Chapter 4
Network Layer

Docente: Dott.ssa Chiara Petrioli

Università degli Studi di Roma "La Sapienza"

Computer Networking: A Top Down Approach Featuring the Internet, 2nd edition.
Jim Kurose, Keith Ross
Addison-Wesley, July 2002.

Chapter 4: Network Layer

Chapter goals:

- understand principles behind network layer services:
  - routing (path selection)
  - dealing with scale
  - how a router works
  - advanced topics: IPv6, mobility
- instantiation and implementation in the Internet

Overview:

- network layer services
- routing principles: path selection
- hierarchical routing
- IP
- Internet routing protocols reliable transfer
  - intra-domain
  - inter-domain
- what's inside a router?
- IPv6
- mobility

Chapter 4 roadmap

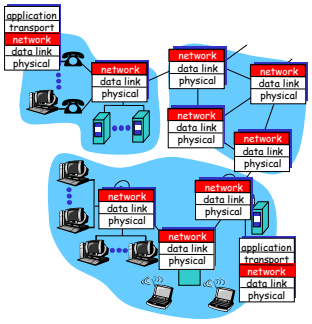
4.1 Introduction and Network Service Models
4.2 Routing Principles
4.3 Hierarchical Routing
4.4 The Internet (IP) Protocol
4.5 Routing in the Internet
4.6 What's Inside a Router
4.7 IPv6
4.8 Multicast Routing
4.9 Mobility

Network layer functions

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

three important functions:

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



Network service model

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

service abstraction

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

The most important abstraction provided by network layer:

virtual circuit or datagram?

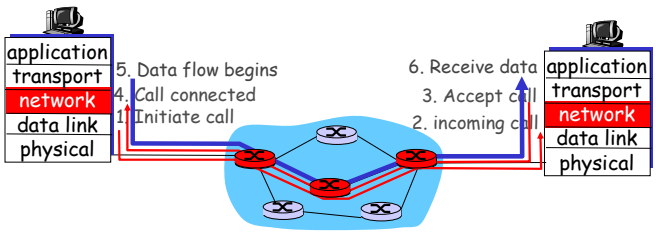
Virtual circuits

"source-to-dest path behaves much like telephone circuit"
- performance-wise
- network actions along source-to-dest path

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

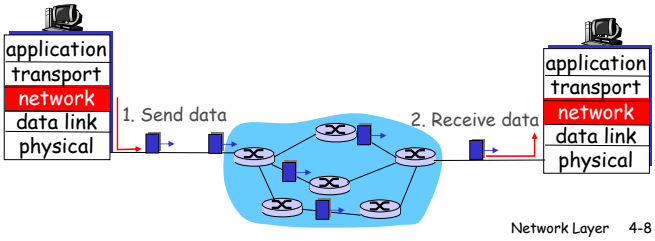
Virtual circuits: signaling protocols

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



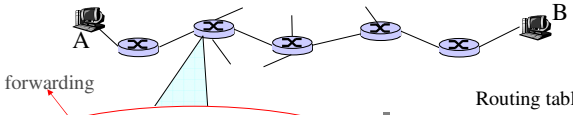
Datagram networks: the Internet model

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



Router

- Forward a chunk of information (called *packet*) arriving on one of its communication links to one of its outgoing communications link (the *next hop* on the source-to-destination path)



forwarding

-Receives the packet
-Based on a routing table and the destination address, computes the 'next hop' to the destination
-Forwards the packet to the next hop
-The process of computing and maintaining the routing table is called Routing

Routing table

Dest. Address	Next Hop

Network layer service models:

Network Architecture	Service Model	Guarantees ? Bandwidth	Loss	Order	Timing	Congestion feedback
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

- Internet model being extended: Intserv, Diffserv
  - Chapter 6 (qualcosa nella parte del corso su vocesuIP, 21 e 22 maggio)

Datagram or VC network: why?

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

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

VC vs. Datagram

Issue	Datagram	VC
Circuit setup	Not needed	Required
Addressing	Full source and destination address	only a short VC number
State info	None	Required VC table space
Routing	each packet independently	route chosen at setup same route followed by all the packets
Router failure	Low effect(packets lost in crash)	All VC passing through the router terminated
Congestion Control	Difficult	Easier

Chapter 4 roadmap

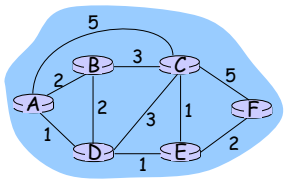
4.1 Introduction and Network Service Models
4.2 Routing Principles
- Link state routing
- Distance vector routing

4.3 Hierarchical Routing
4.4 The Internet (IP) Protocol
4.5 Routing in the Internet
4.6 What's Inside a Router
4.7 IPv6
4.8 Multicast Routing
4.9 Mobility

Routing

Routing protocol

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



Graph abstraction for routing algorithms:

- graph nodes are routers
- graph edges are physical links
  - link cost: delay, $ cost, or congestion level

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

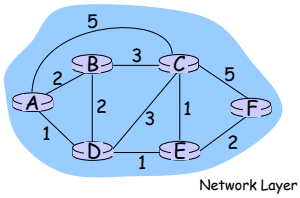
Summary on graphs…(1/3)

A graph G=(N,A) is made of a finite nonempty set of nodes N and a set of edges A. An edge is an unordered pair (i,j) with i and j in N.

A walk in a graph G is a sequence of nodes (n_1 n_2 … n_l) such that (n_i,n_{i+1}) is an edge of G. A walk with no repeated nodes is a path. A walk with $n_1 = n_l$ and no repeated nodes is called a cycle.

A graph is connected if for any pair of nodes i,j in N there is a path between them.

A weighted graph is a graph where each edge (i,j) has associated a weight $w_{i,j}$

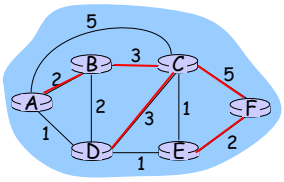


Summary on graphs…(2/3)

Given a graph G=(N,A) we say that a graph G'=(N',A') is a subgraph of G if G' is a graph, N'⊂ N, A'⊂A.

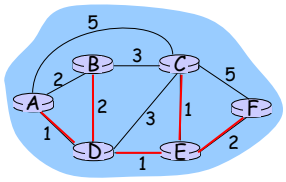
A tree is a connected acyclic graph. A spanning tree of a graph G is a subgraph of G which is a tree and includes all nodes in G.

In a spanning tree there are |N|-1 edges, and only one path between each pair of nodes.

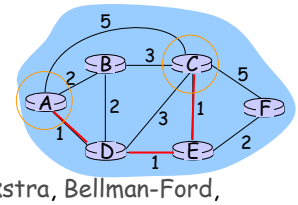


Summary on graphs…(3/3)

A minimum weight spanning tree is a spanning tree with minimum sum of arc weights.



Given two nodes i and j, the shortest path between i and j is a path such that the sum of arc weights is minimum.

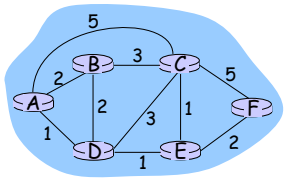


Shortest path algorithms: Dijkstra, Bellman-Ford, Floyd-Warshall

Routing

Routing protocol

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



Graph abstraction for routing algorithms:

- graph nodes are routers
- graph edges are physical links
  - link cost: delay, $ cost, or congestion level

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

Routing Algorithm classification

Global or decentralized information?

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

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

Static or dynamic?

Static:
- routes change slowly over time

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

A Link-State Routing Algorithm

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

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

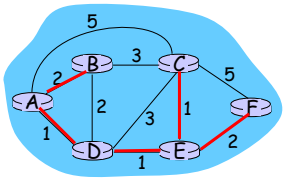
Algorithm assumption: No negative weights!!

Dijsktra's Algorithm

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


Dijkstra's algorithm: example (shortest paths from A to other nodes)

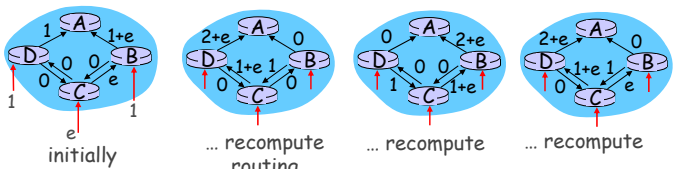
Step	start N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	infinity	infinity
1	AD	2,A	4,D		2,D	infinity
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
5	ADEBCF					



Dijkstra's algorithm, discussion

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

Oscillations possible:
- e.g., link cost = amount of carried traffic



Usually metric adopted in routing simply the number of hops

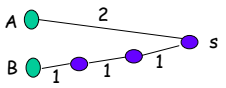
Dijsktra's Algorithm (complexity)

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

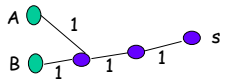
n

Shortest paths vs. Min. weight Spanning tree

- The shortest paths form a (spanning) tree
- Is it a minimum weight spanning tree? NO.
  - Example:



Shortest path tree



Minimum weight spanning tree

Why not a shortest path tree? s-A has length 3 instead of 2!!
But the overall sum of arc weights is 4 vs. 4 in the shortest path tree

Bellman-Ford

Given a graph G and a node *s* finds the shortest path from *s* to every node in N.

A shortest walk from *s* to i subject to the constraint that the walk contains at most h arcs and goes through node *s* only once, is denoted shortest(<=h) walk and its length is D^h_i.

Bellman-Ford rule:

Initialization D^h_s=0, for all h; $w_{i,j}$ = infinity if (i,j) NOT in A; $w_{k,k}$ =0; D^0_i=infinity for all i!=s.

Iteration:

$$D^{h+1}_i = \min_k [w_{i,k} + D^h_k]$$

The Bellman-Ford algorithm first finds the one-arc shortest walk lengths, then the two-arc shortest walk length, then the three-arc…etc. →distributed version used for routing

Distance Vector Routing Algorithm

iterative:

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

asynchronous:

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

distributed:

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

Assumption: no negative cycles

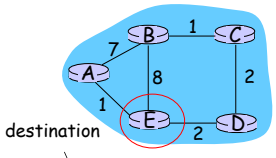
Distance Table data structure

- each node has its own
- row for each possible destination
- column for each directly-attached neighbor to node
- example: in node X, for dest. Y via neighbor Z:

$$D^X(Y,Z) = \text{distance } \textit{from } X \textit{ to } Y, \textit{ via } Z \text{ as next hop} = c(X,Z) + \min_w\{D^Z(Y,w)\}$$

Info maintained at Z. Min must be communicated

Distance Table: example



destination

$D^E(C,D) = c(E,D) + \min_w\{D^D(C,w)\}$
$= 2+2 = 4$

$D^E(A,D) = c(E,D) + \min_w\{D^D(A,w)\}$
$= 2+3 = 5$ loop!

$D^E(A,B) = c(E,B) + \min_w\{D^B(A,w)\}$
$= 8+6 = 14$ loop!

Path B-C-D-E-A

Distance table in node E after the algorithm has converged

cost to destination via

$D^E()$	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

destination

First example

Distance table gives routing table

cost to destination via

$D^E()$	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

destination

	Outgoing link to use, cost
A	A,1
B	D,5
C	D,4
D	D,2

destination

Distance table → Routing table

Distance Vector Routing: overview

Iterative, asynchronous: each local iteration caused by:

- local link cost change
- message from neighbor: its least cost path change from neighbor

Distributed:

- each node notifies neighbors *only* when its least cost path to any destination changes
  - neighbors then notify their neighbors if necessary

Each node:

wait for (change in local link cost of msg from neighbor)

↓

recompute distance table

↓

if least cost path to any dest has changed, *notify* neighbors

Distance Vector Algorithm:

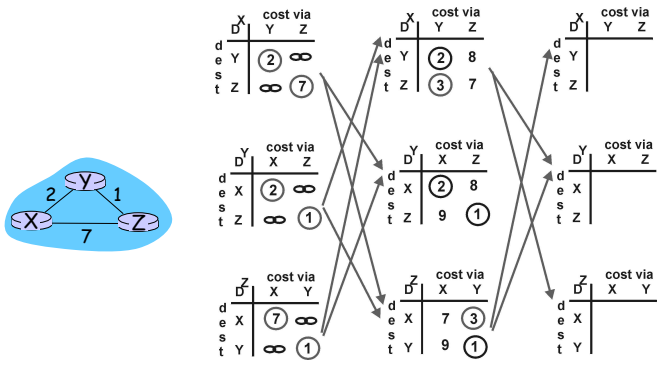
At all nodes, X:

```
1  Initialization:
2  for all adjacent nodes v:
3     D^X(*,v) = infinity       /* the * operator means "for all rows" */
4     D^X(v,v) = c(X,v)
5  for all destinations, y
6     send min_w D^X(y,w) to each neighbor  /* w over all X's neighbors */
```

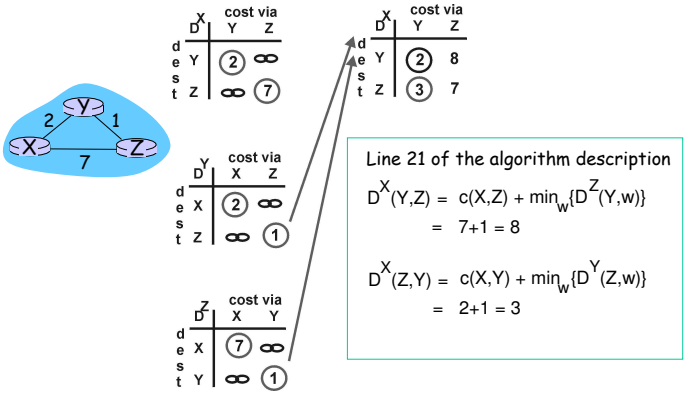

Distance Vector Algorithm (cont.):

```
8  loop
9   wait (until I see a link cost change to neighbor V
10       or until I receive update from neighbor V)
11
12  if (c(X,V) changes by d)
13   /* change cost to all dest's via neighbor v by d */
14   /* note: d could be positive or negative */
15   for all destinations y:  D^X(y,V) =  D^X(y,V) + d
16
17  else if (update received from V wrt destination Y)
18   /* shortest path from V to some Y has changed  */
19   /* V has sent a new value for its  min_w DV(Y,w) */
20   /* call this received new value is "newval"     */
21   for the single destination y: D^X(Y,V) = c(X,V) + newval
22
23  if we have a new min_w D^X(Y,w)for any destination Y
24    send new value of min_w D^X(Y,w) to all neighbors
25
26  forever
```


Distance Vector Algorithm: example



Distance Vector Algorithm: example



Algoritmo Bellman-Ford distribuito

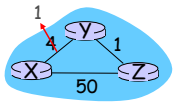
- Completamente asincrono
- Converge agli shortest path (dimostrazione sul libro di Bertsekas-Gallager). Qualche ipotesi? Molto lasche (non limitanti)
  - Nodi continuano a mandare messaggi secondo l'algoritmo
  - Possono partire da stime diverse, non accurate degli shortest path, NON negative
  - I messaggi inviati arrivano in tempo finito

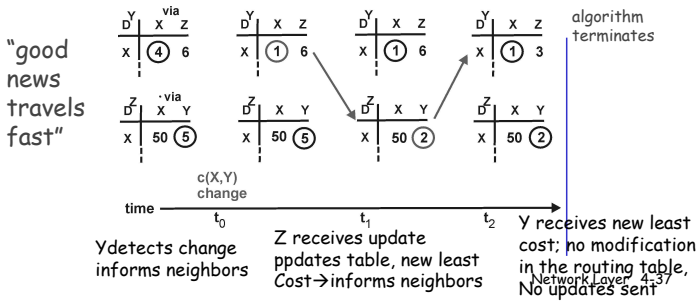
  (e se si verificano errori nella trasmissione sul canale? Nelle implementazioni comunque invio updates periodici)

Distance Vector: link cost changes

Link cost changes:

- node detects local link cost change
- updates distance table (line 15)
- if cost change in least cost path, notify neighbors (lines 23,24)

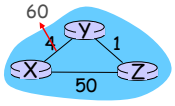


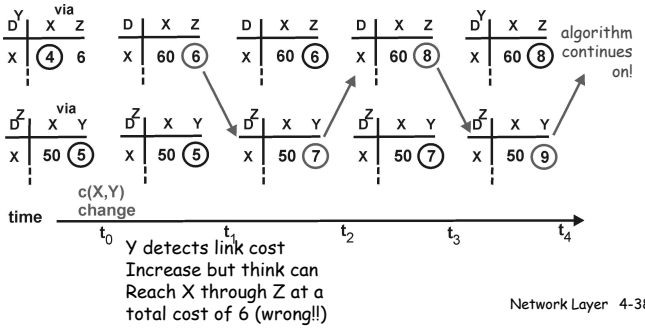


Distance Vector: link cost changes

Link cost changes:

- good news travels fast
- bad news travels slow - "count to infinity" problem!

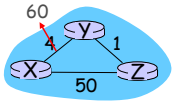


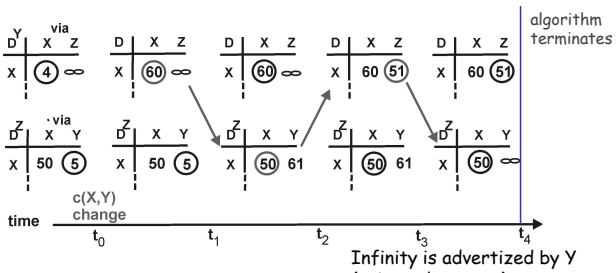


Distance Vector: poisoned reverse

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

Message complexity

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

Speed of Convergence

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

Robustness: what happens if router malfunctions?

LS:

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

DV:

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

Flooding

- Obiettivo: inviare a tutti i nodi un messaggio con un determinato Sequence Number
- Quando un messaggio con SN arriva ad un nodo questo lo riforwarda verso tutti i suoi vicini
  - Tranne il nodo da cui il messaggio è stato ricevuto
  - Se non ha ancora mai forwardato messaggi con sequence number >=SN
- Estremamente semplice. Richiede un numero di messaggi O(numero di link).
- Meno messaggi se si usa uno spanning tree MA necessario mantenere lo spanning tree, MENTRE flooding può essere effettuato immediatamente, richiede solo la conoscenza dei vicini.

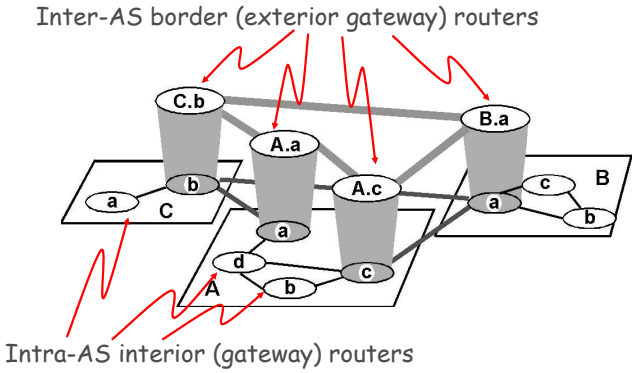
Chapter 4 roadmap

Routing in the Internet

- The Global Internet consists of Autonomous Systems (AS) interconnected with each other:
  - **Stub AS**: small corporation: one connection to other AS's
  - **Multihomed AS**: large corporation (no transit): multiple connections to other AS's
  - **Transit AS**: provider, hooking many AS's together
- Two-level routing:
  - **Intra-AS**: administrator responsible for choice of routing algorithm within network
  - **Inter-AS**: unique standard for inter-AS routing: BGP

Autonomous system: administered (or at leats some degree of technical And administrative control) by a single entity, characterized by a given Routing technology.

Internet AS Hierarchy



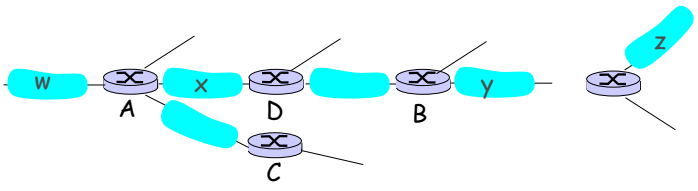
Intra-AS Routing

- 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)

RIP (Routing Information Protocol)

- Distance vector algorithm
- Included in BSD-UNIX Distribution in 1982
- designed for IGP,moderate size networks
- Distance metric: # of hops (max = 15 hops)
  - *Can you guess why?*
- Distance vectors: exchanged among neighbors every 30 sec via Response Message (also called **advertisement**)
- Each advertisement: list of up to 25 destination nets within AS

RIP: Example



Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	B	7
x	--	1
….	….	….

Routing table in D

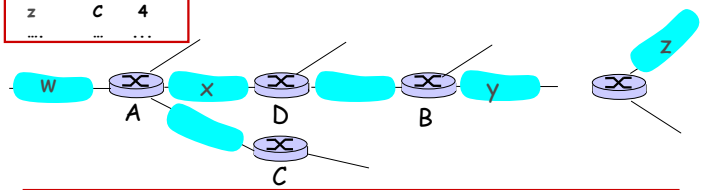
Info: dest. Address, next router, interface to the next router, metric value
Timer(amount of time since last update of the entry)

RIP: Example

Dest	Next	hops
w	-	-
x	-	-
z	C	4
….	…	…

Advertisement from A to D

1)takes track only of the 'best route'



Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	~~B~~ A	~~7~~ 5
x	--	1
….	….	….

Routing table in D

RIP logic

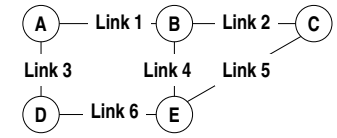
- *Distance Vector routing protocol*
  - Bellman-Ford algorithm
    - METRIC = distance
    - STATE INFO = vector
- each router maintains a table with:
  - best known distance (in hop count) to each destination
  - link to use to reach the destination
- fully distributed protocol
  - vector (table) updates via periodic (30 s) communication with neighbors
  - when an update arrives from neighbors G' compare if cost of network shared with G'+ advertised cost smaller than what stored currently in the table. In case update the table. Also update if G' is the router through which the current route in the table goes trough or if there is no entry in the table.
- What if the next-hop router crashes? Timeouts→when it expires and no updates is received from the router, the route is deleted from the table.

RIP: Link Failure and Recovery

If no advertisement heard after 180 sec --> neighbor/link declared dead

- routes via neighbor invalidated
- new advertisements sent to neighbors
- neighbors in turn send out new advertisements (if tables changed)
- link failure info quickly propagates to entire net
- poison reverse used to prevent ping-pong loops (infinite distance = 16 hops)

RIP operation example (1)



All routers start from empty tables

Router A		
dst	hop	lnk
A	0	loc

Router B		
dst	hop	lnk
B	0	loc

Router C		
dst	hop	lnk
C	0	loc

Router D		
dst	hop	lnk
D	0	loc

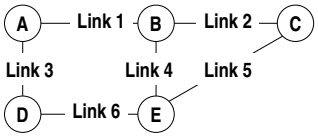
Router E		
dst	hop	lnk
E	0	loc

Router A emits message (A,0) to adjacent routers (B,D), which update table as:

Router B		
dst	hop	lnk
B	0	loc
A	1	1

Router D		
dst	hop	lnk
D	0	loc
A	1	3

RIP operation example (2)



New step: B propagates its updated routing table to neighbohrs A, C, E

Router B		
dst	hop	lnk
B	0	loc
A	1	1

Router A		
dst	hop	lnk
A	0	loc
B	1	1

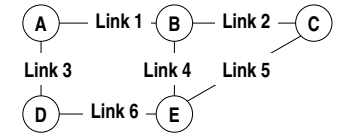
Router B		
dst	hop	lnk
B	0	loc
A	1	1

Router C		
dst	hop	lnk
C	0	loc
B	1	2
A	2	2

Router D		
dst	hop	lnk
D	0	loc
A	1	3

Router E		
dst	hop	lnk
E	0	loc
B	1	4
A	2	4

RIP operation example (3)



Step 3: D propagates its routing table to A, E

Router D		
dst	hop	lnk
D	0	loc
A	1	3

Router A		
dst	hop	lnk
A	0	loc
B	1	1
D	1	3

Router B		
dst	hop	lnk
B	0	loc
A	1	1

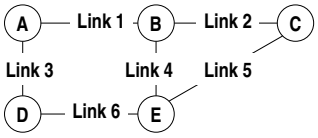
Router C		
dst	hop	lnk
C	0	loc
B	1	2
A	2	2

Router D		
dst	hop	lnk
D	0	loc
A	1	3

Router E		
dst	hop	lnk
E	0	loc
B	1	4
A	2	4
D	1	6

Already updated!

RIP operation example (4)



Step 4: A propagates its routing table to B,D

Router A		
dst	hop	lnk
A	0	loc
B	1	1
D	1	3

Router A		
dst	hop	lnk
A	0	loc
B	1	1
D	1	3

Router B		
dst	hop	lnk
B	0	loc
A	1	1
D	2	1

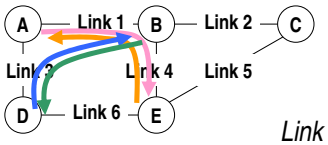
Router C		
dst	hop	lnk
C	0	loc
B	1	2
A	2	2

Router D		
dst	hop	lnk
D	0	loc
A	1	3
B	2	3

Router E		
dst	hop	lnk
E	0	loc
B	1	4
A	2	4
D	1	6

...ETC ETC ETC...

RIP operation example
final routing tables



Step 5: C -> B, E
Step 6: E-> B, C, D
Step 7: B-> A, C, E

Link 6 under-utilized!!

Router A		
dst	hop	lnk
A	0	loc
B	1	1
C	2	1
D	1	3
E	2	1

Router B		
dst	hop	lnk
A	1	1
B	0	loc
C	1	2
D	2	1
E	1	4

Router C		
dst	hop	lnk
A	2	2
B	1	2
C	0	loc
D	2	5
E	1	5

Router D		
dst	hop	lnk
A	1	3
B	2	3
C	2	6
D	0	loc
E	1	6

Router E		
dst	hop	lnk
A	2	4
B	1	4
C	1	5
D	1	6
E	0	loc

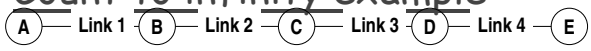
Apparent limits of RIP

- Hop count is a too simple metric!
  - But Bellman-ford algorithm does not require to operate with hop count! Can be trivially extended to different distance metric: the core of the algorithm does not change!
    - queue length on considered hop
    - time delay in ms
    - packet loss ratio measured
    - etc…
- Slow convergence
  - routers distant N hops need N steps to update their tables
- Limited to small network sizes
  - as infinite=16, nodes cannot be more than 15 hops far away
    - but just raise infinite to 32…

Real limit of RIP
"count to infinity" problem

- Insane transient reaction to node/link failures!
  - Convergence still remains, but very slow
  - Loops may occur while routing tables stabilize
  - the slower, the higher value infinite is chosen!!
    - Values higher than 16 are terrible
  - An intrinsic and unavoidable drawback for all Distance Vector schemes

Count to infinity example



Router A		
dst	hop	lnk
A	0	loc

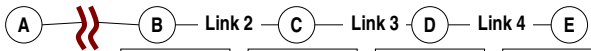
Router B		
dst	hop	lnk
A	1	1

Router C		
dst	hop	lnk
A	2	2

Router D		
dst	hop	lnk
A	3	3

Router E		
dst	hop	lnk
A	4	4

LINK 1 breaks



B does not get updates from A,route timeout.
Then updates from B, that tell A=3!
Why? B believes that there is a route through C (but the route was through B!!)

Next steps:

Router B		
dst	hop	lnk
A	3	2
A	3	2
A	5	2
A	5	2
A	7	2

Router C		
dst	hop	lnk
A	2	2
A	4	2
A	4	2
A	6	2
A	6	2

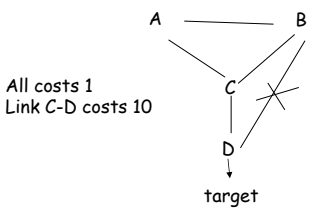
Router D		
dst	hop	lnk
A	3	3
A	3	3
A	5	3
A	5	3
A	7	3

Router E		
dst	hop	lnk
A	4	4
A	4	4
A	4	4
A	6	4
A	6	4

The count to infinity problem

- The problem is that NO ROUTERS have a value more than 1 + the minimum of adjacent routers
- situation stabilizes only when count gets to infinity
- it is more critical the higher infinity is set! (→infinity set to 16 in RIP!!)

Count to infinity another example



All costs 1
Link C-D costs 10

Routes to the target:
D:direc. Conn cost 1;
B: via D, cost 2;
C: via B, cost 3;
A: via B, cost 3.

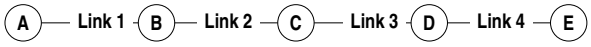
Now the link from B to D fails.
What happens?

time →

D:dir, 1	dir,1	dir,1	dir,1	..dir,1	dir,1
B:unreach	C,4	C,5	C,6	..C,11	C,12
C:B,3	A,4	A,5	A,6	..A,11	D,11
A:B,3	C,4	C,5	C,6	..C,11	C,12

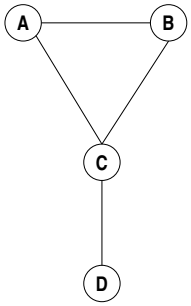
Count to infinity!!!!

Split horizon with poison reverse"solution"



- The distance is NOT reported on the line from which information comes
  - C tells correct distance to D, but lies (says infinity) to B
- discovers link failure in 1 hop

Split horizon poison reverse failure



Line CD goes down…

1) because of split horizon rule, A and B tell C that dist(D)=inf
2) C concludes that D is unreachable and reports this to A and B
3) but A knows from B that dist(D)=2, and sets its dist=3
4) similarly, B knows from A distance from D…
C estimates new value 4; A and B again through C estimate a value of 5….then again 1)
… etc until distance = infinite

Regardless the hack used, there is always a network topology that makes the trick fail!

RIP Solution: trigger updates

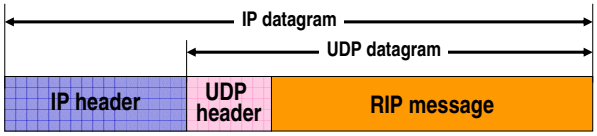
•Just an attempt to speed up convergence

•Whenever a gateway change the metric for a route it is required to send an update message almost immediately (even if it is not time for the Regular periodic update message)

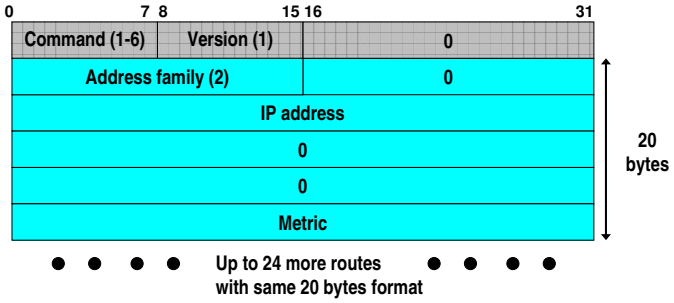
•→ In case of (and only if) changes modifying the best route or the best route metric value updates are sent→indeed trigger updates lead to a cacade of updates→fast convergence

Routing Information Protocol

- Most widely used
  - and most criticized…
- Official specification: RFC 1058 (1988)
  - but used from several years before
- Uses UDP to exchange messages
  - well known UDP port = 520



RIP message



Command: 1=request to send all or part of the routing table; 2=reply (3-6 obsolete or non documented)
Address family: 2=IP addresses
metric: distance of *emitting router* from the specified IP address in *number of hops (valid from 1 to 15; 16=infinite)*

Message size

- 8 UDP header
- 4 bytes RIP header
- 20 bytes x up to 25 entries

- total: maximum of 512 bytes UDP datagram

- 25 entries: too little to transfer an entire routing table
  - more than 1 UDP datagram generally needed

Initialization

- When routing daemon started, send special RIP request on every interface
  - command = 1 (request)
  - address family = 0 (instead of 2)
  - metric set to 16 (infinite)
- This asks for complete routing table from all connected routers
  - allows to discover adjacent routers!

Operation after initialization

- Request:
  - asks for response relative to specific IP addresses listed in the request message
- Response:
  - return list of IP addresses with associated metric
  - if router does not have a route to the specified destination, returns 16
- Regular update:
  - routers send part (or all) of their table every 30s to adjacent routers
  - a router deletes (set metric to 16) an entry from its routing table if not refreshed within 6 cycles (180s)
    - deletion after additional 60s to ensure propagation of entry invalidation
- triggered update:
  - upon change of metric for a route (transmits only entries changed)

RIP Table example (continued)

Router: *giroflee.eurocom.fr*

Destination	Gateway	Flags	Ref	Use	Interface
127.0.0.1	127.0.0.1	UH	0	26492	lo0
192.168.2.	192.168.2.5	U	2	13	fa0
193.55.114.	193.55.114.6	U	3	58503	le0
192.168.3.	192.168.3.5	U	2	25	qaa0
224.0.0.0	193.55.114.6	U	3	0	le0
default	193.55.114.129	UG	0	143454	

- Three attached class C networks (LANs)
- Router only knows routes to attached LANs
- Default router used to "go up"
- Route multicast address: 224.0.0.0
- Loopback interface (for debugging)

Chapter 4 roadmap

Hierarchical Routing

Our routing study thus far - idealization
- all routers identical
- network "flat"

… *not* true in practice

scale: with 200 million destinations:
- can't store all dest's in routing tables!
- routing table exchange would swamp links!

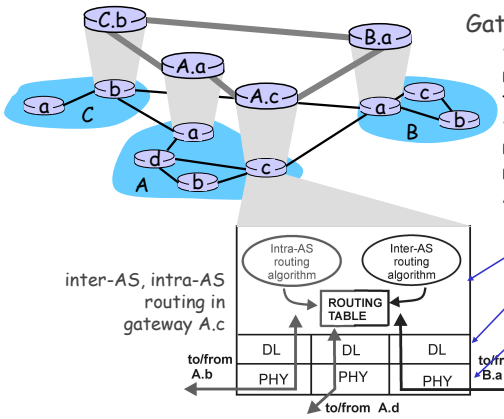
administrative autonomy
- internet = network of networks
- each network admin may want to control routing in its own network

Hierarchical Routing

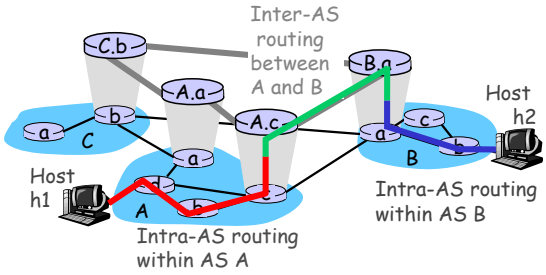
- aggregate routers into regions, "autonomous systems" (AS)
- routers in same AS run same routing protocol
  - "intra-AS" routing protocol
  - routers in different AS can run different intra-AS routing protocol

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

Intra-AS and Inter-AS routing

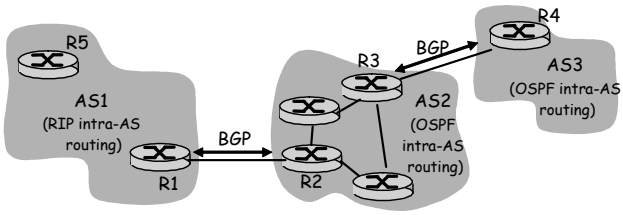


Intra-AS and Inter-AS routing



- We'll examine specific inter-AS and intra-AS Internet routing protocols shortly

Inter-AS routing in the Internet: BGP



AS2:
BGP used to determine routes to destinations outside AS2
OSPF routes to destinations within AS2

Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): *the* de facto standard
- **Path Vector** protocol:
  - similar to Distance Vector protocol
  - each Border Gateway broadcast to neighbors (BGP peers) *entire path* (i.e., sequence of AS's) to destination
  - BGP routes to networks (ASs), not individual hosts
  - E.g., Gateway X may send its path to dest. Z:

Path (X,Z) = X,Y1,Y2,Y3,…,Z

Internet inter-AS routing: BGP

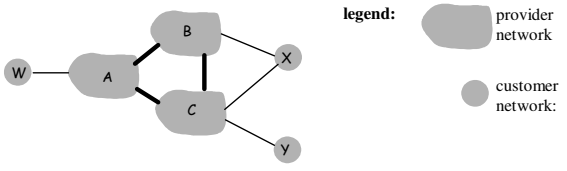
Suppose: gateway X send its path to peer gateway W

- W may or may not select path offered by X
  - cost, policy (don't route via competitors AS), loop prevention reasons.
- If W selects path advertised by X, then:

Path (W,Z) = w, Path (X,Z)

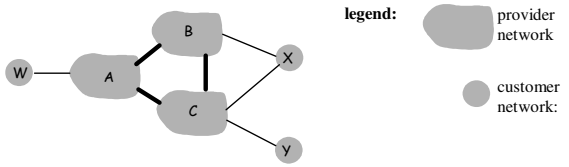
- Note: X can control incoming traffic by controlling it route advertisements to peers:
  - e.g., don't want to route traffic to Z -> don't advertise any routes to Z

BGP: controlling who routes to you



- A,B,C are provider networks
- X,W,Y are customer (of provider networks)
- X is dual-homed: attached to two networks
  - X does not want to route from B via X to C
  - .. so X will not advertise to B a route to C

BGP: controlling who routes to you



- A advertises to B the path AW
- B advertises to W the path BAW
- Should B advertise to C the path BAW?
  - No way! B gets no "revenue" for routing CBAW since neither W nor C are B's customers
  - B wants to force C to route to w via A
  - B wants to route *only* to/from its customers!

BGP operation

Q: What does a BGP router do?

- Receiving and filtering route advertisements from directly attached neighbor(s).
- Route selection.
  - To route to destination X, which path)of several advertised) will be taken?
- Sending route advertisements to neighbors.

BGP messages

- BGP messages exchanged using TCP.
- BGP messages:
  - OPEN: opens TCP connection to peer and authenticates sender
  - 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

Why different Intra- and Inter-AS routing ?

Policy:

- Inter-AS: admin wants control over how its traffic routed, who routes through its net.
- Intra-AS: single admin, so no policy decisions needed

Scale:

- hierarchical routing saves table size, reduced update traffic

Performance:

- Intra-AS: can focus on performance
- Inter-AS: policy may dominate over performance

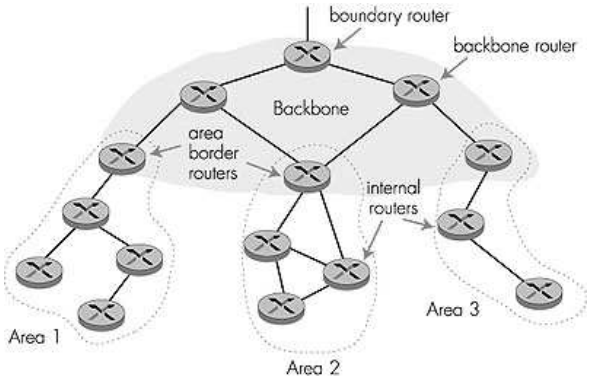
OSPF (Open Shortest Path First)

- "open": publicly available
- Uses Link State algorithm
  - LS packet 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

OSPF "advanced" features (not in RIP)

- Security: all OSPF messages authenticated (to prevent malicious intrusion)
- Multiple same-cost paths allowed (only one path in RIP)
- Load balancing
- 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 data base as OSPF
- Hierarchical OSPF in large domains.

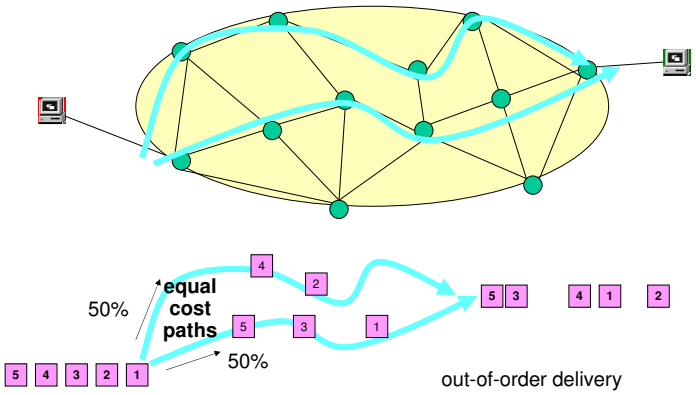
Hierarchical OSPF



Hierarchical OSPF

- Two-level hierarchy: local area, backbone.
  - Link-state advertisements only in area
  - each nodes has detailed area topology; only know direction (shortest path) to nets in other areas.
- **Area border routers:** "summarize" distances to nets in own area, advertise to other Area Border routers.
- **Backbone routers:** run OSPF routing limited to backbone.
- **Boundary routers:** connect to other AS's.

Load Balancing drawbacks



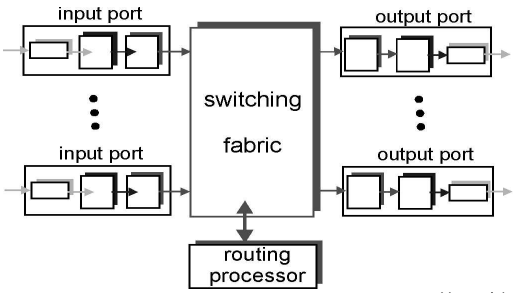
Chapter 4 roadmap

4.1 Introduction and Network Service Models
4.2 Routing Principles
4.3 Hierarchical Routing
4.4 The Internet (IP) Protocol
4.5 Routing in the Internet
4.6 What's Inside a Router?
4.7 IPv6
4.8 Multicast Routing
4.9 Mobility

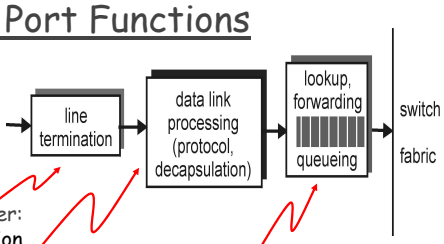
Router Architecture Overview

Two key router functions:

- run routing algorithms/protocol (RIP, OSPF, BGP)
- *switching* datagrams from incoming to outgoing link



Input Port Functions



Physical layer:
bit-level reception

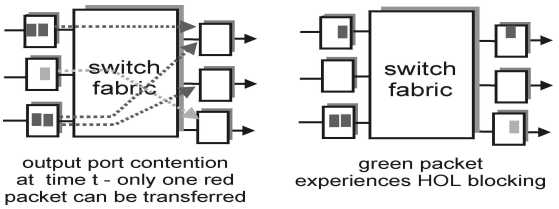
Data link layer:
e.g., Ethernet
see chapter 5

Decentralized switching:

- given datagram dest., lookup output port using routing table in input port memory
- goal: complete input port processing at 'line speed'
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Input Port Queuing

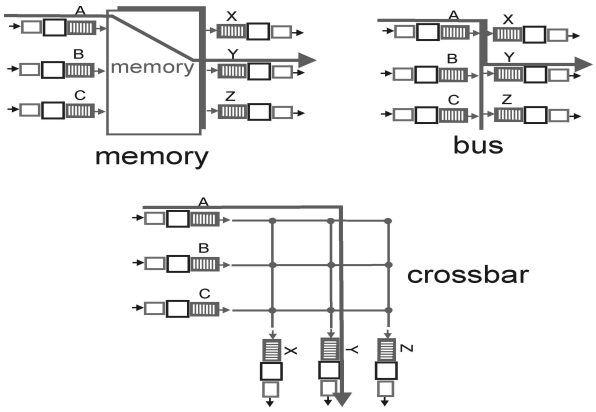
- Fabric slower that input ports combined -> queueing may occur at input queues
- Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward
- *queueing delay and loss due to input buffer overflow!*



output port contention at time t - only one red packet can be transferred

green packet experiences HOL blocking

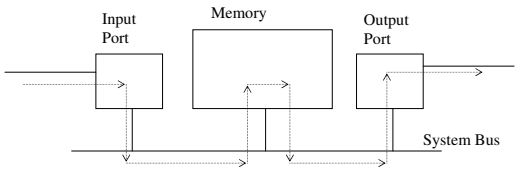
Three types of switching fabrics



Switching Via Memory

First generation routers:

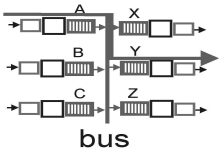
- packet copied by system's (single) CPU
- speed limited by memory bandwidth (2 bus crossings per datagram)



Modern routers:

- input port processor performs lookup, copy into memory
- Cisco Catalyst 8500

Switching Via a Bus

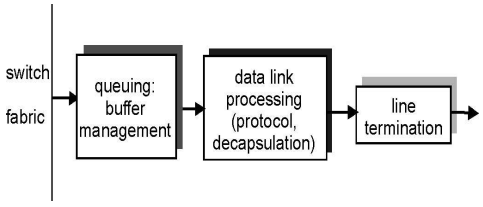


- datagram from input port memory to output port memory via a shared bus
- bus contention: switching speed limited by bus bandwidth
- 1 Gbps bus, Cisco 1900: sufficient speed for access and enterprise routers (not regional or backbone)

Switching Via An Interconnection Network

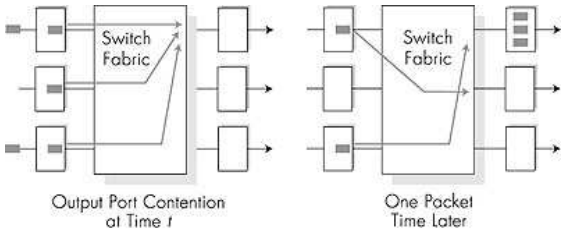
- overcome bus bandwidth limitations
- Banyan networks, other interconnection nets initially developed to connect processors in multiprocessor
- Advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches Gbps through the interconnection network

Output Ports



- *Buffering* required when datagrams arrive from fabric faster than the transmission rate
- *Scheduling discipline* chooses among queued datagrams for transmission (typical FCFS, other disciplines configurable, e.g. for QoS support, priority queueing, WFQ)
- What in case of buffer overflow? Discarding. Schemes introduced (e.g. RED to react before the buffer overflow

Output port queueing



- buffering when arrival rate via switch exceeds output line speed
- *queueing (delay) and loss due to output port buffer overflow!*

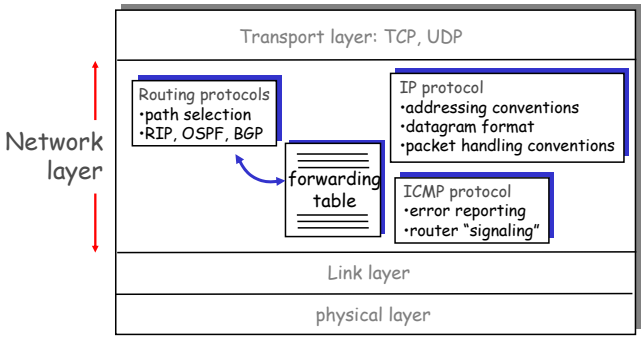
Chapter 4 roadmap

4.1 Introduction and Network Service Models
4.2 Routing Principles
4.3 Hierarchical Routing
4.4 The Internet (IP) Protocol
- 4.4.1 IPv4 addressing
- 4.4.2 Moving a datagram from source to destination
- 4.4.3 Datagram format
- 4.4.4 IP fragmentation
- 4.4.5 ICMP: Internet Control Message Protocol
- 4.4.6 DHCP: Dynamic Host Configuration Protocol
- 4.4.7 NAT: Network Address Translation

4.5 Routing in the Internet
4.6 What's Inside a Router
4.7 IPv6
4.8 Multicast Routing
4.9 Mobility

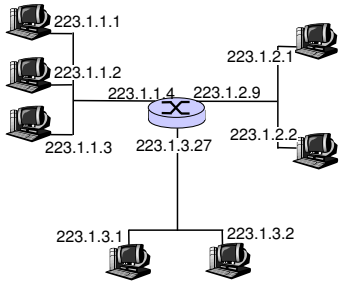
The Internet Network layer

Host, router network layer functions:



IP Addressing: introduction

- IP address: 32-bit identifier for host, router *interface*
- *interface:* connection between host/router and physical link
  - router's typically have multiple interfaces
  - host may have multiple interfaces
  - IP addresses associated with each interface



223.1.1.1 = 11011111 00000001 00000001 00000001
223 1 1 1

Gli indirizzi IP

- **Sono costituiti da 32 bit solitamente raggruppati in gruppi di 8 bit (byte)**

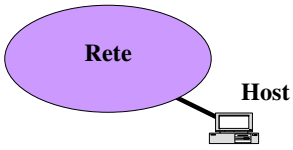
- **i byte sono usualmente riportati in notazione decimale divisi da punti e possono assumere valori compresi tra 0 e 255**

131.175.21.1

Gli indirizzi IP

- L'indirizzo è diviso in due parti

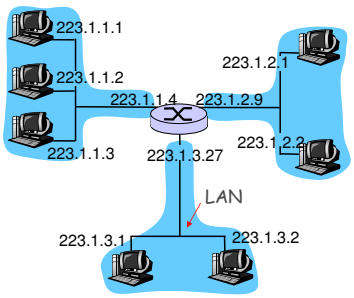
- La NetID (indirizzo di rete) identifica la rete
- La HostID (indirizzo di host) identifica l'host nella rete



- **tutti gli host all'interno della stessa rete hanno lo stesso indirizzo di rete**

IP Addressing

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

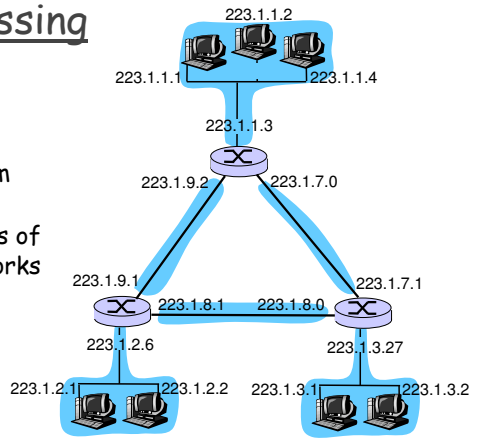


network consisting of 3 IP networks (for IP addresses starting with 223, first 24 bits are network address)

IP Addressing

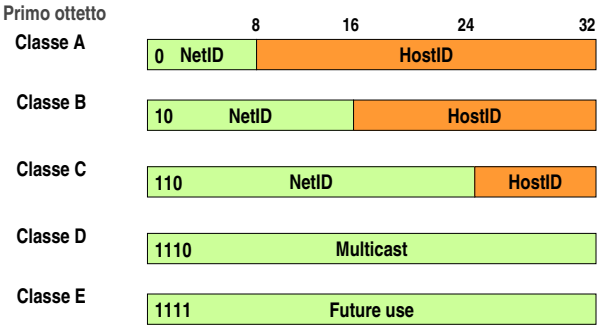
How to find the networks?

- Detach each interface from router, host
- create "islands of isolated networks



Interconnected system consisting of six networks

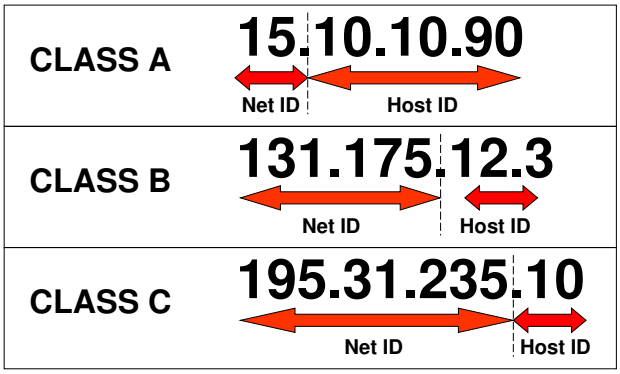
Le classi



Dotted Decimal Ranges

Address Class	Dotted Decimal ranges
Class A	1.xxx.xxx.xxx through 126.xxx.xxx.xxx
Class B	128.0.xxx.xxx through 191.255.xxx.xxx
Class C	192.0.0.xxx through 223.255.255.xxx
Class D (mcast)	224.xxx.xxx.xxx through 239.xxx.xxx.xxx
Class E (exper)	240.xxx.xxx.xxx through 255.xxx.xxx.xxx

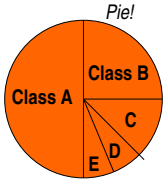
Examples



Counting up

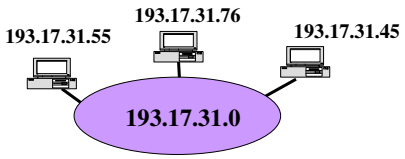
- 32 bit IP address:
  - 2^{32} = 4.294.967.296 theoretical IP addresses
- class A:
  - 2^7-2 =126 networks [0.0.0.0 and 127.0.0.0 reserved]
  - 2^{24}-2 = 16.777.214 maximum hosts
    - **2.113.928.964** addressable hosts (49,22% of max)
- class B
  - 2^{14}=16.384 networks
  - 2^{16}-2 = 65.534 maximum hosts
    - **1.073.709.056** addressable hosts (24,99% of max)
- class C
  - 2^{21}=2.097.152 networks
  - 2^8-2 = 254 maximum hosts
    - **532.676.608** addressable hosts (12,40% of max)

The IP address Pie!



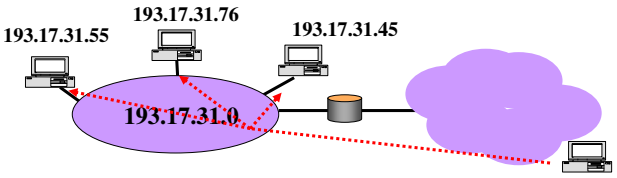
Indirizzi speciali

- Indirizzo di rete:
  - L'indirizzo con il campo HostID posto a 0 serve ad indicare la rete il cui indirizzo è contenuto nel campo NetID (usato solo nelle tabelle di instradamento)
  - esempio:
    - rete in classe B: 131.175.0.0
    - rete in classe C: 193.17.31.0



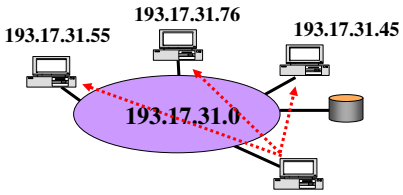
Indirizzi speciali

- Indirizzo broadcast diretto:
  - Un indirizzo con il campo HostID di soli 1 assume il significato di indirizzo broadcast della rete indicata nel campo NetID.
  - esempio: 193.17.31.255



Indirizzi speciali

- Indirizzo broadcasting limitato:
  - Un indirizzo di soli 1 assume il significato di indirizzo broadcast nella stessa rete di chi invia il pacchetto. Il pacchetto non può oltrepassare dei router: 255.255.255.255

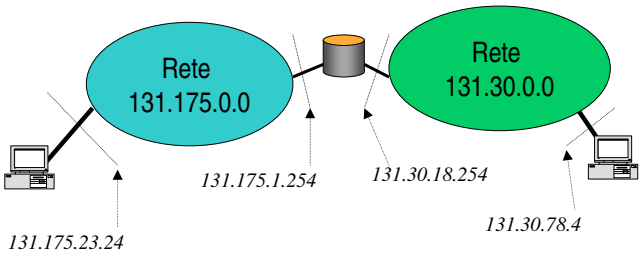


Indirizzi speciali

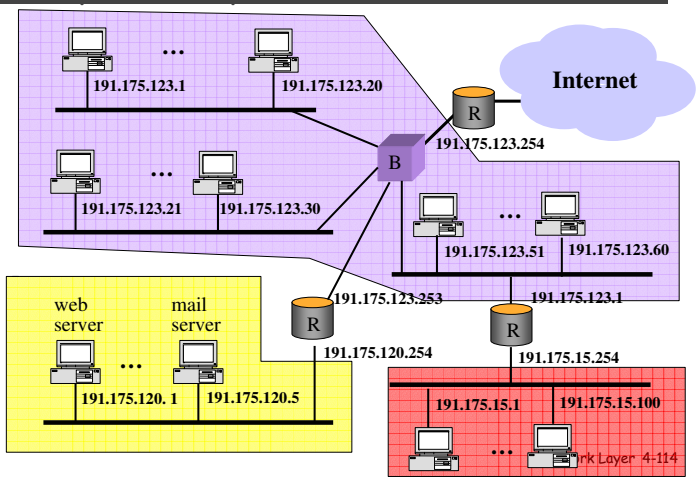
- Quando il campo NetID è posto a zero, l'indirizzo indica l'host il cui indirizzo è contenuto nel campo host sulla stessa rete del mittente.
  - esempio: 0.0.21.173 (in una rete in classe B)
- Se anche il campo host è posto a zero l'indirizzo indica il mittente stesso del pacchetto (usato quando l'host non conosce il proprio indirizzo).
  - esempio: 0.0.0.0
- Infine, l'indirizzo con il primo ottetto pari a 127 e gli altri campi qualsivoglia indica il loopback sullo stesso host (usato nei sistemi operativi per testare le funzionalità di rete→il pacchetto non e' inviato in rete ma trattato dall'host come un pacchetto in arrivo).
  - esempio: 127.0.0.0

Indirizzamento IP

- L'indirizzo IP indica l'interfaccia di un dispositivo con la rete
- Se un dispositivo ha più interfacce su più reti deve avere un indirizzo per ciascuna interfaccia



Esempio di un piano di indirizzamento IP



Inoltro (forwarding) e instradamento (routing)

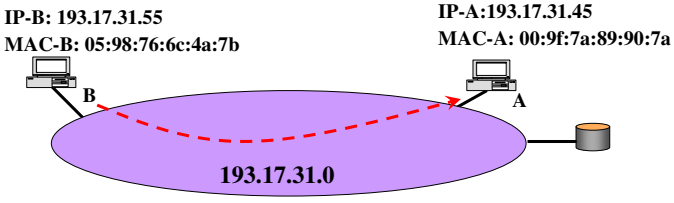
- Tecnica di inoltro:
  - *definisce le regole con le quali un pacchetto viene inoltrato verso l'uscita (normalmente sulla base della lettura di una tabella di instradamento)*
- Algoritmo di routing:
  - *definisce le regole con le quali viene scelto un percorso in rete tra sorgente e destinazione (sulla base delle quali vengono scritte le tabelle di instradamento)*
- Protocollo di routing:
  - *definisce i messaggi che si scambiano i nodi di rete per implementare l'algoritmo di routing*

Inoltro dei pacchetti

- IP è una tecnica di internetworking, quindi nell' instradamento dei pacchetti tra un router/host e un altro si serve della capacità di instradamento della reti (locali) che collega
- Inoltro diretto:
  - quando la destinazione è nella stessa rete (locale)
- Inoltro indiretto:
  - quando la destinazione non è nella stessa rete (locale)

Inoltro diretto

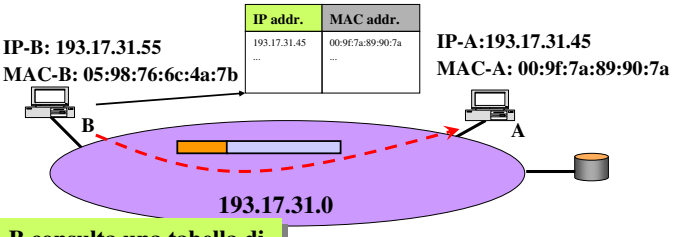
Rete locale coincidente con rete / sottorete IP



1. L'entità IP di B deve spedire un pacchetto all'indirizzo IP-A

2. B conosce l'indirizzo IP-B della propria interfaccia e dal confronto con IP-A capisce che A si trova nella stessa rete

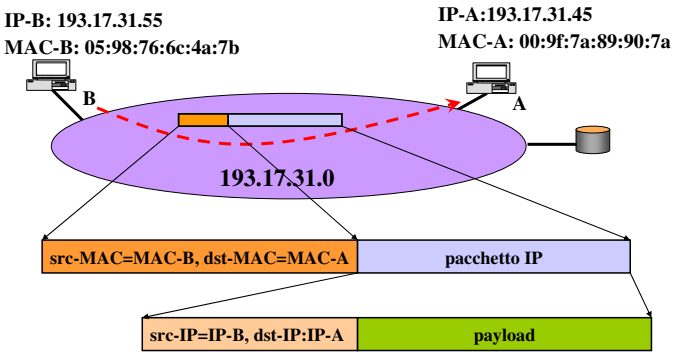
Inoltro diretto



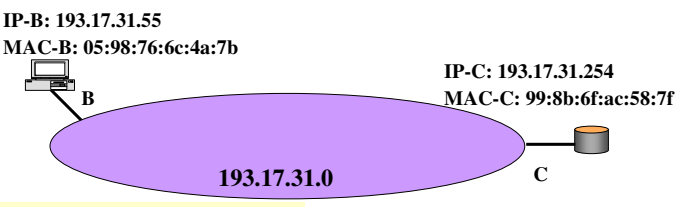
3. B consulta una tabella di corrispondenza tra indirizzi IP e indirizzi della rete (indirizzi MAC nel caso di rete locale) per reperire l'indirizzo MAC-A

4. L'entità IP di B passa il pacchetto al livello inferiore che crea un pacchetto con destinazione MAC-A

Inoltro diretto



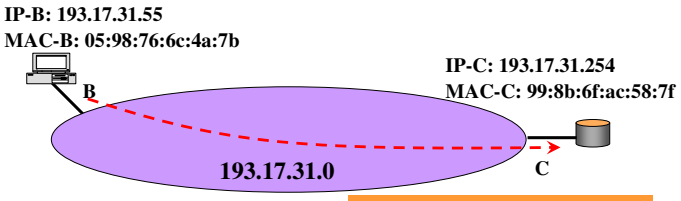
Inoltro indiretto (da un host)



1. L'entità IP di B deve spedire un pacchetto all'indirizzo *IP-D=131.17.23.4*

2. B conosce l'indirizzo IP-B della propria interfaccia e dal confronto con IP-D capisce che D NON si trova nella stessa rete

Inoltro indiretto

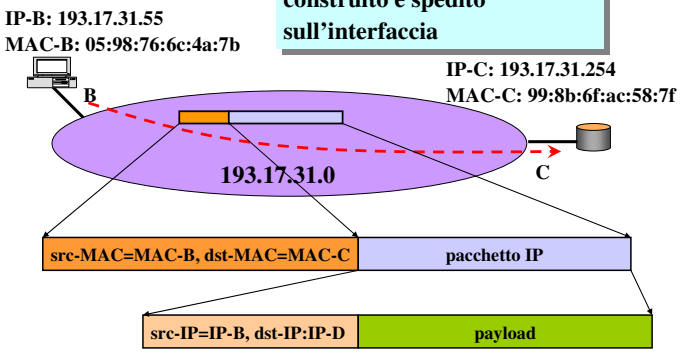


3. B deve dunque inoltrare il pacchetto ad un router (di solito è configurato un solo default router)

4. B recupera l'indirizzo MAC del router nella tabella di corrispondenza e passa il pacchetto al livello inferiore

Inoltro indiretto

5. il pacchetto viene construito e spedito sull'interfaccia



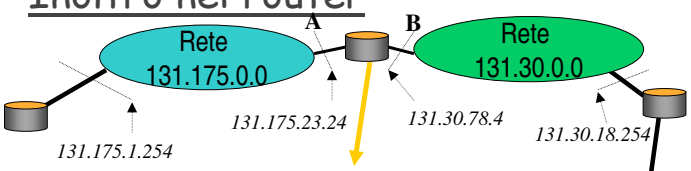
Inoltro nei router

- Anche i router seguono le tecniche di inoltro diretto ed indiretto ma:
  - hanno di solito più di una interfaccia dove poter effettuare l'inoltro diretto
  - hanno delle tabelle di routing dove sono indicati i router a cui passare i pacchetti nel caso di inoltro indiretto

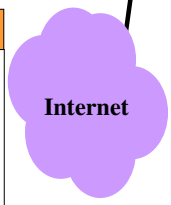
Inoltro nei router

- L'inoltro IP è basato sul solo indirizzo di destinazione (destination-based)
- in particolare sul solo NetID di destinazione (tutti gli host della stessa rete sono considerati insieme)
- nelle tabelle di routing per ogni rete di destinazione è indicato solo il prossimo router (next-hop) nel percorso verso la destinazione (next-hop routing)

Inoltro nei router

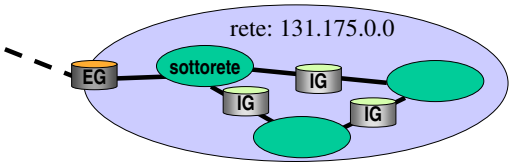


destination network	next-hop
194.34.23.0	131.175.1.254
194.34.34.0	131.175.1.254
140.56.0.0	131.30.18.254
141.56.0.0	131.30.18.254
131.175.0.0	interface A
131.30.0.0	interface B
…	…
default	131.30.18.254



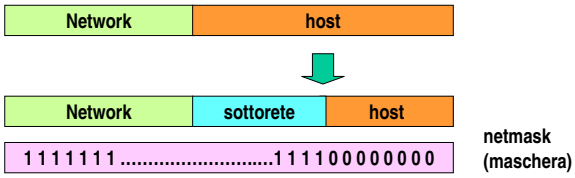
Subnetting

- Con la veloce diffusione delle reti IP, la divisione in classi è divenuta non più soddisfacente a causa della rigità della divisione
- Le organizzazioni private con un indirizzo di rete in classe B ($2^{16}-2$=65534 indirizzi di host) hanno sviluppato proprie Intranet con sotto-reti locali di poche centinaia (o decine) di host



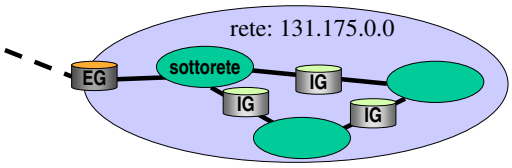
Subnetting

- L'indirizzo di sottorete viene ricavato dividendo ulteriormente il campo host in un campo sottorete e un nuovo campo host



- **La divisione viene determinata in modo flessibile mediante una *netmask* formata da una sequenza di 1 (che indicano la parte rete + sottorete) e una sequenza di 0 (che indicano la parte host)**

Subnetting



- In generale i router esterni alla Intranet continueranno ad avere nella tabella di routing un sola riga per la rete (131.175.0.0)
- mentre i router interni alla rete dovranno gestire anche le sottoreti mediante le netmask

Netmask

- Più in generale la netmask rappresenta un modo per avere un confine mobile tra campo host e campo rete non legato alle classi
- Anche la netmask viene di solito indicata in

netmask: 255.255.255.0

rete: 131.175.0.0

sottoreti: 131.175.0.0, 131.175.1.0, …, 131.175.254.0, 131.75.255.0

Netmask

- Con netmask continue i valori decimali possono assumere i valori:

255	1	1	1	1	1	1	1	1
254	1	1	1	1	1	1	1	0
248	1	1	1	1	1	0	0	0
240	1	1	1	1	0	0	0	0
224	1	1	1	0	0	0	0	0
192	1	1	0	0	0	0	0	0
128	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

- in alternativa la netmask può essere indicata con il numero di 1 consecutivi (prefisso):
  - 131.175.21.0/24

Esempio di subnetting (1)

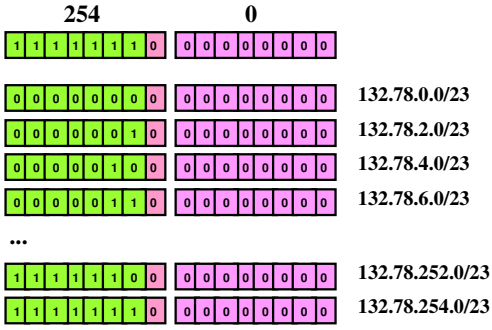
- indirizzo originario della rete: 132.78.0.0
- occorre creare reti con almeno 500 host

- il campo rete ha 16 bit
- 2^9=512, quindi servono 9 bit per il campo host
- rimangono 7 bit per il la sottorete

255.255.254.0

Esempio di subnetting (1)

- il numero di sottoreti disponibili è 2^7=128 ciascuna con 2^9-2=510 host



Esempio di subnetting (2)

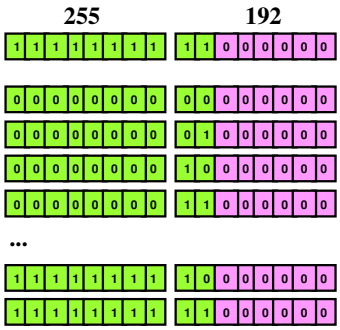
- indirizzo originario della rete: 128.234.0.0
- occorre creare almeno 1000 piccole sottoreti

- il campo rete ha 16 bit
- 2^{10}=1024, quindi servono 10 bit per il campo subnet
- rimangono 6 bit per il campo host
- la netmask dovrà dunque avere 16+10=26

255.255.255.192

Esempio di subnetting (2)

- il numero di host per ciascuna delle 1024 reti è 2^6-2=62 host



Configurazione delle interfacce

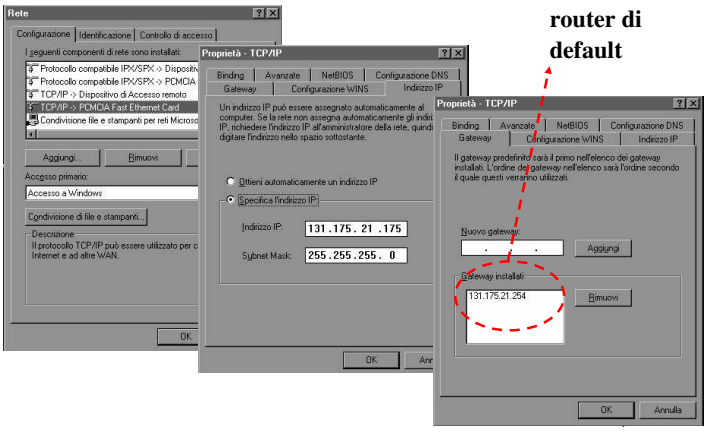
- Per configurare una interfaccia (di un host o di un router) è necessario indicare sia l'indirizzo IP che la netmask

interfaccia: ethernet 0
address: 131.175.21.96
netmask: 255.255.255.0

- Nelle tabelle di intradamento ad ogni indirizzo di rete va associata una netmask per poter conoscere la lunghezza del campo rete+sottorete

network	netmask	next-hop
--	--	--

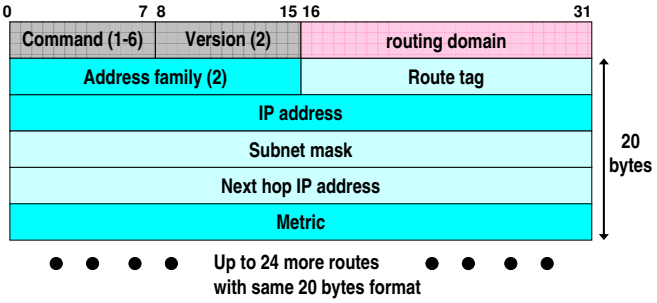
Configurazione degli host



RIP 2

- Does not change the protocol operation
- simply adds information in the all 0s fields of the RIP message
- It is designed to maintain full compatibility with RIP routers
  - al least if they don't get confused from the non 0 entries

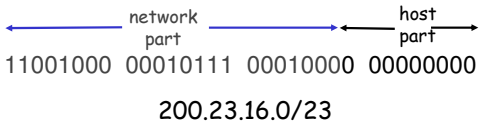
RIP 2 message format



Most important modification: **subnet mask** (allows use with VLSM and CIDR)
Next hop address: specifies where packet should be sent when addressed to Ipaddr
details in RFC 1388

IP addressing: CIDR

- Classful addressing:
  - inefficient use of address space, address space exhaustion
  - e.g., class B net allocated enough addresses for 65K hosts, even if only 2K hosts in that network
- CIDR: Classless InterDomain Routing
  - network portion of address of arbitrary length
  - address format: a.b.c.d/x, where x is # bits in network portion of address



200.23.16.0/23

Inoltro diretto e indiretto con le netmask

- Per inoltrare un pacchetto occorre capire se appartiene alla sottorete di una delle interfacce
- per effettuare la verifica si fa un AND bit a bit tra indirizzo dell'interfaccia e netmask e tra indirizzo di destinazione e netmask
- se i due risultati coincidono allora la sottorete è la stessa e si procede all'inoltro

destinazione: (131.175.21.77) AND (255.255.255.0) = 131.175.21.0

confronto positivo

interfaccia: (131.175.21.96) AND (255.255.255.0) = 131.175.21.0

Tabelle di routing con le netmask

- Se i confronti con le interfacce sono negativi occorre procedere ad un inoltro indiretto
- Se siamo in un router occorre analizzare la tabella di routing
- Il confronto riga per riga si effettua allo stesso modo usando la netmask relativa a ciascuna riga
- Se il confronto dà esito positivo per più righe della tabella viene selezionata la tabella con la netmask che ha il maggior numero di 1 (si dice comunemente che vale il principio del prefisso più lungo).

Tabelle di routing con le netmask

network	netmask	first hop
131.175.21.0	255.255.255.0	131.17.123.254
131.175.16.0	255.255.255.0	131.17.78.254
131.56.0.0	255.255.0.0	131.17.15.254
131.155.0.0	255.255.0.0	131.17.15.254
0.0.0.0	0.0.0.0	131.17.123.254

interface eth0

IP address	131.17.123.1
netmask	255.255.255.0

interface eth1

IP address	131.17.78.1
netmask	255.255.255.0

interface eth2

IP address	131.17.15.12
netmask	255.255.255.0

default router: il confronto dà sempre esito positivo ma la netmask è lunga 0 bit

Tabelle di routing: esempio (1)

network	netmask	first hop
131.175.15.0	255.255.255.0	131.175.21.1
131.175.16.0	255.255.255.0	131.175.21.2
131.175.17.0	255.255.255.0	131.175.21.3
131.180.23.0	255.255.255.0	131.175.21.4
131.180.18.0	255.255.255.0	131.175.21.4
131.180.21.0	255.255.255.0	131.175.21.4
131.180.0.0	255.255.0.0	131.175.21.5
0.0.0.0	0.0.0.0	131.175.12.254

131.175.21.86

interfaccia 1: 131.175.21.254, 255.255.255.0
interfaccia 2: 131.175.12.254, 255.255.255.0

Tabelle di routing: esempio (2)

network	netmask	first hop	
131.175.15.0	255.255.255.0	131.175.21.1	x
131.175.16.0	255.255.255.0	131.175.21.2	OK
131.175.17.0	255.255.255.0	131.175.21.3	x
131.180.23.0	255.255.255.0	131.175.21.4	x
131.180.18.0	255.255.255.0	131.175.21.4	x
131.180.21.0	255.255.255.0	131.175.21.4	x
131.180.0.0	255.255.0.0	131.175.21.5	x
0.0.0.0	0.0.0.0	131.175.12.254	OK

131.175.16.65

interfaccia 1: 131.175.21.254, 255.255.255.0
interfaccia 2: 131.175.12.254, 255.255.255.0

Tabelle di routing: esempio (3)

network	netmask	first hop	
131.175.15.0	255.255.255.0	131.175.21.1	x
131.175.16.0	255.255.255.0	131.175.21.2	x
131.175.17.0	255.255.255.0	131.175.21.3	x
131.180.23.0	255.255.255.0	131.175.21.4	x
131.180.18.0	255.255.255.0	131.175.21.4	x
131.180.21.0	255.255.255.0	131.175.21.4	OK
131.180.0.0	255.255.0.0	131.175.21.5	OK
0.0.0.0	0.0.0.0	131.175.12.254	OK

131.180.21.78

interfaccia 1: 131.175.21.254, 255.255.255.0
interfaccia 2: 131.175.12.254, 255.255.255.0

Tabelle di routing: esempio (4)

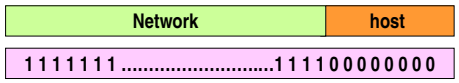
network	netmask	first hop	
131.175.15.0	255.255.255.0	131.175.21.1	x
131.175.16.0	255.255.255.0	131.175.21.2	x
131.175.17.0	255.255.255.0	131.175.21.3	x
131.180.23.0	255.255.255.0	131.175.21.4	x
131.180.18.0	255.255.255.0	131.175.21.4	x
131.180.21.0	255.255.255.0	131.175.21.4	x
131.180.0.0	255.255.0.0	131.175.21.5	x
0.0.0.0	0.0.0.0	131.175.12.254	OK

200.45.21.84

interfaccia 1: 131.175.21.254, 255.255.255.0
interfaccia 2: 131.175.12.254, 255.255.255.0

Supernetting

- La netmask può essere vista non solo come un modo per creare un campo sottorete, ma più in generale come un modo per creare un confine variabile tra il campo NetID e il campo HostID



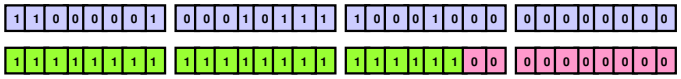
- Mediante la netmask dunque è possibile raggruppare più indirizzi in classe C per formare una rete più grande

Supernetting

- Se ad esempio sono solo disponibili indirizzi in classe C e una organizzazione ha bisogno di circa 1000 indirizzi è possibile raggruppare 4 reti in classe C contigue a formare una super-rete con 1024 indirizzi:

193.23.136.0
193.23.137.0
193.23.138.0
193.23.139.0

193.23.136.0/22



Routing classless

- I router esterni alla super-rete potranno avere una riga per ciascuna delle reti in classe C o una sola per la super-rete (routing classless)
- con l'indirizzamento classless le classi non hanno più significato e i gruppi di indirizzi vengono assegnati come coppie indirizzo+netmask
- è possibile dunque assegnare un numero di indirizzi (potenza di 2) in modo flessibile
- una volta assegnato il gruppo di indirizzi all'interno della intranet è possibile usare un'altra netmask più corta per suddividere la rete in sottoreti

The 1992 Internet scenario

- Near-term exhaustion of class B address space
  - In early years, Class B addresses given away!
  - Unefficient division into A, B, C classes
    - byte-word: unwise choice (class C too little, class B too big)
    - The aftermath: much better, e.g. C=10 bits, B=14 bits
  - Projections at the time: class B exhaustion by 1994/95

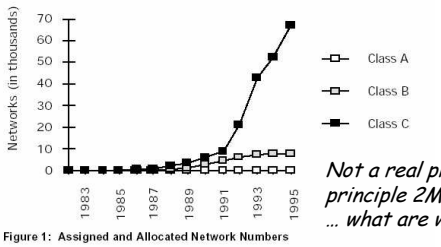


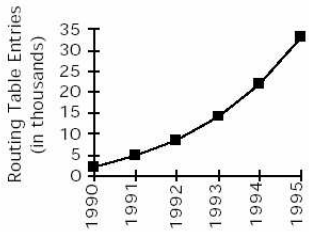
Figure 1: Assigned and Allocated Network Numbers

Not a real problem: there are in principle 2M class C addresses! … what are we missing??

The 1992 Internet scenario

➔ **Exponential growth of routing tables**

- ⇨ Multiple class C allocation dramatic for routing tables
  - → necessary because of Class B exhaustion
  - → 100.000 entries highly critical for performance
    - » *2M class C: WAY OUT of the capabilities of routing sw & hw*



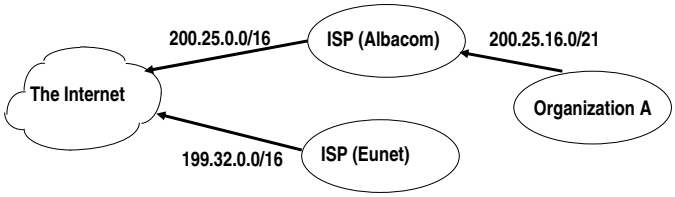
⇨ Projections at the time
- → End 1990: 2190 routes; end 1992: 8500 routes;
- → End 1995 projection: 70000 routes (critical);
- → End 1995 factual: 30000 routes thanks to classless routing
- → Mid 1999: 50000 routes

Figure 2: Growth of Internet Routing Tables

CIDR—basics (a temporary solution)

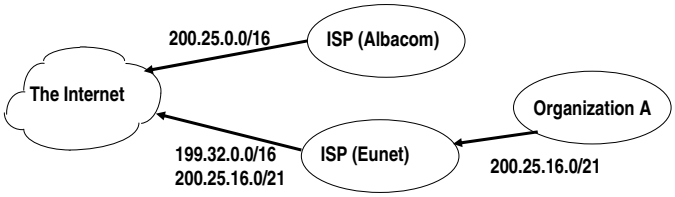
- Residual IP addresses assigned to ISP; ISP allocate bunch of contiguous IP addresses to a customer based on geografic customer location, and needs (simpler decentralized allocation of IP addresses, summarization→shorter routing tables; better use of available IP addresses)
- ISP advertize: IP addresses allocated to them, IP addresses of their customers not in their bunch (of networks which got the IP address from a different ISP but that moved to this ISP), multi-homed stub networks for which it acts as backup ...
- longer prefix based routing: out of the network addresses in the routing table matching the request the one with the longer common prefix is used for sake of routing (see previous examples).

Common exception route case



- At a point in time, organization A selects Eunet as new ISP!
  - Best thing to do (for the Internet): obtain a new block of addresses and renumber
  - virtually impossible for a reasonably complex organization...
    - and even think to organizations that re-sells subnets...

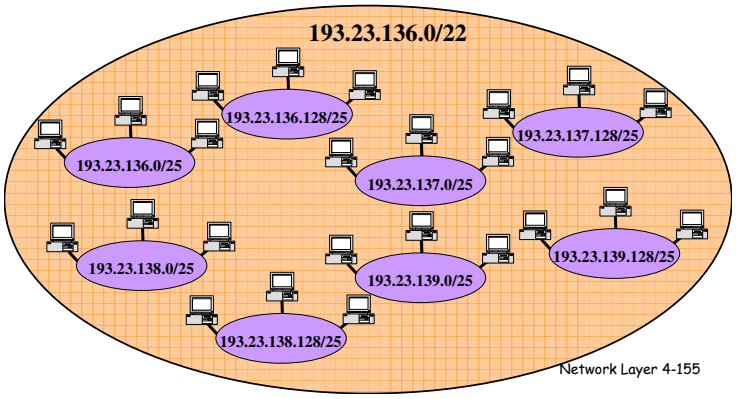
Common exception route case



- Then organization A keeps the same address block
  - Eunet is in charge to advertise the new block, too, by injecting in the internet more specific route infos
  - This has created a new entry in routing tables, to be solved with longest match

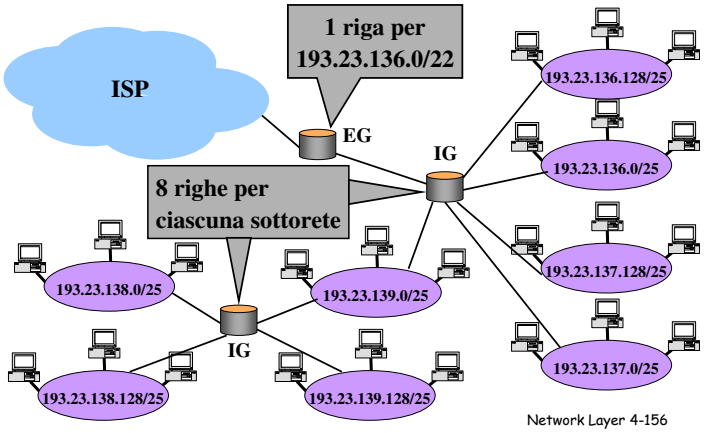
Routing classless

- La super-rete da 1022 indirizzi è suddivisa in 8 sottoreti da 126 indirizzi



Routing classless

- Esempio di una possibile architettura:



IP addresses: how to get one?

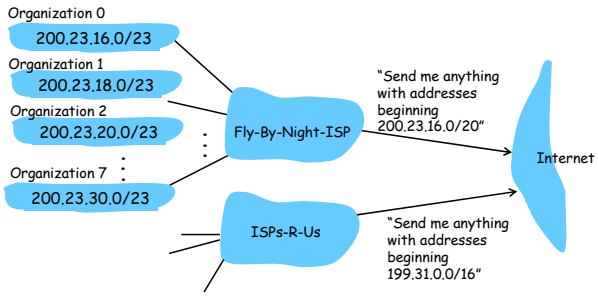
Q: How does *network* get network part of IP addr?

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

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

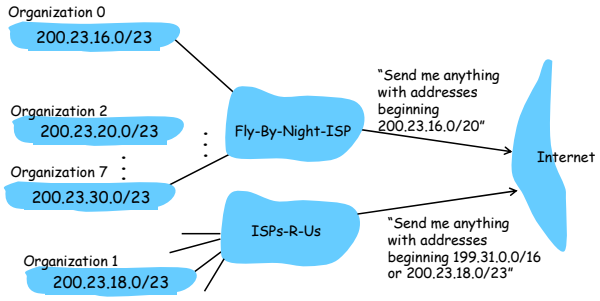
Hierarchical addressing: route aggregation

Hierarchical addressing allows efficient advertisement of routing information:



Hierarchical addressing: more specific routes

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



IP addresses: how to get one?

Q: How does *host* get IP address?

- hard-coded by system admin in a file
  - Wintel: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  - "plug-and-play"

  (more shortly)

Intranet IP address

- Intranets can reuse the same IP addressing space (not ambiguous)
- No need to 'ask for a part of the IP addressing space'
- Class A net 10 used for Intranet purposes
- What does it mean? Can we have a PC in an Intranet in Rome been labeled 10.0.0.5 and use the same address for a PC in an Intranet in Paris? Yes.

IP addressing: the last word...

Q: How does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers
- allocates addresses
- manages DNS
- assigns domain names, resolves disputes

Getting a datagram from source to dest.

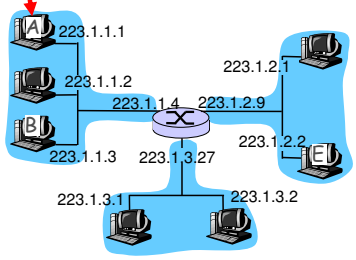
IP datagram:

misc fields	source IP addr	dest IP addr	data

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

forwarding table in A

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



Getting a datagram from source to dest.

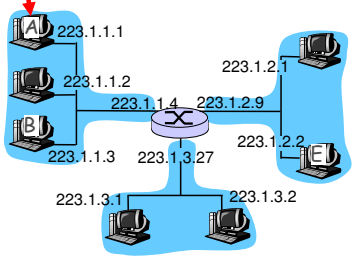
misc fields	223.1.1.1	223.1.1.3	data

Starting at A, send IP datagram addressed to B:

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

forwarding table in A

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



Getting a datagram from source to dest.

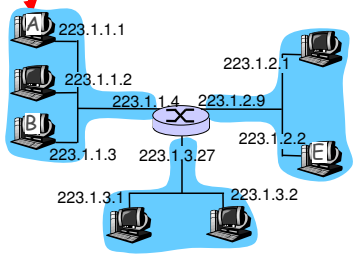
misc fields	223.1.1.1	223.1.2.3	data

Starting at A, dest. E:

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

forwarding table in A

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



Getting a datagram from source to dest.

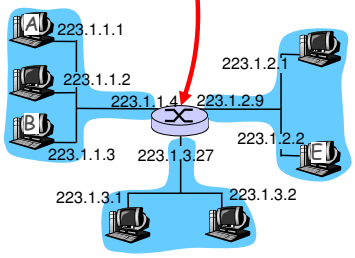
misc fields	223.1.1.1	223.1.2.3	data

Arriving at 223.1.4, destined for 223.1.2.2

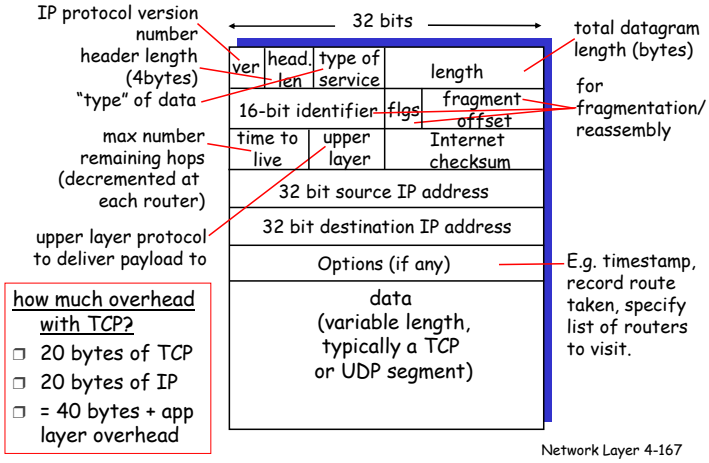
- look up network address of E in router's forwarding table
- E on *same* network as router's interface 223.1.2.9
  - router, E directly attached
- link layer sends datagram to 223.1.2.2 inside link-layer frame via interface 223.1.2.9
- datagram arrives at 223.1.2.2!!! (hooray!)

forwarding table in router

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



IP datagram format



how much overhead with TCP?

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

ICMP: Internet Control Message Protocol

- used by hosts, routers, gateways to communication network-level information
  - error reporting: unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- network-layer "above" IP:
  - ICMP msgs carried in IP datagrams
- ICMP message: type, code plus first 8 bytes of IP datagram causing error

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

La trama (pacchetto) IP

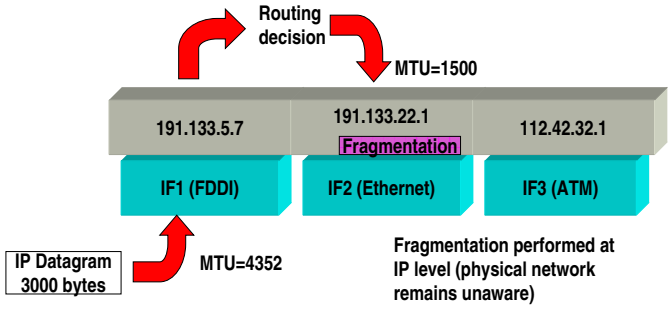
- Ver (4 bit):
  - version: indica la versione del protocollo; quella che noi studiamo è la versione 4
- HLEN (4 bit)
  - header length: indica la lunghezza dell'header del pacchetto espressa in parole da 32 bit
- TOS (8 bit)
  - type of service: un campo che adesso prende il nome di DS field e può essere per la gestione delle priorità nelle code dei router.
  - Originariamente: tre bit di precedenza (0 normale, 7 pacchetti di controllo), tre bit di flag (delay, throughput, reliability) per indicare quale/i aspetto più importante, 2 bit inutilizzati.
- Total length (16 bit):
  - indica la lunghezza totale del pacchetto in byte: valore massimo 2^{16}=65536; una volta sottratta la dimensione dell'header dà la lunghezza del payload

La frammentazione

- Identification, Flags, Frag. Offset
  - Alcuni protocolli di livello inferiore a cui IP si appoggia richiedono una dimensione massima del pacchetto (MTU) inferiore a 65536 bytes (tipico l'esempio di Ethernet che accetta pacchetti fino a 1500 bytes). Diversi tipi di link hanno diverse MTU.
  - prima di passare il pacchetto al livello inferiore IP divide il pacchetto in frammenti ciascuno con il suo header
  - i frammenti verranno ricomposti dall'entità IP del destinatario
  - i campi Identification, Flags e Frag. Offset sono usati per questo scopo

Why fragmentation

physical networks have different Maximum Transmission Units (MTU)

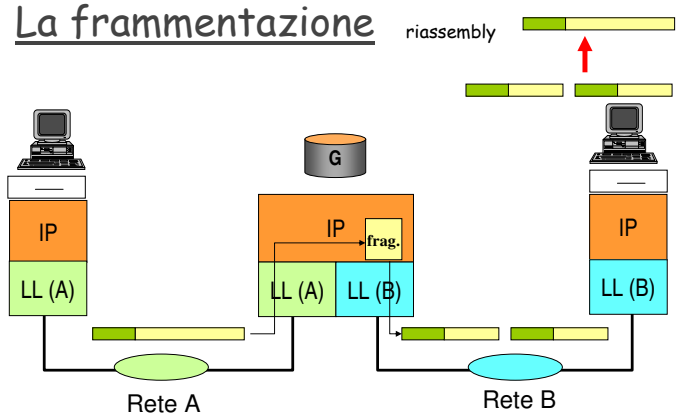


MTU examples

RFC 1191

Network	MTU (bytes)
IBM token ring 16Mbps	17914
4 Mbps Token Ring (IEEE 802.5)	4464
FDDI	4352
Ethernet	1500
IEEE 802.2 802.3	1492
X.25	576
point to point (PPP, SLIP)	May be set to 296 for interactive use

La frammentazione



La frammentazione

- Identification
  - è un campo che identifica tutti i frammenti di uno stesso pacchetto in modo univoco. E' scelto dall'IP che effettua la frammentazione
- Frag. Offset
  - I byte del pacchetto originale sono numerati da 0 al valore della lunghezza totale. Il campo Frag. Offset di ogni frammento riporta il numero di sequenza del primo byte del frammento.
  - Ad esempio: se un pacchetto di 2000 byte viene diviso in due da 1000 il primo frammento avrà un offset pari a 0 e il secondo pari a 1000

La frammentazione

DF=don't fragment
MF=more fragments

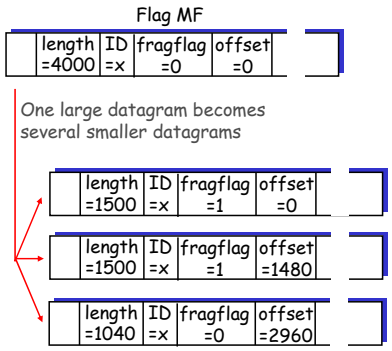
- Flags

-	DF	MF

  - il bit M (More) è pari a 0 solo nell'ultimo frammento
  - il bit D viene posto a 1 quando non si vuole che lungo il percorso venga applicata la frammentazione
    - in questo caso se la frammentazione fosse necessaria non viene applicata ma viene generato un messaggio di errore per indicare il mancato corretto recapito del pacchetto.

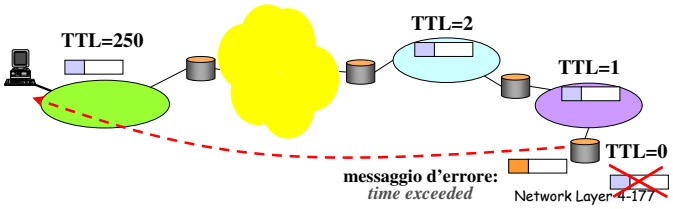
IP Fragmentation and Reassembly

Example
- 4000 byte datagram
- MTU = 1500 bytes



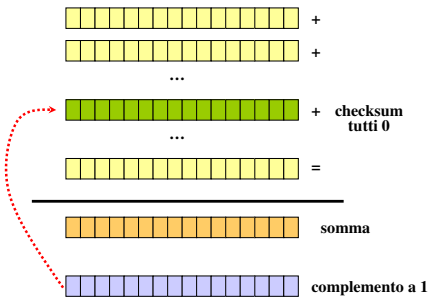
TTL (Time To Live)

- Il campo TTL viene settato ad un valore elevato da chi genera il pacchetto e viene decrementato da ogni router attraversato
- Se un router decrementa il valore e questo va a zero, il pacchetto viene scartato e viene generato un messaggio di errore verso la sorgente



Checksum

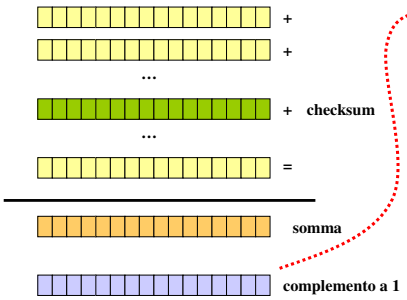
- serve per individuare eventuali errori nell'header
- viene calcolato dal mittente e controllato dal destinatario



- l'header viene diviso in blocchi di 16 bit
- viene fatta la somma modulo 2 dei bit corrispondenti in ciascun blocco
- il risultato viene complementato e inserito nel campo checksum

Checksum

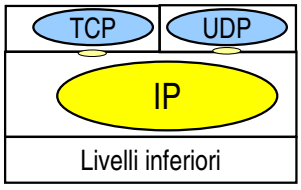
- In ricezione si calcola la somma e si verifica il complemento:



- se sono tutti 0 il pacchetto viene accettato
- altrimenti viene scartato!

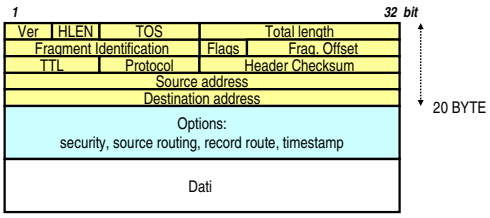
Protocol

- E' un codice che indica il protocollo di livello superiore
- più protocolli di livello superiore possono usare IP (multiplazione)
- il codice identifica il SAP (Service Access Point) tra IP e il protocollo di livello superiore

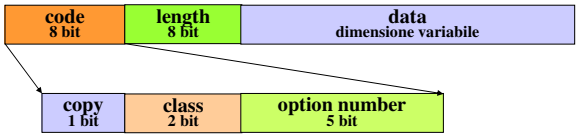


Le opzioni

- La parte iniziale dell'header IP è di 20 byte ed è sempre presente
- In aggiunta è possibile la presenza di campi aggiuntivi (le opzioni) che possono allungare l'header fino ad un massimo di 60 byte



Le opzioni

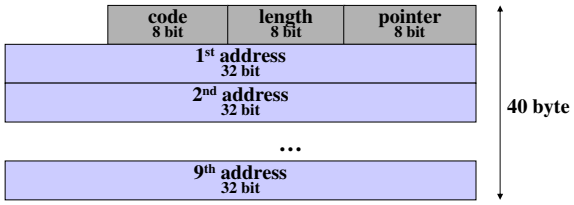


Must Copy:
0 nel primo o unico frammento
1 anche negli altri

Class:
00 controllo del datagram
10 debugging e misure

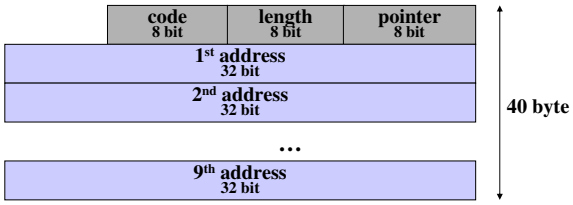
Option number:
00000 end of option (1 byte)
00001 no operation (1 byte) tra opzioni per allin earsi all'inizio di una parola
00011 loose source route
00100 time stamp
00111 record route
01001 strict source route

Record route



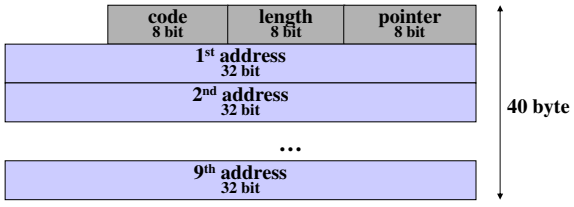
- Tutti i campi address sono inizialmente vuoti e il pointer vale 4 (punta al primo campo address)
- ogni volta che viene attraversato un router viene registrato l'indirizzo nel campo puntato e il puntatore viene aumentato di 4
- (per conoscere il percorso verso una destinazione esiste la possibilità di usare pacchetti ICMP come vedremo in seguito)

Strict source route



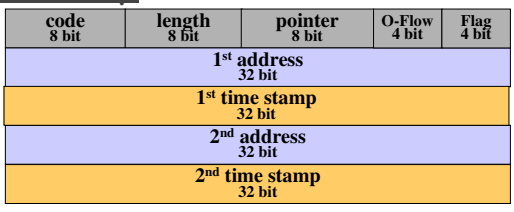
- Implementa un meccanismo di source routing (percorso scelto dalla sorgente)
- Tutti i campi address sono inizialmente pieni e indicano i router che si vuole vengano attraversati
- il puntatore viene incrementato di 4 ad ogni hop
- se viene raggiunto un router non previsto il pacchetto viene scartato e viene generato un messaggio di errore
- (usata molto raramente!!!)

Loose source route



- come la precedente, ma è possibile visitare anche altri router (il pacchetto non viene scartato)
- (usata molto raramente!!!)

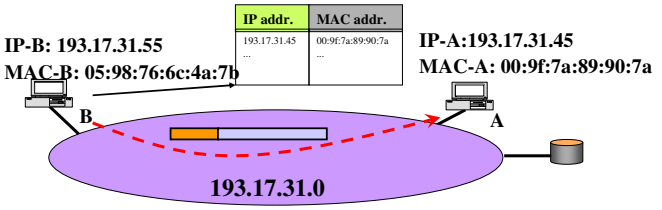
Time stamp



- misura il tempo assoluto di uscita del pacchetto in un router
- il campo Over-Flow indica i router sul percorso che non hanno potuto aggiungere il timestamp
- il campo Flag indica la modalità operativa stabilita dal mittente (address riempiti dal mittente o dai router, solo timestamp o IP address+timestamp ecc.)
- N.B. In generale un problema la dimensione limitata delle opzioni (<=40 bytes)!!

Corrispondenza tra indirizzi IP e indirizzi fisici

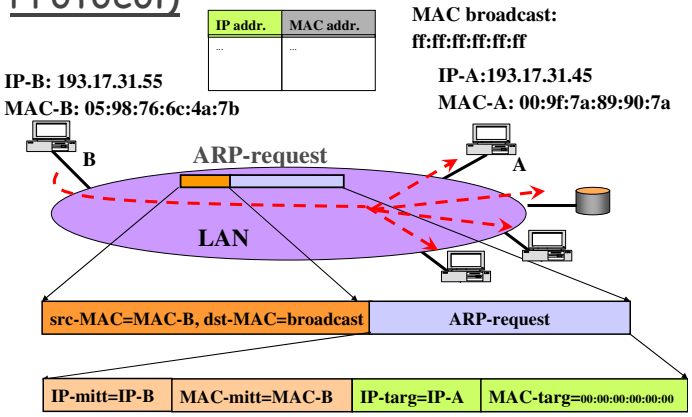
- Illustrando le tecniche di inoltro abbiamo ipotizzato la presenza di una tabella di corrispondenza tra indirizzi IP e indirizzi di livello inferiore (indirizzi fisici)
- Queste tabelle vengono create dinamicamente da ciascun host mediante il protocollo ARP



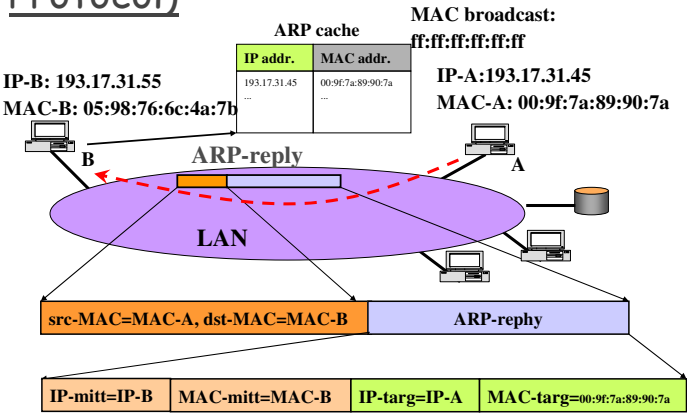
ARP (Address Resolution Protocol)

- Il meccanismo si basa sulla capacità di indirizzamento broadcast della rete locale
- quando nella tabella memorizzata nell'host (denominata ARP-cache) non è presente l'indirizzo cercato viene generato un messaggio di ARP-request
- La ARP-request viene inviata in broadcast e contiene l'indirizzo IP di cui si chiede il corrispondente indirizzo MAC
- L'host che riconosce l'indirizzo IP come proprio invia una ARP-reply direttamente a chi aveva inviato la richiesta con l'indicazione dell'indirizzo MAC

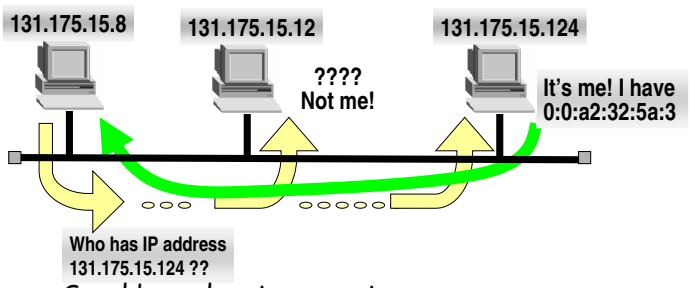
ARP (Address Resolution Protocol)



ARP (Address Resolution Protocol)



ARP idea



- Send broadcast request
- receive unicast response

Ottimizzazioni: caching dei mapping ricevuti da parte di tutti gli host
Caching del mapping MAC address—IP address della sorgente

Formato dei pacchetti ARP

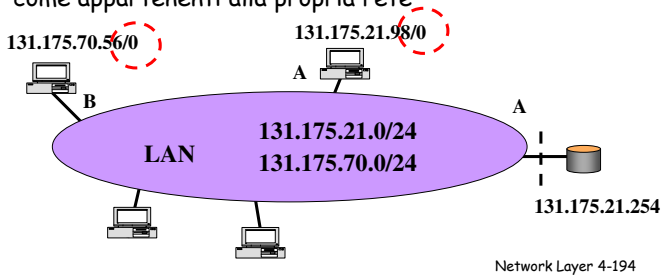
- **ARP può essere usato per altri protocolli di livello 2 e livello 3 quindi occorre indicare il tipo di protocollo (IP nel nostro caso) e il tipo di hardware (ethernet per esempio)**

Domini di broadcast e reti IP

- Per il funzionamento del meccanismo di inoltro e dell'ARP abbiamo fin qui ipotizzato che una sottorete IP corrisponda uno a uno con una rete locale (Dominio di Broadcast)
- In realtà un'unica rete locale può corrispondere a diverse sottoreti IP (per es. perché la numerazione disponibile per una non è sufficiente)
- Non è possibile che più reti locali possano coesistere in un'unica sottorete IP perché non potrebbero comunicare

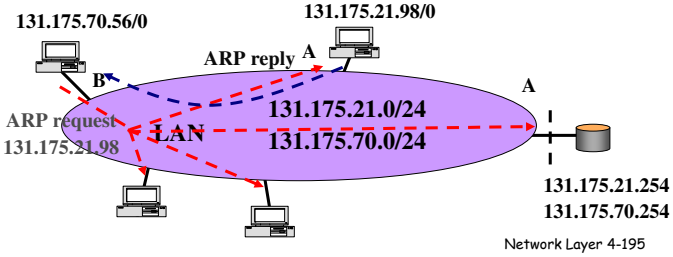
Domini di broadcast e reti IP: proxy ARP

- Un alternativa è quella dell'installazione di un *proxy ARP* nel router
- Tutti gli host della rete vengono configurati con una netmask di tutti 0
- Come conseguenza considerano tutti gli indirizzi come appartenenti alla propria rete



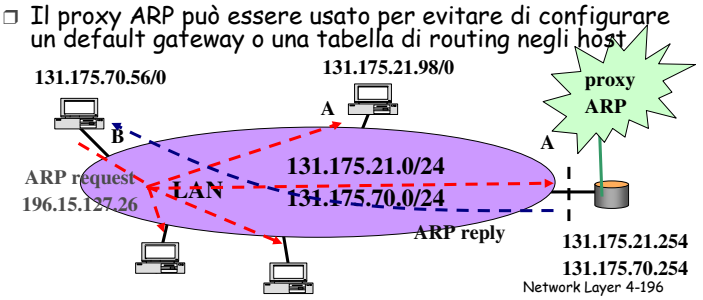
Domini di broadcast e reti IP: proxy ARP

- Quindi, quando deve essere spedito un pacchetto (anche esterno alla rete) viene comunque prima inviata una ARP-request
- se la richiesta è per un indirizzo di uno degli host sulla stessa rete locale (dominio di broadcast), sarà l'host destinatario a rispondere con un ARP-reply



Domini di broadcast e reti IP: proxy ARP

- Al contrario, se la richiesta è per un indirizzo esterno alla rete locale sarà il proxy ARP del router a rispondere con l'indirizzo MAC dell'interfaccia del router
- Per funzionare il proxy ARP ha solo bisogno di conoscere gli indirizzi delle reti raggiunte dall'interfaccia e le relative netmask
- Il proxy ARP può essere usato per evitare di configurare un default gateway o una tabella di routing negli host



ARP cache

- Avoids arp request for every IP datagram!
  - Entry lifetime defaults to 20min
    - deleted if not used in this time
    - 3 minutes for "incomplete" cache entries (i.e. arp requests to non existent host)
    - it may be changed in some implementations
      - in particularly stable (or dynamic) environments
  - ***arp -a*** to display all cache entries

try a traceroute or ping to check ARP caching!

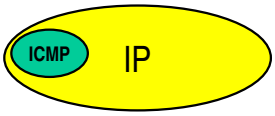
- First packet generally delays more
- includes an ARP request/reply!

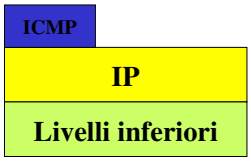
Gratuitous ARP

- ARP request issued by an IP address and addressed to *the same* IP address!!
  - Clearly nobody else than ME can answer!
  - WHY asking the network which MAC address do I have???
- Two main reasons:
  - determine if another host is configured with the same IP address
    - in this case respond occurs, and MAC address of duplicated IP address is known.
  - Use gratuitous ARP when just changed hardware address
    - all other hosts update their cache entries!
    - A problem is that, despite specified in RFC, not all ARP cache implementations operate as described….

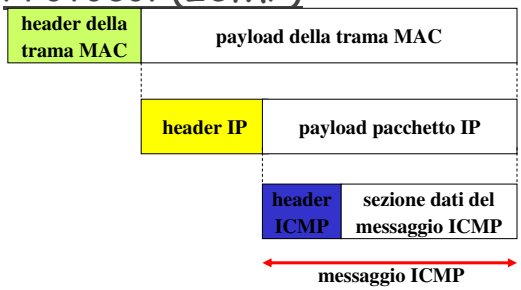
Internet Control Message Protocol (ICMP)

- E' un protocollo per messaggi di servizio fra host e router per informazioni su errori e fasi di attraversamento della rete
- da questo punto di vista può essere considerato come parte di IP
- i messaggi ICMP sono incapsulati e trasportati da IP, e quindi da questo punto di vista può essere considerato un utente di IP



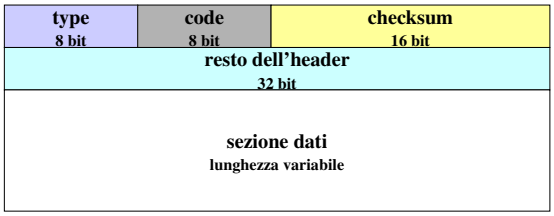


Internet Control Message Protocol (ICMP)



- Nel pacchetto IP il campo protocol indica il codice dell'ICMP
- il messaggio ICMP viaggia all'interno del pacchetto IP

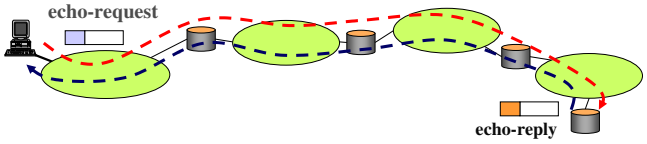
Messaggi ICMP



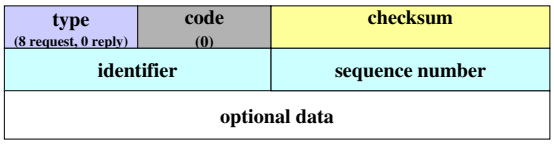
Type		Type	
0	Echo reply	11	Parameter problem
3	Destination unreachable	13	Timestamp request
4	Source Quench	14	Timestamp reply
5	Redirect (change a route)	17	Address mask request
8	Echo request	18	Address mask reply
11	Time exceeded		

Echo

- I messaggi di Echo-request e Echo-reply sono usati per verificare la raggiungibilità e lo stato di un host o un router
- quando un nodo IP riceve un messaggio di Echo-request risponda immediatamente con un messaggio di Echo reply

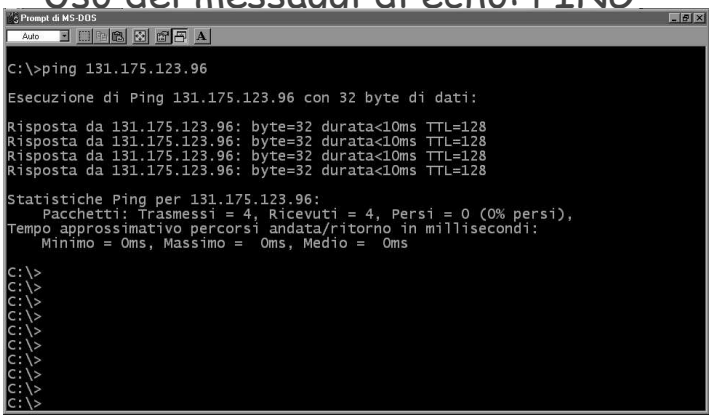


Echo



- Il campo identifier viene scelto dal mittente della richiesta
- nella risposta viene ripetuto lo stesso identifier della richiesta
- più richieste consecutive possono avere lo stesso identifier e differire per il sequence number
- una sequenza arbitraria può essere aggiunta dal mittente nel campo optional data e deve essere riportata uguale nella risposta

Uso dei messaggi di echo: PING



Destination unreachable

type (3)	code (0-12)	checksum
non usato (0)		
header + primi 64 bit del pacchetto IP che ha causato il problema		

- Quando un router scarta un pacchetto per qualche motivo normalmente genera un messaggio di errore che invia alla sorgente del pacchetto
- nel campo code è codificato il motivo che ha causato l'errore
- ovviamente la generazione del messaggio avviene solo nei casi in cui il router può accorgersi del problema
- il motivo più comune è il fatto che la destinazione non è presente nella tabella di routing. Un messaggio destination unreachable può anche essere inviato se è necessario frammentare un pacchetto ed il flag DF è a 1.

Destination unreachable

type (3)	code (0-12)	checksum
non usato (0)		
header + primi 64 bit del pacchetto IP che ha causato il problema		

Alcuni Code:

0 network unreachable
1 host unreachable
2 protocol unreachable
3 port unreachable
4 fragmentation needed and DF set
5 source route failed
...

Time exceeded

type (11)	code (0-1)	checksum
non usato (0)		
header + primi 64 bit del pacchetto IP che ha causato il problema		

- Code 0
  - Il messaggio di time exceeded viene usato quando il router decrementando il TTL lo pone a 0
  - il messaggio di time exceeded viene inviato alla sorgente del pacchetto
- Code 1
  - viene usato dalla destinazione quando non tutti i frammenti di un pacchetto arrivano entro un tempo massimo

Parameter problem

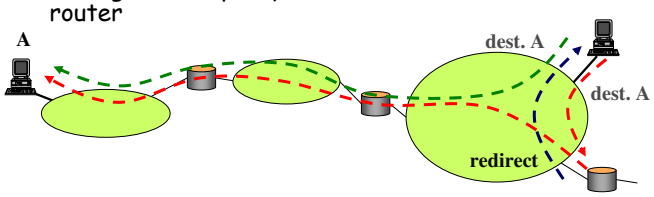
type (12)	code (0-1)	checksum
pointer	non usato (0)	
header + primi 64 bit del pacchetto IP che ha causato il problema		

- Code 0
  - se l'header di un pacchetto IP ha una incongruenza in qualcuno dei suoi campi viene inviato il messaggio di parameter problem; il campo pointer punta al byte del pacchetto che ha causato il problema
- Code 1
  - viene usato quando un'opzione non è implementata e non può essere soddisfatta

Redirect

type (5)	code (0-3)	checksum
indirizzo IP del router		
header + primi 64 bit del pacchetto IP		

- Questo messaggio viene usato quando si vuole che la sorgente usi per quella destinazione un diverso router



Timestamp request e reply

type (13 request, 14 reply)	code (0)	checksum
identifier	sequence number	
originate timestamp		
receive timestamp		
transmit timestamp		

- Questo messaggio viene usato per scambiarsi informazioni sul clock di sorgente e destinazione
- *originate timestamp:* viene riempito dalla sorgente
- *receive timestamp*: viene riempito dalla destinazione appena ricevuto il pacchetto
- *transmit timestamp:* viene riempito dalla destinazione immediatamente prima di inviare il pacchetto di risposta

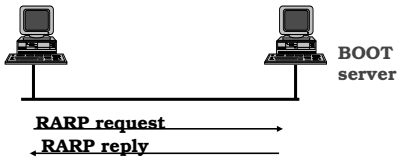
Address mask request e reply

type (17 request, 18 reply)	code (0)	checksum
identifier		sequence number
address mask		

- Questo messaggio viene usato per conoscere la netmask di un host/router
- Il campo address mask viene riempito da invia la risposta

RARP (Reverse ARP)

- Il protocollo ARP consente di associare ad un indirizzo IP noto un indirizzo fisico non noto usando la capacità di broadcast della rete sottostante
- il protocollo RARP (Reverse ARP) è in grado di effettuare l'operazione inversa:
  - un host che conosce il proprio indirizzo fisico chiede di sapere il proprio indirizzo IP
  - utile per macchine diskless che effettuano il bootstrap in rete
  - *ma non è più usato !!!*



RARP problems

- Network traffic
  - for reliability, multiple RARP servers need to be configured on the same Ethernet
    - to allow bootstrap of terminals even when one server is down
  - But this implies that ALL servers simultaneously respond to RARP request
    - contention on the Ethernet occurs
- RARP requests not forwarded by routers
  - being hardware level broadcasts...

RARP fundamental limit

- Allows only to retrieve the IP address information
  - and what about all the remaining full set of TCPIP configuration parameters???
    - Netmask?
    - name of servers, proxies, etc?
    - other proprietary/vendor/ISP-specific info?
- This is the main reason that has driven to engineer and use BOOTP and DHCP

BOOTP/DHCP approach

- Requests/replies encapsulated in UDP datagrams
  - may cross routers
  - no more dependent on physical medium
- request addressing:
  - destination IP = 255.255.255.255
  - source IP = 0.0.0.0
  - destination port (BOOTP): 67
  - source port (BOOTP): 68
- router crossing:
  - router configured as BOOTP relay agent
  - forwards broadcast UDP requests with destination port 67

BOOTP parameters exchange

- Many more parameters
  - client IP address (when static IP is assigned)
  - your IP address (when dynamic server assignment)
  - gateway IP address (bootp relay agent - router - IP)
  - server hostname
  - boot filename
- Fundamental: vendor-specific information field (64 bytes)
  - seems a lot of space: not true!
  - DHCP uses a 312 vendor-specific field!

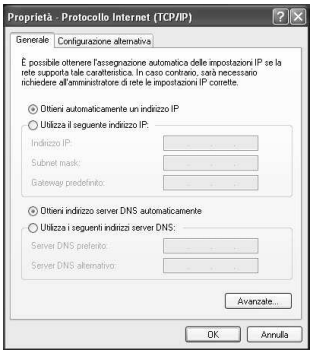
Indirizzi dinamici

- l'uso di procedure di questo tipo ha suggerito la possibilità di usare procedure per associare in modo flessibile gli indirizzi IP agli indirizzi fisici
- può essere comodo non configurare i singoli host con l'indirizzo IP, ma usare un server per memorizzare tutte le configurazioni
- in molti casi non è necessario avere un'associazione stabile tra i due indirizzi ma si può usare un'associazione dinamica (più host degli indirizzi disponibili):
  - host spesso inattivi (es. collegamenti remoti con rete d'accesso telefonica)
  - host che usano IP solo per rari scambi di informazioni
  - Mobilità degli utenti (casa, campus)

Indirizzi dinamici

- Supponiamo di avere un server in grado di fornire l'indirizzo IP ad un host su richiesta
- sono possibili diversi casi:
  - associazioni statica: il server ha una tabella di corrispondenza tra indirizzi fisici e indirizzi IP e all'arrivo di una richiesta consulta la tabella e invia la risposta
  - associazione automatica: la procedura di corrispondenza nella tabella è automatizzata dal server
  - associazione dinamica: l'insieme di indirizzi IP è più piccolo degli host che possono usarlo

Indirizzi dinamici

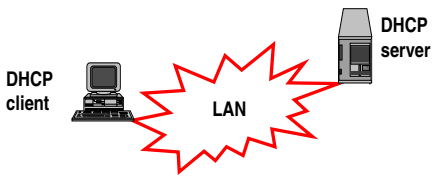


Associazione dinamica

- Il caso dell'allocazione dinamica è utile in situazioni nelle quali gli host non necessitano di avere sempre un indirizzo IP
- L'associazione deve essere temporanea (uso di timeout o procedure di rilascio esplicito)
- è possibile che all'arrivo di una richiesta non vi siano indirizzi disponili (rifiuto della richiesta)
- il dimensionamento del numero di indirizzi IP segue gli stessi principi del dimensionamento di un fascio di circuiti in telefonia

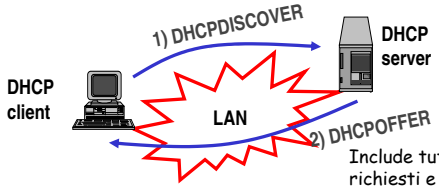
Dynamic Host Configuration Protocol (DHCP)

- per la configurazione di indirizzi IP non si usa il RARP, ma un protocollo più evoluto derivato dal BOOTP
- è un protocollo di tipo client-server



DHCP

- Un client che deve configurare il proprio stack IP invia in broadcast un messaggio di DHCPDISCOVER contenete il proprio indirizzo fisico
- Il server risponde con un messaggio di DHCPOFFER contente un proprio identificativo e un indirizzo IP proposto

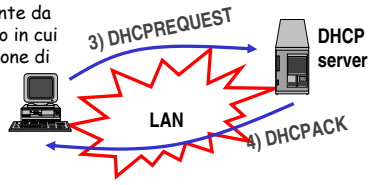


Include tutti i parametri richiesti e il tempo di durata del lease

DHCP

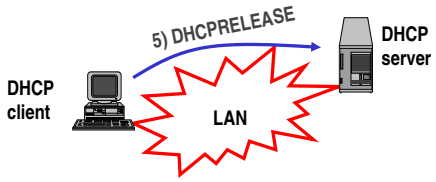
- Il client può accettare l'offerta inviando una DHCPREQUEST contenente l'identificativo del server (anche questo messaggio viene inviato in broadcast)
- Il server crea l'associazione con l'indirizzo IP e manda un messaggio di DHCPACK contenente tutte le informazioni di configurazione necessarie

Si parte direttamente da questa fase nel caso in cui si richieda l'estensione di un lease



DHCP

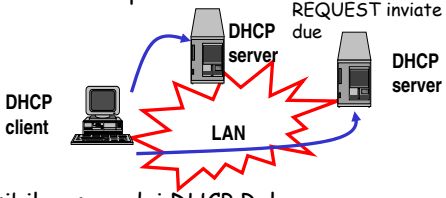
- Parametri di configurazione
  - IP address
  - Netmask
  - Gateway
  - DNS server
- Il rilascio dell'indirizzo avviene con l'invio di un messaggio di DHCPRELEASE da parte del client



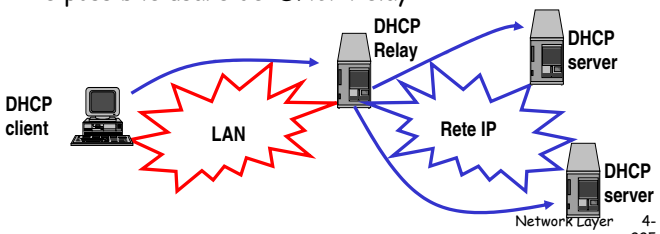
DHCP

- è possibile avere più server:

Il DHCP client può scegliere tra i parametri di config. 'proposti' dai sue server. Il messaggio di REQUEST inviate solo ad uno dei due

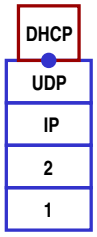


- è possibile usare dei DHCP Relay



Trasporto dei messaggi

- DHCP si appoggia su UDP per il trasporto dei messaggi
- I messaggi dei client fino all'assegnamento dell'indirizzo IP hanno:
  - ind. di sorgente: 0.0.0.0
  - ind. di destinazione: 255.255.255.255
  - porta sorgente: 68
  - porta destinazione: 67



Messaggi

OP	HTYPE	HLEN	HOPS
XID			
SECS		FLAGS	
CIADDR			
YIADDR			
SIADDR			
GIADDR			
CHADDR			
SNAME			
FILE			
OPTIONS			

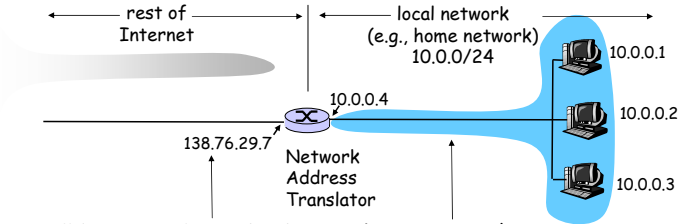
CAMPO	BYTE	DESCRIZIONE
op	1	Tipo di messaggio (1 = BOOTREQUEST, 2 = BOOTREPLY)
htype	1	Tipo di indirizzo fisico (1 = Eth 10Mb)
hlen	1	Lunghezza ind. fisico ('6' per Eth 10Mb)
hops	1	Settato dal client a 0 e incrementato dai relay agests
xid	4	Numero casuale settato dal client e usato per evitare ambiguit\`a
secs	2	Settato dal client, numero di sec dall'inizio della procedura
flags	2	Flags (si usa solo il primo bit per chiedere una risposta multicast o unicast).
ciaddr	4	Indirizzo IP del client (settato dal client, zero se non noto)
yiaddr	4	Indirizzo IP del client (settato dal server)
siaddr	4	Indirizzo IP del server
giaddr	4	Indirizzo del relay agent
chaddr	16	Indirizzo fisico del client
sname	64	Stringa Nome del server (opzionale)
file	128	Stringa nome del file di boot (opzionale)
options	312	Lista di opzioni per il trasferimento di altre informazioni

associa richieste risposte

e.g. DHCP message type

NAT: Network Address Translation

Short term solution to IP addresses depletion. Idea: address reuse!!
A network Address Translator is placed at the borders of stub domains. Inside The domain local IP addressing is used (i.e. these IP addresses are reused). Only one globally unique IP address needed at the border. The NAT will be able to translate from the IP address and port to local address and port number.



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

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

NAT: Network Address Translation

- Motivation: local network uses just one IP address as far as outside word is concerned:
  - no need to be allocated range of addresses from ISP: - just one IP address is used for all devices
  - can change addresses of devices in local network without notifying outside world
  - can change ISP without changing addresses of devices in local network
  - devices inside local net not explicitly addressable, visible by outside world (a security plus).
- Relies on the fact that a very small percentage of hosts in a stub domain are communicating outside of the domain at any given time. Has the disadvantage of taking away the end-to-end significance of an IP address, and making up for it with increased state in the network.

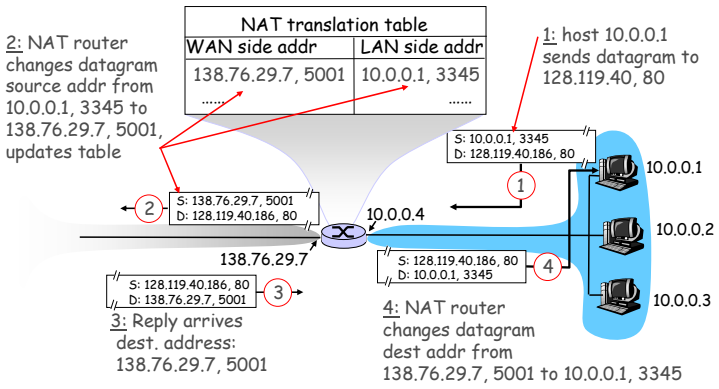
NAT: Network Address Translation

Implementation: NAT router must:

- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
  - . . . remote clients/servers will respond using (NAT IP address, new port #) as destination addr.
- *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 dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

N.B. In addition to modifying the IP address, NAT must _at least_ modify the IP checksum and the TCP checksum.

NAT: Network Address Translation



NAT: Network Address Translation

- 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 app designers, eg, P2P applications
  - Requires to translate any occurrence of the IP address; IP address cannot be encrypted; more difficult to identify the source of an attack if behind a NAT box.
  - address shortage should instead be solved by IPv6

Chapter 4 roadmap

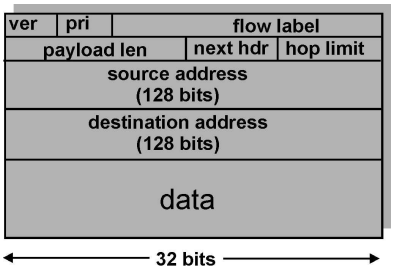
4.1 Introduction and Network Service Models
4.2 Routing Principles
4.3 Hierarchical Routing
4.4 The Internet (IP) Protocol
4.5 Routing in the Internet
4.6 What's Inside a Router?
4.7 IPv6
4.8 Multicast Routing
4.9 Mobility

IPv6

- Initial motivation: 32-bit address space completely allocated by 2008.
- Additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS
  - new "anycast" address: route to "best" of several replicated servers
- IPv6 datagram format:
  - fixed-length 40 byte header
  - no fragmentation allowed

IPv6 Header (Cont)

Priority: identify priority among datagrams in flow
Flow Label: identify datagrams in same "flow."
(concept of"flow" not well defined).
Next header: identify upper layer protocol for data



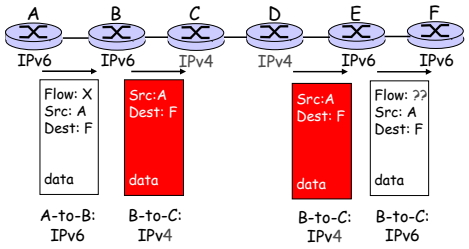
Other Changes from IPv4

- *Checksum*: removed entirely 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

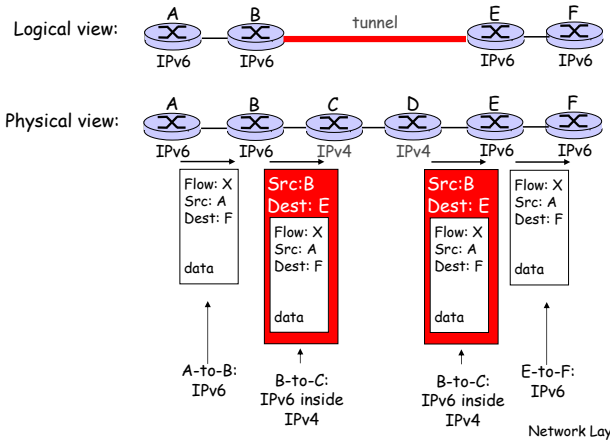
Transition From IPv4 To IPv6

- Not all routers can be upgraded simultaneous
  - no "flag days"
  - How will the network operate with mixed IPv4 and IPv6 routers?
- Two proposed approaches:
  - *Dual Stack*: some routers with dual stack (v6, v4) can "translate" between formats
  - *Tunneling:* IPv6 carried as payload in IPv4 datagram among IPv4 routers

Dual Stack Approach



Tunneling

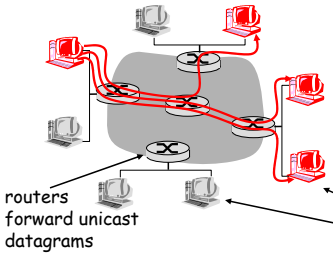


Chapter 4 roadmap

4.1 Introduction and Network Service Models
4.2 Routing Principles
4.3 Hierarchical Routing
4.4 The Internet (IP) Protocol
4.5 Routing in the Internet
4.6 What's Inside a Router?
4.7 IPv6
4.8 Multicast Routing
4.9 Mobility

Multicast: one sender to many receivers

- Multicast: act of sending datagram to multiple receivers with single "transmit" operation
  - analogy: one teacher to many students
- Question: how to achieve multicast

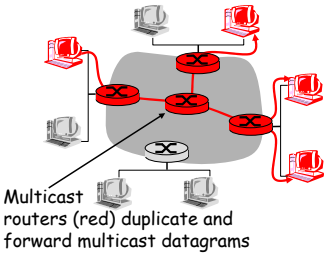


Multicast via unicast

- source sends N unicast datagrams, one addressed to each of N receivers

Multicast: one sender to many receivers

- Multicast: act of sending datagram to multiple receivers with single "transmit" operation
  - analogy: one teacher to many students
- Question: how to achieve multicast

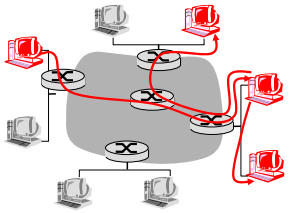


Network multicast

- Router actively participate in multicast, making copies of packets as needed and forwarding towards multicast receivers

Multicast: one sender to many receivers

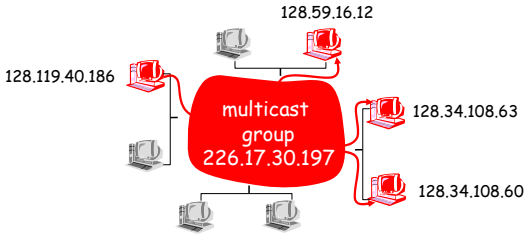
- Multicast: act of sending datagram to multiple receivers with single "transmit" operation
  - analogy: one teacher to many students
- Question: how to achieve multicast



Application-layer multicast

- end systems involved in multicast copy and forward unicast datagrams among themselves

Internet Multicast Service Model



multicast group concept: use of indirection

- hosts addresses IP datagram to multicast group
- routers forward multicast datagrams to hosts that have "joined" that multicast group

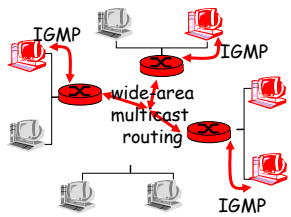
Multicast groups

- class D Internet addresses reserved for multicast:

- host group semantics:
  - anyone can "join" (receive) multicast group
  - anyone can send to multicast group
  - no network-layer identification to hosts of members
- *needed:* infrastructure to deliver mcast-addressed datagrams to all hosts that have joined that multicast group

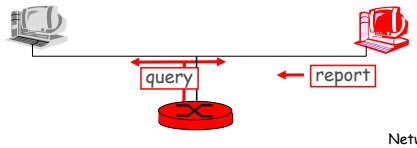
Joining a mcast group: two-step process

- *local:* host informs local mcast router of desire to join group: IGMP (Internet Group Management Protocol)
- *wide area:* local router interacts with other routers to receive mcast datagram flow
  - many protocols (e.g., DVMRP, MOSPF, PIM)



IGMP: Internet Group Management Protocol

- *host:* sends IGMP report when application joins mcast group
  - IP_ADD_MEMBERSHIP socket option
  - host need not explicitly "unjoin" group when leaving
- *router:* sends IGMP query at regular intervals
  - host belonging to a mcast group must reply to query



IGMP

IGMP version 1

- *router:* Host Membership Query msg broadcast on LAN to all hosts
- *host:* Host Membership Report msg to indicate group membership
  - randomized delay before responding
  - implicit leave via no reply to Query
- RFC 1112

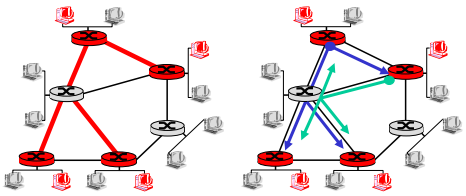
IGMP v2: additions include

- group-specific Query
- Leave Group msg
  - last host replying to Query can send explicit Leave Group msg
  - router performs group-specific query to see if any hosts left in group
  - RFC 2236

IGMP v3: under development as Internet draft

Multicast Routing: Problem Statement

- ***Goal:*** find a tree (or trees) connecting routers having local mcast group members
  - *tree:* not all paths between routers used
  - *source-based:* different tree from each sender to rcvrs
  - *shared-tree:* same tree used by all group members



Shared tree | Source-based trees

Approaches for building mcast trees

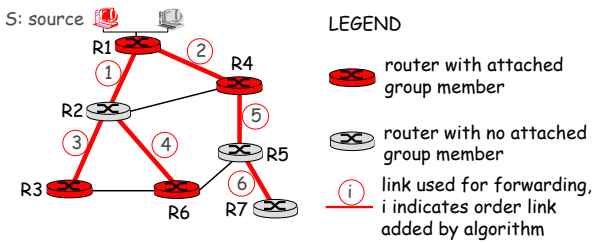
Approaches:

- source-based tree: one tree per source
  - shortest path trees
  - reverse path forwarding
- group-shared tree: group uses one tree
  - minimal spanning (Steiner)
  - center-based trees

…we first look at basic approaches, then specific protocols adopting these approaches

Shortest Path Tree

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

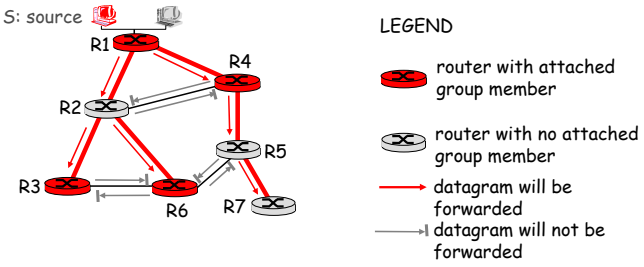


Reverse Path Forwarding

- 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 shortest path back to center)
> ***then*** flood datagram onto all outgoing links
> ***else*** ignore datagram

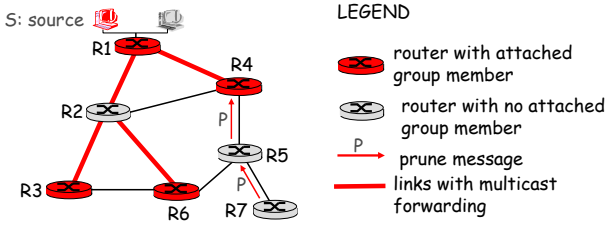
Reverse Path Forwarding: example



- result is a source-specific *reverse* SPT
  - may be a bad choice with asymmetric links

Reverse Path Forwarding: pruning

- forwarding tree contains subtrees with no mcast group members
  - no need to forward datagrams down subtree
  - "prune" msgs sent upstream by router with no downstream group members



Shared-Tree: Steiner Tree

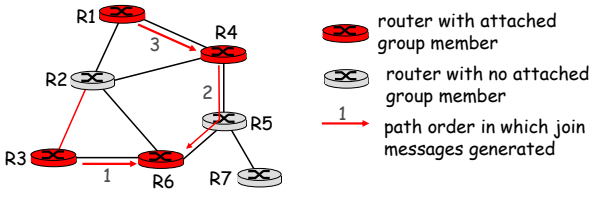
- Steiner Tree: minimum cost tree connecting all routers with attached group members
- problem is NP-complete
- excellent heuristics exists
- not used in practice:
  - computational complexity
  - information about entire network needed
  - monolithic: rerun whenever a router needs to join/leave

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

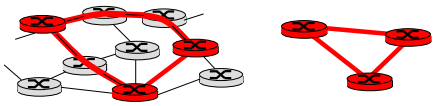
- DVMRP: distance vector multicast routing protocol, RFC1075
- *flood and prune:* reverse path forwarding, source-based tree
  - RPF tree based on DVMRP's own routing tables constructed by communicating DVMRP routers
  - no assumptions about underlying unicast
  - initial datagram to mcast group flooded everywhere via RPF
  - routers not wanting group: send upstream prune msgs

DVMRP: continued…

- *soft state:* DVMRP router periodically (1 min.) "forgets" branches are pruned:
  - mcast data again flows down unpruned branch
  - 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 routing done using DVMRP

Tunneling

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



physical topology logical topology

- mcast datagram encapsulated inside "normal" (non-multicast-addressed) datagram
- normal IP datagram sent thru "tunnel" via regular IP unicast to receiving mcast router
- receiving mcast router unencapsulates to get mcast datagram

PIM: Protocol Independent Multicast

- 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:
- # networks with group members small wrt # interconnected networks
- group members "widely dispersed"
- bandwidth not plentiful

Consequences of Sparse-Dense Dichotomy:

Dense
- group membership by routers *assumed* until routers explicitly prune
- *data-driven* construction on mcast tree (e.g., RPF)
- bandwidth and non-group-router processing *profligate*

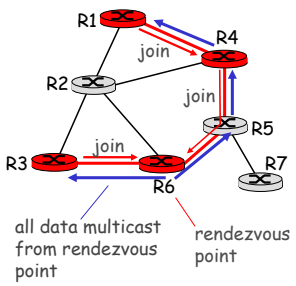
Sparse:
- no membership until routers explicitly join
- *receiver- driven* construction of mcast tree (e.g., center-based)
- bandwidth and non-group-router processing *conservative*

PIM- Dense Mode

flood-and-prune RPF, similar to DVMRP but
- underlying unicast protocol provides RPF info for incoming datagram
- less complicated (less 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

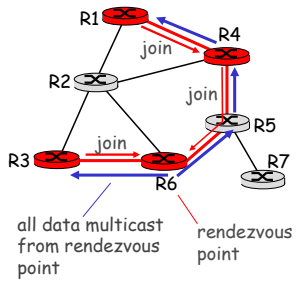
- center-based approach
- router sends *join* msg to rendezvous point (RP)
  - intermediate routers update state and forward *join*
- after joining via RP, router can switch to source-specific tree
  - increased performance: less concentration, shorter paths



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* msg if no attached receivers
  - "no one is listening!"

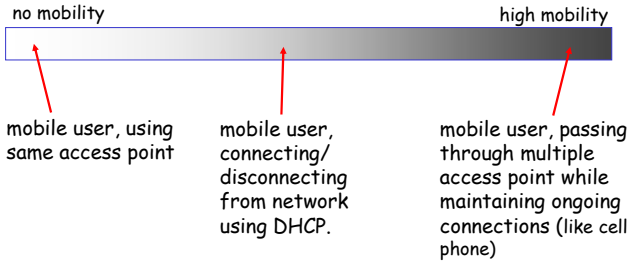


Chapter 4 roadmap

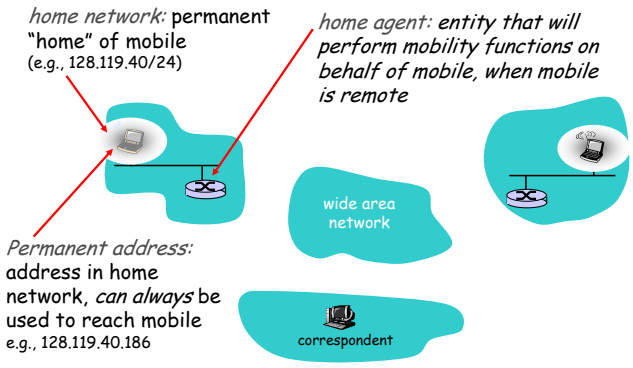
4.1 Introduction and Network Service Models
4.2 Routing Principles
4.3 Hierarchical Routing
4.4 The Internet (IP) Protocol
4.5 Routing in the Internet
4.6 What's Inside a Router?
4.7 IPv6
4.8 Multicast Routing
4.9 Mobility

What is mobility?

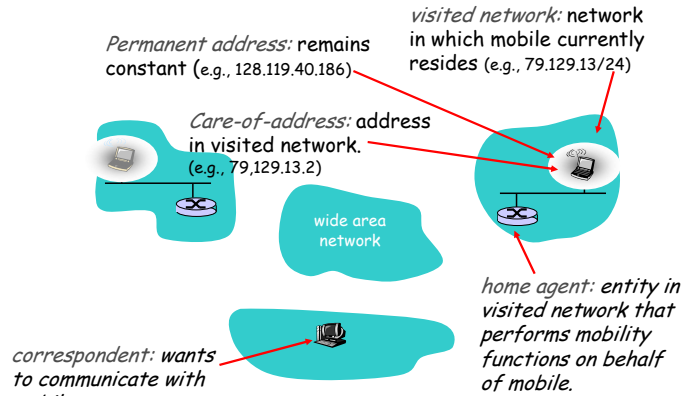
- spectrum of mobility, from the *network* perspective:



Mobility: Vocabulary



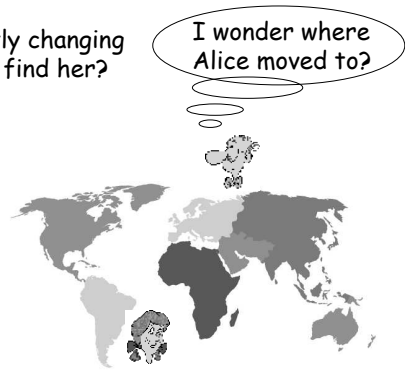
Mobility: more vocabulary



How do *you* contact a mobile friend:

Consider friend frequently changing addresses, how do you find her?

- search all phone books?
- call her parents?
- expect her to let you know where he/she is?



Mobility: approaches

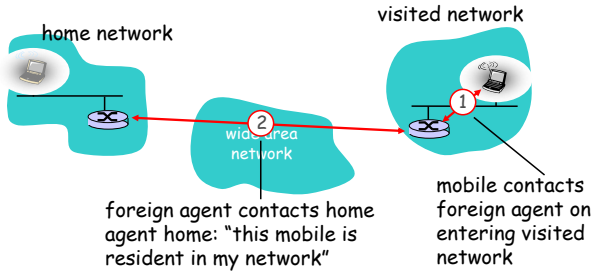
- *Let routing handle it:* routers advertise permanent address of mobile-nodes-in-residence via usual routing table exchange.
  - routing tables indicate where each mobile located
  - no changes to end-systems
- *Let end-systems handle it:*
  - *indirect routing:* communication from correspondent to mobile goes through home agent, then forwarded to remote
  - *direct routing:* correspondent gets foreign address of mobile, sends directly to mobile

Mobility: approaches

- *Let routing handle it:* routers advertise permanent address of mobile-nodes-in-residence via usual routing table exchange.
  - routing tables indicate where each mobile located
  - no changes to end-systems
- *let end-systems handle it:*
  - *indirect routing:* communication from correspondent to mobile goes through home agent, then forwarded to remote
  - *direct routing:* correspondent gets foreign address of mobile, sends directly to mobile

not scalable to millions of mobiles

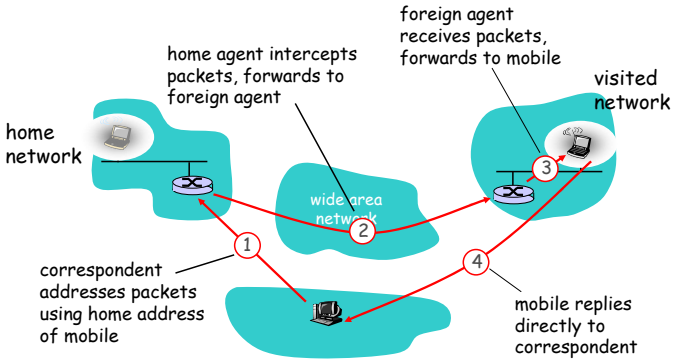
Mobility: registration



End result:

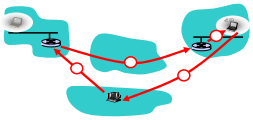
- Foreign agent knows about mobile
- Home agent knows location of mobile

Mobility via Indirect Routing

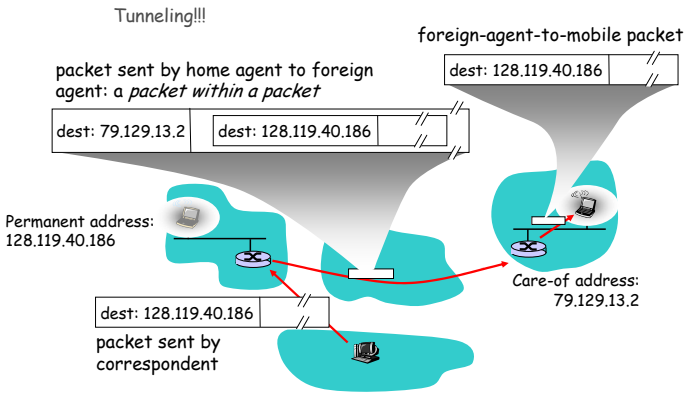


Indirect Routing: comments

- Mobile uses two addresses:
  - permanent address: used by correspondent (hence mobile location is *transparent* to correspondent)
  - care-of-address: used by home agent to forward datagrams to mobile
- foreign agent functions may be done by mobile itself
- triangle routing: correspondent-home-network-mobile
  - inefficient when correspondent, mobile are in same network



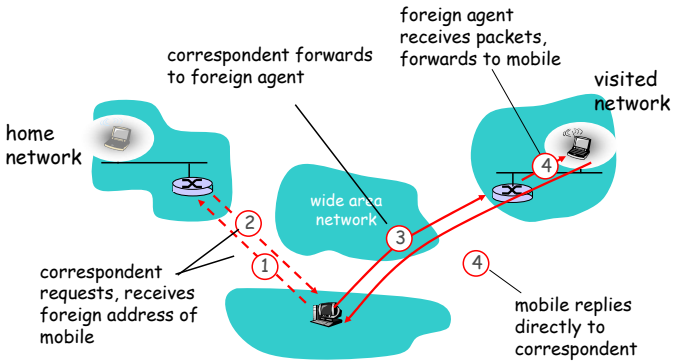
Forwarding datagrams to remote mobile



Indirect Routing: moving between networks

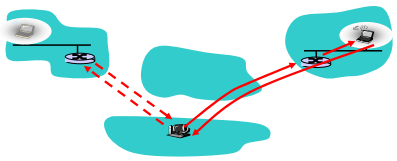
- suppose mobile user moves to another network
  - registers with new foreign agent
  - new foreign agent registers with home agent
  - home agent update care-of-address for mobile
  - packets continue to be forwarded to mobile (but with new care-of-address)
- Mobility, changing foreign networks transparent: *on going connections can be maintained!*

Mobility via Direct Routing



Mobility via Direct Routing: comments

- overcome triangle routing problem
- non-transparent to correspondent: correspondent must get care-of-address from home agent
  - What happens if mobile changes networks?

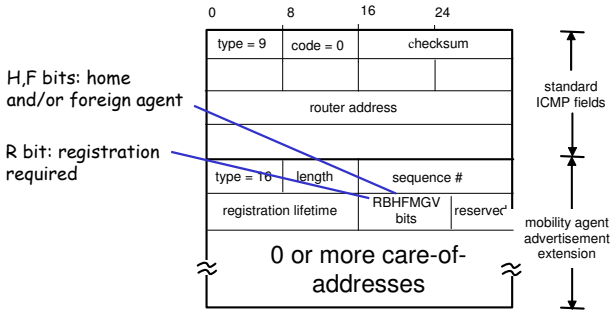


Mobile IP

- RFC 3220
- has many features we've seen:
  - home agents, foreign agents, foreign-agent registration, care-of-addresses, encapsulation (packet-within-a-packet)
- three components to standard:
  - agent discovery
  - registration with home agent
  - indirect routing of datagrams

Mobile IP: agent discovery

- agent advertisement: foreign/home agents advertise service by broadcasting ICMP messages (typefield = 9)



Mobile IP: registration example