

- where are the subnets?
- what are the /24 subnet addresses?



IP v4 Addresses

Addresses beginning with 127 are special:
- They are reserved to reference the station itself;
- 127.0.0.1 – **localhost**

Addresses with all “station” bits set to 0 (zero) :
- Represent the **network address**;
- Example: the address 193.136.128.41 belongs to the class C network 193.136.128.**0/24**

Addresses with all “station” bits set to 1 (one) :
- Represent a **broadcast address**;
- Example: 193.136.128.**255**

IP Addressing: Subnet Part

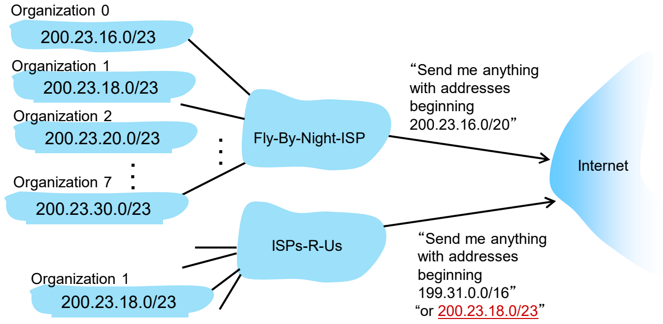
Q: How does the *network* get the subnet part of the IP address?
A: It gets allocated a portion of its provider ISP’s address space.

ISP's block	11001000 00010111 0001*0000 00000000*	200.23.16.0/20
Organization 0	11001000 00010111 0001**000***0 00000000*	200.23.16.0/23
Organization 1	11001000 00010111 0001**001***0 00000000*	200.23.18.0/23
Organization 2	11001000 00010111 0001**010***0 00000000*	200.23.20.0/23
...	....	....
Organization 7	11001000 00010111 0001**111***0 00000000*	200.23.30.0/23

Hierarchical Addressing: Route Aggregation

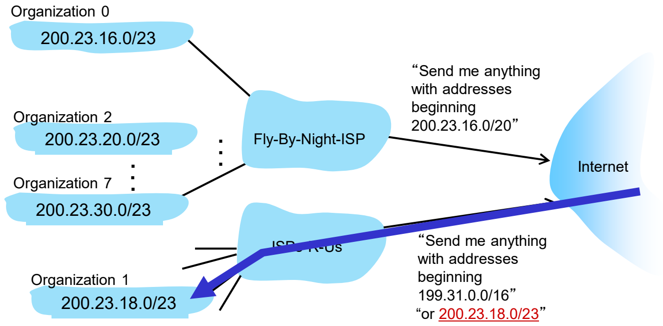
Hierarchical addressing allows efficient advertisement of routing information

- Organization 1 moves from Fly-By-Night-ISP to ISPs-R-Us
- ISPs-R-Us now advertises a more specific route to Organization 1



Hierarchical Addressing: Route Aggregation

- Organization 1 moves from Fly-By-Night-ISP to ISPs-R-Us
- ISPs-R-Us now advertises a more specific route to Organization 1



Datagrams – Forwarding Table
Longest Prefix Matching

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

Examples

Dest.Address: 11001000 00010111 00010110 10100001 Which interface?

Dest.Address: 11001000 00010111 00011000 10101010 Which interface?

Addressing

So far we have seen the usage of **3 types of addresses**:

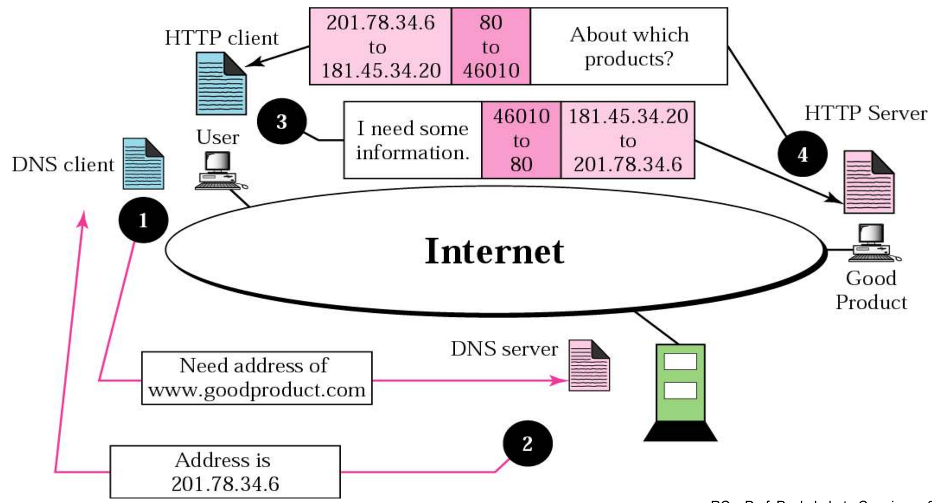
- **Application** layer address (e.g.: www.tecnico.ulisboa.pt – web server);
- **Transport** layer address (e.g.: ports 52132 and 80 – web client and server ports, respectively);
- IP address at the **network** layer address (e.g.: 193.136.222.20 and 193.136.128.1 – origin and destination, respectively);

The user only knows the first type of address (application server), but for packets to be transmitted the source and destination ports and IP addresses need to be known.

- Destination port is specific of the application in usage (e.g.: 80 for HTTP);
- Origin port number is temporarily assigned by the station, which also knows its IP address;
- Destination IP address is obtained from the application layer address using the DNS (*Domain Name System*).

TÉCNICO LISBOA

Addressing: Example



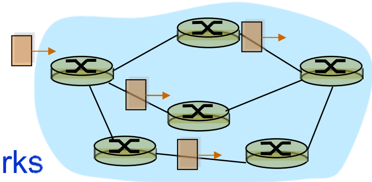
38

TÉCNICO LISBOA

39

TÉCNICO LISBOA

Outline



- Introduction
- Virtual circuit and datagram networks
- IPv4 addressing and forwarding tables
- Internet Protocol (IP) - Datagram format, Fragmentation
- ICMP, DHCP
- NAT, IPv6
- Routing algorithms
  - Link state, Distance Vector
- Routing in the Internet
  - Hierarchical routing, RIP, OSPF, BGP
- Broadcast and multicast routing

TÉCNICO LISBOA

Forwarding Tables

Row	Network/ Subnet	Mask (/Prefix)	Metric (Cost)	Interface	Next *Router*
1	128.171.0.0	255.255.0.0 (/16)	47	2	G
2	172.30.33.0	255.255.255.0 (/24)	0	1	Local
3	192.168.6.0	255.255.255.0 (/24)	12	2	G

Usually, forwarding tables do not contain one entry for each host: their size would be huge.
It is enough to include the destination network address.
That network is then responsible to deliver the message to the destination host (e.g., by diffusion).

Forwarding Tables: Masks

1. Masking

Information	1	1	0	0
Mask	1	0	1	0
Result	1	0	0	0

2. Usual Values

Binary	Decimal
00000000	0
11111111	255

3. Example 1

IP Address	172.	30.	22.	7
Mask	255.	0.	0.	0
Result	172.	0.	0.	0

4. Example 2

IP Address	172.	30.	22.	7
Mask	255.	255.	0.	0
Result	172.	30.	0.	0

To check to which network an address belongs, a binary mask is applied (logical "***AND***"), to remove the host component of the IP address.

Forwarding Tables: Masks

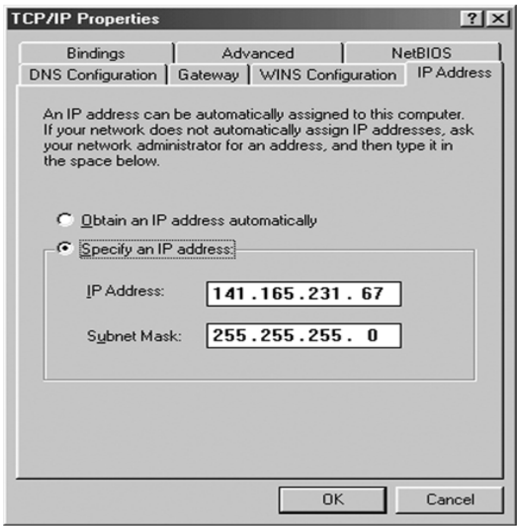
Example of applying a binary mask to the class C IP address: `234.136.25.50` to identify the network and host components.

IP Address		
11101010.10001000.00011001.	00110010	
Subnet Mask		
11111111.11111111.11111111.	00000000	/24
Network	Host	
11101010.10001000.00011001	00110010	

TÉCNICO LISBOA

Forwarding Tables

IP address configuration in the *Windows* environment.



TÉCNICO LISBOA

Forwarding

Example: Destination IP address = **172.30.33.6**

Row	Network/ Subnet	Mask (/Prefix)*	Metric (Cost)	Interface	Next-Hop Router
1	128.171.0.0	255.255.0.0 (/16)	47	2	G
2	172.30.33.0	255.255.255.0 (/24)	0	1	Local
3	192.168.6.0	255.255.255.0 (/24)	12	2	G

Router tests **1st row** of the forwarding table:

IP address = 172.30.33.6

Mask = 255.255.0.0

Result = 172.30.0.0

This result is different from the Network/Subnet value: 128.171.0.0

TÉCNICO LISBOA

Forwarding

Example: Destination IP address = **172.30.33.6**

Row	Network/ Subnet	Mask (/Prefix)*	Metric (Cost)	Interface	Next-Hop Router
1	128.171.0.0	255.255.0.0 (/16)	47	2	G
2	172.30.33.0	255.255.255.0 (/24)	0	1	Local
3	192.168.6.0	255.255.255.0 (/24)	12	2	G

Router tests **2nd row** of the forwarding table:

IP address = 172.30.33.6
Mask = 255.255.255.0
Result = 172.30.33.0

This result matches the Network/Subnet field value: 172.30.33.0

TÉCNICO LISBOA

Forwarding

TPC: Prob. 8

Row	Network/ Subnet	Mask (/Prefix)*	Metric (Cost)	Interface	Next-Hop Router
15	0.0.0.0	0.0.0.0 (/0)	5	3	H

If the mask takes the value 0.0.0.0 there is always a (length 0) matching – the result is always 0.0.0.0:

- This allows to define the ***default routing***, assumed when no other line(s) provide a match.

Forwarding

For each packet:

- First, for each routing table row, apply the mask and look for matching:
  - Analyse the packet's destination IP address;
  - Apply the mask of that routing table's row;
  - Compare masking result with the value of the *Network/Subnet* field of that row;
  - If there is a positive match:
    - Add this row to the list of candidates to route this packet;
    - Else, ignore this row.

Forwarding

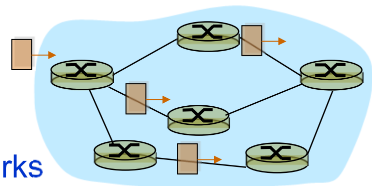
- Second, search for the best (longest) matching:
  - If there is only one match, that is the best one;
  - If there is only one longest matching, that is the best one;
  - If there are several matchings with the longest length, select the row with the lowest cost metric:
    - It can be the lowest value (e.g.: cost);
    - It can be the largest value (e.g.: throughput);
- Third, forward the packet to a network interface:
  - Send the packet to the network interface indicated in the selected row;
  - In that network or subnet, send the packet to:
    - The *next-hop-router*, or:
    - The destination station, if the *next-hop router field* contains the value "local".

Forwarding

Summary:

- A forwarding decision requires that each routing table row is tested, for each packet, to choose the best path;
  - Lengthy operation;
- Each packet is separately processed;
  - Router must have high processing power;
- With alternative routes, there may be several alternatives to forward/route a packet;
  - Choice will depend on the metric values in each row.

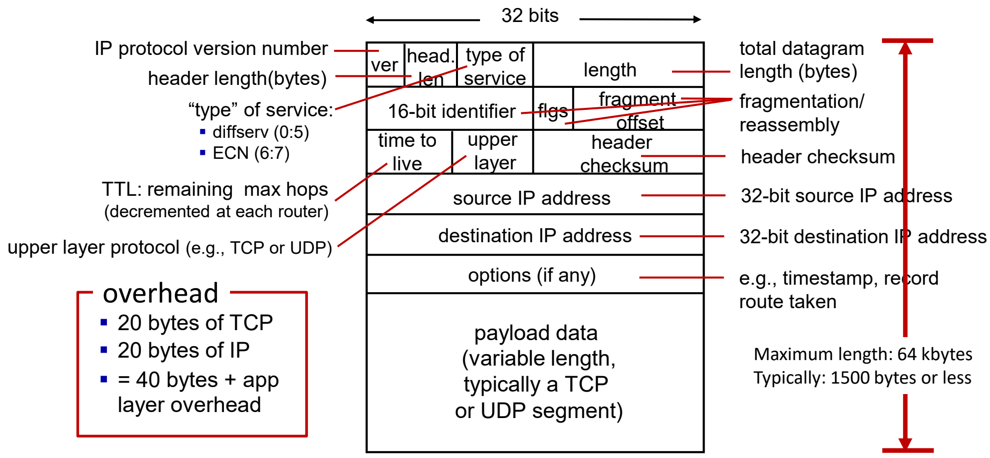
Outline



- Introduction
- Virtual circuit and datagram networks
- IPv4 addressing and forwarding tables
- Internet Protocol (IP) - Datagram format, Fragmentation
- ICMP, DHCP
- NAT, IPv6
- Routing algorithms
  - Link state, Distance Vector
- Routing in the Internet
  - Hierarchical routing, RIP, OSPF, BGP
- Broadcast and multicast routing

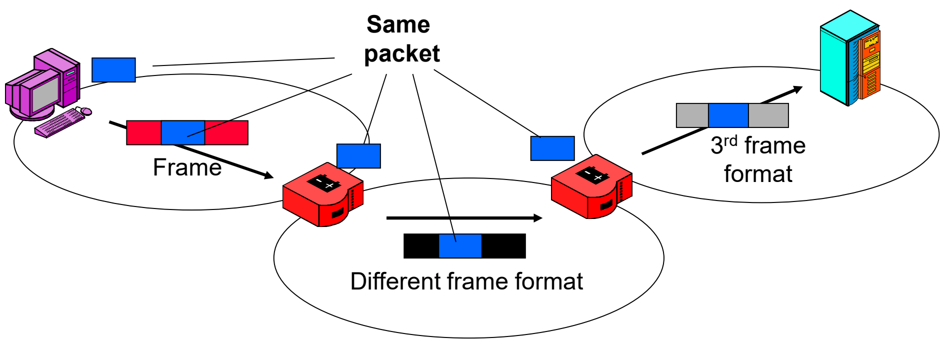
TÉCNICO LISBOA

IP Datagram format



TÉCNICO LISBOA

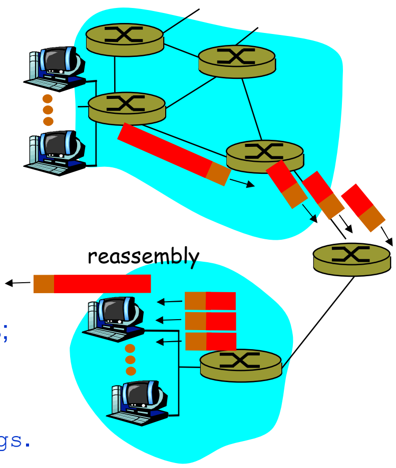
IP Fragmentation and Reassembly: Why?



A router removes the packet from the origin network and retransmits it in the following network of the selected path (***store-and-forward***), using in each different network the appropriate frame format.

IP Fragmentation and Reassembly

- Maximum Transmission Unit (MTU) is the largest possible data link frame:
  - **Different link types → Different MTU values.**
- Large IP datagrams are "fragmented" within the net:
  - One datagram becomes several datagrams;
  - "Reassembled" only at final destination;
  - IP header bits used to identify and reorder fragments, using `fragment offset + flags`.

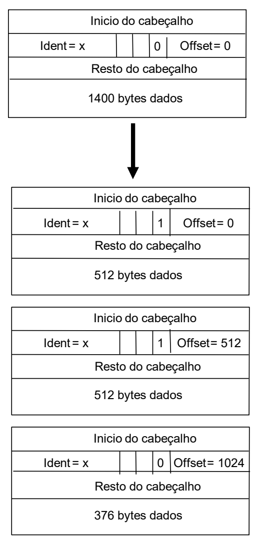


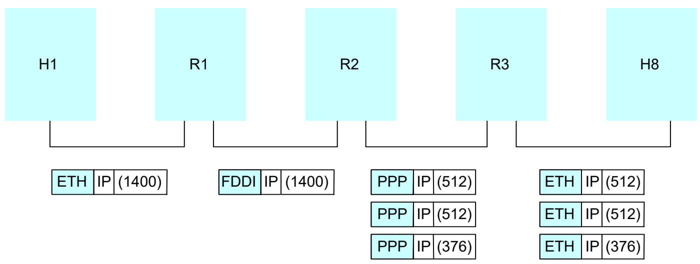
Fragmentation:
in: one large datagram
out: 3 smaller datagrams

IP Fragmentation and Reassembly

Each network has its MTU value;
Strategy:

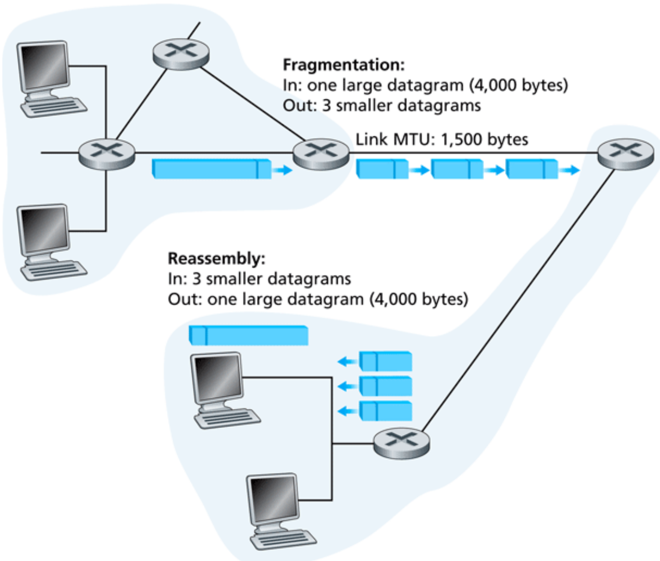
- Fragment when needed;
- Refragmentation is possible;
- Each fragment composes a datagram (same ID);
- Reconstruction is only done at the destination;
- **Offset** indicates the number of the previous fragments' byte, in multiples of 8 bytes.





IP Fragmentation and Reassembly

Example:
- 4000 byte datagram;
- 20 byte IP header;
- MTU = 1500 bytes.



IP Fragmentation and Reassembly

Example:
- 4000 byte datagram (total length):
  - 20 byte IP header
  - 3980 data bytes
- MTU = 1500 bytes

length	ID	fragflag	offset
=4000	=x	=0	=0

3980 bytes in data field

One large datagram becomes several smaller datagrams

length	ID	fragflag	offset
=1500	=x	=1	=0
=1500	=x	=1	=185
=1040	=x	=0	=370

total length = MTU = 1500 bytes
IP header (20) + data field (1480)

offset =
= 1480/8 = 185

What if offset is not integer?
e.g.: MTU = 1320 → offset = 1300/8 = 162.5
how to represent in binary (13 bits)?

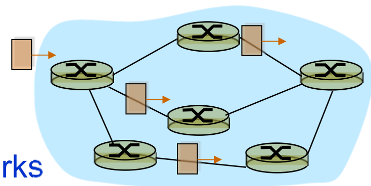
IP Fragmentation and Reassembly

TPC: Prob. 9

Example:
- 4000 byte datagram;
- 20 byte IP header;
- MTU = 1500 bytes.

Fragment	Bytes	ID	Offset	Flag
1st fragment	1,480 bytes in the data field of the IP datagram	identification = 777	offset = 0 (meaning the data should be inserted beginning at byte 0)	flag = 1 (meaning there is more)
2nd fragment	1,480 bytes of data	identification = 777	offset = 185 (meaning the data should be inserted beginning at byte 1,480. Note that 185 · 8 = 1,480)	flag = 1 (meaning there is more)
3rd fragment	1,020 bytes (= 3,980–1,480–1,480) of data	identification = 777	offset = 370 (meaning the data should be inserted beginning at byte 2,960. Note that 370 · 8 = 2,960)	flag = 0 (meaning this is the last fragment)

Outline

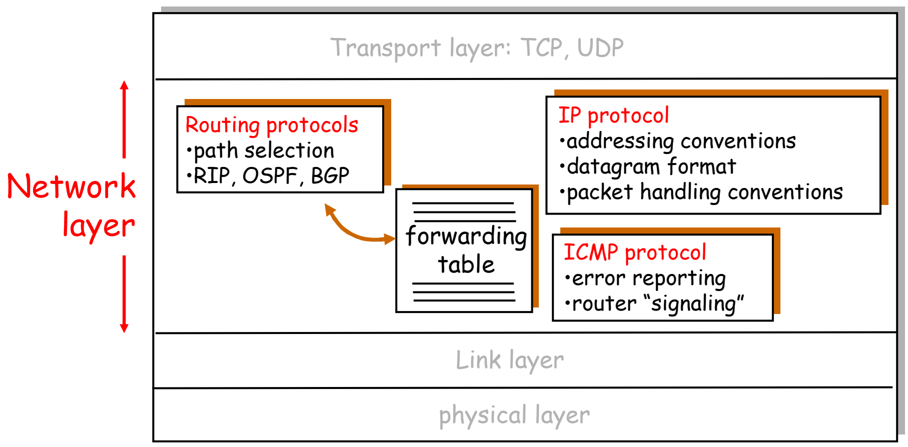


- Introduction
- Virtual circuit and datagram networks
- IPv4 addressing and forwarding tables
- Internet Protocol (IP) - Datagram format, Fragmentation
- ICMP, DHCP
- NAT, IPv6
- Routing algorithms
  - Link state, Distance Vector
- Routing in the Internet
  - Hierarchical routing, RIP, OSPF, BGP
- Broadcast and multicast routing

TÉCNICO LISBOA

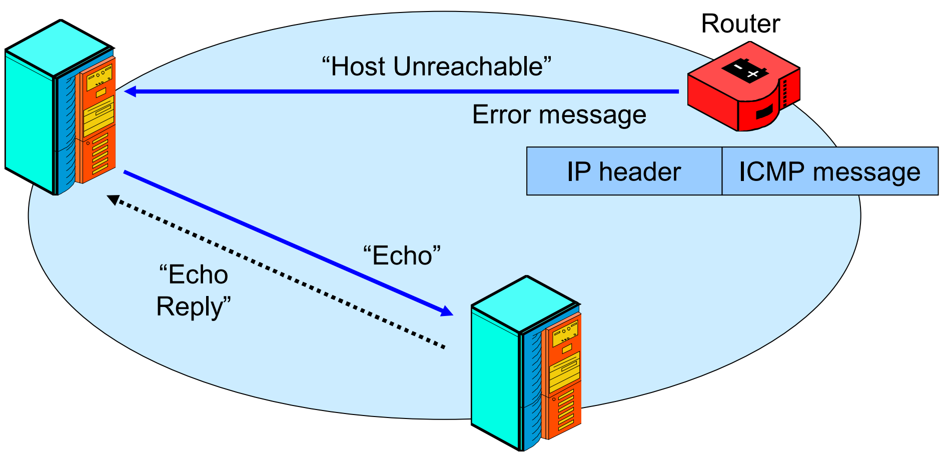
The Internet Network Layer

Host, router network layer functions:



TÉCNICO LISBOA

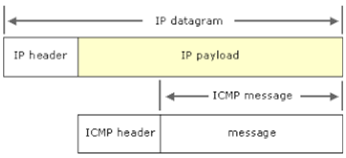
Internet Control Message Protocol (ICMP)



Internet Control Message Protocol (ICMP)

ICMP (RFC 792):

- Used by hosts and routers to communicate network-level information:
  - Error reporting: unreachable host, network, port, protocol;
  - Echo request/reply (used by ping).
- Network-layer "above" IP:
  - ICMP messages carried in IP datagrams;

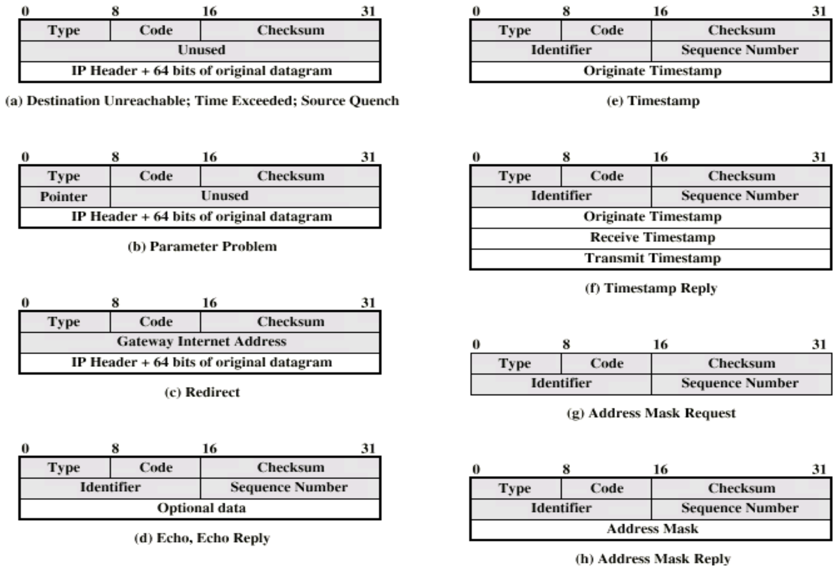


- ICMP message:
  - Type and code;
  - First 8 bytes of IP datagram causing error.

Internet Control Message Protocol (ICMP)

Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

Internet Control Message Protocol (ICMP)



Traceroute and ICMP

- Source sends series of UDP segments to destination:
  - First has TTL =1;
  - Second has TTL=2, etc.
  - Unlikely port number.
- When n^{th} datagram arrives to n^{th} router (TTL):
  - Router discards datagram;
  - Router sends to source an ICMP message (type 11, code 0);
  - Message includes name of router and IP address;
- When ICMP message arrives, source calculates RTT;
- Traceroute does this 3 times.

Stopping criterion:

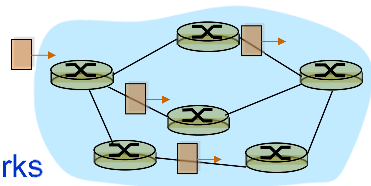
- UDP segment eventually arrives at destination host;
- Destination returns ICMP “host unreachable” packet (type 3, code 3);
- When source gets this ICMP, stops.

Traceroute

```
> traceroute www.google.com
traceroute to www.google.com (74.125.39.103), 30 hops max, 40 byte packets

 1  gt-ci-tn.ist.utl.pt (193.136.138.254)  1.797 ms  2.207 ms  2.441 ms
 2  gatekeeper1.ci.ist.utl.pt (192.168.253.3)  0.260 ms  0.277 ms  0.217 ms
 3  brites2.utl.pt (193.136.134.11)  0.360 ms  0.445 ms  0.348 ms
 4  Router3.GE.Lisboa.fccn.pt (193.136.1.89)  0.554 ms  0.564 ms  0.528 ms
 5  ROUTER4.10GE.Lisboa.fccn.pt (193.137.0.20)  0.614 ms  0.557 ms  0.640 ms
 6  fccn.rt1.mad.es.geant2.net (62.40.124.97)  38.229 ms  38.005 ms  37.992 ms
 7  so-7-2-0.rt1.gen.ch.geant2.net (62.40.112.25)  33.593 ms  33.689 ms  33.744 ms
 8  so-3-3-0.rt1.fra.de.geant2.net (62.40.112.70)  41.916 ms  41.926 ms  42.056 ms
 9  TenGigabitEthernet7-3.ar1.FRA4.gblx.net (207.138.144.45)  42.218 ms  42.236 ms
   42.395 ms
10  74.125.50.189 (74.125.50.189)  42.117 ms  42.125 ms  42.223 ms
11  209.85.255.170 (209.85.255.170)  42.302 ms  42.299 ms  42.277 ms
12  209.85.254.116 (209.85.254.116)  42.729 ms  42.748 ms  42.762 ms
13  209.85.249.162 (209.85.249.162)  42.820 ms 209.85.249.166 (209.85.249.166)
   42.980 ms  42.981 ms
14  fx-in-f103.google.com (74.125.39.103)  42.842 ms  43.061 ms  43.040 ms
```


Outline



- Introduction
- Virtual circuit and datagram networks
- IPv4 addressing and forwarding tables
- Internet Protocol (IP) - Datagram format, Fragmentation
- ICMP, DHCP
- NAT, IPv6
- Routing algorithms
  - Link state, Distance Vector
- Routing in the Internet
  - Hierarchical routing, RIP, OSPF, BGP
- Broadcast and multicast routing

TÉCNICO LISBOA

IP Addresses: How to Get One?

Q: How does a *host* get an IP address?

- Hard-coded by system administrator in a file:
  - Windows: control-panel->network->configuration->TCP/IP->properties;
  - UNIX: /etc/rc.config.
- **DHCP**: Dynamic Host Configuration Protocol
  (RFC 2131, updated by: 3396, 4361, 5494, 6842):
  - Dynamically get an address from server;
  - "Plug-and-play".

TÉCNICO LISBOA

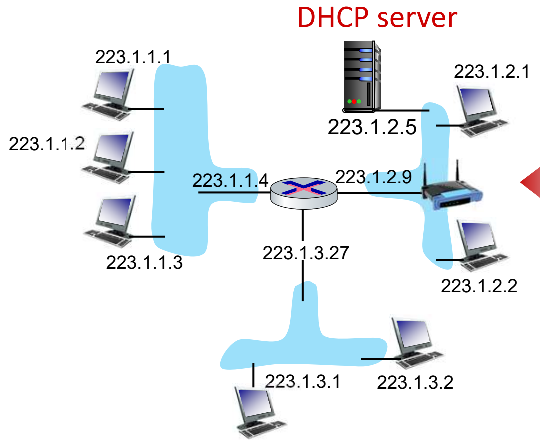
DHCP: Dynamic Host Configuration Protocol

Goal: allow host to *dynamically* obtain its IP address from a network server when it joins the network:
- Allows reuse of addresses (only hold address while connected and "on");
- Can renew its lease on address in use;
- Support for mobile users who want to join network.

DHCP overview:
- Host broadcasts "DHCP discover" message;
- DHCP server responds with "DHCP offer" message;
- Host requests IP address: "DHCP request" message;
- DHCP server sends address: "DHCP ACK" message.

TÉCNICO LISBOA

DHCP Client-Server Scenario

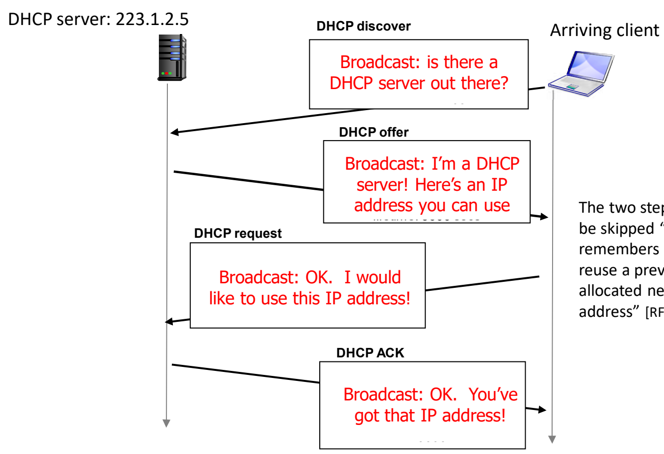


Typically, DHCP server will be co-located in router, serving all subnets to which router is attached

arriving DHCP client needs address in this network

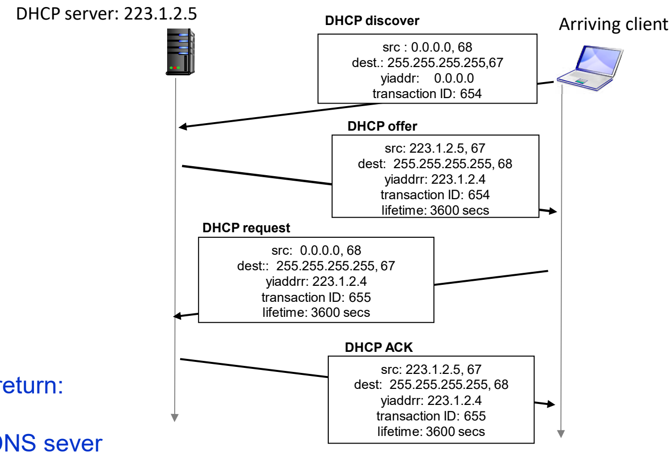
TÉCNICO LISBOA

DHCP Client-Server Scenario



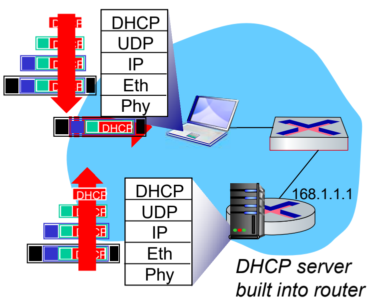
The two steps above can be skipped "if a client remembers and wishes to reuse a previously allocated network address" [RFC 2131]

DHCP Client-Server Scenario



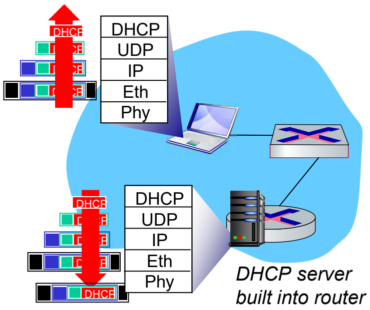
DHCP can also return:
- network mask
- IP address of DNS sever
- address of first-hop router

DHCP: example



- Connecting laptop will use DHCP to get IP address (+ address of first-hop router, + address of DNS server).
- DHCP REQUEST message encapsulated in UDP segment, encapsulated in IP packet, encapsulated in Ethernet frame
- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server

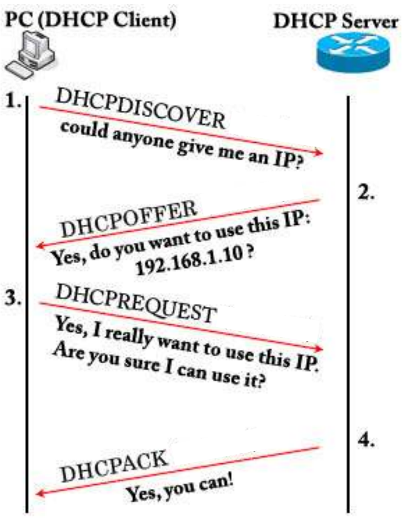
DHCP: example

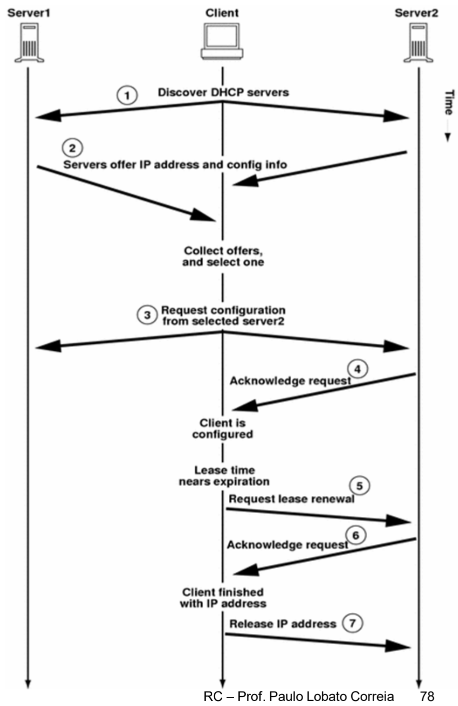


- DCP server formulates DHCP ACK containing client' s IP address, + IP address of first-hop router,
- + name & IP address of DNS server

77

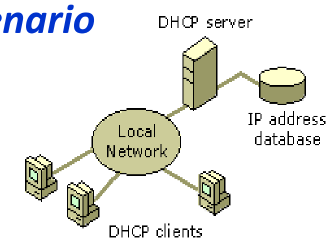
DHCP





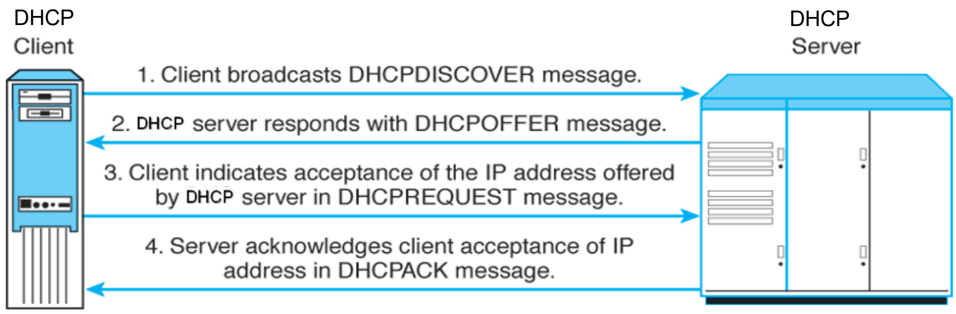
78

DHCP Client-Server Scenario

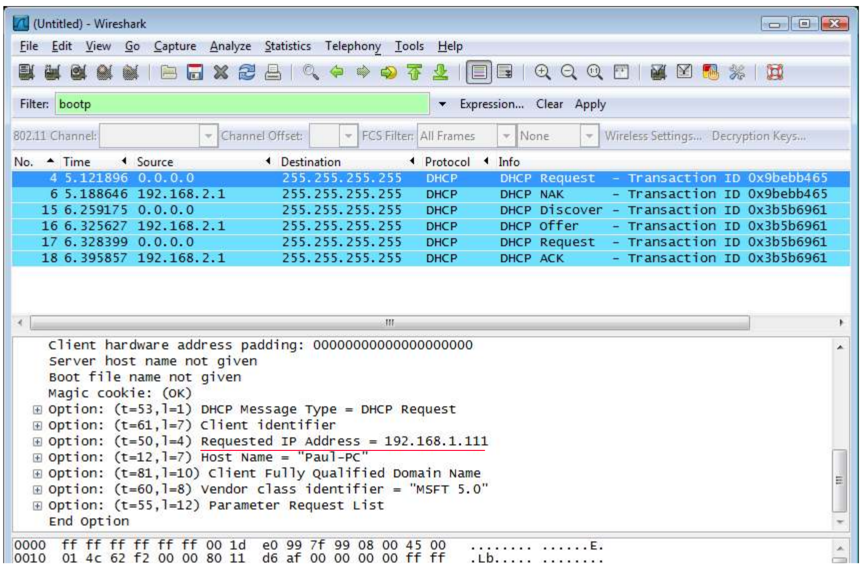


- DHCP uses UDP.
- Client packets: source port 68, destination port 67;
- Server packets: source port 67, destination port 68.

DHCPv6 uses UDP port number 546 for clients and port number 547 for servers.



DHCP

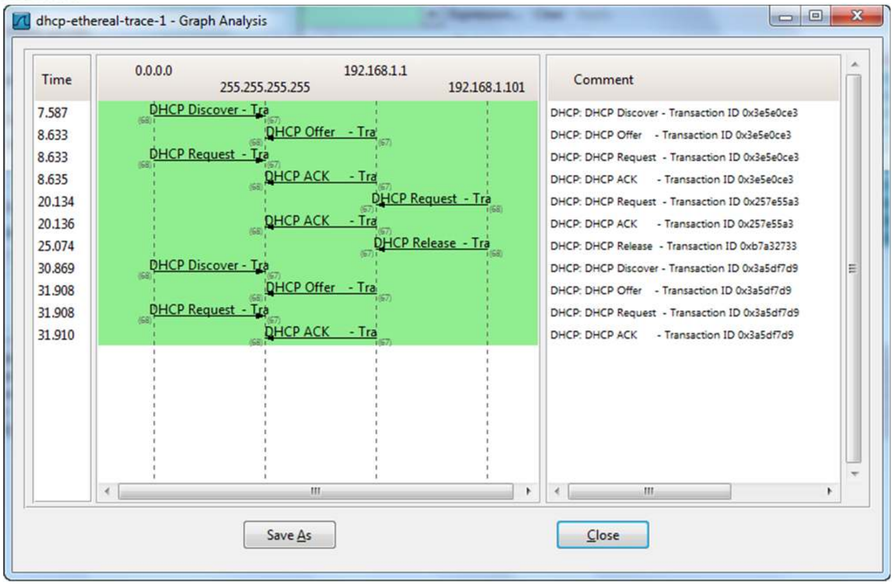


Address Leasing

- Address leasing:
  - *T1 Time* (***50%*** of *Lease Time*) – time after which the terminal should **try to renew** the address leasing;
  - *T2 Time* (***85%*** of *Lease Time*) – time after which the terminal should **try to renew** the address leasing **again**, if the first trial has failed;
  - *Lease Time* – time after which the terminal should **stop using** the leased address, if not renewed in the meantime.

It is common for the server to set the lease time to several hours or days [Droms 2002].

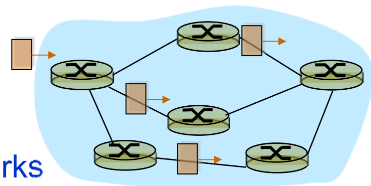
DHCP



Other DHCP messages

- *DHCP Decline*:
  - The client rejects the offer he was made and restarts the process;
- *DHCP Nack*:
  - The server informs it cannot satisfy the request done with the *DHCP Request* message;
- *DHCP Release*:
  - The client informs that he wants to terminate the address lease;
- *DHCP Inform*:
  - The client asks for just a few parameters (in this case, the client already has an IP address, but he asks, for instance, the address of a DNS server).

Outline



- Introduction
- Virtual circuit and datagram networks
- IPv4 addressing and forwarding tables
- Internet Protocol (IP) - Datagram format, Fragmentation
- ICMP, DHCP
- NAT, IPv6
- Routing algorithms
  - Link state, Distance Vector
- Routing in the Internet
  - Hierarchical routing, RIP, OSPF, BGP
- Broadcast and multicast routing

Private IP Addresses

A number of address blocks are assigned for private use. They are not recognized globally.

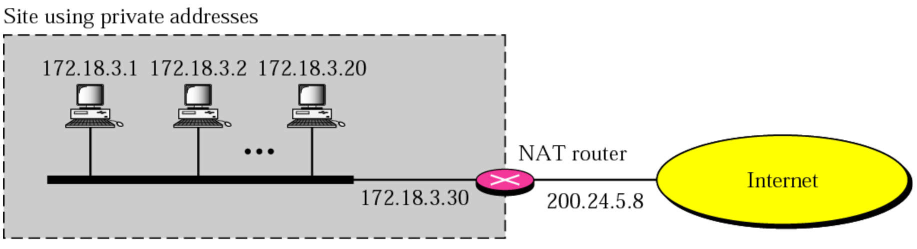
The Internet Assigned Numbers Authority (IANA) has reserved the following three blocks of the IP address space for private internets (RFC 1918):

10.0.0.0	-	10.255.255.255	(10.0.0.0/8 prefix)
172.16.0.0	-	172.31.255.255	(172.16.0.0/12 prefix)
192.168.0.0	-	192.168.255.255	(192.168.0.0/16 prefix)

Network Address Translation (NAT)

***Network Address Translation* (NAT):**

- Allows a (local) network to use a set of private addresses, for communication inside the network;
- Communication to outside the private network uses a set of (at least one) global IP address;
- All inside addresses are hidden from the outside – increased security.

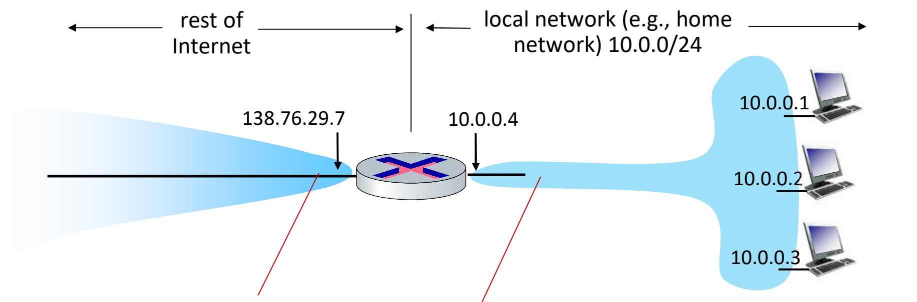


Network Address Translation (NAT)

- Inside the network a set of *unregistered, or private, IP addresses* can be used (10.0.0.0 – 10.255.255.255; 172.16.0.0 – 172.31.255.255; 192.168.0.0 – 192.168.255.255).
- When a private network user sends a packet, the NAT replaces the internal sender IP address by the network external IP address. The correspondence is memorized.
- When a reply is received, the NAT restores the internal address after checking the memory.
- The NAT may use a fixed mapping table to support the reception of packets sent from outside the private network.
- If the internal address is not in memory, the packet is discarded.
- Can change ISP without changing addresses of devices in local network.

Network Address Translation (NAT)

NAT: all devices in local network share just one IPv4 address as far as outside world is concerned



all datagrams *leaving* local network have *same* source NAT IP address: 138.76.29.7, but *different* source port numbers

datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

Network Address Translation (NAT)

Implementation – NAT router must:

- *Outgoing datagrams:*

  Replace **`(source IP address, port #)`** of every outgoing datagram with **`(NAT IP address, new port #);`**

  Remote host responds using **`(NAT IP address, new port #)`** as destination address.

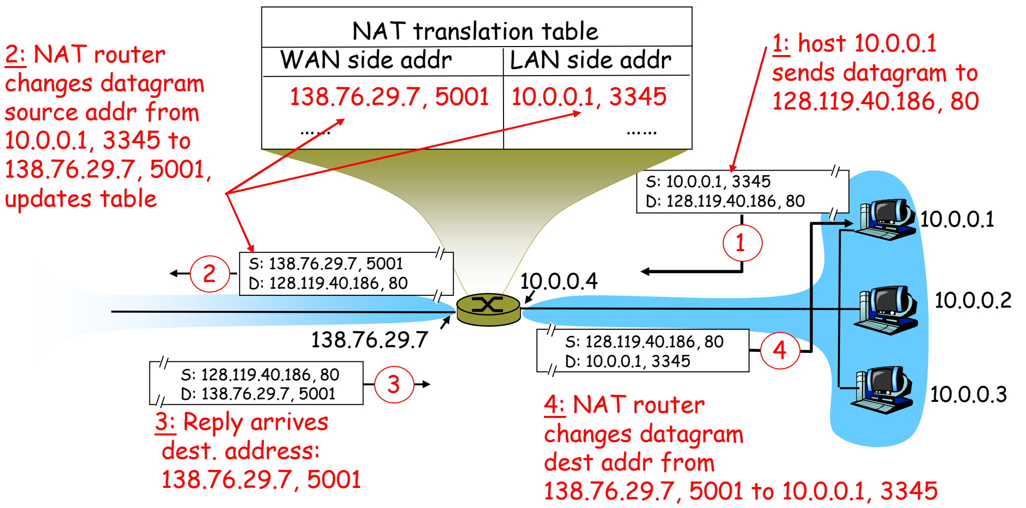
- *Remember (in* ***NAT translation table****):*

  Every **`(source IP address, port #)`** to **`(NAT IP address, new port #)`** translation pair.

- *Incoming datagrams:*

  Replace **`(NAT IP address, new port #)`** in destination fields of every incoming datagram with corresponding **`(source IP address, port #)`** stored in the NAT table.

Network Address Translation (NAT)

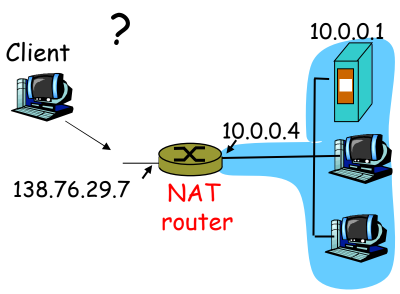


Network Address Translation (NAT)

- 16-bit port-number field:
  - > 60,000 simultaneous connections with a single LAN-side address!
- NAT is controversial:
  - Routers should only process up to layer 3;
  - Violates end-to-end argument:
    - NAT possibility must be taken into account by application designers, e.g., P2P applications;
  - Address shortage should instead be solved by IPv6…
- NAT is here to stay:
  - Extensively used in home and institutional nets, 4G/5G cellular nets

NAT Traversal Problem

- Client wants to connect to server with address 10.0.0.1:
  - Server address 10.0.0.1 is local to LAN (client can't use it as destination address);
  - Only one externally visible NATted address: 138.76.29.7.
- Solution 1: statically configure NAT to forward incoming connection requests at given port to server:
  - E.g., (123.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000.

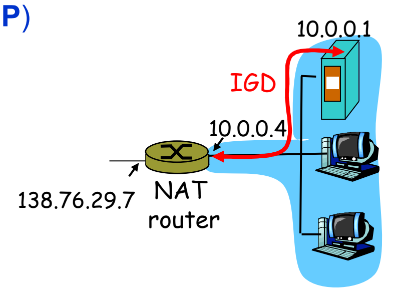


NAT Traversal Problem

- Solution 2: Universal Plug and Play (**UPnP**) Internet Gateway Device (IGD) Protocol.

  Allows NATted host to:
  - Learn public IP address (138.76.29.7);
  - Add/Remove port mappings (with lease times);

  It automates static NAT port map configuration.

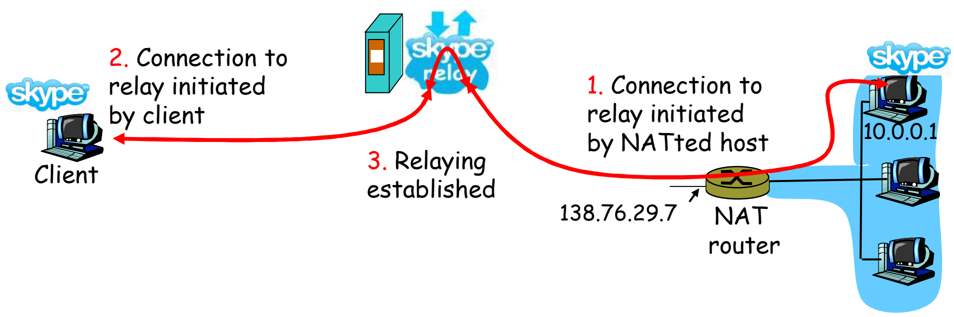


RC – Prof. Paulo Lobato Correia 93

93

NAT Traversal Problem

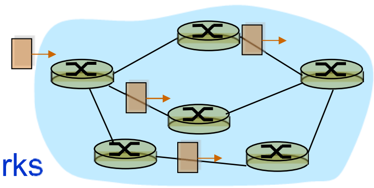
- Solution 3 – relaying (used in Skype):
  - NATed client establishes connection to relay;
  - External client connects to relay;
  - Relay bridges packets between two connections.



RC – Prof. Paulo Lobato Correia 94

94

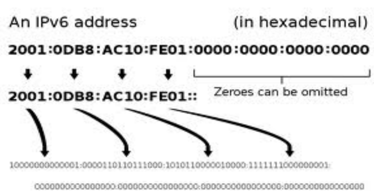
Outline



- Introduction
- Virtual circuit and datagram networks
- IPv4 addressing and forwarding tables
- Internet Protocol (IP) - Datagram format, Fragmentation
- ICMP, DHCP
- NAT, IPv6
- Routing algorithms
  - Link state, Distance Vector
- Routing in the Internet
  - Hierarchical routing, RIP, OSPF, BGP
- Broadcast and multicast routing

IP v6 Addresses

- IP v4 addresses (32 bits) shortage:
  - Network and station components “waste” addresses;
  - Stations “use” addresses even when not connect to the Internet;
  - Internet keeps growing.
- **IP v6**:
  - New IP generation;
  - Addresses represented using **128 bits**;
  - Auto-configuration of addresses;
  - Header: simpler and more efficient;
  - Allow fast packet forwarding and routing;
  - Quality of service (QoS);
  - Authentication and encryption.



369A:54B4:9856:1256:7531:AAD2:FA01:1F21

0:0:0:0:0:0:FA01:1F21

::FA01:1F21

IPv6 Unicast Addresses

There are four types of unicast addresses:

- **Global unicast** – conventional, publicly routable address (like IPv4 public addresses);
- **Link-local** – similar to private IPv4 addresses, to be used inside a single network segment.
- **Unique local** – also for private addressing, with the addition of being unique – joining two subnets will not cause address collisions.
- **Special** – loopback addresses; IPv4-address mapped spaces; 6-to-4 addresses (for crossing from an IPv4 network to an IPv6 network).

Types of Addresses and Assigned Prefixes

Allocation	Prefix binary	Prefix hex
Unassigned	0000 0000	::0/8
Reserved	0000 001	0200/7
Loopback address		::1/128
Global unicast	001	2000::/3
Link-local unicast	1111 1110 10	FE80::/10
Reserved (formerly site-local unicast)	1111 1110 10	FEC0::/10
Local IPv6 address	1111 110	FC00::/7
Private administration	1111 1101	FD00::/8
Multicast	1111 1111	FF00::/8

Local IPv6 addresses = IPv4 private addresses;

- Not routable in the global IPv6 Internet.

TÉCNICO LISBOA

Link-Local Unicast

- For use on a single link
- Not routable
- Allow address autoconfiguration

10 bits	54 bits	64 bits
1111 1110 10	**0**	**Interface ID**

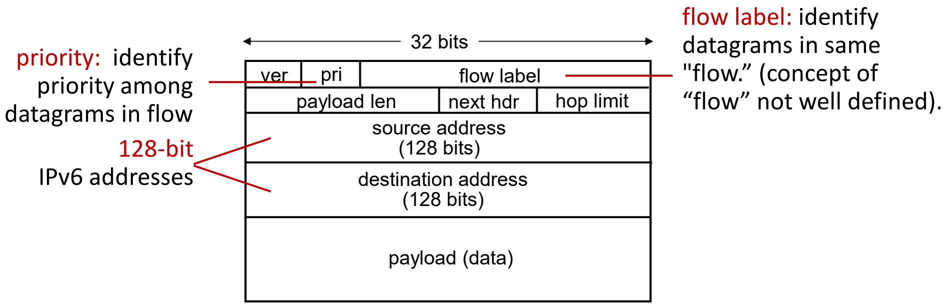
101

TÉCNICO LISBOA

Address Configuration

- Manual configuration
- Using DHCPv6
- Auto-configuration using the interface ID (last 64 bits of IPv6 unicast address)
  - Can be based on MAC address
  - Can be random number!
  - Network prefix always given by router

IPv6 datagram format



What’s missing (compared with IPv4):

- no checksum (to speed processing at routers)
- no fragmentation/reassembly
- no options (available as upper-layer, next-header protocol at router)

Other Changes from IPv4

- *Checksum*:
  - Removed to reduce processing time at each hop.
- *Options:*
  - Allowed, but outside of header;
  - Indicated by “Next Header” field.
- *ICMPv6* (new version of ICMP):
  - Additional message types, e.g. “*Packet too Big*”;
  - Multicast group management functions.

Other Changes from IPv4

- *Fragmentation:*

  IPv6 routers do not fragment, but drop the packets that are larger than the MTU (*min MTU = 1280 bytes*).

  IPv6 hosts are required to determine the optimal Path MTU before sending packets.

To send a packet larger than the path MTU:

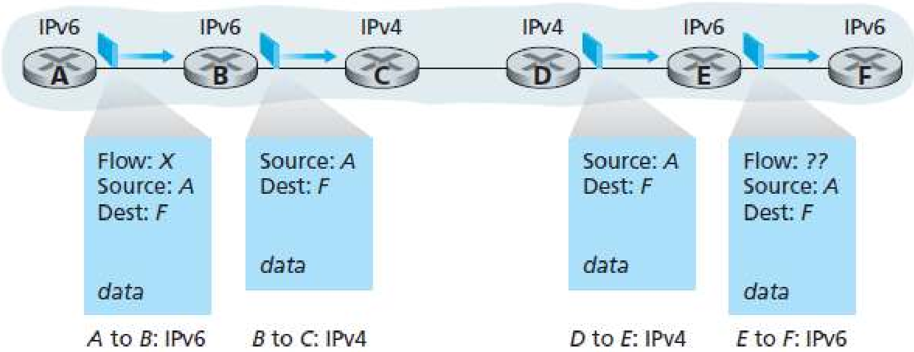
- ***sending node*** **splits the packet into fragments;**
- *Fragment extension header* carries the information necessary to reassemble the original (i.e., unfragmented) packet (including *offset* value in bytes and the *more fragments* flag).

Transition From IPv4 To IPv6

- Not all routers can be upgraded simultaneously:
  - No “flag days”;
  - How will the network operate with mixed IPv4 and IPv6 routers?
- *Double stack:* Some routers can translate between IPv4 and IPv6 headers.
- *Tunneling:* IPv6 carried as payload in IPv4 datagram among IPv4 routers.

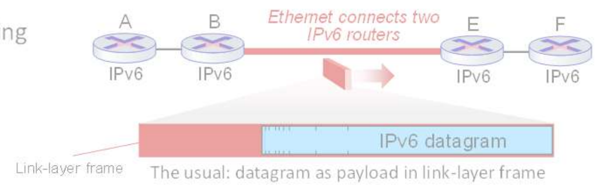
Double Stack

Double Stack: Some information present in the original IPv6 Header is lost when translating to an IPv4 datagram!

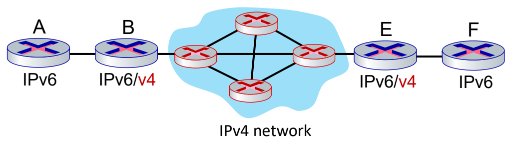


Tunneling and Encapsulation

Ethernet connecting two IPv6 routers:

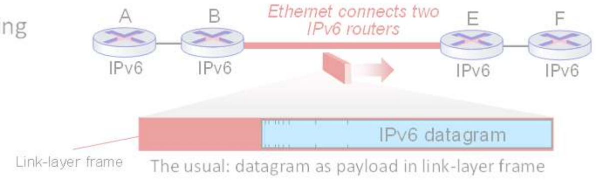


IPv4 network connecting two IPv6 routers

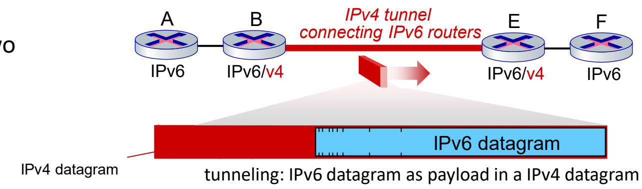


Tunneling and Encapsulation

Ethernet connecting two IPv6 routers:

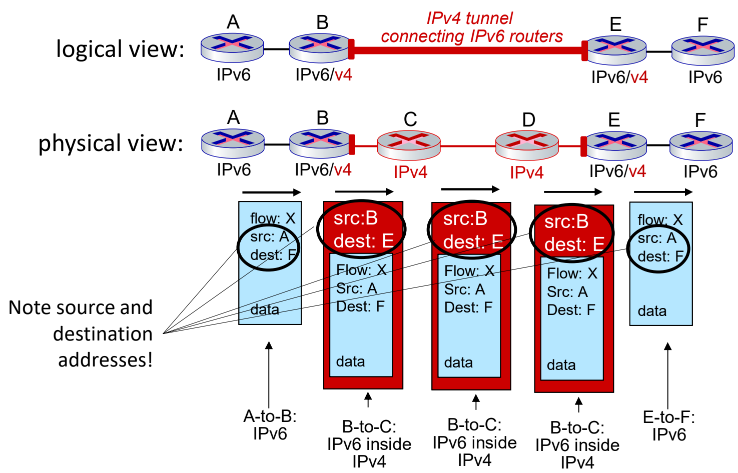


IPv4 tunnel connecting two IPv6 routers



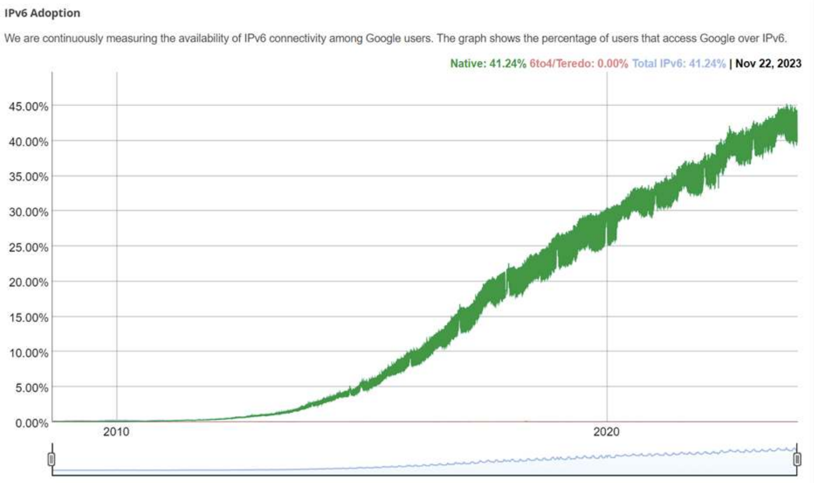
113

Tunneling



114

IPv6 Adoption



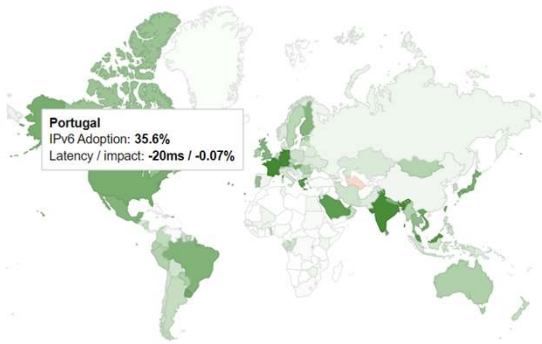
Google: ~ 40% of clients access services via IPv6

https://www.google.com/intl/en/ipv6/statistics.html

115

IPv6 Adoption

Per-Country IPv6 adoption



World | Africa | Asia | Europe | Oceania | North America | Central America | Caribbean | South America

The chart above shows the availability of IPv6 connectivity around the world.

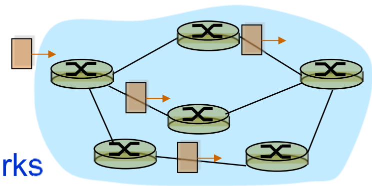
Regions where IPv6 is more widely deployed (the darker the green, the greater the deployment) and users experience infrequent issues connecting to IPv6-enabled websites.

Google: ~ 40% of clients access services via IPv6

https://www.google.com/intl/en/ipv6/statistics.html

116

Outline



- Introduction
- Virtual circuit and datagram networks
- IPv4 addressing and forwarding tables
- Internet Protocol (IP) - Datagram format, Fragmentation
- ICMP, DHCP
- NAT, IPv6
- Routing algorithms
  - Link state, Distance Vector
- Routing in the Internet
  - Hierarchical routing, RIP, OSPF, BGP
- Broadcast and multicast routing

The Internet Network Layer

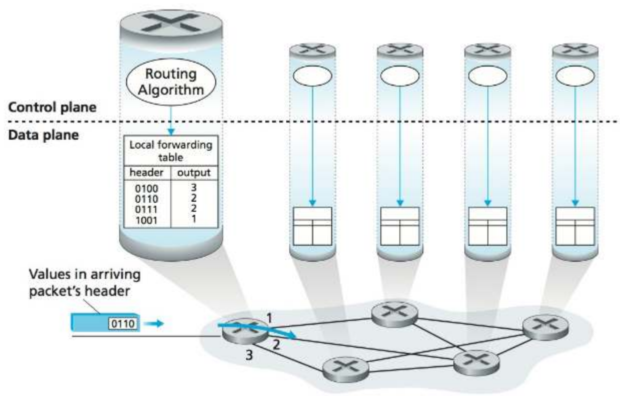
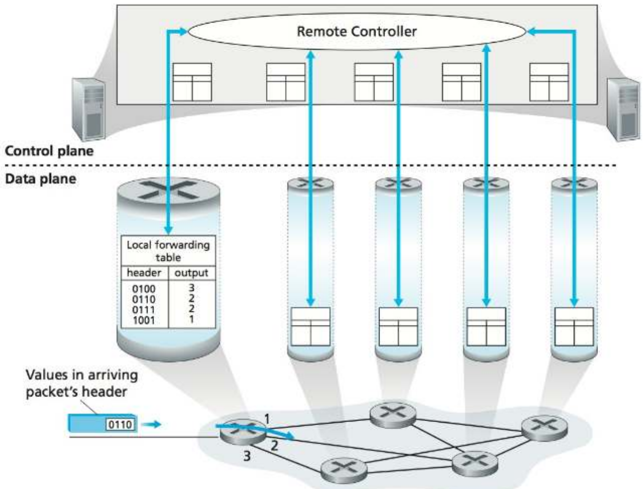


Figure 4.2 ♦ Routing algorithms determine values in forward tables

Traditional router includes control and data planes.

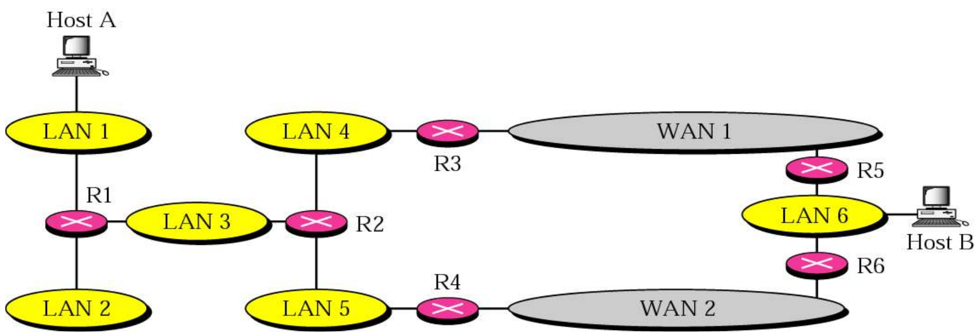
The Internet Network Layer



SDN (software defined network) router separates control plane (remote) from data plane (local).

127

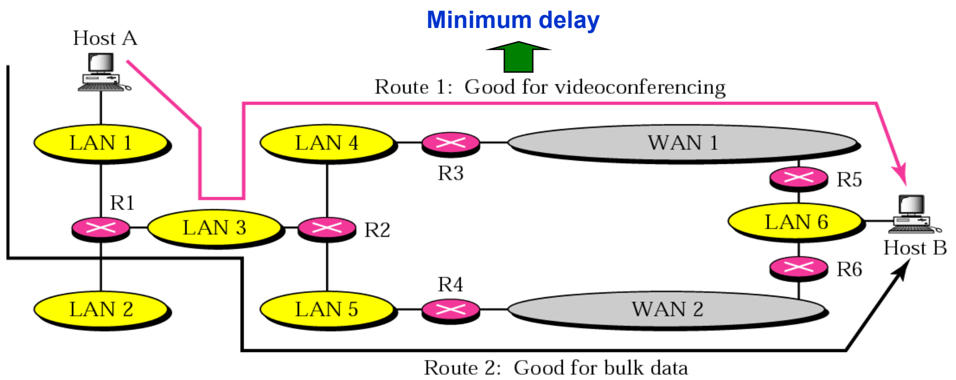
Routing



Which is the best path from A to B?

128

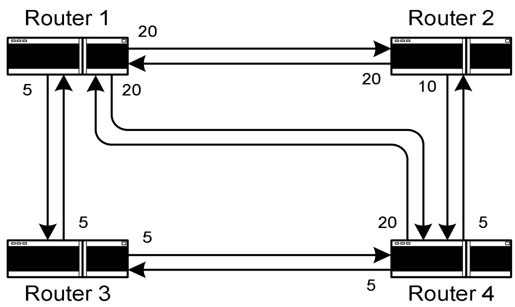
Routing

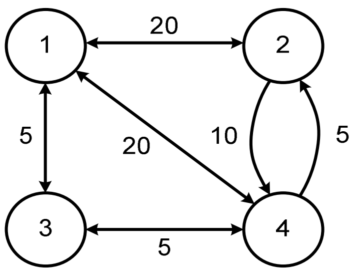


Best path depends on the type of service.

For instance, minimal delay possible vs. reliability.

Representation of Networks Through Graphs





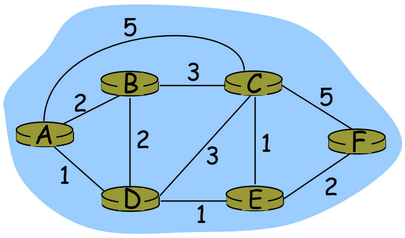
$G = (N, L)$

$N = \{1,2,3,4\}$

$L = \{(1,2), (2,1), (1,3), (3,1), (1,4), (4,1), (2,4), (4,2), (3,4), (4,3)\}$

TÉCNICO LISBOA

Graph Abstraction: Costs



- **c(x,x')** = cost of link (x,x')
  - e.g., c(C,F) = 5
- Cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

Cost of path (x_1, x_2, x_3,…, x_p) = c(x_1,x_2) + c(x_2,x_3) + … + c(x_{p-1},x_p)

Question: What's the least-cost path between A and F ?

Routing algorithm: algorithm that finds least-cost path.

TÉCNICO LISBOA

Routing Algorithm Classification

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

How fast do routes change?

static: routes change slowly over time

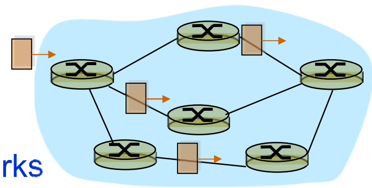
dynamic*:* routes change more quickly
- periodic updates or in response to link cost changes

decentralized: iterative process of computation, exchange of info with neighbors
- routers initially only know link costs to attached neighbors
- "distance vector" algorithms

global or decentralized information?

TÉCNICO LISBOA

Outline



- Introduction
- Virtual circuit and datagram networks
- IPv4 addressing and forwarding tables
- Internet Protocol (IP) - Datagram format, Fragmentation
- ICMP, DHCP
- NAT, IPv6
- Routing algorithms
  - Link state, Distance Vector
- Routing in the Internet
  - Hierarchical routing, RIP, OSPF, BGP
- Broadcast and multicast routing

TÉCNICO LISBOA

A Link-State Routing Algorithm

Dijkstra's algorithm

- Net topology and link costs known to all nodes:
  - Accomplished via "link state packet (**LSP**) broadcast".
  - All nodes have the same information.
- Computes least cost paths from one node ("source") to all other nodes:
  - Obtains **forwarding table** for that node.
- Iterative: after k iterations, know least cost path to k destinations.

Notation:

c(x,y): Link cost from node x to y; = ∞ if not direct neighbors.

D(v): Current value of cost of path from source to destination V.

p(v): Predecessor node along path from source to V.

N': Set of nodes whose least cost path definitively known.

Dijsktra's Algorithm

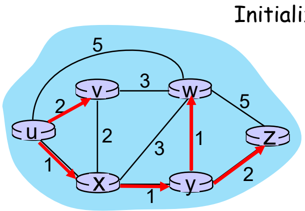
```
Initialization (computing costs for node A):
  N' = {A}
  for all nodes n
    if n adjacent to A
      then D(n) = c(A,n)
    else D(n) = ∞

Loop
   find m not in N' such that D(m) is a minimum
    add m to N'
   update D(n) for all n adjacent to m and not in N' :
      D(n) = min( D(n), D(m) + c(m,n) )
   /* new cost to n is either old cost to n or known
    shortest path cost to m plus cost from m to n */
until all nodes in N'
```


Dijkstra's Algorithm: an example

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

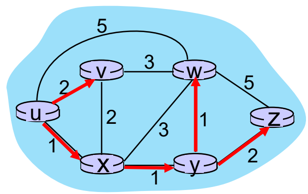
Initialization (step 0): For all *a*: if *a* adjacent to u then $D(a) = c_{u,a}$



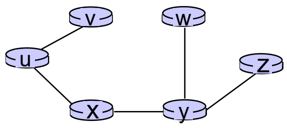
find *a* not in *N'* such that *D(a)* is a minimum
add *a* to *N'*
update *D(b)* for all *b* adjacent to *a* and not in *N'* :
$D(b) = \min (D(b), D(a) + c_{a,b})$

TÉCNICO LISBOA

Dijkstra's Algorithm: an example



Resulting least-cost-path tree from u:



Resulting forwarding table in u:

destination	outgoing link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

route from *u* to *v* directly

route from u to all other destinations via *x*

TÉCNICO LISBOA

Dijkstra's Algorithm: Discussion

Algorithm complexity: *n* nodes

- each of *n* iteration: need to check all nodes, *w*, not in *N*
- $n(n+1)/2$ comparisons: $O(n^2)$ complexity
- more efficient implementations possible: $O(n\log n)$

Message complexity:

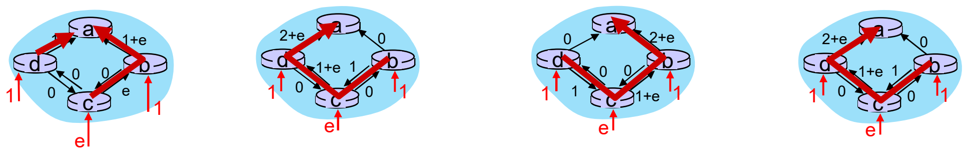
- each router must *broadcast* its link state information to other *n* routers
- efficient (and interesting!) broadcast algorithms: $O(n)$ link crossings to disseminate a broadcast message from one source
- each router's message crosses $O(n)$ links: overall message complexity: $O(n^2)$

TÉCNICO LISBOA

Dijkstra's Algorithm: Oscillations Possible

TPC: Prob. 10

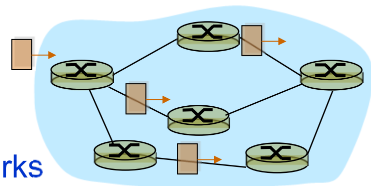
- When link costs depend on traffic volume, route oscillations possible
- Sample scenario:
  - routing to destination a, traffic entering at d, c, e with rates 1, e (<1), 1
  - link costs are directional, and volume-dependent



initially | given these costs, find new routing…. resulting in new costs | given these costs, find new routing…. resulting in new costs | given these costs, find new routing…. resulting in new costs

TÉCNICO LISBOA

Outline



- Introduction
- Virtual circuit and datagram networks
- IPv4 addressing and forwarding tables
- Internet Protocol (IP) - Datagram format, Fragmentation
- ICMP, DHCP
- NAT, IPv6
- What's inside a router
- Routing algorithms
  - Link state, Distance Vector
- Routing in the Internet
  - Hierarchical routing, RIP, OSPF, BGP
- Broadcast and multicast routing

Distance Vector Algorithm

Based on *Bellman-Ford* (BF) equation (dynamic programming):

Bellman-Ford equation

Let $D_x(y)$: cost of least-cost path from x to y.

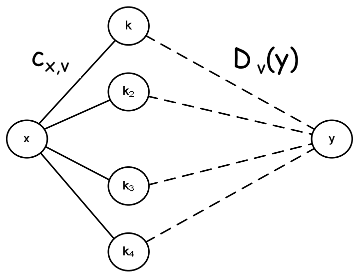
Then:

$$D_x(y) = \min_v \{ c_{x,v} + D_v(y) \}$$

min taken over all neighbors v of x

direct cost of link from x to v

v's estimated least-cost-path cost to y



Distance Vector Algorithm

Basic idea:

- From time-to-time, each node sends its own distance vector (DV) estimate to its neighbors.
- Asynchronous distribution of distance vector estimates.
- When a node x receives new DV estimate from a neighbor, it updates its own DV using the Bellman-Ford equation:

$$D_x(y) \leftarrow \min_v\{c(x,k) + D_k(y)\} \quad \text{for each node } y \in N$$

- Under natural conditions, with minor changes, the estimate $D_x(y)$ *converges to the actual least costs:* $d_x(y)$.

Distance Vector Algorithm

Iterative, asynchronous

Local iterations are caused by:

- Local link cost change;
- DV update message from neighbor.

Distributed

- Each node notifies neighbors *only* when its DV changes:
  - Neighbors then notify their neighbors, if necessary.

Each node:

wait for (change in local link cost or message from neighbor);

recompute estimates;

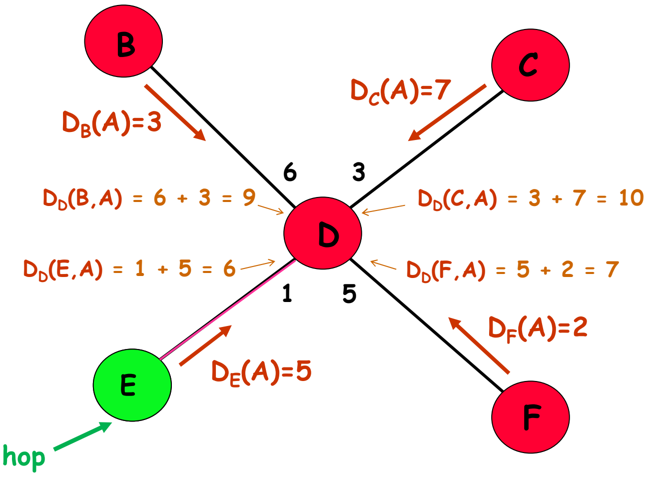
If DV to any destination has changed, *notify* neighbors.

RC – Prof. Paulo Lobato Correia 148

148

From centralized to distributed routing

Best path from D to A?



Selected next hop

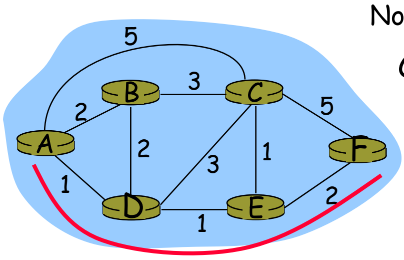
$D_D(A) = 6$

$D_D(k,A) = c(D,k) + Dk(A)$ – distance estimate from D to A, via neighbor k

RC – Prof. Paulo Lobato Correia 150

150

Bellman-Ford Example



Node A - $D_A(F)$:

Clearly, $D_B(F) = 5$, $D_D(F) = 3$, $D_C(F) = 3$

Bellman-Ford equation says:

$$D_A(F) = \min \{ c(A,B) + D_B(F), \; c(A,D) + D_D(F), \; c(A,C) + D_C(F) \}$$
$$= \min \{2 + 5, \; 1 + 3, \; 5 + 3\} = 4$$

Node that achieves minimum is next hop in shortest path → forwarding table

Node A:

Destination	Next router
F	D

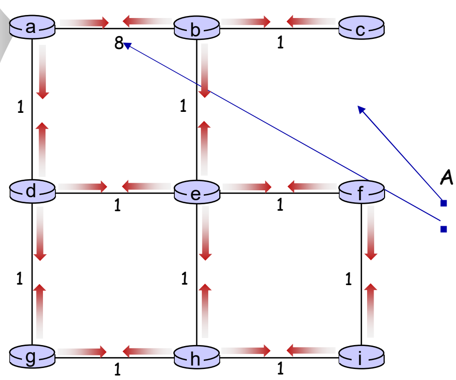
Distance vector: example

t=0

- All nodes have distance estimates to nearest neighbors (only)
- All nodes send their local distance vector to their neighbors

DV in

$D_a(a)=0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$

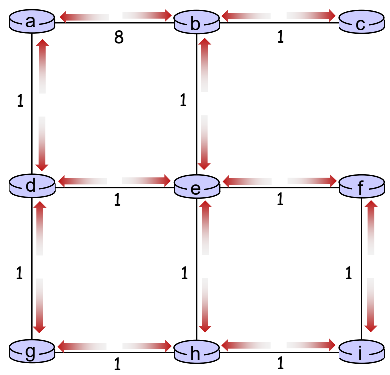


A few asymmetries:
- missing link
- larger cost

Distance vector example: iteration

t=1

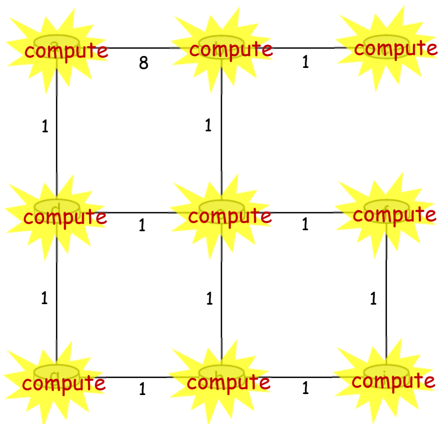
All nodes:
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



Distance vector example: iteration

t=1

All nodes:
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



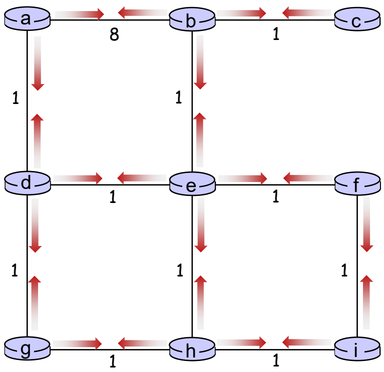
TÉCNICO LISBOA

Distance vector example: iteration

t=1

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



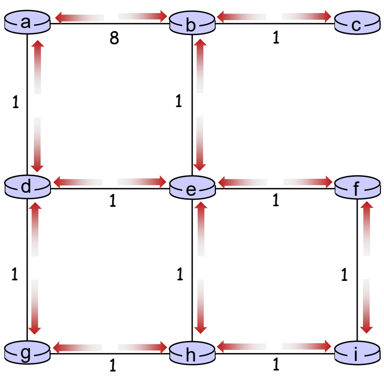
TÉCNICO LISBOA

Distance vector example: iteration

t=2

All nodes:

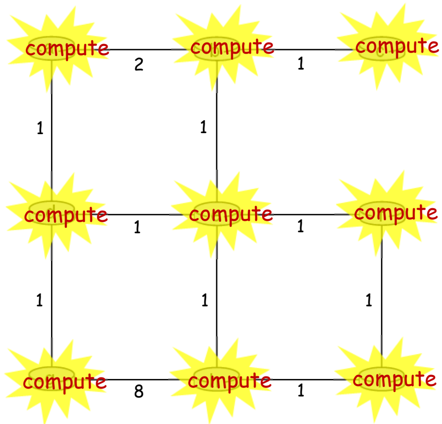
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



Distance vector example: iteration

t=2

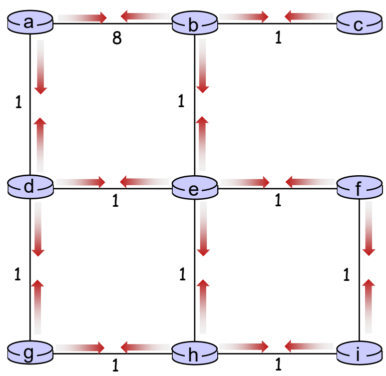
All nodes:
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



Distance vector example: iteration

t=2

All nodes:
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



Distance vector example: *iteration*

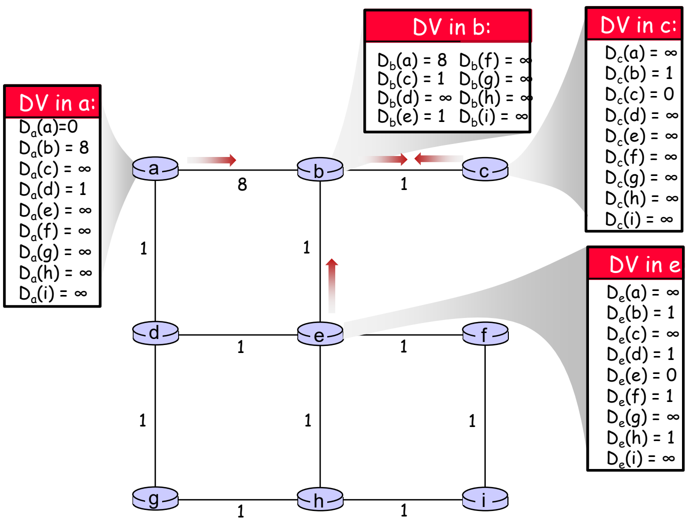
…. and so on

Let's next take a look at the iterative *computations* at nodes

Distance vector example: computation

t=1

- c receives DVs from b



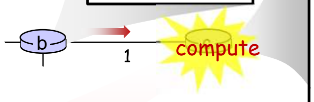
Distance vector example: computation

t=1

- c receives DVs from b computes:

DV in b:

$D_b(a) = 8$ $D_b(f) = \infty$
$D_b(c) = 1$ $D_b(g) = \infty$
$D_b(d) = \infty$ $D_b(h) = \infty$
$D_b(e) = 1$ $D_b(i) = \infty$



DV in c:

$D_c(a) = \infty$
$D_c(b) = 1$
$D_c(c) = 0$
$D_c(d) = \infty$
$D_c(e) = \infty$
$D_c(f) = \infty$
$D_c(g) = \infty$
$D_c(h) = \infty$
$D_c(i) = \infty$

$D_c(a) = \min\{c_{c,b}+D_b(a)\} = 1 + 8 = 9$
$D_c(b) = \min\{c_{c,b}+D_b(b)\} = 1 + 0 = 1$
$D_c(d) = \min\{c_{c,b}+D_b(d)\} = 1+ \infty = \infty$
$D_c(e) = \min\{c_{c,b}+D_b(e)\} = 1 + 1 = 2$
$D_c(f) = \min\{c_{c,b}+D_b(f)\} = 1+ \infty = \infty$
$D_c(g) = \min\{c_{c,b}+D_b(g)\} = 1+ \infty = \infty$
$D_c(h) = \min\{c_{bc,b}+D_b(h)\} = 1+ \infty = \infty$
$D_c(i) = \min\{c_{c,b}+D_b(i)\} = 1+ \infty = \infty$

DV in c:

$D_c(a) = 9$
$D_c(b) = 1$
$D_c(c) = 0$
$D_c(d) = 2$
$D_c(e) = \infty$
$D_c(f) = \infty$
$D_c(g) = \infty$
$D_c(h) = \infty$
$D_c(i) = \infty$

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

161

Distance vector: state information diffusion

Iterative communication, computation steps diffuses information through network:

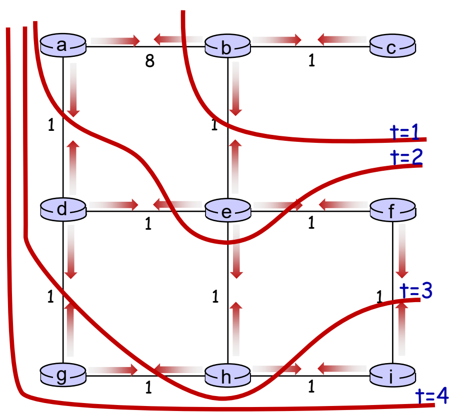
t=0 c's state at t=0 is at c only

t=1 c's state at t=0 has propagated to b, and may influence distance vector computations up to **1** hop away, i.e., at b

t=2 c's state at t=0 may now influence distance vector computations up to **2** hops away, i.e., at b and now at a, e as well

t=3 c's state at t=0 may influence distance vector computations up to **3** hops away, i.e., at b,a,e and now at c,f,h as well

t=4 c's state at t=0 may influence distance vector computations up to **4** hops away, i.e., at b,a,e, c, f, h and now at g,i as well

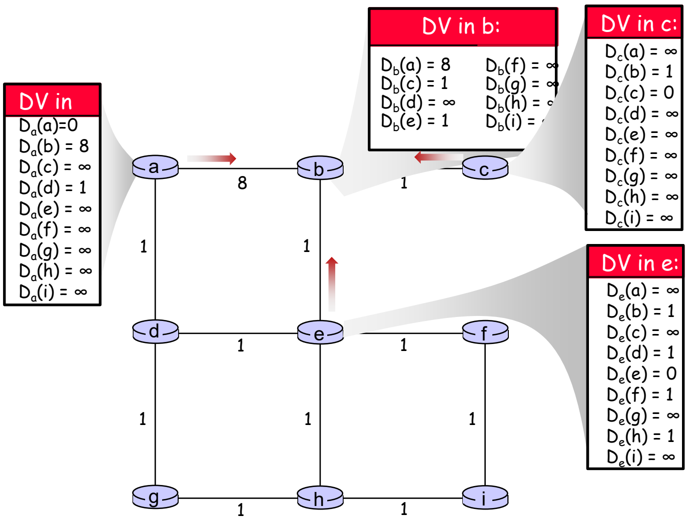


162

Distance vector example: computation

t=1

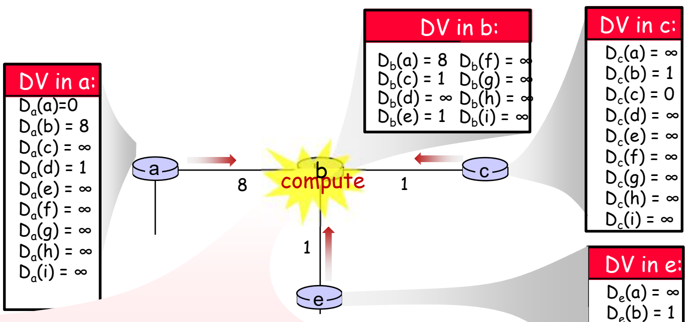
- b receives DVs from a, c, e



Distance vector example: computation

t=1

- b receives DVs from a, c, e, computes:



$D_b(a) = \min\{c_{b,a}+D_a(a),\ c_{b,c}+D_c(a),\ c_{b,e}+D_e(a)\} = \min\{8,\infty,\infty\} = 8$

$D_b(c) = \min\{c_{b,a}+D_a(c),\ c_{b,c}+D_c(c),\ c_{b,e}+D_e(c)\} = \min\{\infty,1,\infty\} = 1$

$D_b(d) = \min\{c_{b,a}+D_a(d),\ c_{b,c}+D_c(d),\ c_{b,e}+D_e(d)\} = \min\{9,2,\infty\} = 2$

$D_b(e) = \min\{c_{b,a}+D_a(e),\ c_{b,c}+D_c(e),\ c_{b,e}+D_e(e)\} = \min\{\infty,\infty,1\} = 1$

$D_b(f) = \min\{c_{b,a}+D_a(f),\ c_{b,c}+D_c(f),\ c_{b,e}+D_e(f)\} = \min\{\infty,\infty,2\} = 2$

$D_b(g) = \min\{c_{b,a}+D_a(g),\ c_{b,c}+D_c(g),\ c_{b,e}+D_e(g)\} = \min\{\infty,\infty,\infty\} = \infty$

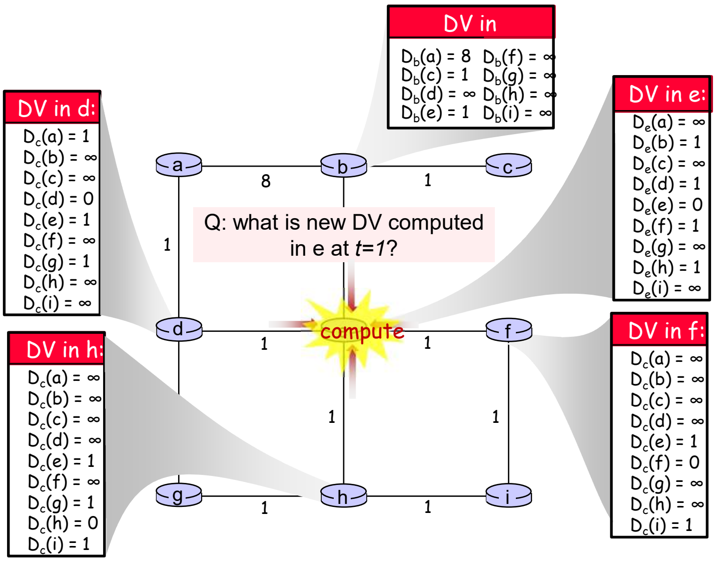
$D_b(h) = \min\{c_{b,a}+D_a(h),\ c_{b,c}+D_c(h),\ c_{b,e}+D_e(h)\} = \min\{\infty,\infty,2\} = 2$

$D_b(i) = \min\{c_{b,a}+D_a(i),\ c_{b,c}+D_c(i),\ c_{b,e}+D_e(i)\} = \min\{\infty,\infty,\infty\} = \infty$

Distance vector example: computation

t=1

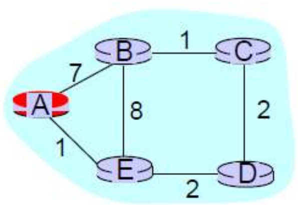
- e receives DVs from b, d, f, h



165

Distance Table: Example

Distances to Node A



neighbor

B

d[,]	A	E	C
A	7	∞	∞

d[,]	A	E	C
A	7	9	∞

C

d[,]	B	D
A	∞	∞

d[,]	B	D
A	8	∞

D

d[,]	C	E
A	∞	∞

d[,]	C	E
A	∞	3

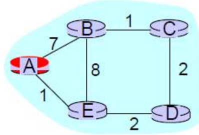
E

d[,]	A	B	D
A	1	∞	∞

d[,]	A	B	D
A	1	15	∞

170

Distance Table: Example



Distances to Node A

B

d[,]	A	E	C
A	7	∞	∞
A	7	9	∞
A	7	9	9
A	7	9	6

C

d[,]	B	D
A	∞	∞
A	8	∞
A	8	5
A	8	5

D

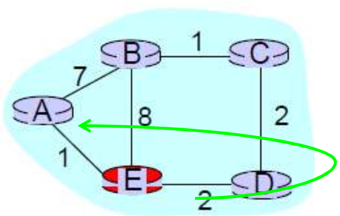
d[,]	C	E
A	∞	∞
A	∞	3
A	10	3
A	7	3

E

d[,]	A	B	D
A	1	∞	∞
A	1	15	∞
A	1	15	5
A	1	15	5

171

Distance Table: Example



Distance Table for Node E

neighbor

$dtab_E$[,]	A	B	D	
A	1	14	5	Loop
B	7	8	5	
C	6	9	4	
D	4	11	2	

destination

$dtab_E[D,C]$ = c[E,D] + 2 = 4

$\boxed{dtab_E[D,A] = c[E,D] + 3 = 5}$ Loop !

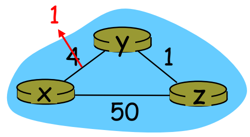
$dtab_E[B,A]$ = c[E,B] + 6 = 14 Loop !

$dtab_E[k,Y]$ - distance from E to Y, going through k

172

TÉCNICO LISBOA

Distance Vector: Link Cost Changes



Link cost changes:

- Node detects local link cost change;
- Updates routing info, recalculates distance vector;
- If DV changes, notify neighbors.

"good news travel fast"

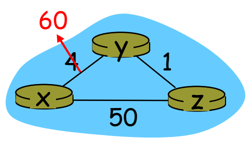
At time t_0, *y* detects the link-cost change, updates its DV, and informs its neighbors.

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

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

TÉCNICO LISBOA

Distance Vector: Link Cost Changes



Link cost changes:

- Good news travel fast.
- Bad news travel slow – "**count to infinity**" problem!
- 44 iterations before algorithm stabilizes…

y→x	z→x
4(x)	5(y)
6(z)	5(y)
6(z)	7(y)
8(z)	7(y)
8(z)	9(y)
…	…

Solution → Poisoned reverse:

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

Comparison of LS and DV Algorithms

TPC: Prob. 11

Message complexity:

LS: With n nodes, E links, O(nE) messages sent.

DV: Exchange between neighbors only – Convergence time varies.

Speed of Convergence:

LS: algorithm requires O(nE) messages.

- May have oscillations.

DV: Convergence time varies:

- May temporarily have routing loops;
- Count-to-infinity problem.

Robustness: what happens if router malfunctions?

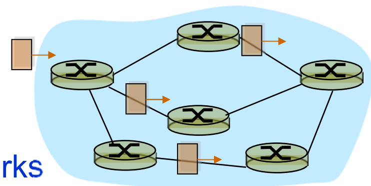
LS:

- Node can advertise incorrect *link* cost.
- Each node computes only its *own* table.

DV:

- DV node can advertise incorrect *path* cost.
- Each node's table used by others – errors propagate through network.

Outline



- Introduction
- Virtual circuit and datagram networks
- IPv4 addressing and forwarding tables
- Internet Protocol (IP) - Datagram format, Fragmentation
- ICMP, DHCP
- NAT, IPv6
- Routing algorithms
  - Link state, Distance Vector
- Routing in the Internet
  - Hierarchical routing, RIP, OSPF, BGP
- Broadcast and multicast routing

Hierarchical Routing

The routing study so far was based on an idealization:

- All routers assumed identical;
- "Flat" network;

… *not* true in practice

Scale – with >18 000 million destinations:

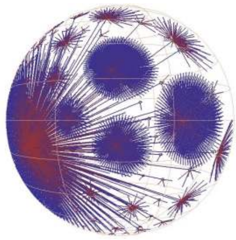
[http://blogs.cisco.com/news/cisco-connections-counter]

- **Can't store all destinations in routing tables!**
- **Routing table exchange would swamp links!**

Solution: Administrative autonomy

- *Internet = network of networks*;
- **Each network administration may want to control routing in its own network**.

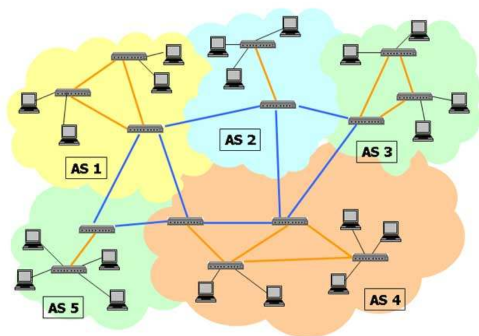
Hierarchical Routing



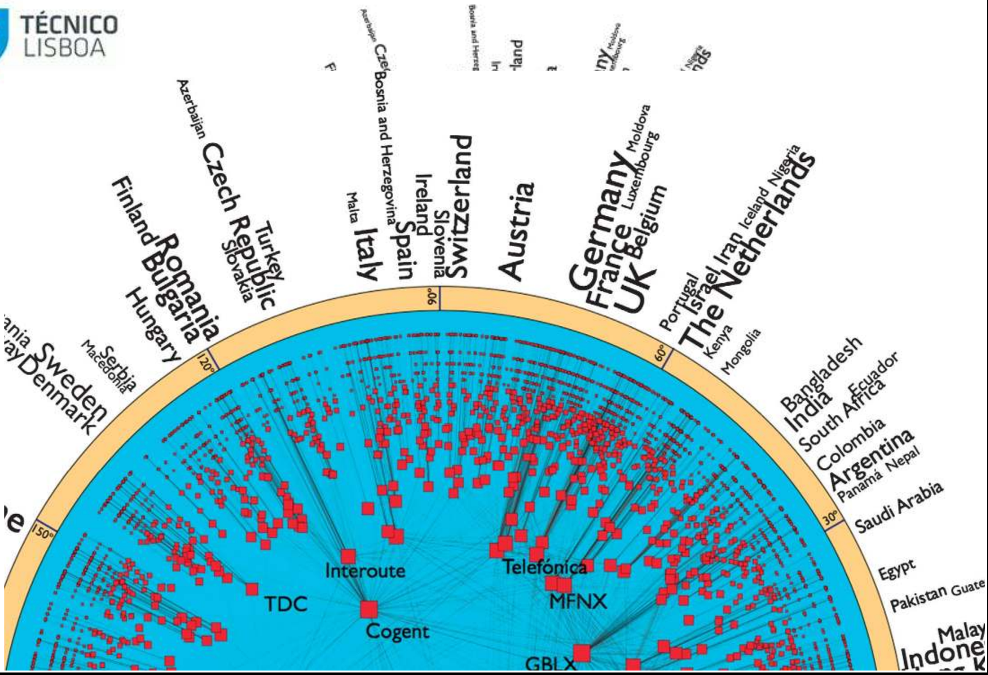
- Aggregate routers into regions:
  "**Autonomous systems**" (AS)
- Routers in same AS run the same routing protocol:
  - "**Intra-AS**" **routing** protocol;
  - Routers in different AS can run different intra-AS routing protocols.

Gateway router:

- Direct links to routers in other ASs.



TÉCNICO LISBOA



179

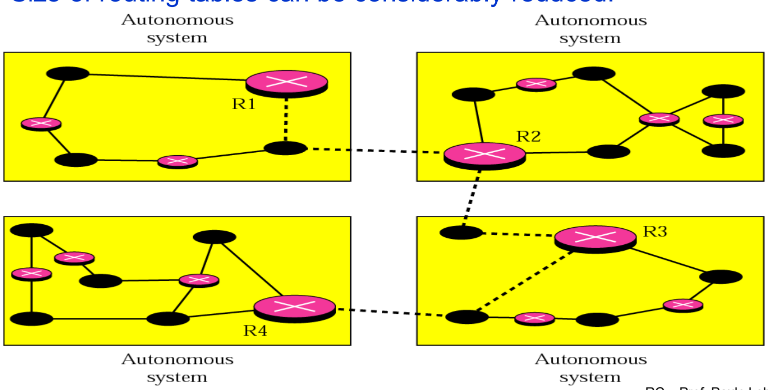
TÉCNICO LISBOA

Hierarchical Routing

Hierarchical routing:

- Delivery is first done to the **AS**;
- Then, to the **destination network**;
- Finally packets are delivered to the **destination host** (*using data link layer*).

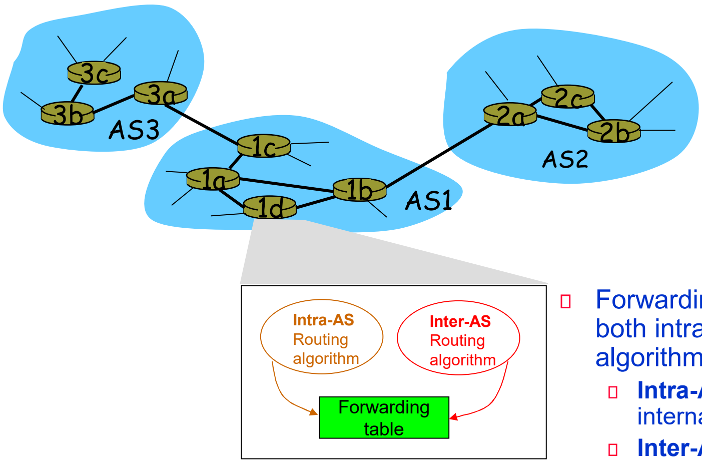
Size of routing tables can be considerably reduced!



RC – Prof. Paulo Lobato Correia 180

180

Interconnected ASes



- Forwarding table configured by both intra- and inter-AS routing algorithms:
  - **Intra-AS** sets entries for internal destinations.
  - **Inter-AS** & **intra-AS** set entries for external destinations.

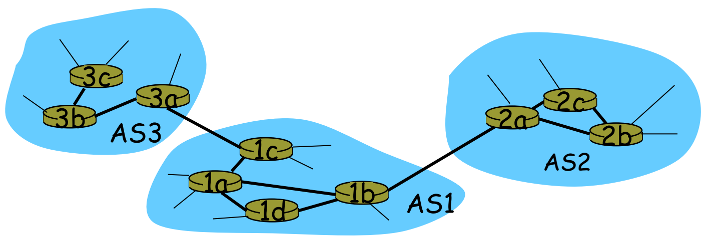
Inter-AS Tasks

- AS1 router receives datagram with destination outside of AS1:
  - **Forward** packet **to which gateway router?**

AS1 must:

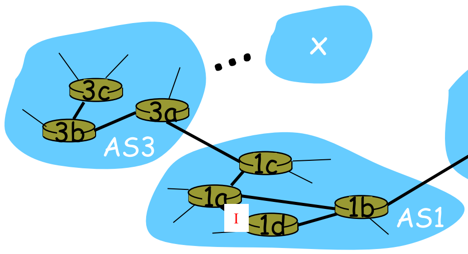
1. Learn which destinations are reachable through AS2 and which through AS3;
2. **Propagate** this **reachability information to all AS1 routers**.

Job of **inter-AS routing**!



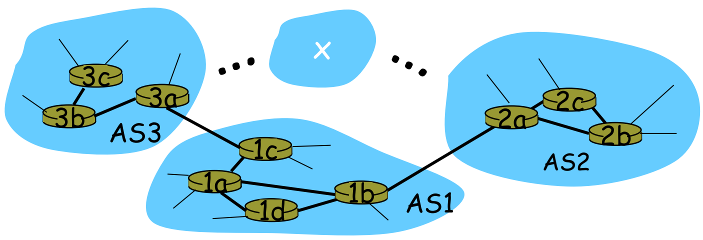
Setting Forwarding Table in Router `1d`

- AS1 learns (via inter-AS protocol) that **subnet *x* is reachable via AS3 (gateway 1c)** but not via AS2.
- **Inter-AS** protocol **propagates reachability information** to all internal routers.
- Router 1d uses intra-AS routing information to find the least cost path to 1c, via its interface *I* ;
  - Router 1d installs in its forwarding table the entry: *(x,I)*.



Choosing among Multiple ASes

- If AS1 learns (inter-AS protocol) that subnet *x* is reachable from AS3 *and* from AS2.
- To configure forwarding table, router 1d must determine towards which gateway it should forward packets for destination x.



TÉCNICO LISBOA

Choosing among Multiple ASes

- If AS1 learns (inter-AS protocol) that subnet *x* is reachable from AS3 *and* from AS2.
- To configure forwarding table, router 1d must determine towards which gateway it should forward packets for destination x.

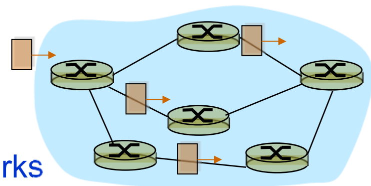
- **Hot potato routing**: send packet towards closest (lowest cost) of two gateway routers.

From inter-AS protocol learn that **subnet x** is **reachable via multiple gateways** → **Use intra-AS routing information to determine costs to each of the gateways** → **Hot potato routing: Choose the gateway that has the smallest cost** → **Determine from forwarding table the interface I that leads to least-cost gateway. Enter (x,I) in forwarding table**

185

TÉCNICO LISBOA

Outline



- Introduction
- Virtual circuit and datagram networks
- IPv4 addressing and forwarding tables
- Internet Protocol (IP) - Datagram format, Fragmentation
- ICMP, DHCP
- NAT, IPv6
- Routing algorithms
  - Link state, Distance Vector
- Routing in the Internet
  - Hierarchical routing, RIP, OSPF, BGP
- Broadcast and multicast routing

186

TÉCNICO LISBOA

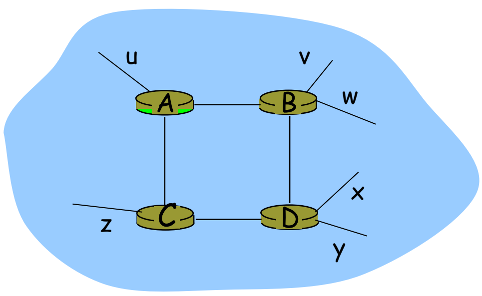
Intra-AS Routing

- Also known as Interior Gateway Protocols (IGP).
- Most common Intra-AS routing protocols:
  - **RIP**: Routing Information Protocol;
  - **OSPF**: Open Shortest Path First;
  - **IGRP**: Interior Gateway Routing Protocol (Cisco proprietary).

TÉCNICO LISBOA

RIP (Routing Information Protocol)

- Uses a **distance vector algorithm**;
- Distance metric: number of hops (***max distance = 15 hops, 16=∞***);



From router A to subnets:

destination	hops
u	1
v	2
w	2
x	3
y	3
z	2

RIP Advertisements

- ***Distance vectors:***
  exchanged among neighbors (**every 30 sec**,
  via ‘Response Message’, also called advertisement);

- Each advertisement:
  list of up to 25 destination subnets within the AS.

RIP: Link Failure and Recovery

If no advertisement heard after 180 sec → neighbor/link declared dead:

- Routes via neighbor invalidated;
- New advertisements sent to remaining neighbors;
- Neighbors in turn send out new advertisements (if tables changed);
- Link failure info quickly propagates to entire net;

Poisoned reverse used to prevent *ping-pong* loops.

In RIP: infinite distance = 16 hops.

OSPF (Open Shortest Path First)

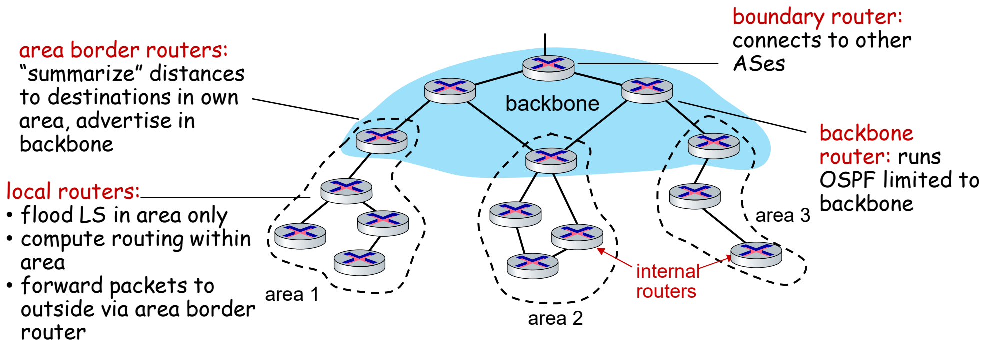
- "Open": publicly available;
- Uses **Link State** algorithm:
  - Link state packet (LSP) dissemination;
  - Topology map at each node;
  - Route computation using Dijkstra's algorithm.
- OSPF advertisement carries one entry per neighbor router.
- Advertisements disseminated to entire AS (via flooding):
  - Carried in **OSPF messages directly over IP** (rather than TCP or UDP). Process executing OSPF: **`gated`**.

OSPF "Advanced" Features (not available in RIP)

- Security: all OSPF messages authenticated (to prevent malicious intrusion);
- Multiple same-cost paths allowed (only one path in RIP);
- For each link, multiple cost metrics for different TOS (e.g., satellite link cost set "low" for best effort; "high" for real time);
- Integrated uni- and multicast support:
  - Multicast OSPF (MOSPF) uses same topology database as OSPF;
- Hierarchical OSPF in large domains.

Hierarchical OSPF

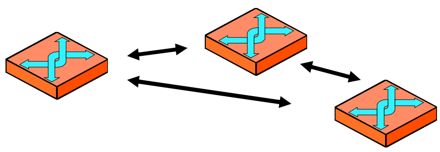
- Two-level hierarchy: local area, backbone.
  - link-state advertisements flooded only in area, or backbone
  - each node has detailed area topology; only knows direction to reach other destinations



RIP vs OSPF

The choice is done by the AS manager;

- RIP is appropriate for small networks:
  - Easy to implement;
  - 15 hops is not a problem;
  - Table diffusion (*even to hosts, interrupting them*) is not a big problem;
  - Distance vector.
- OSPF is scalable:
  - Works with networks of any dimension;
  - Link-state;
  - Management complexity is compensated by the better efficiency in larger networks.



Internet Inter-AS Routing: BGP

- BGP (Border Gateway Protocol): *the* de facto inter-domain routing protocol
  - "glue that holds the Internet together"

- Allows subnet to advertise its existence, and the destinations it can reach, to rest of Internet: *"I am here, here is who I can reach, and how"*

- BGP provides each AS a means to:
  - eBGP: obtain subnet reachability information from **neighboring ASes**
  - iBGP: propagate reachability information to all **AS-internal routers**.
  - determine "good" routes to other networks based on reachability information and *policy*

Path Attributes and BGP Routes

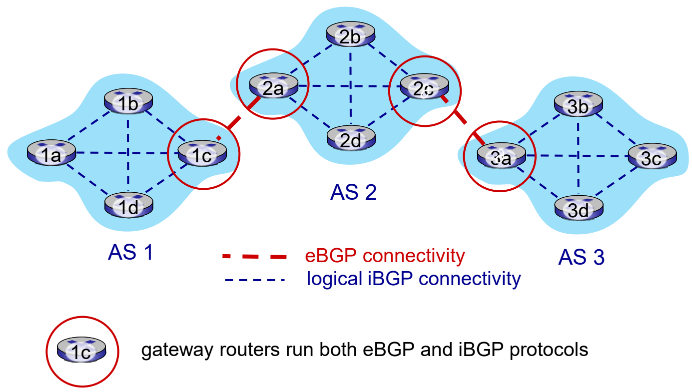
- BGP advertised route: prefix + attributes
  - prefix: destination being advertised
  - two important attributes:
    - AS-PATH: list of ASes through which prefix advertisement has passed
    - NEXT-HOP: indicates specific internal-AS router to next-hop AS

- Policy-based routing:
  - gateway receiving route advertisement uses ***import policy* to accept/decline path** (e.g., never route through AS Y).
  - AS policy also determines whether to *advertise* path to other other neighboring ASes

BGP Messages

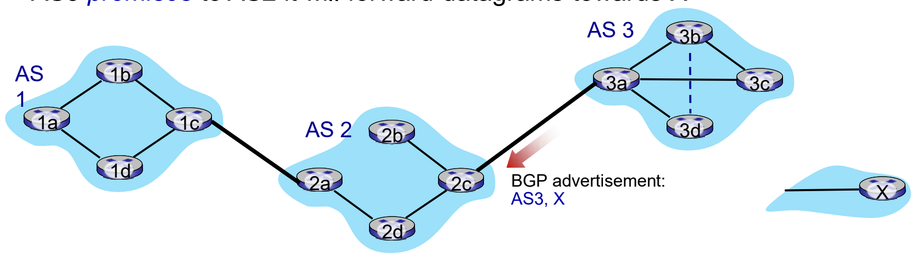
- BGP messages exchanged between peers over TCP connection
- BGP messages:
  - OPEN: opens TCP connection to remote BGP peer and authenticates sending BGP peer
  - UPDATE: advertises new path (or withdraws old)
  - KEEPALIVE: keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - NOTIFICATION: reports errors in previous msg; also used to close connection

eBGP, iBGP Connections

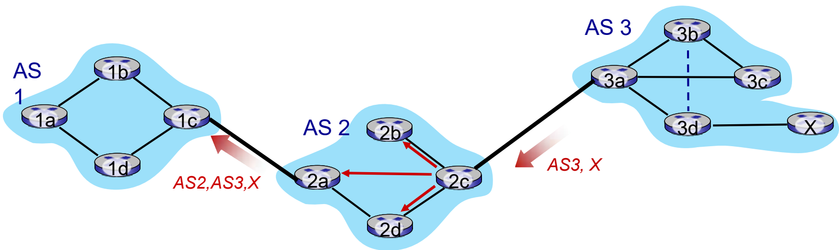


BGP Basics

- BGP session: two BGP routers ("peers") exchange BGP messages over semi-permanent TCP connection:
  - advertising *paths* to different destination network prefixes (BGP is a "path vector" protocol)
- when AS3 gateway 3a advertises path AS3,X to AS2 gateway 2c:
  - AS3 *promises* to AS2 it will forward datagrams towards X



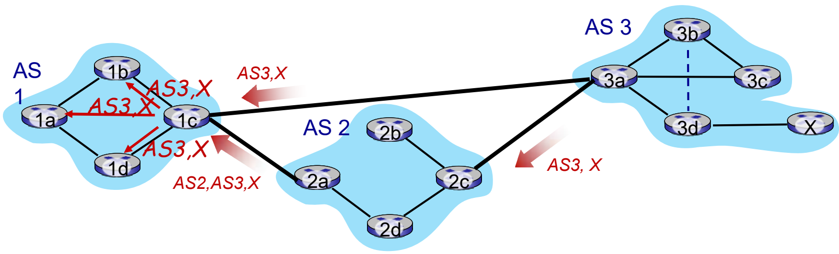
BGP Path Advertisement



- AS2 router 2c receives path advertisement AS3,X (via eBGP) from AS3 router 3a
- based on AS2 policy, AS2 router 2c accepts path AS3,X, propagates (via iBGP) to all AS2 routers
- based on AS2 policy, AS2 router 2a advertises (via eBGP) path AS2, AS3, X to AS1 router 1c

TÉCNICO LISBOA

BGP Path Advertisement (more)

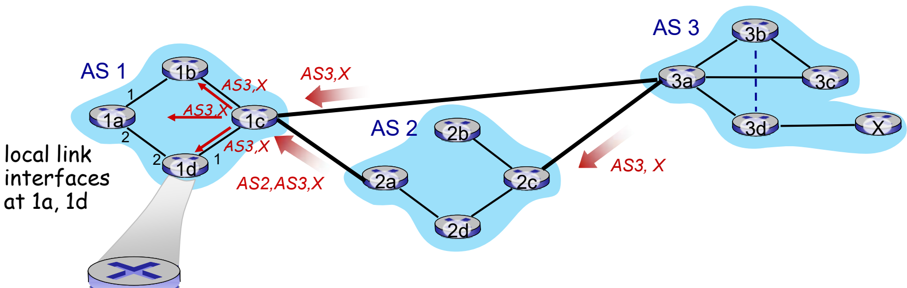


Gateway router may learn about multiple paths to destination:

- AS1 gateway router 1c learns path *AS2,AS3,X* from 2a
- AS1 gateway router 1c learns path *AS3,X* from 3a
- based on *policy,* AS1 gateway router 1c chooses path *AS3,X* and advertises path within AS1 via iBGP

TÉCNICO LISBOA

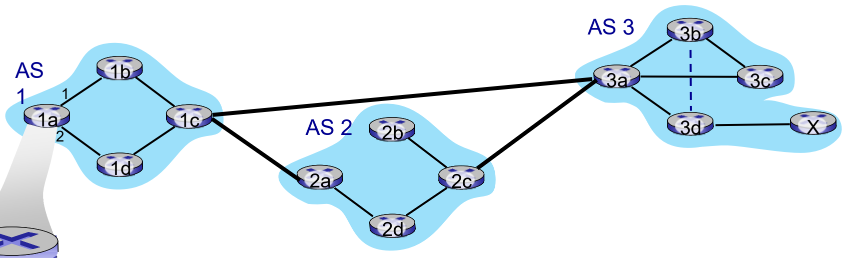
BGP Path Advertisement

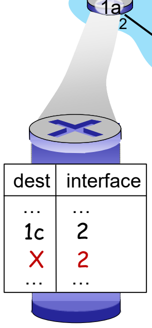


dest	interface
…	…
1c	1
X	1
…	…

- recall: 1a, 1b, 1d learn via iBGP from 1c: "path to X goes through 1c"
- at 1d: OSPF intra-domain routing: to get to 1c, use interface 1
- at 1d: to get to X, use interface 1

BGP Path Advertisement





- recall: 1a, 1b, 1d learn via iBGP from 1c: “path to X goes through 1c”
- at 1d: OSPF intra-domain routing: to get to 1c, use interface 1
- at 1d: to get to X, use interface 1
- at 1a: OSPF intra-domain routing: to get to 1c, use interface 2
- at 1a: to get to X, use interface 2

Why Different Intra-, Inter-AS Routing ?

Policy:

- inter-AS: admin wants control over how its traffic routed, who routes through its network
- intra-AS: single admin, so policy less of an issue

Scale:

- hierarchical routing saves table size, reduced update traffic

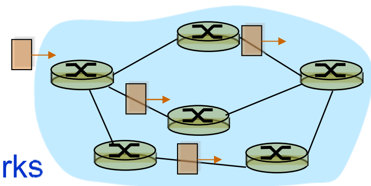
Performance:

- intra-AS: can focus on performance
- inter-AS: policy dominates over performance

BGP Route Selection

- Router may learn about more than one route to destination AS, selects route based on:
  1. local preference value attribute: policy decision
  2. shortest AS-PATH
  3. closest NEXT-HOP router: hot potato routing
  4. additional criteria

Outline

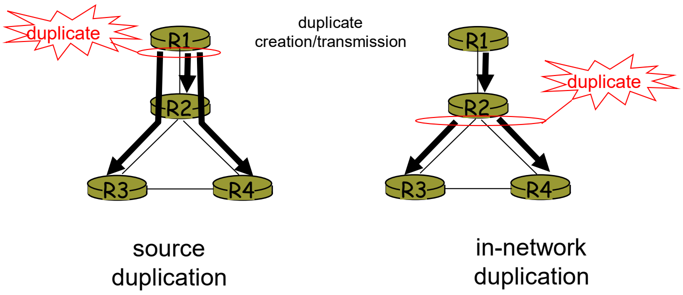


- Introduction
- Virtual circuit and datagram networks
- IPv4 addressing and forwarding tables
- Internet Protocol (IP) - Datagram format, Fragmentation
- ICMP, DHCP
- NAT, IPv6
- Routing algorithms
  - Link state, Distance Vector
- Routing in the Internet
  - Hierarchical routing, RIP, OSPF, BGP
- Broadcast and multicast routing

Broadcast Routing

- Deliver packets from source to all other nodes.

Source duplication is inefficient:

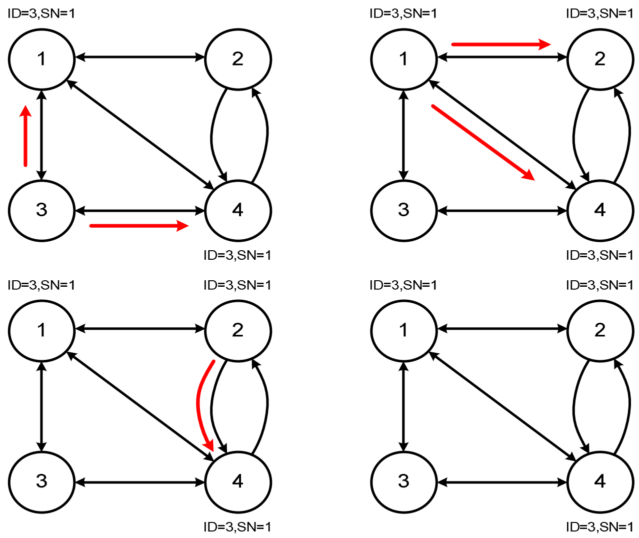


Source duplication:
How does source determine recipient addresses?

In-Network Duplication

- Flooding - when node receives broadcast packet, sends copy to all neighbors:
  - Problems: cycles & broadcast storm.
- Controlled flooding - node only broadcasts packet if it hasn't broadcast the same packet before:
  - Node keeps track of **packet IDs** already broadcasted;
  - Or, **reverse path forwarding** (RPF): only forward packet if it arrived on the shortest path between node and source.
- Spanning tree:
  - No redundant packets are received by any node.

Broadcast Routing



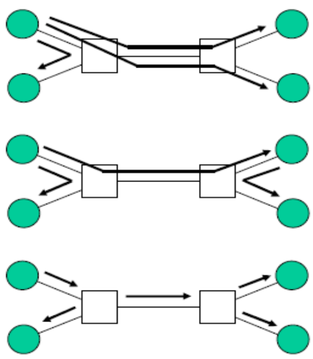
controlled flooding, broadcast of first message.

Multicast

TPC: Prob. 12

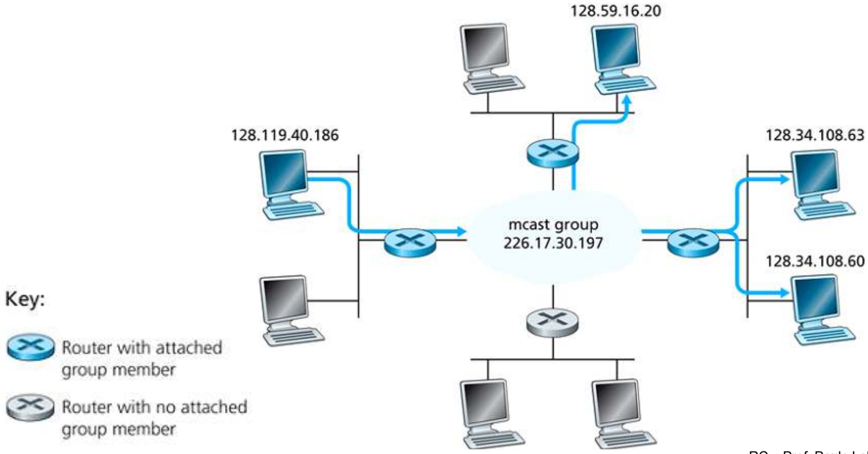
When a source application generates a message, a copy of that message should be delivered to each one of the destinations belonging to the *multicast* group.

- *Multicast* emulation:
  - Source establishes *unicast* sessions with each one of the destinations.
- *Multicast* in the application layer:
  - Hosts (end stations) build a *multicast* logical tree on top of the network.
- *Multicast* in the network layer:
  - Routers build *multicast* tree.



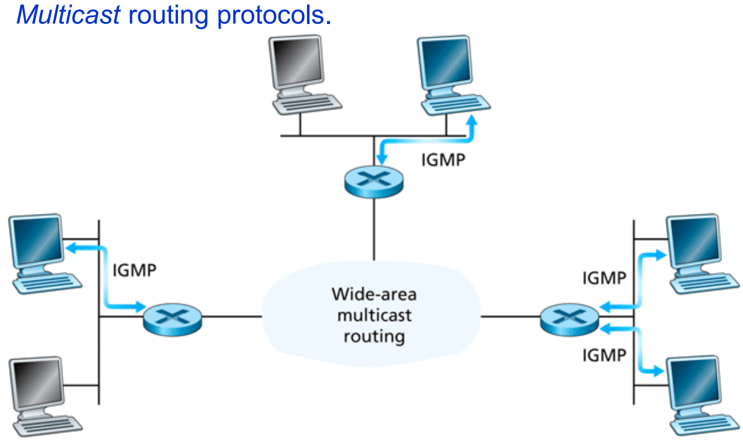
Multicast

Multicast group:
- A datagram addressed to the group is delivered to all members of the *multicast* group.



Multicast

The two components of network-layer multicast in the Internet:
- **Multicast Group Membership Discovery** protocol
  - Internet Group Management Protocol (**IGMP**) - IPv4;
  - Multicast Listener Discovery (**MLD**) - IPv6;
- *Multicast* routing protocols.



Internet Group Management Protocol (IGMP)

All destinations in a *multicast* group share the same IP address (class D).

- *Internet Group Management Protocol* (RFC 1112):
  - Operates between one host and the router to which it is directly connected;
  - Router wants to know, for each interface, which *multicast* groups have members connected to that interface;
  - Router invites hosts to indicate to which *multicast* groups they want to belong.

0 – 3	4 – 7	8 – 15	16 – 31
Version	Type	Unused	Checksum
Group Address (Class D IPv4 address)			

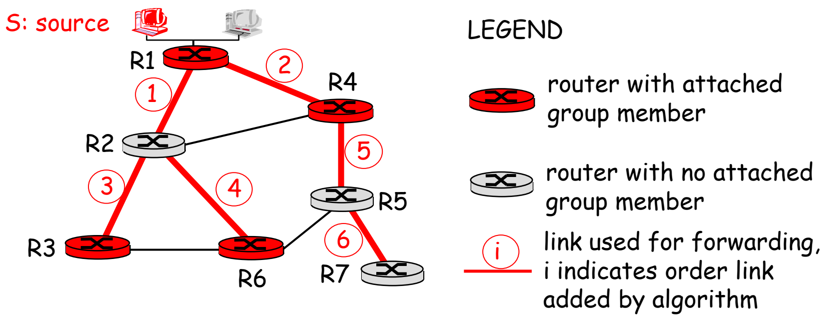
Approaches for Building Mcast Trees

Approaches:

- Source-based tree - one tree per source:
  - Shortest path trees (e.g., using Dijkstra);
  - Reverse path forwarding.
- Group-shared tree - group uses one tree:
  - Minimal spanning (Steiner) tree;
  - Center-based trees.

Shortest Path Tree

- Mcast forwarding tree - tree of shortest path routes from source to all receivers:
  - Dijkstra's algorithm.



Reverse Path Forwarding

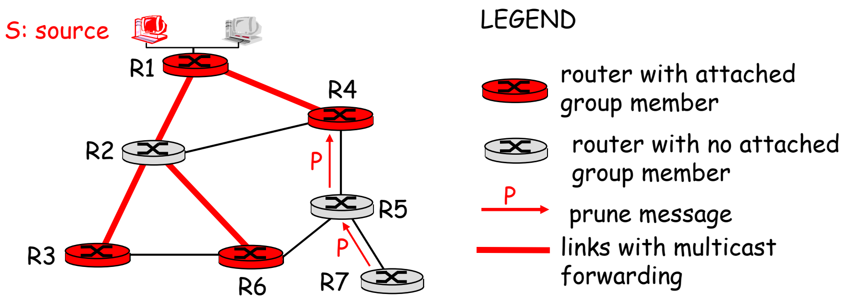
TPC: Prob. 13

- Rely on router's knowledge of unicast shortest path from it to sender;
- Each router has simple forwarding behavior:

> ***if*** (Mcast datagram received on incoming link on the shortest path back to center):
> ***then***
> flood datagram onto all outgoing links;
> ***else***
> ignore datagram.

Reverse Path Forwarding: Pruning

- Forwarding tree contains subtrees with no Mcast group members:
  - No need to forward datagrams down those subtrees;
  - "Prune" messages sent upstream by router with no downstream group members.

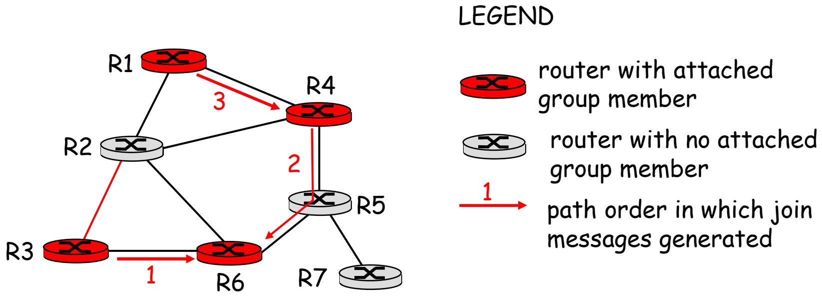


Center-based Trees

- Single delivery tree shared by all.
- One router identified as *"center"* of tree.
- To join:
  - Edge router sends unicast *join-msg* addressed to center router;
  - *Join-msg* "processed" by intermediate routers and forwarded towards center;
  - *Join-msg* either hits existing tree branch for this center, or arrives at center;
  - Path taken by *join-msg* becomes new branch of tree for this router.

Center-based Trees: an Example

- Suppose R6 chosen as center:



Internet Multicasting Routing: DVMRP

- DVMRP: distance vector multicast routing protocol (RFC1075).
- *Flood and prune:*
  Reverse path forwarding, source-based tree:
  - RPF tree based on DVMRP's own routing tables constructed by communicating DVMRP routers;
  - No assumptions about underlying unicast;
- Initial datagram to Mcast group is flooded everywhere via RPF;
- Routers not wanting group: send upstream prune messages.

DVMRP

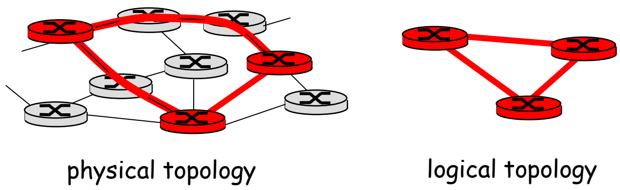
- *Soft state:* DVMRP router periodically (1 min.) "forgets" that some branches were pruned:
  - Mcast data flows down unpruned branch, again;
  - Downstream router: reprune or else continue to receive data.
- Routers can quickly regraft to tree:
  - Following IGMP join at leaf.
- Odds and ends:
  - Commonly implemented in commercial routers;
  - Mbone ("multicast backbone") routing done using DVMRP.

RC – Prof. Paulo Lobato Correia 237

237

Tunneling

Q: How to connect "islands" of multicast routers in a "sea" of unicast routers?



- Mcast datagram encapsulated inside "normal" (non-multicast-addressed) datagram;
- Normal IP datagram sent through "tunnel" via regular IP unicast to receiving Mcast router;
- Receiving Mcast router unencapsulates to get Mcast datagram.

RC – Prof. Paulo Lobato Correia 238

238

Protocol Independent Multicast (PIM)

- Not dependent on any specific underlying unicast routing algorithm (works with all);
- Two different multicast distribution scenarios:

Dense:

- Group members densely packed, in "close" proximity.
- Bandwidth more plentiful.

Sparse:

- Number of networks with group members is small compared to the number of interconnected networks;
- Group members are "widely dispersed";
- Bandwidth not plentiful.

PIM: Sparse-Dense Dichotomy

Dense:

- Group membership by routers *assumed* until routers explicitly prune;
- *Data-driven* construction of Mcast tree (e.g., RPF);
- Bandwidth and non-group-router processing *prodigal.*

Sparse:

- No membership until routers explicitly join;
- *Receiver- driven* construction of Mcast tree (e.g., center-based);
- Bandwidth and non-group-router processing *conservative.*

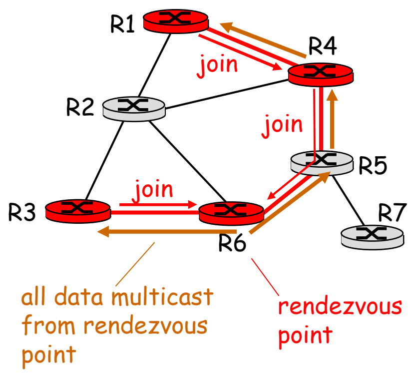
PIM- Dense Mode

Flood-and-prune RPF, similar to DVMRP but:

- Underlying unicast protocol provides RPF info for incoming datagram;
- Less complicated (more efficient) downstream flood than DVMRP reduces reliance on underlying routing algorithm;
- Has protocol mechanism for router to detect it is a leaf-node router.

PIM - Sparse Mode

- Center-based approach;
- Router sends *join* message to rendezvous point (RP):
  - Intermediate routers update state and forward *join;*
- After joining via RP, router can switch to source-specific tree:
  - Increased performance: less concentration, shorter paths.



PIM - Sparse Mode

Sender(s):

- Unicast data to RP, which distributes down RP-rooted tree;
- RP can extend Mcast tree upstream to source;
- RP can send *stop* message if no attached receivers:
  - “No one is listening!”