Chapter 4
Network Layer

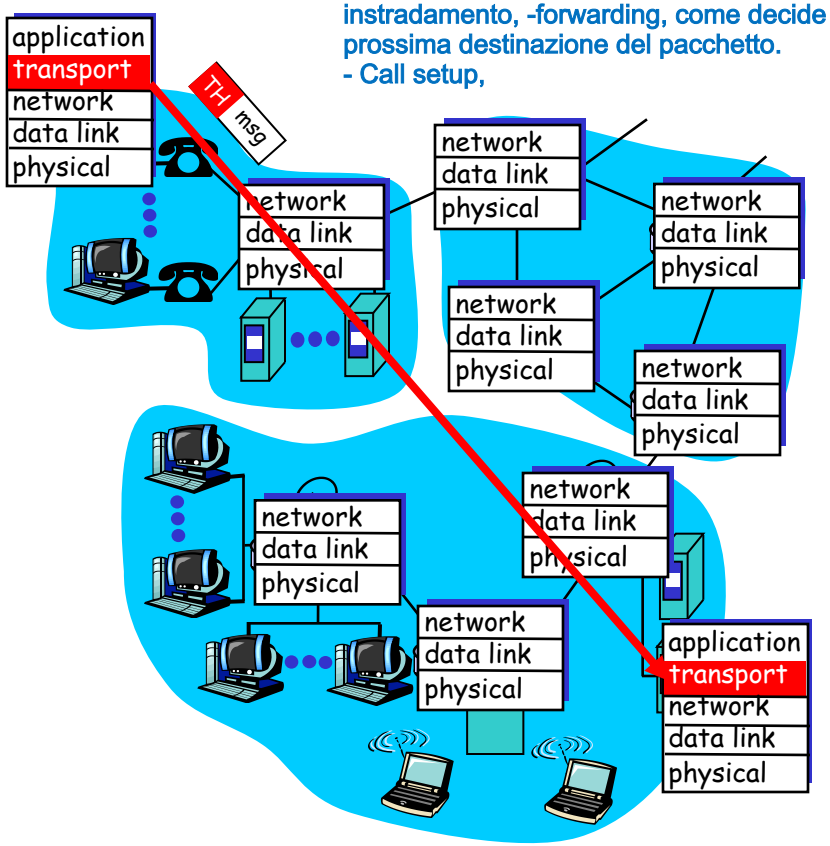
Network layer functions

al livello di trasporto si occupa di entità all'estremo della rete che si scambiano pacchetti
IL livello di rete è il primo livello che a differenza del livello, è in esecuzione su tutti gli host e su tutti i router. I compiti sono: individuare i percorsi, con algoritmi di instradamento, -forwarding, come decide la prossima destinazione del pacchetto.
- Call setup,

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

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 (VC networks):* Set-up routes state before sending packet



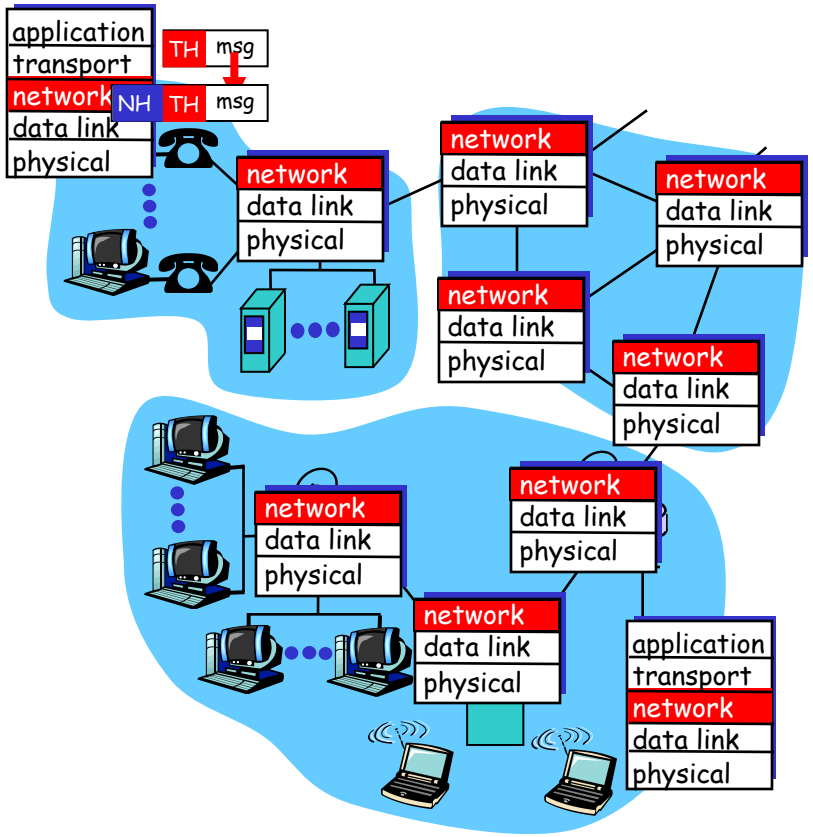
Problema: come i nodi cooperando tra loro riescono a far arrivare i pacchetti da nodi di partenza a nodi di arrivo. Costo minimo nei grafi.

Network layer functions

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

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 (VC networks):* Set-up routes state before sending packet

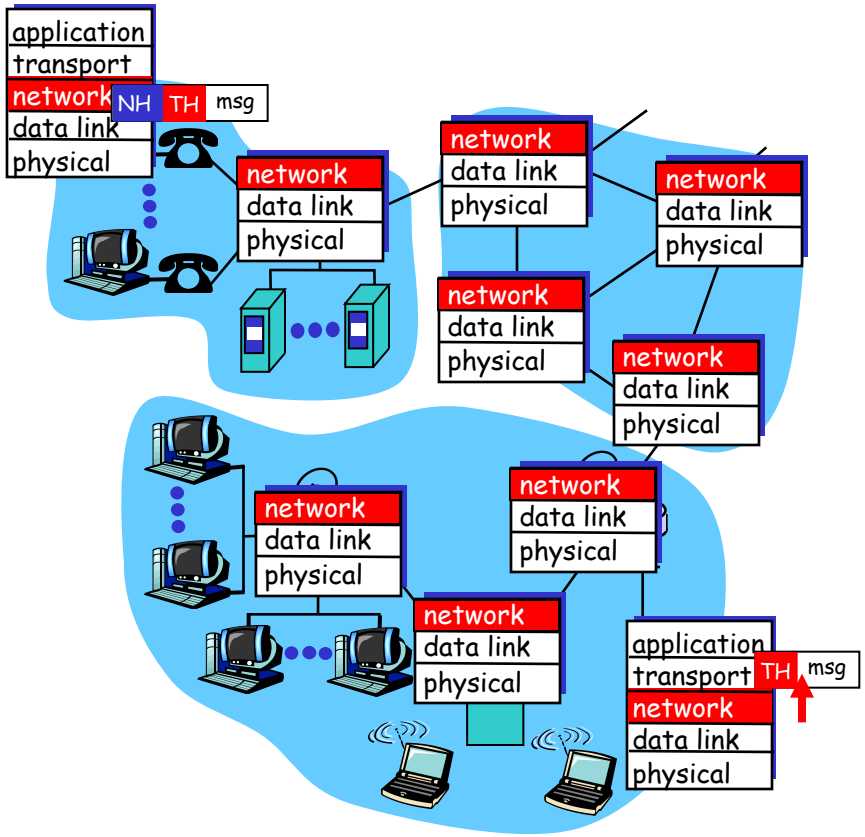


Network layer functions

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

functions:

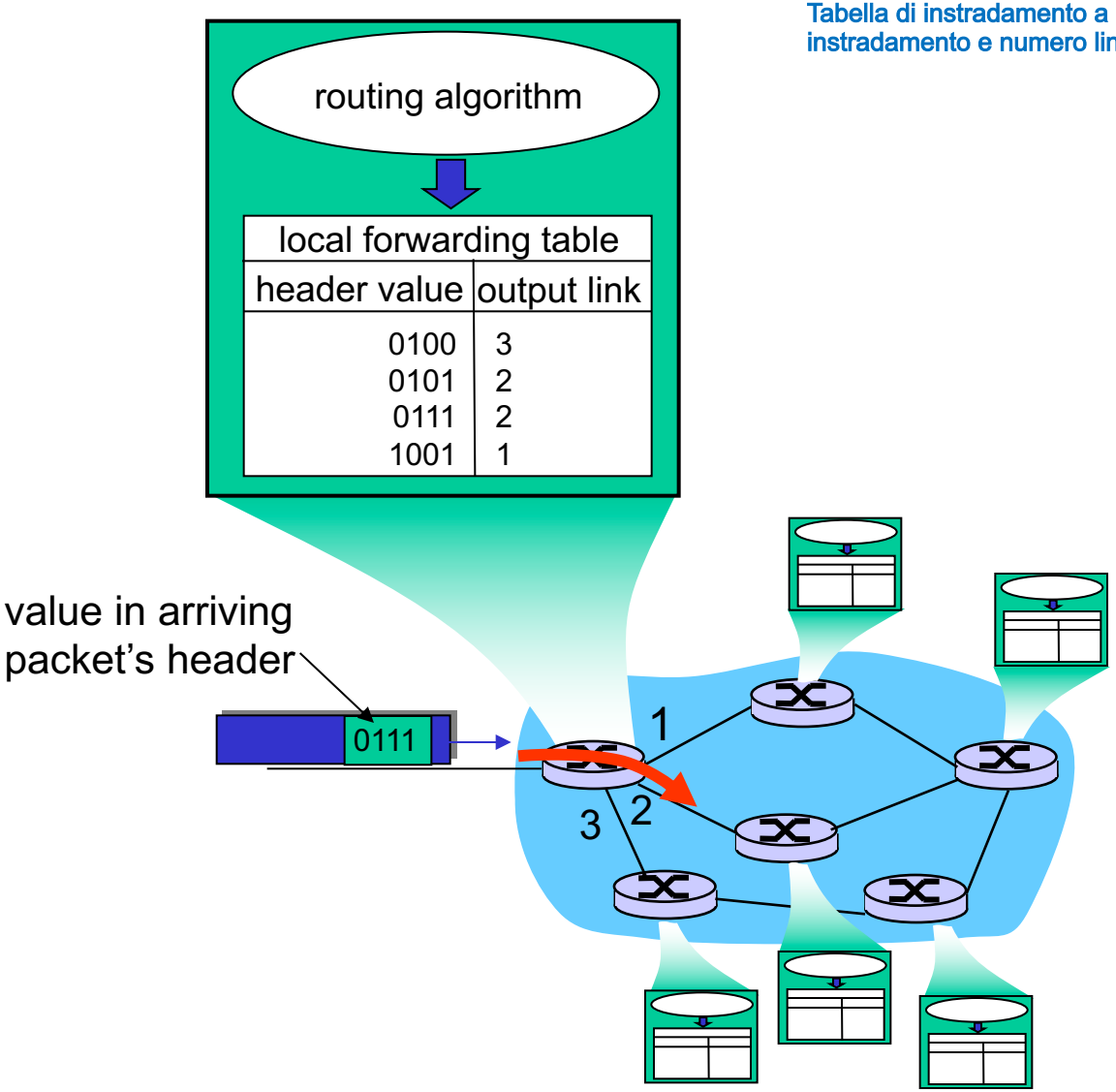
- *path determination:* route taken by packets from source to dest. *Routing algorithms*
- *forwarding:* move packets from router's input to appropriate router output
- *Call setup (VC networks):* Set-up routes state before sending packet



Interplay between routing and forwarding

net op

Tabella di instradamento a due colonne, valore di instradamento e numero link di uscita. Sono infor



Forwarding table

sono tabelle che si trovano nei router e vanno gestite pacchetto per pacchetto

4 billion possible entries

Ipv4

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

si lavora a gruppi di indirizzi

Nel dns devo organizzare uno spazio di nomi organizzato in modo gerarchico

Se si lavora in modo gerarchico la soluzione è più semplice da gestire

Network layer: data plane, control plane

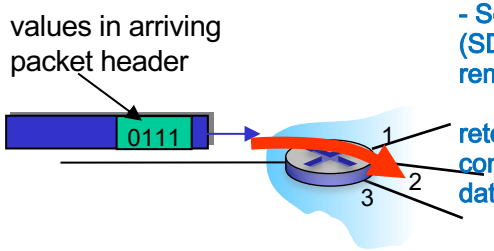
Data plane:

- *local*, per-router function
- determines how datagram arriving on router input port is forwarded to router output port

Data plane: gestione dati come mostrare i pacchetti

control plane: tutte le funzioni gestite dal protocollo per controllare le funzionalità di quel livello

Logia al livello di controllo, determina come i pacchetti percorrono la mia rete.



Ci sono due approcci principali:
- Tradizionale: implementato nei router
- Softwar define networking (SDN) -> implementato da remoto

rete a pacchetto senza connessione -> Ip. datagramma

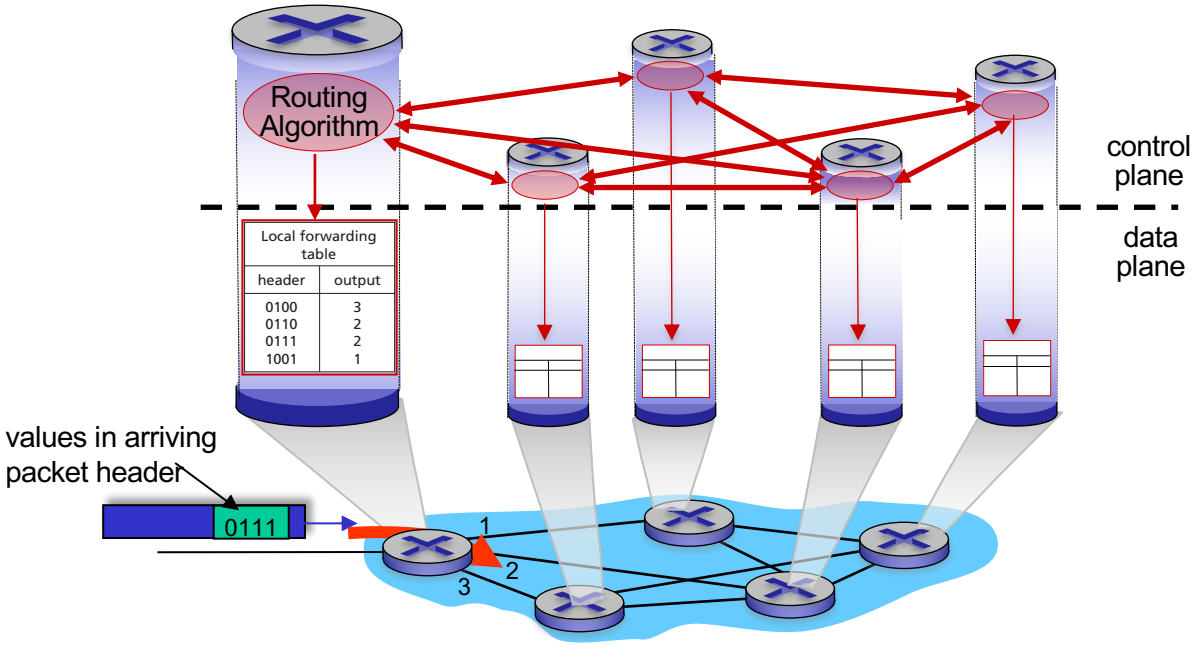
Control plane

- *network-wide* logic
- determines how datagram is routed among routers along end-end path from source host to destination host

- two control-plane approaches:
  - *traditional routing algorithms:* implemented in routers
  - *software-defined networking (SDN):* implemented in (remote) servers

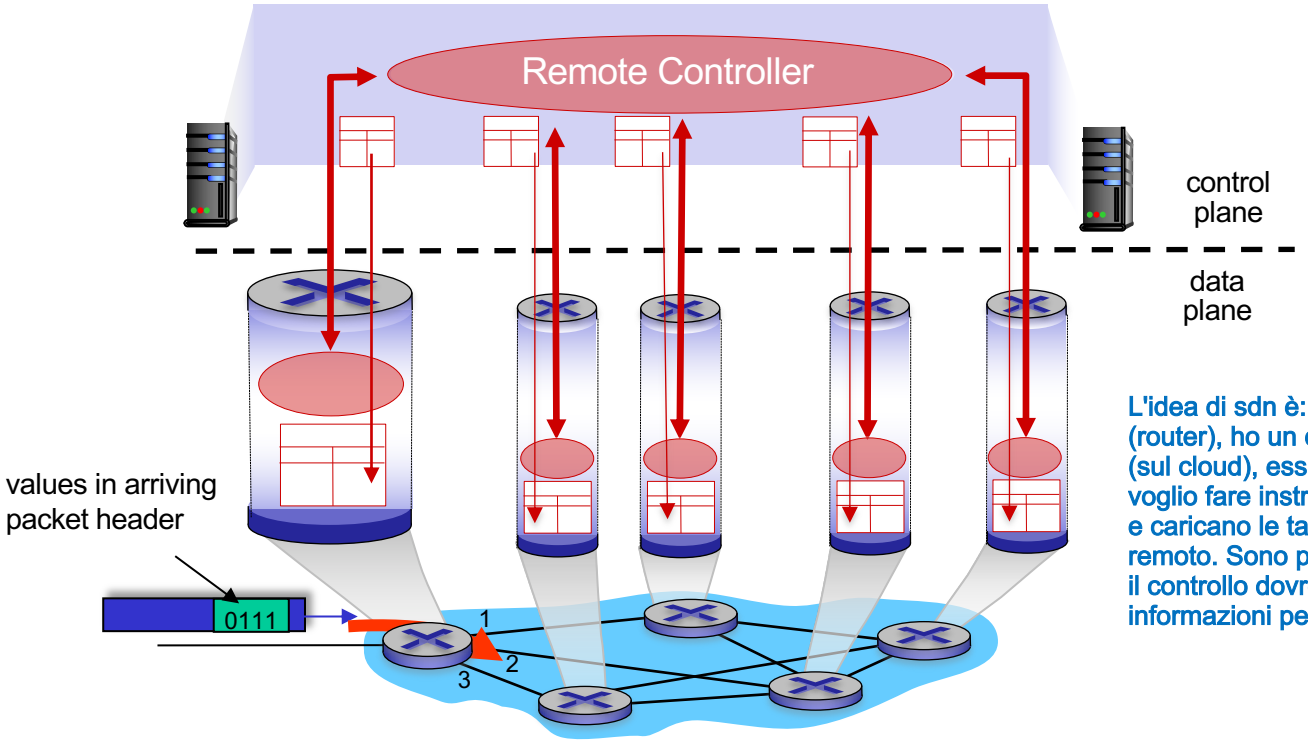
Per-router control plane

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



Software-Defined Networking (SDN) control plane

Remote controller computes, installs forwarding tables in routers



L'idea di sdn è: ho tanti apparati (router), ho un controllore remoto (sul cloud), essi determinano come voglio fare instradamento sulla rete e caricano le tabelle sul controllore remoto. Sono più complessi poiché il controllo dovrebbe avere tutte le informazioni per fare bene i calcoli.

Modello di servizio: ci possono essere diverse architetture di rete che possono chiamare diverse architetture di servizio.

datagramma -> senza connessione

Network service model

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

Example services for individual datagrams:

- guaranteed delivery
- guaranteed delivery with less than 40 msec delay

Example services for a flow of datagrams:

- in-order datagram delivery
- guaranteed minimum bandwidth to flow
- restrictions on changes in inter-packet spacing

Network-layer service model

Network Architecture	Service Model	Quality of Service (QoS) Guarantees ?			
		Bandwidth	Loss	Order	Timing
Internet	best effort	none	no	no	no

Internet "best effort" service model

No guarantees on:

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

Network-layer service model

Network Architecture	Service Model	Quality of Service (QoS) Guarantees ?			
		Bandwidth	Loss	Order	Timing
Internet	best effort	none	no	no	no
ATM	Constant Bit Rate	Constant rate	yes	yes	yes
ATM	Available Bit Rate	Guaranteed min	no	yes	no
Internet	Intserv Guaranteed (RFC 1633)	yes	yes	yes	yes
Internet	Diffserv (RFC 2475)	possible	possibly	possibly	no

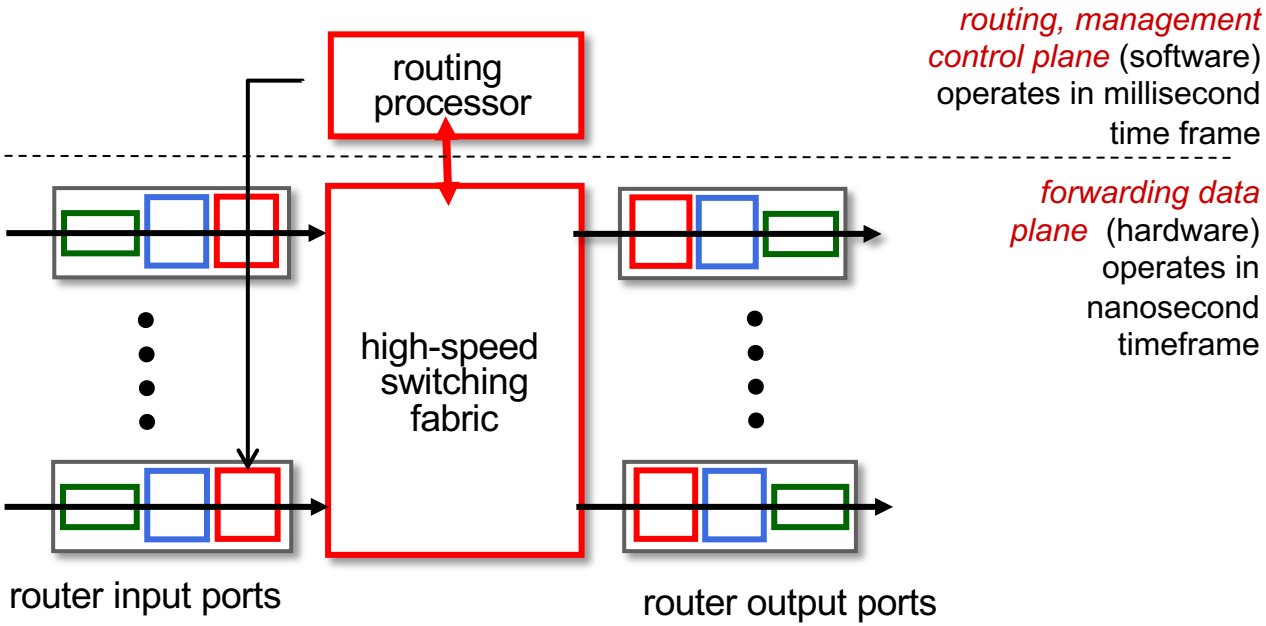
Reflections on best-effort service:

- simplicity of mechanism has allowed Internet to be widely deployed adopted
- sufficient provisioning of bandwidth allows performance of real-time applications (e.g., interactive voice, video) to be "good enough" for "most of the time"
- replicated, application-layer distributed services (datacenters, content distribution networks) connecting close to clients' networks, allow services to be provided from multiple locations
- congestion control of "elastic" services helps

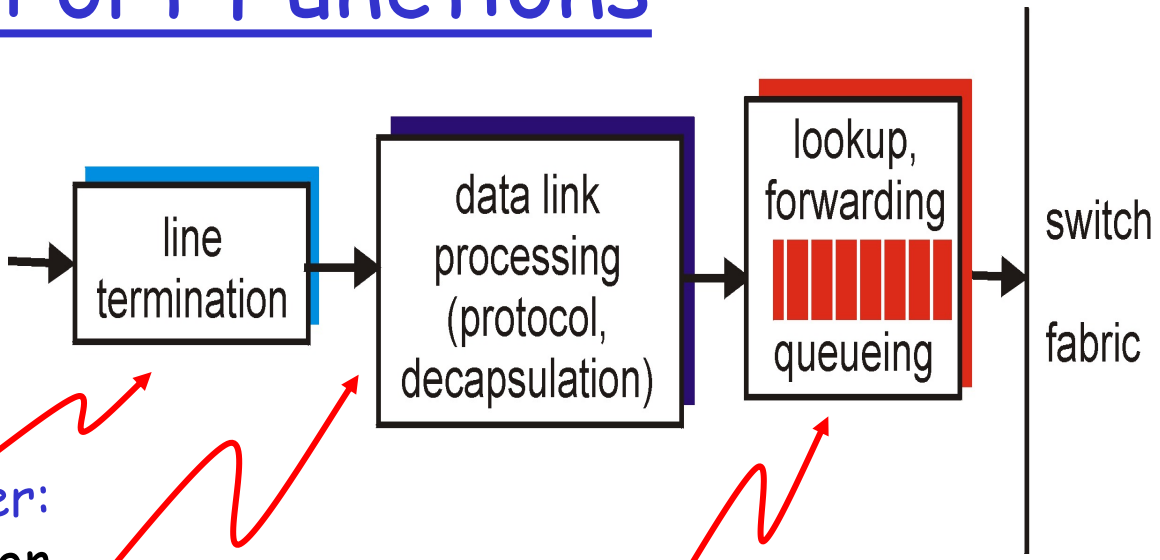
It's hard to argue with success of best-effort service model

Router architecture overview

high-level view of generic router architecture:



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 forwarding table in input port memory
- goal: complete input port processing at 'line speed'
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

3 livelli

il livello 2 individua il traffico inviato ad esso da 1, da qui escono i pacchetti ed essi vengono messi in una porta di ingresso di livello 3 e li vengono accodati perché bisogna capire cosa fare con quel pacchetto.
Qui lavorerà la tabella di instradamento per farli uscire.
Sono spesso operazioni realizzati in hardware poiché ci vuole molto meno tempo. Se i pacchetti non vengono smaltiti così come arrivano si accodano

Destination-based forwarding

forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010000 00000100	0
11001000 00010111 00010000 00000101 through 11001000 00010111 00010000 00000111	3
11001000 00010111 00011000 11111111	
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

Q: but what happens if ranges don't divide up so nicely?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011** ********	2
otherwise	3

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

uso quello dove corrispondono più bit

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011** ********	2
otherwise	3

match!

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011** ********	2
otherwise	3

match!

examples:

11001000 00010111 00010110 10100001	which interface?
11001000 00010111 00011000 10101010	which interface?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011** ********	2
otherwise *	3

match!

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

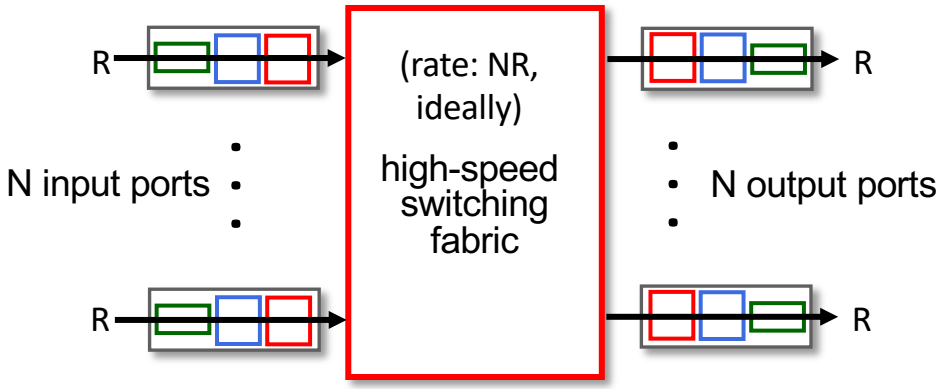
Longest prefix matching

- we'll see *why* longest prefix matching is used shortly, when we study addressing
- longest prefix matching: often performed using ternary content addressable memories (TCAMs)
  - *content addressable:* present address to TCAM: retrieve address in one clock cycle, regardless of table size
  - Cisco Catalyst: ~1M routing table entries in TCAM

Switching fabrics

- transfer packet from input link to appropriate output link

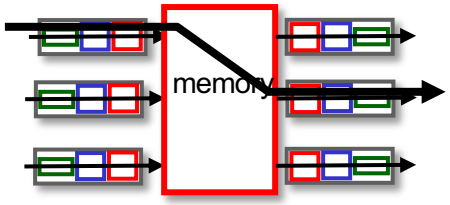
- switching rate: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable



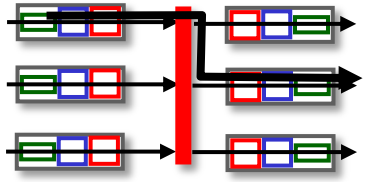
Switching fabrics

- transfer packet from input link to appropriate output link

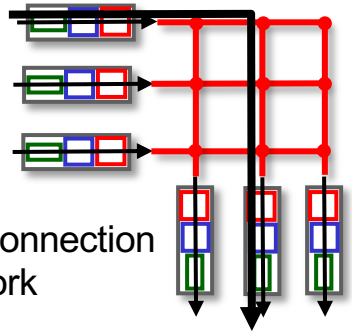
- switching rate: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable
- three major types of switching fabrics:



memory



bus

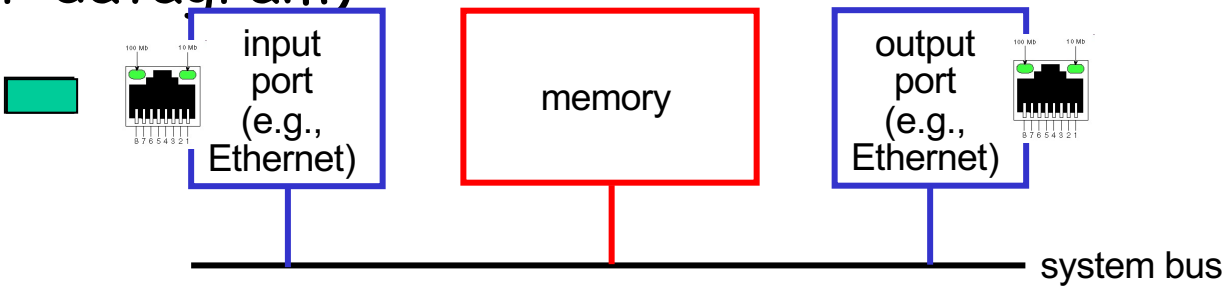


interconnection network

Switching via memory

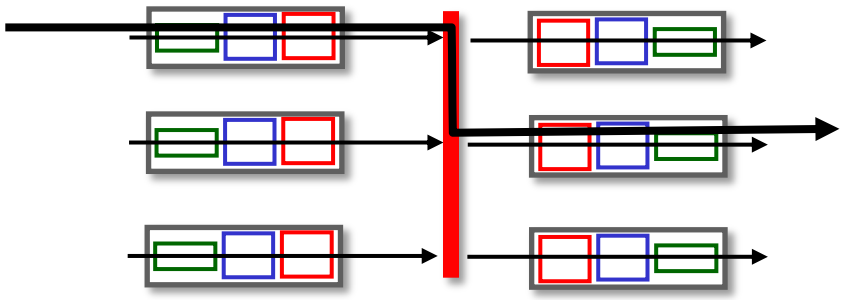
first generation routers:

- traditional computers with switching under direct control of CPU
- packet copied to system's memory
- speed limited by memory bandwidth (2 bus crossings per datagram)



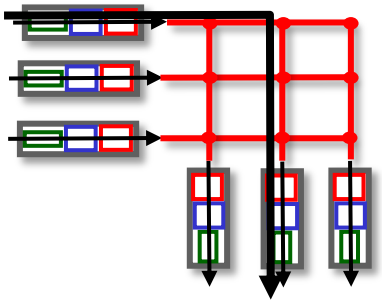
Switching via a bus

- datagram from input port memory to output port memory via a shared bus
- *bus contention:* switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access routers

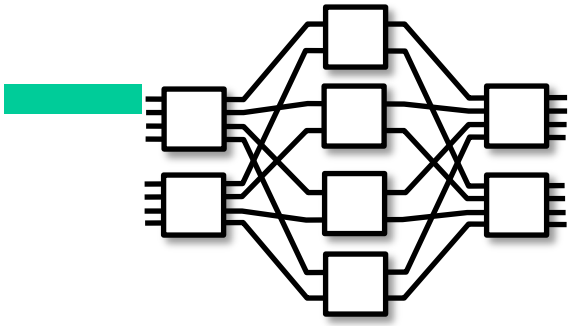


Switching via interconnection network

- Crossbar, Clos networks, other interconnection nets initially developed to connect processors in multiprocessor
- multistage switch: *nxn* switch from multiple stages of smaller switches
- exploiting parallelism:
  - fragment datagram into fixed length cells on entry
  - switch cells through the fabric, reassemble datagram at exit



3x3 crossbar

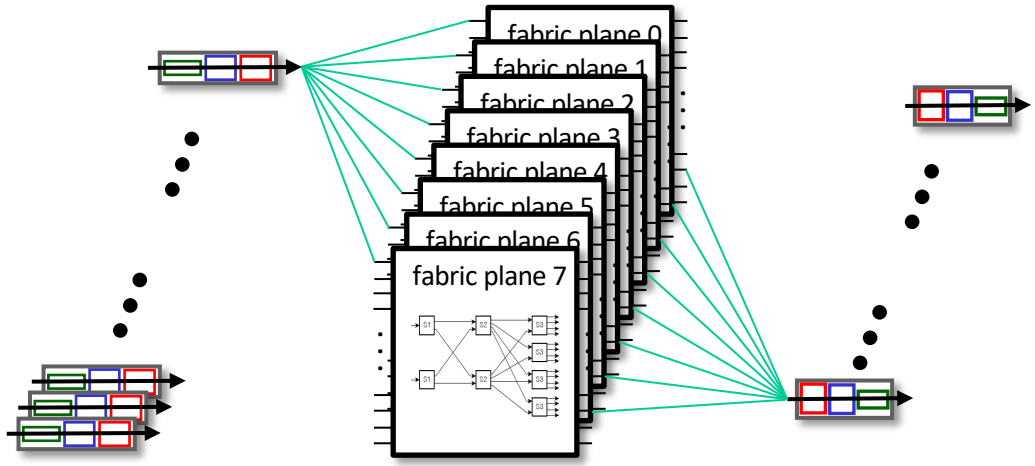


8x8 multistage switch
built from smaller-sized switches

Switching via interconnection network

- scaling, using multiple switching "planes" in parallel:
  - speedup, scaleup via parallelism

- Cisco CRS router:
  - basic unit: 8 switching planes
  - each plane: 3-stage interconnection network
  - up to 100's Tbps switching capacity



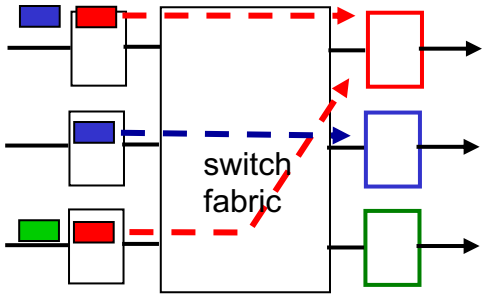
Input port queuing

il pkt viene bloccato per colpa della coda, anche se effettivamente la sua porta è libera (verde)

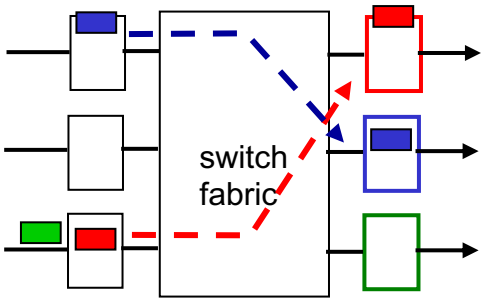
con architetture più complesse, si scandisce la coda affinché si trova un pkt che abbia la porta libera

- If switch fabric slower than input ports combined -> queueing may occur at input queues
  - queueing delay and loss due to input buffer overflow!

- Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward



output port contention: only one red datagram can be transferred. lower red packet is *blocked*

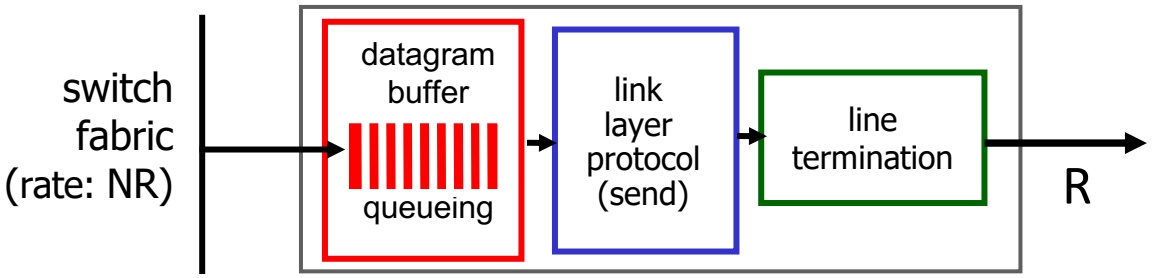


one packet time later: green packet experiences HOL blocking

drop policy, come scarto il pacchetto

Scheduling, ogni volta che devo tx un pkt, quale trasmetto? FIFO

Output port queuing



This is a really important slide

- *Buffering* required when datagrams arrive from fabric faster than link transmission rate. *Drop policy:* which datagrams to drop if no free buffers?

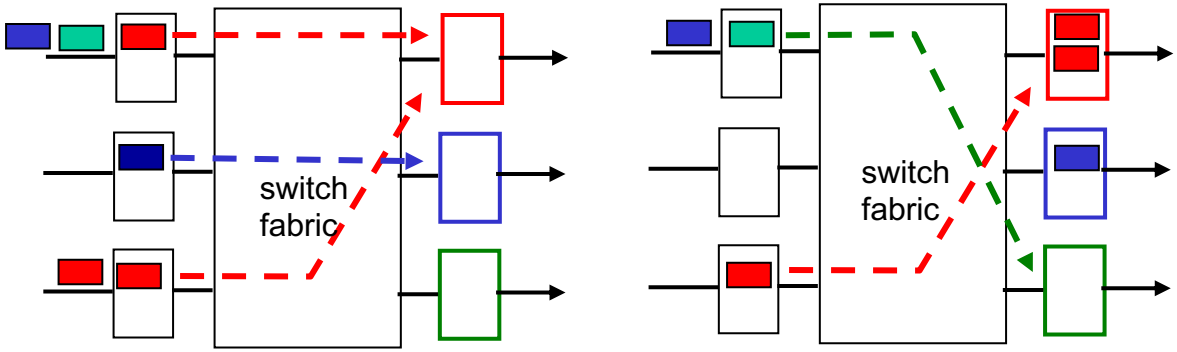
→ Datagrams can be lost due to congestion, lack of buffers

- *Scheduling discipline* chooses among queued datagrams for transmission

Si scarta dalla coda (nulla di hardware), o secondo un principio di priorità (sforzo hardware)

→ Priority scheduling – who gets best performance, network neutrality

Output port queuing



at *t*, packets more from input to output

one packet time later

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

How much buffering?

suggerimenti

- RFC 3439 rule of thumb: average buffering equal to "typical" RTT (say 250 msec) times link capacity C
  - e.g., C = 10 Gbps link: 2.5 Gbit buffer

- more recent recommendation: with N flows, buffering equal to

$$\frac{RTT \cdot C}{\sqrt{N}}$$

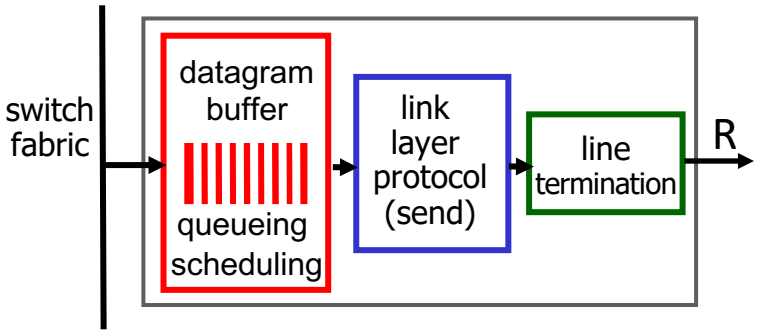
buffer maggiori implicano che potrebbero aumentare i ritardi di coda.

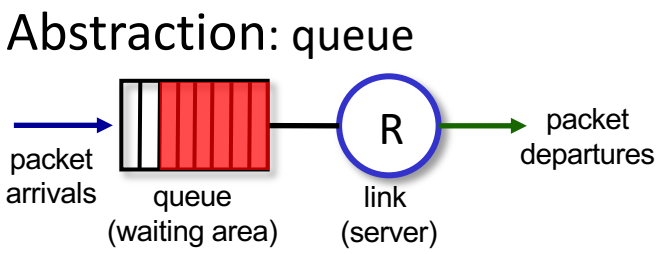
- but *too* much buffering can increase delays (particularly in home routers)
  - long RTTs: poor performance for realtime apps, sluggish TCP response
  - recall delay-based congestion control: "keep bottleneck link just full enough (busy) but no fuller"

tcp si comporta meglio per buffer minori, tcp non vuole ritardi/perdite.

flusso in ipv6 -> Più connessioni ho più il buffer è piccolo

Buffer Management





buffer management:

- drop: which packet to add, drop when buffers are full
  - tail drop: drop arriving packet
  - priority: drop/remove on priority basis
- marking: which packets to mark to signal congestion (ECN, RED)

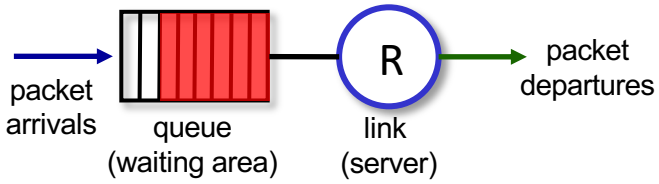
marking: meccanismo diverso per ulteriori versioni di tcp.
Ecn una volta che il buffer è riempito entro un certo livello ogni pacchetto che mi arriva lo marco (Avviso il mittente che c'è congestione).
Red non marca i pacchetti quando il buffer è congestionato, ma quando arrivano i pkt alla mia coda marca i pacchetti a caso indipendentemente da quanto sia pieno il pacchetto.
Red funziona meglio, facile implementazione.

Packet Scheduling: FCFS

packet scheduling:
deciding which packet to send next on link

- first come, first served
- priority
- round robin
- weighted fair queueing

Abstraction: queue



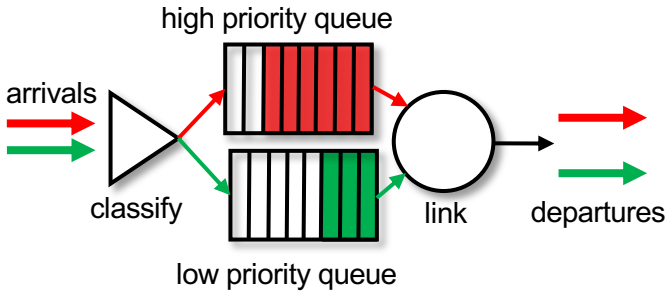
FCFS: packets transmitted in order of arrival to output port

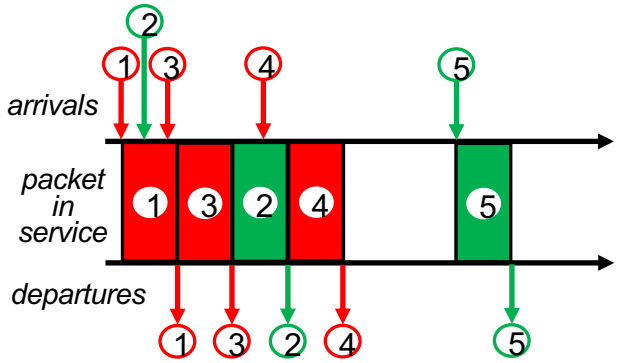
- also known as: First-in-first-out (FIFO)
- real world examples?

Scheduling policies: priority

Priority scheduling:

- arriving traffic classified, queued by class
  - any header fields can be used for classification
- send packet from highest priority queue that has buffered packets
  - FCFS within priority class



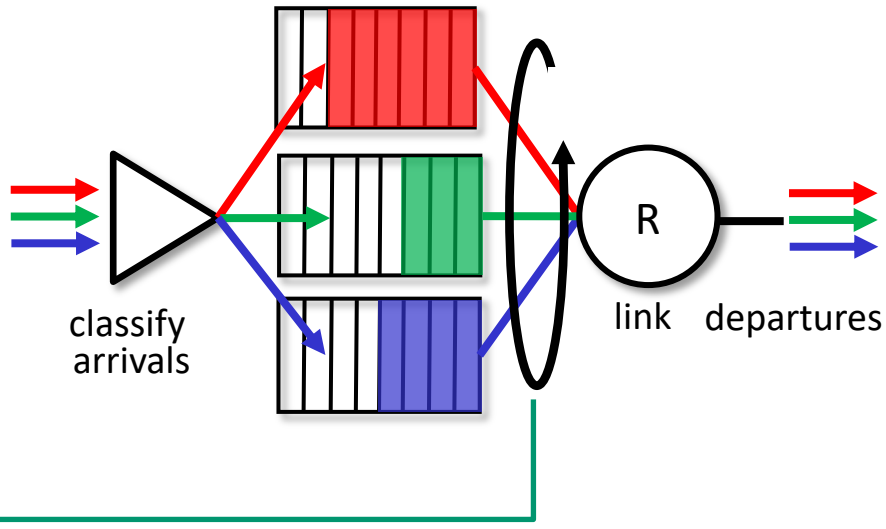


Scheduling policies: round robin

Round Robin (RR) scheduling:

- arriving traffic classified, queued by class
  - any header fields can be used for classification

- server cyclically, repeatedly scans class queues, sending one complete packet from each class (if available) in turn



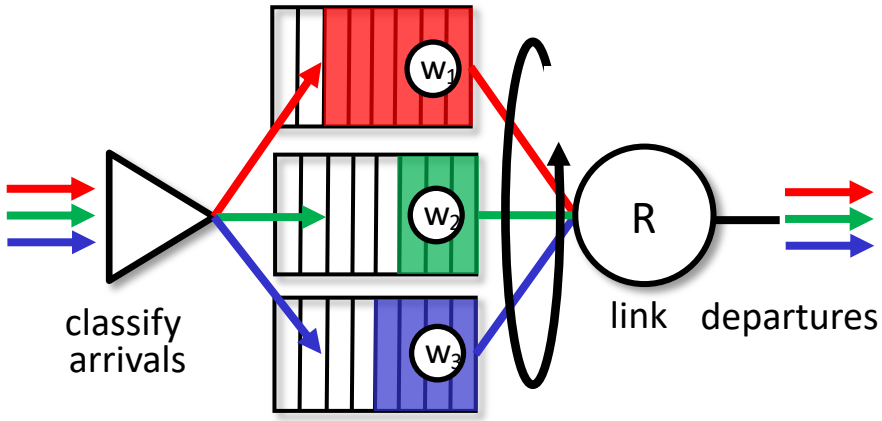
Scheduling policies: weighted fair queueing

Weighted Fair Queuing (WFQ):

- generalized Round Robin
  - each class, *i*, has weight, w_i, and gets weighted amount of service in each cycle:

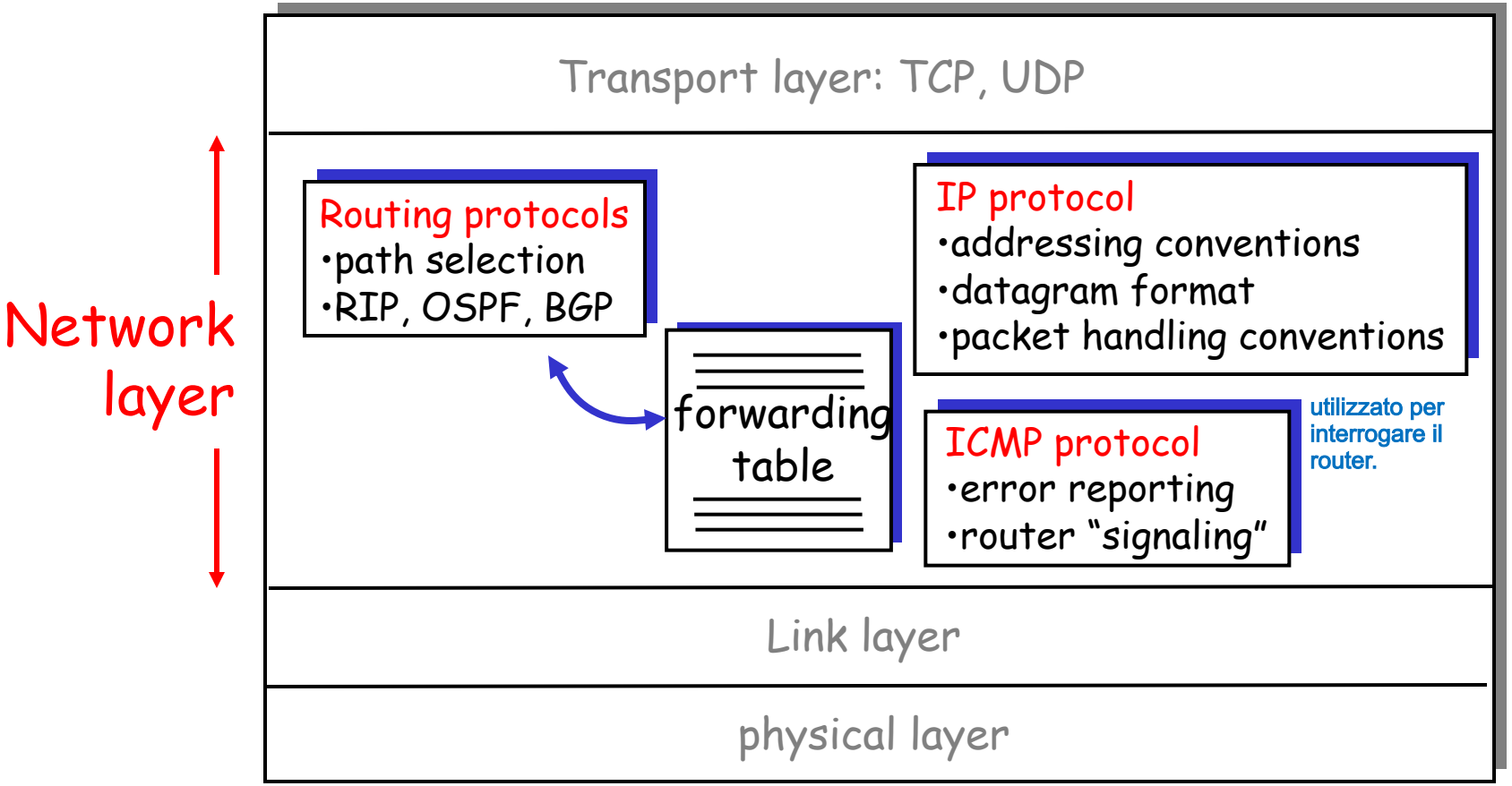
$$\frac{w_i}{\sum_j w_j}$$

  - minimum bandwidth guarantee (per-traffic-class)



The Internet Network layer

Host, router network layer functions:



IP datagram format

I router: 1. Controllano checksum, se il pck passa l'esame viene modificato time to lieve e poi deve essere ricaricato il checksum (diminuito di uno). i router veloci ultimamente non lo fanno.

IP protocol version number

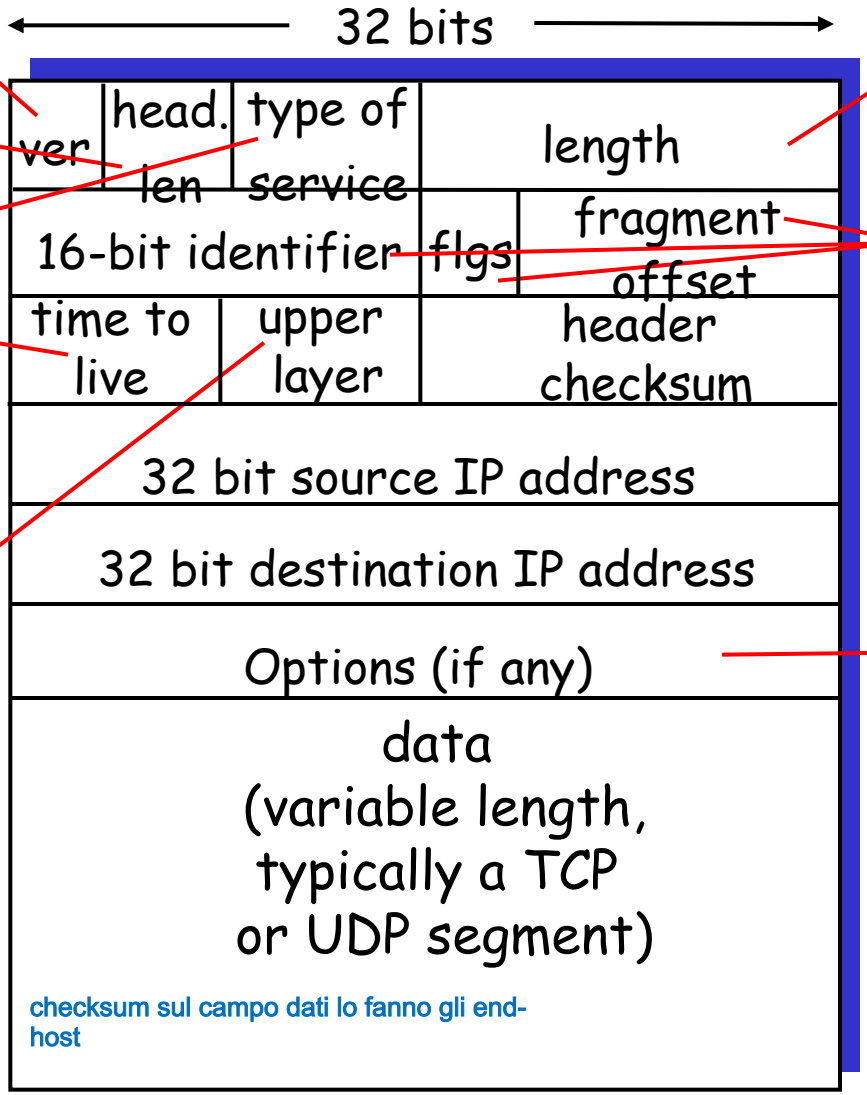
header length (bytes)

"type" of data

per evitare cicli, routing loop

max number remaining hops (decremented at each router)

upper layer protocol to deliver payload to



total datagram length (bytes)

for fragmentation/ reassembly

calcolato solo sui 20byte dell'intestazione e non tutto il pck. questa operazione di checksum viene effettuata da ogni router, se non supera l'esame il router pensa che l'errore potrebbe essere nel campo destinazione e quindi viene buttato. Operazione pesante per i router.

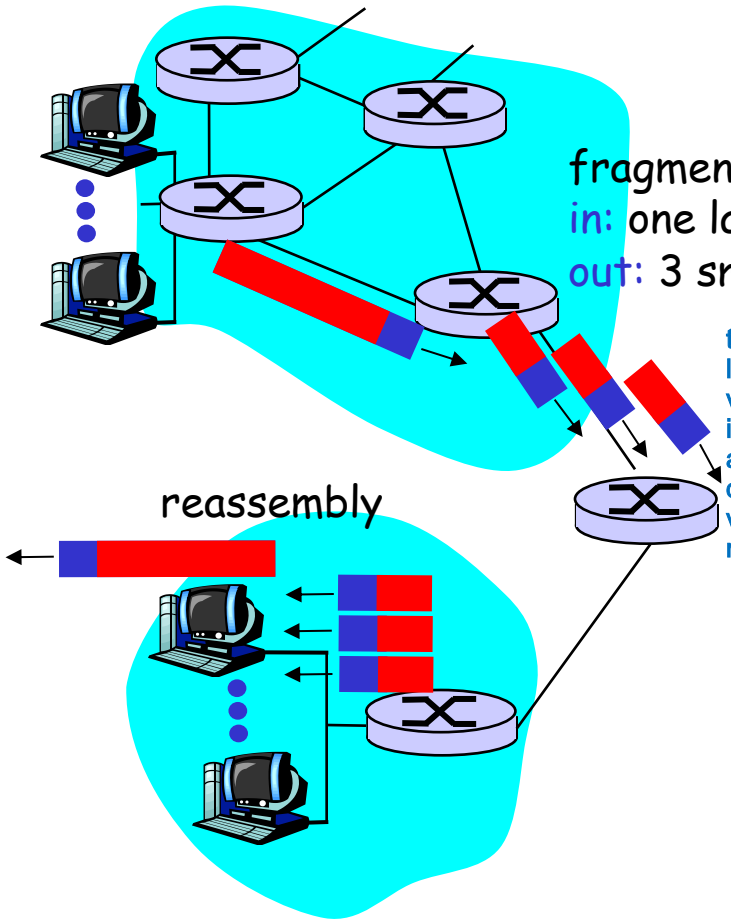
E.g. timestamp, record route taken, specify list of routers to visit.

how much overhead with TCP?

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

IP Fragmentation & Reassembly

- network links have MTU (max.transfer size) - largest possible link-level frame.
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify, order related fragments



tutti la stessa intestazione, la parte dati ognuno la sua. viaggiano come pacchetti indipendenti. Solo dopo aver raggiunto la destinazione finale verranno ricomposti (il ricevente lo assembla)

flag per capire se è un frammento

fragment offset, quale frammento sono

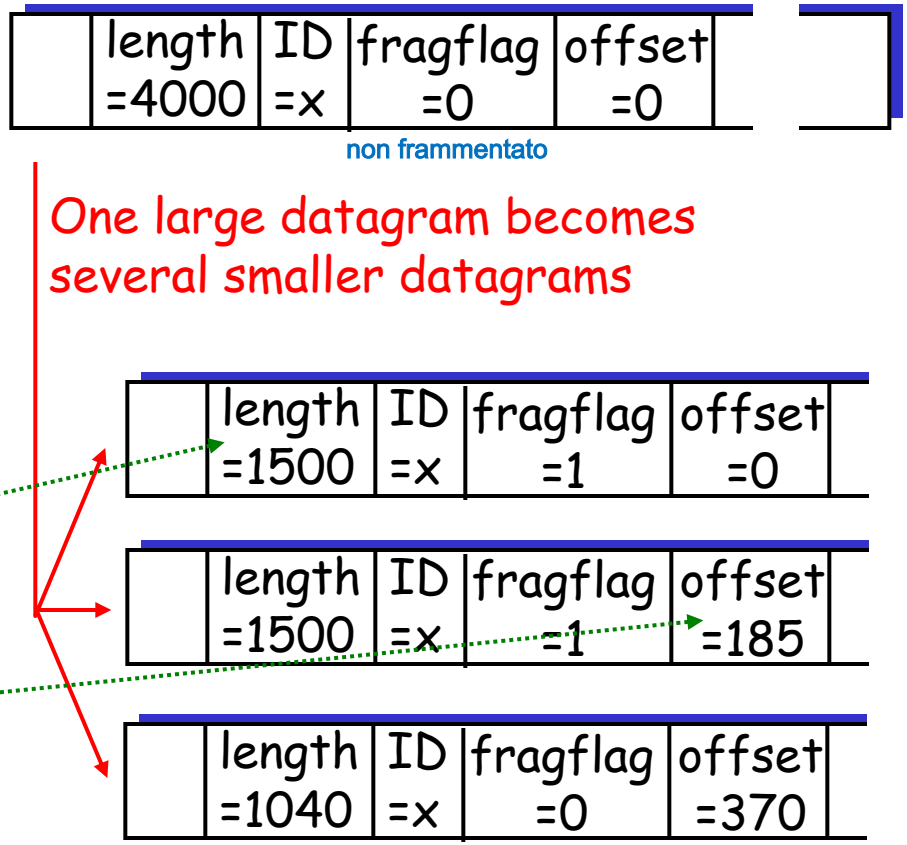
Se si perde un solo frammento si considera tutto il pacchetto come perso

in ipv6 la frammentazione è stata tolta, prima si invia un pacchetto per chiedere la dim max supportata e gestisco i miei pacchetti di conseguenza

IP Fragmentation and Reassembly

Example

- 4000 byte datagram
- MTU = 1500 bytes
  pacchetti non più grandi di 1500 bytes



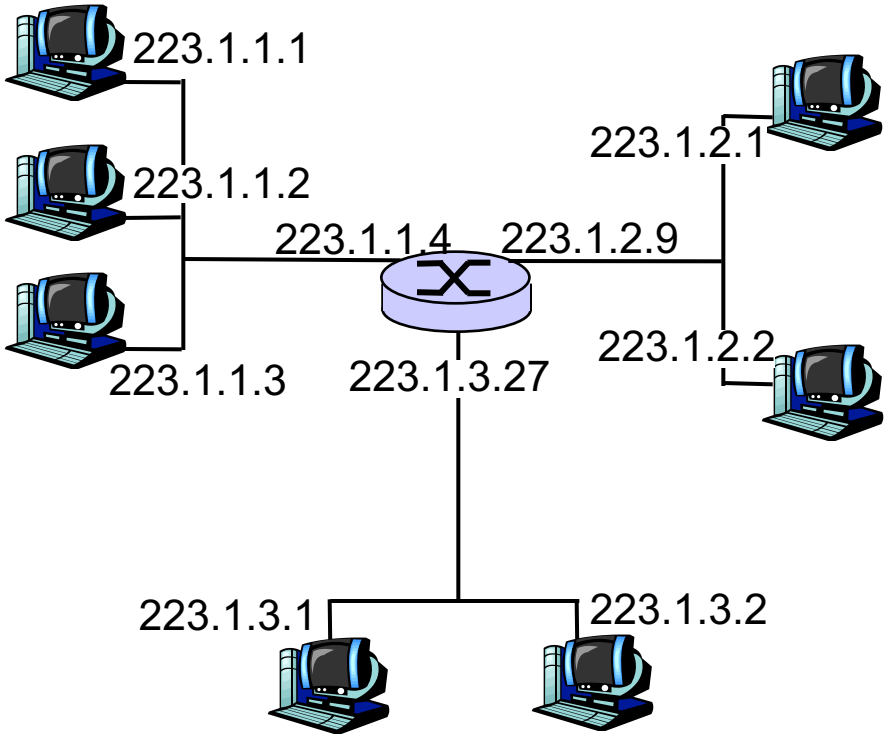
la somma non è 4000, 3 intestazioni, 2 in più, 20 bytes ogniuna

Capiamo che sono tutti di un pacchetto perché l'id è sempre lo stesso, flag = 1 per tutti tranne per l'ultimo, l'ultimo ha flag 0 e si distingue da un pacchetto non frammentato grazie all'offset, che non è zero perché inficia da quale byte io parto.

per questioni di efficienza si scrive l'offset/8
shift right 3

IP Addressing: introduction

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



223.1.1.1 = 11011111 00000001 00000001 00000001

223 . 1 . 1 . 1

sottorete | host

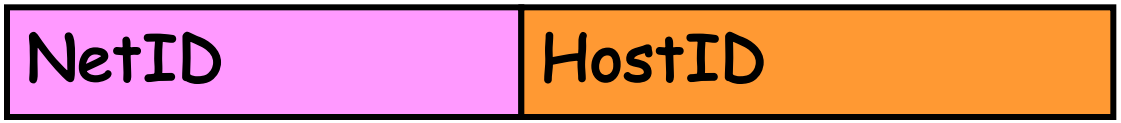
gli indirizzi sono associati alle interfaccie di rete, ogni nodo ha un suo indirizzo ip. I 32bit vengono descritti utilizzando dot decimal, byte per byte si da l'intero che corrisponde a quei 10 bit

IP Addressing

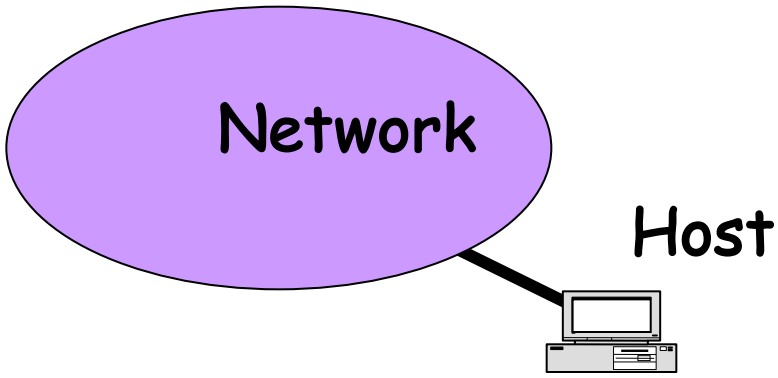
Ogni indirizzo ip è suddiviso in due parti (dim mobile) i bit più significativi identificano la rete all'interno del quale è presente il nodo. e gli altri identificano l'host.

Rete -> la sottorete ip, un sottoinsieme dei nodi il cui indirizzo ip è lo stesso.

- Address can be divided in two parts



- NetID identifies the network
- HostID identifies the host within the network



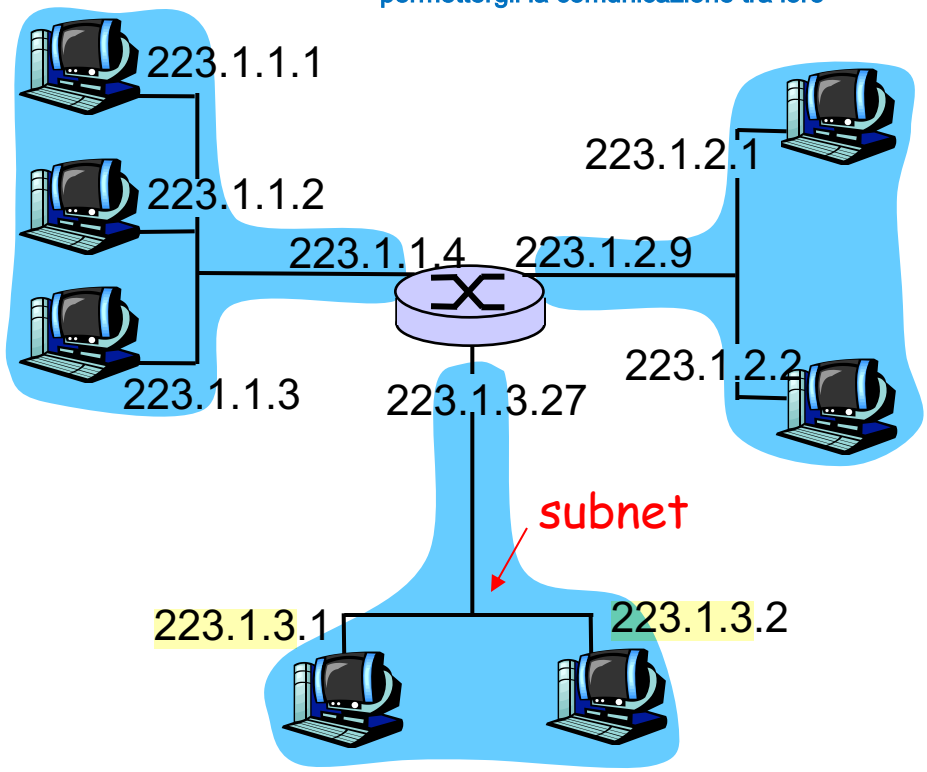
Hosts within the same network have the same NetId

La sottorete si identifica poiché i nodi possono comunicare inviandosi pacchetti che non devono passare nel router (tramite il livello due ad esempio). I router sono quindi apparati che permettono di comunicare tra sottoreti diverse.

ip nasce per connettere reti già esistenti e per permettergli la comunicazione tra loro

Subnets

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



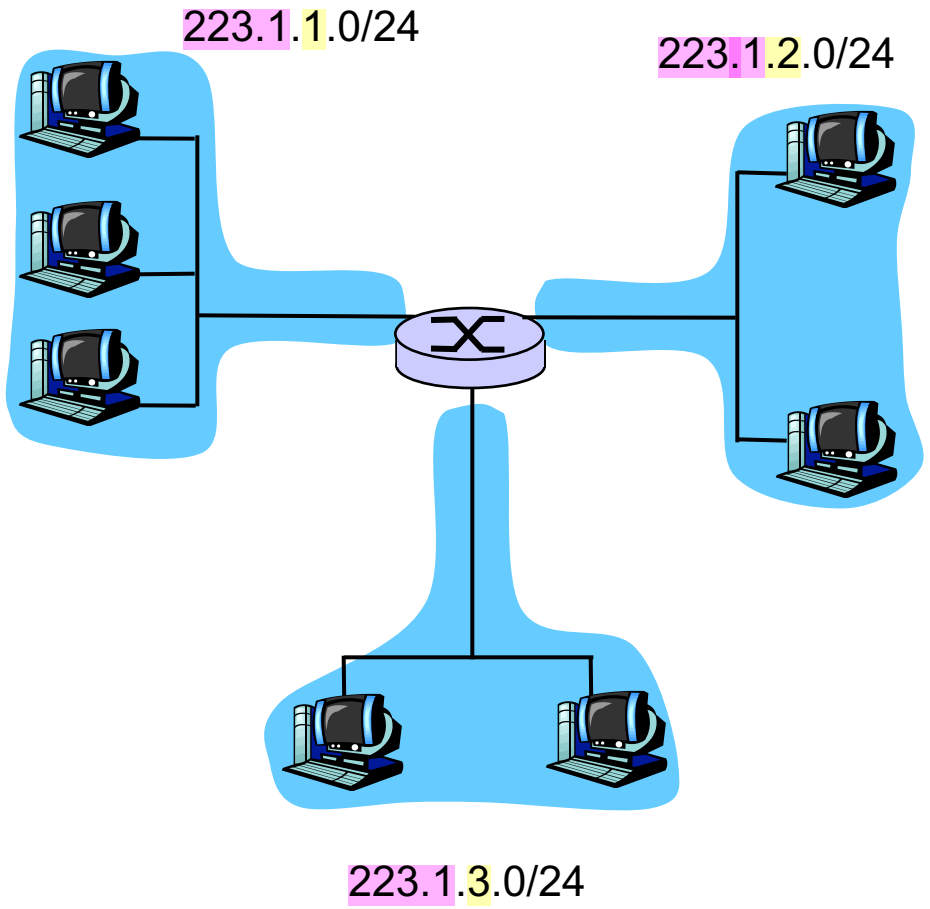
network consisting of 3 subnets

Subnets

Recipe

- To determine the subnets, detach each interface from its host or router, creating islands of isolated networks. Each isolated network is called a subnet.

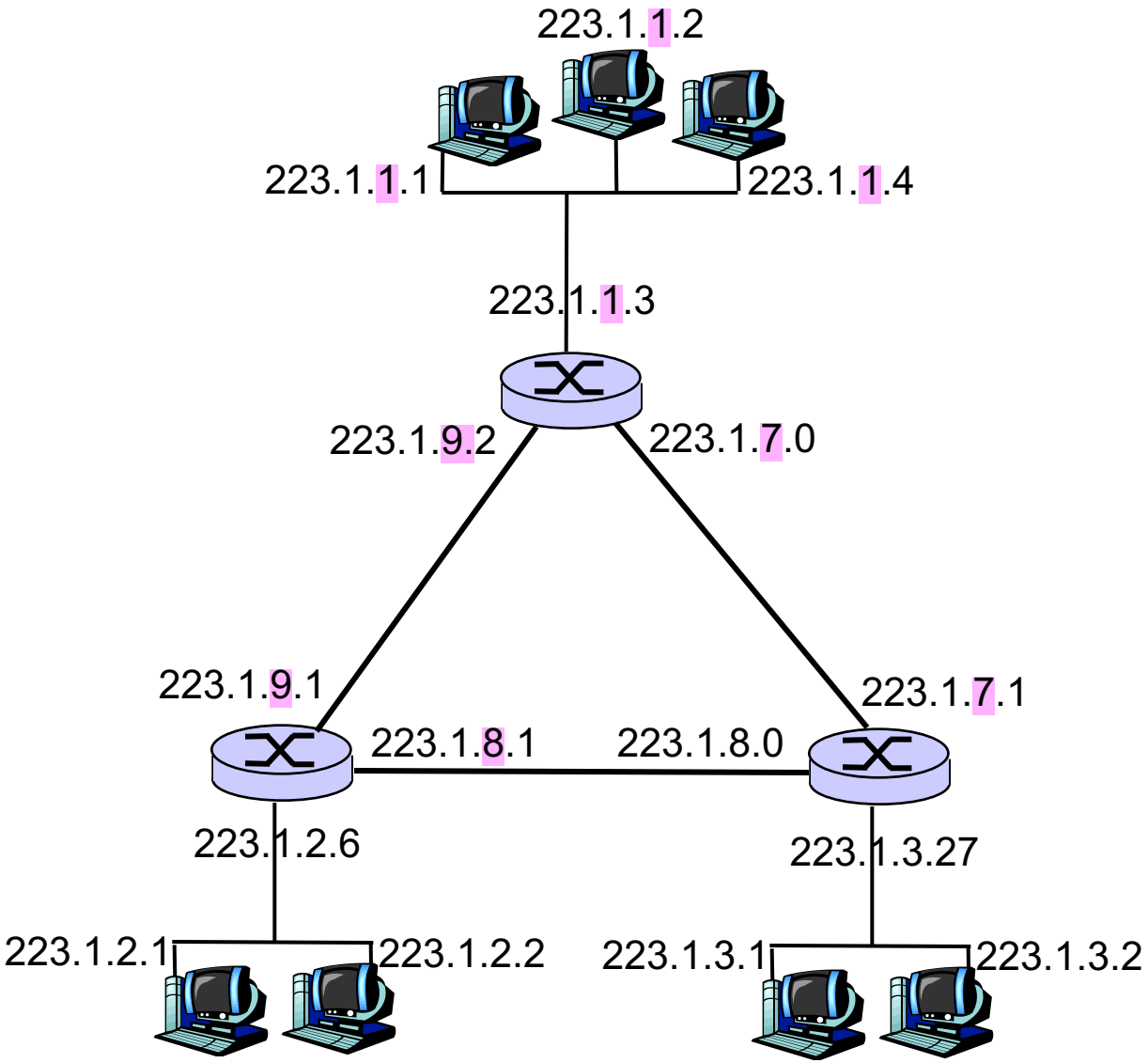
Identificare il numero di sottoreti in una rete



Subnet mask: /24

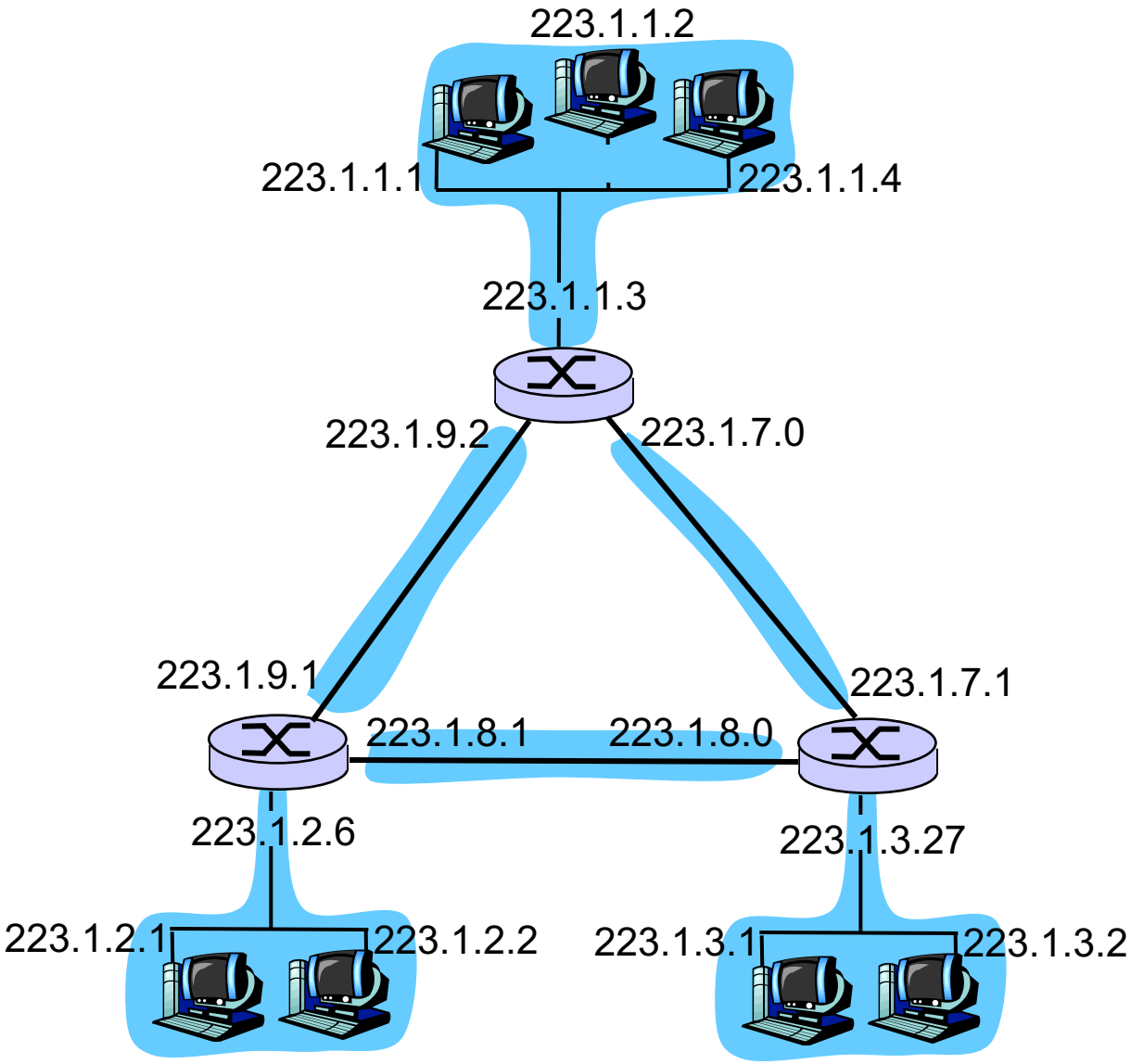
Subnets

How many?



Subnets

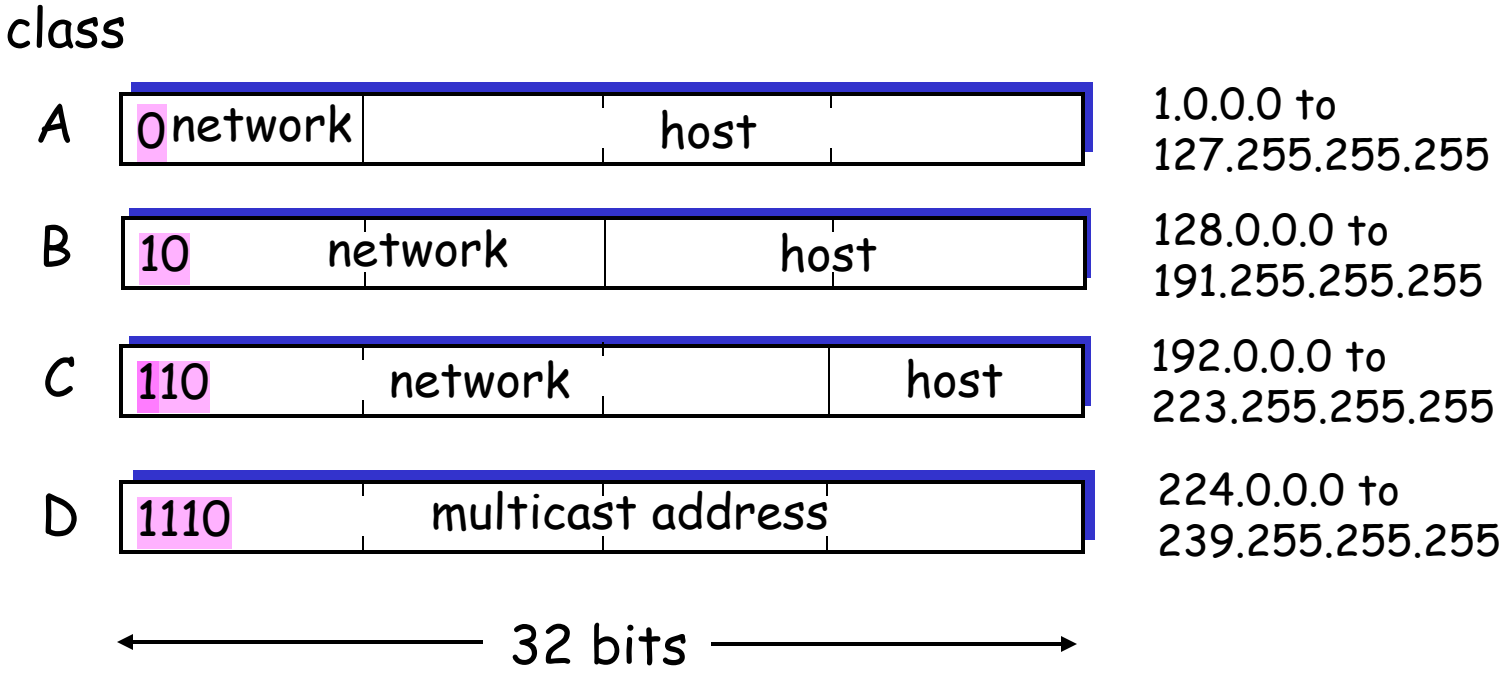
How many?



IP Addresses

given notion of "network", let's re-examine IP addresses:

"class-full" addressing:

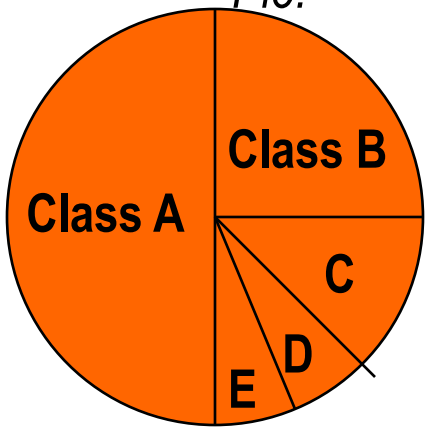


L'instradamento non va fatto tra host ma tra reti, le tabelle di routing essendo stato scelto come parte di rete e parte di host, devono mantenere solamente informazioni di rete e non quelle di host.

Counting up

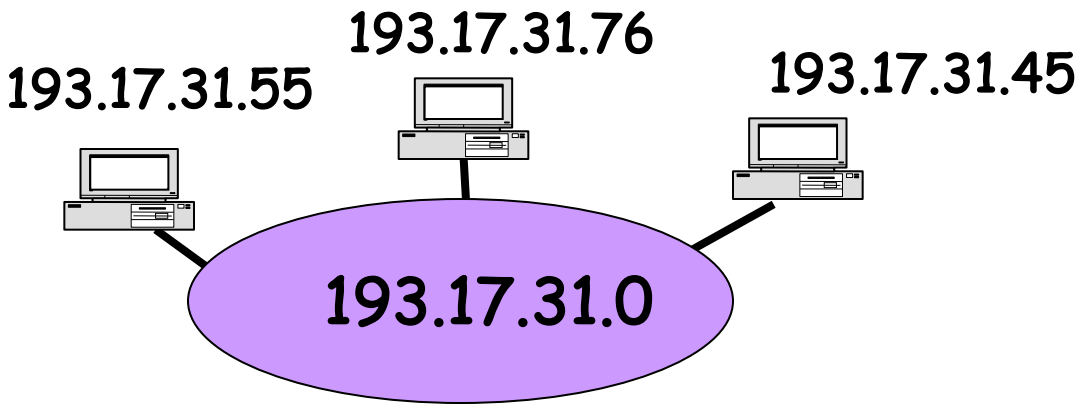
- 32 bit IP address:
  - 2^{32} = 4.294.967.296 theoretical IP addresses
- class A:
  - 2^{7}-2 =126 networks [0.0.0.0 and 127.0.0.0 reserved]
  - 2^{24}-2 = 16.777.214 maximum hosts
    - **2.113.928.964** addressable hosts (49,22% of max)
- class B
  - 2^{14}=16.384 networks
  - 2^{16}-2 = 65.534 maximum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



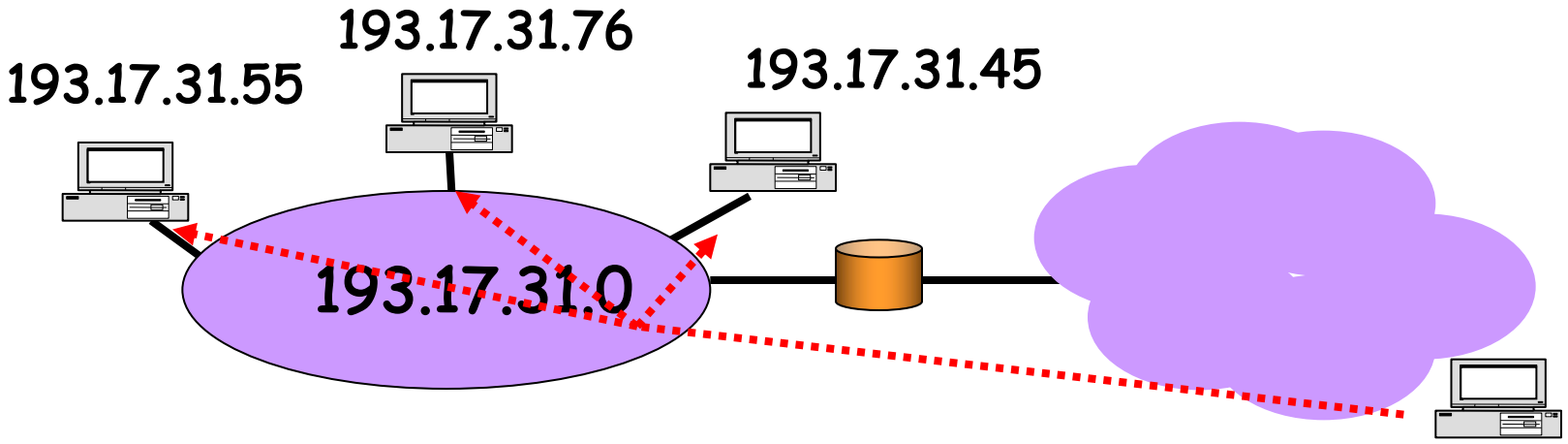
Special Addresses

- Network Address:
  - An address with the HostID bits set to 0 identifies the network with the given NetID (used in routing tables)
  - examples:
    - class B network: 131.175.0.0
    - class C network: 193.17.31.0



Special Addresses

- Direct Broadcast Address:
  - Address with HostID bit set to 1 is the broadcast address of the network identified by NetID.
  - example: 193.17.31.255



IP addressing: CIDR

Ogni volta che definiamo una rete definiamo la lunghezza di bit arbitraria

CIDR: Classless InterDomain Routing

- subnet portion of address of arbitrary length
- address format: a.b.c.d/x, where x is # bits in subnet portion of address

200.23.16.0/23

IP addresses: how to get one?

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

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

L'indirizzp ip del nodo dipende quindi da dove si trova, nel livello 3 il lavoro svolto è che tramite l'indirizzo vengono estrapolate le informazioni per l'instradamento (raggiungere quel nodo).

DHCP: Dynamic Host Configuration Protocol

Goal: allow host to *dynamically* obtain its IP address from network server when it joins network

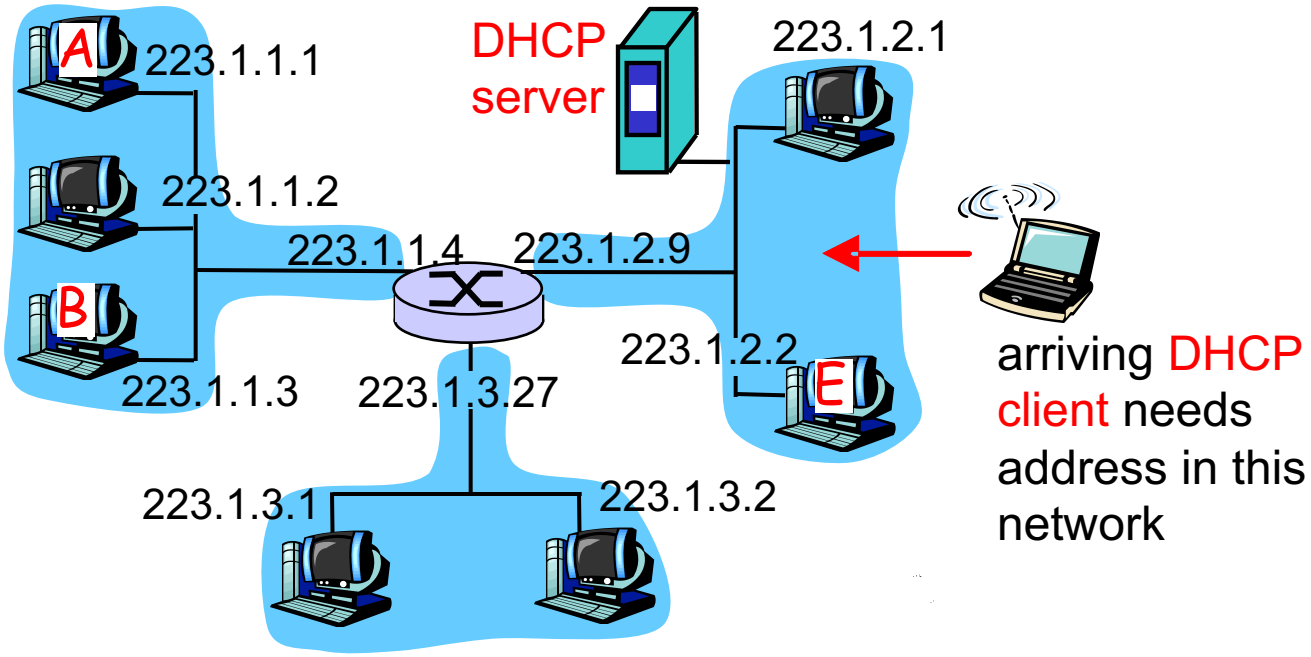
- Can renew its lease on address in use
- Allows reuse of addresses (only hold address while connected an "on")
- Support for mobile users who want to join network (more shortly)

DHCP overview:

- host broadcasts "DHCP discover" msg
- DHCP server responds with "DHCP offer" msg
- host requests IP address: "DHCP request" msg
- DHCP server sends address: "DHCP ack" msg

dhcp server, è un server che ha il compito di permettere ai nodi di farsi assegnare indirizzi consistenti alla rete in cui è collegato.

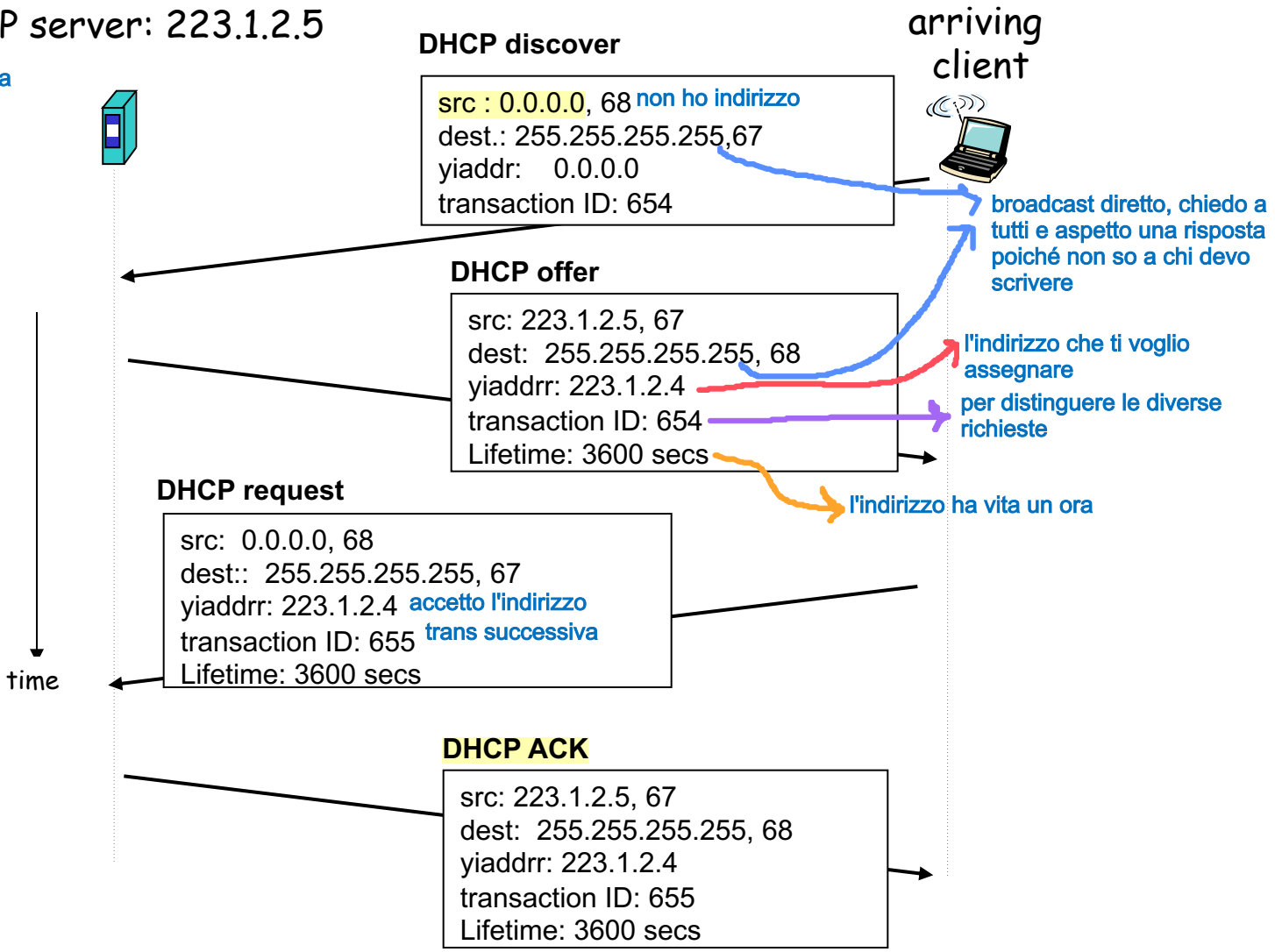
DHCP client-server scenario



DHCP client-server scenario

il protocollo è più complicato poiché potrebbero esserci più server che rispondono

1. Un client che arriva alla rete fa una richiesta di dhcp Discovery (chiede se esiste un server che assegna gli indirizzi, se non ci fosse andrebbe selezionato a mano)
2.



Getting a datagram from source to dest.

netstat -r chiede le informazioni di routing

le tabelle di instradamento sono contenute in tutti i router e in tutti gli host

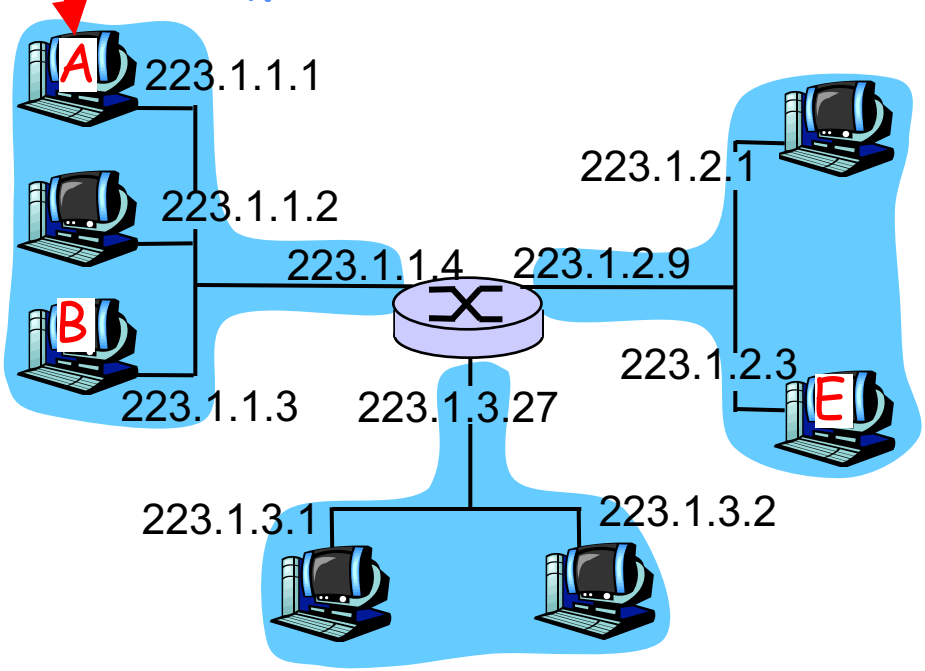
forwarding table in A

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
OTHER		

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



Getting a datagram from source to dest.

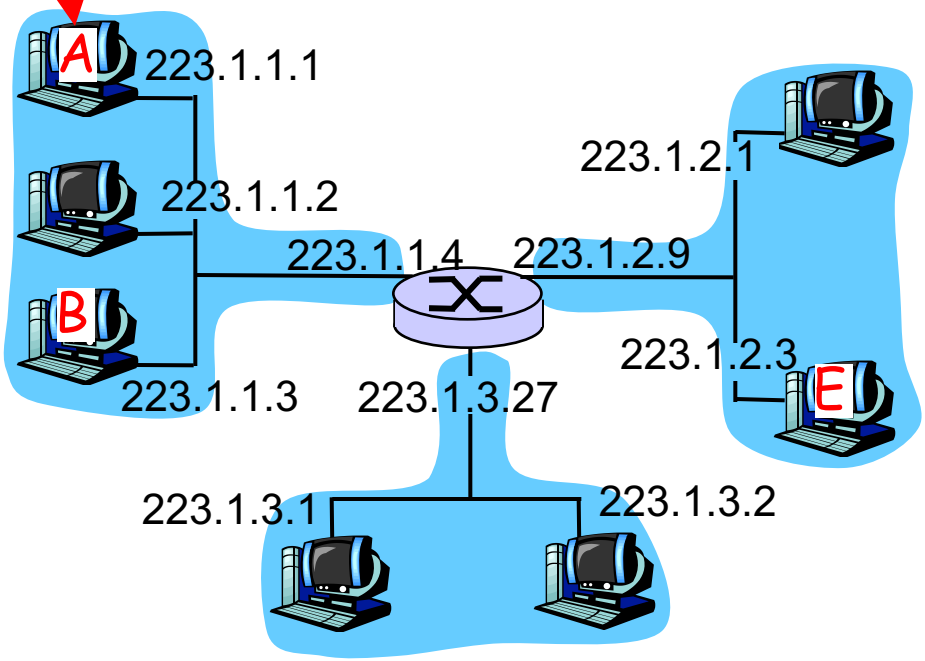
misc fields	223.1.1.1	223.1.1.3	data

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



caso in cui devo inviare un pkt nella mia sotto rete, guardo l'indirizzo, vedo nella mia tebella a quale caso corrisponde e notando che sta nella mia sottorete A utilizza il livello 2 (Lan/Wifi) senza dover passare in nessun router

Getting a datagram from source to dest.

Destinatario E, A guarda la sua tab di instradamento, nota che va inviato in un altra rete ip, e lo invia quindi al router tramite il livello 2.

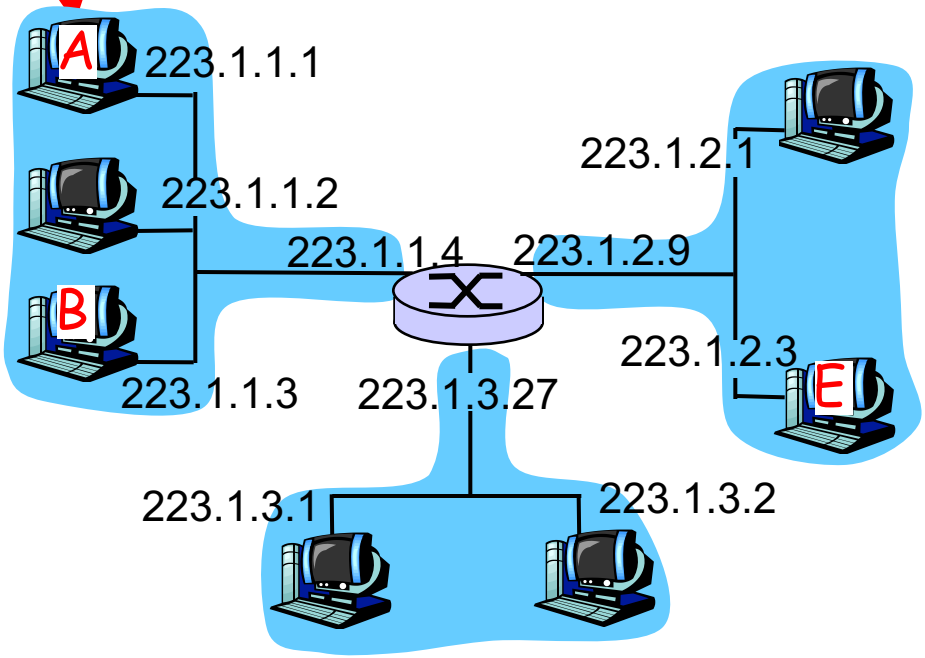
misc fields	223.1.1.1	223.1.2.3	data

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
223.1.1.0		1
223.1.2.0	223.1.1.4	2
223.1.3.0	223.1.1.4	2



Getting a datagram from source to dest.

il router la maggior parte dei casi riceve pacchetti non destinati a lui, e usa la sua tab di instradamento per inviarli. noto che E corrisponde al secondo caso

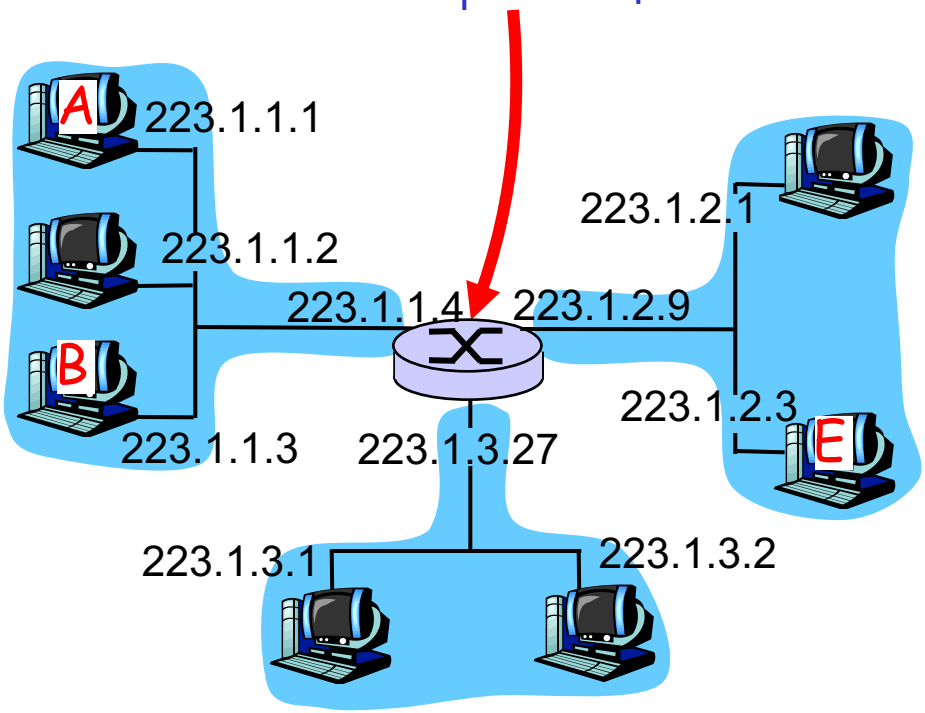
misc fields	223.1.1.1	223.1.2.3	data

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 223.1.2.3 inside link-layer frame via interface 223.1.2.9
- datagram arrives at 223.1.2.3!!! (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 addressing: last words …

Q: how does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers http://www.icann.org/

- allocates IP addresses, through 5 regional registries (RRs) (who may then allocate to local registries)
- manages DNS root zone, including delegation of individual TLD (.com, .edu , …) management

Q: are there enough 32-bit IP addresses?

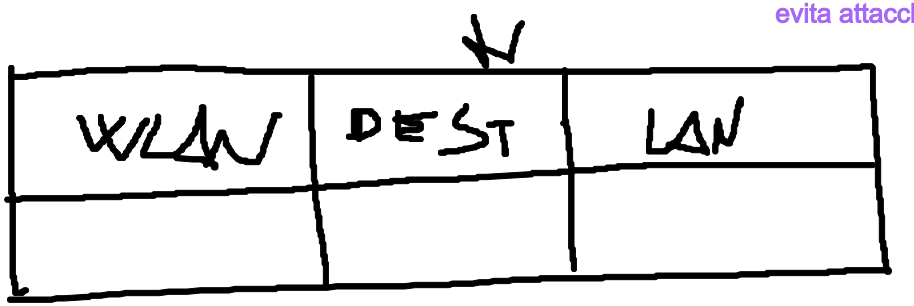
- ICANN allocated last chunk of IPv4 addresses to RRs in 2011
- NAT (next) helps IPv4 address space exhaustion
- IPv6 has 128-bit address space

"Who the hell knew how much address space we needed?" Vint Cerf (reflecting on decision to make IPv4 address 32 bits long)

Con l'utilizzo del nat il mio router "cambierà" il mio indirizzo ip, e se qualsiasi macchina esterna provasse ad inviare traffico a nuovo indirizzo con la stessa porta, il traffico mi arriverà. Questo è un problema alla sicurezza. Per ovviare a questo problema la tabella deve essere più aggiornata, con la variabile "chi sto contattando?". Aggiungendo un indirizzo ip del destinatario, per cui accetterò solo traffico inviato da lui.

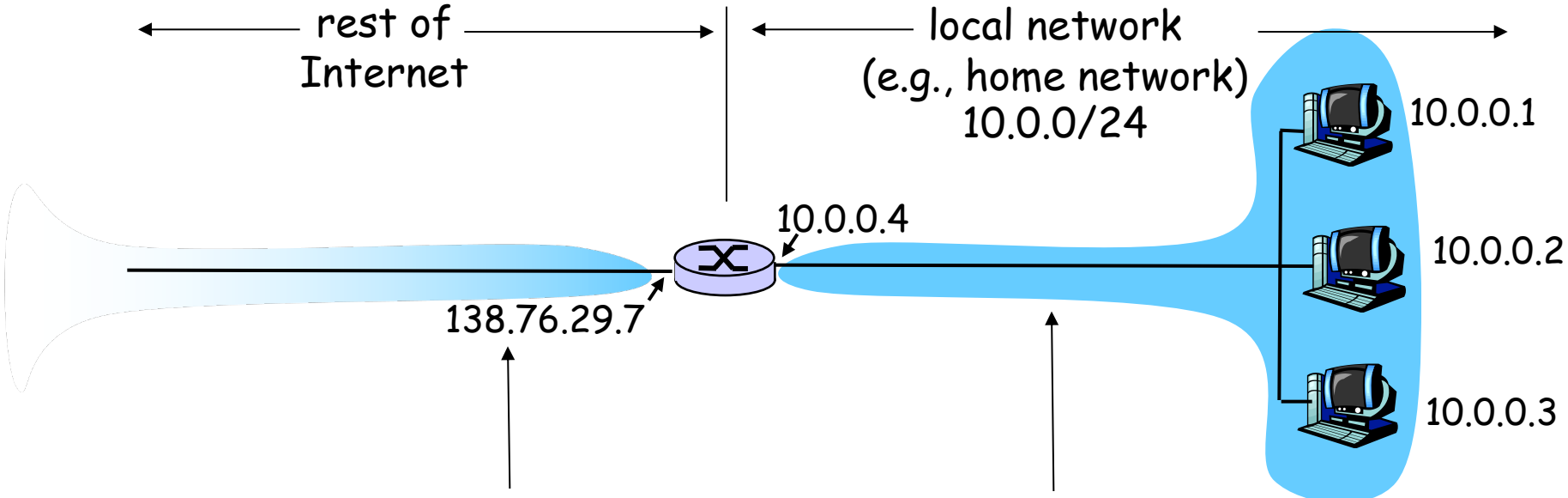
solo questo nodo può inviarci del traffico. NAT Simmetrica. Se A comunica con B, allora B può comunicare con A. Ma C non comunica con A.

nuova versione che evita attacchi



Un ulteriore attacco a questa nuova versione, è utilizzare come indirizzo mittente quello del campo destinazione della tabella

NAT: Network Address Translation



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)

tutto il traffico generato dalla rete locale inviato, quando il router se effettua una op di Nat, modifica l'indirizzo di rete sorgente sostituendolo con il proprio. Tutti quelli che ricevono pacchetti vedono come indirizzo sorgente quello del router e risponderanno a lui. Tutto ciò viene effettuato senza che source e dest si accorgano di qualcosa

NAT: Network Address Translation

- Motivation: local network uses just one IP address as far as outside world is concerned:
  - range of addresses not needed from ISP: just one IP address 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).

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

NAT: Network Address Translation

nodi dietro la nat non possono essere contattati dall'esterno. Skype, i client comunicano tramite un server, evitando la nat.

la nat va programmata molto bene.

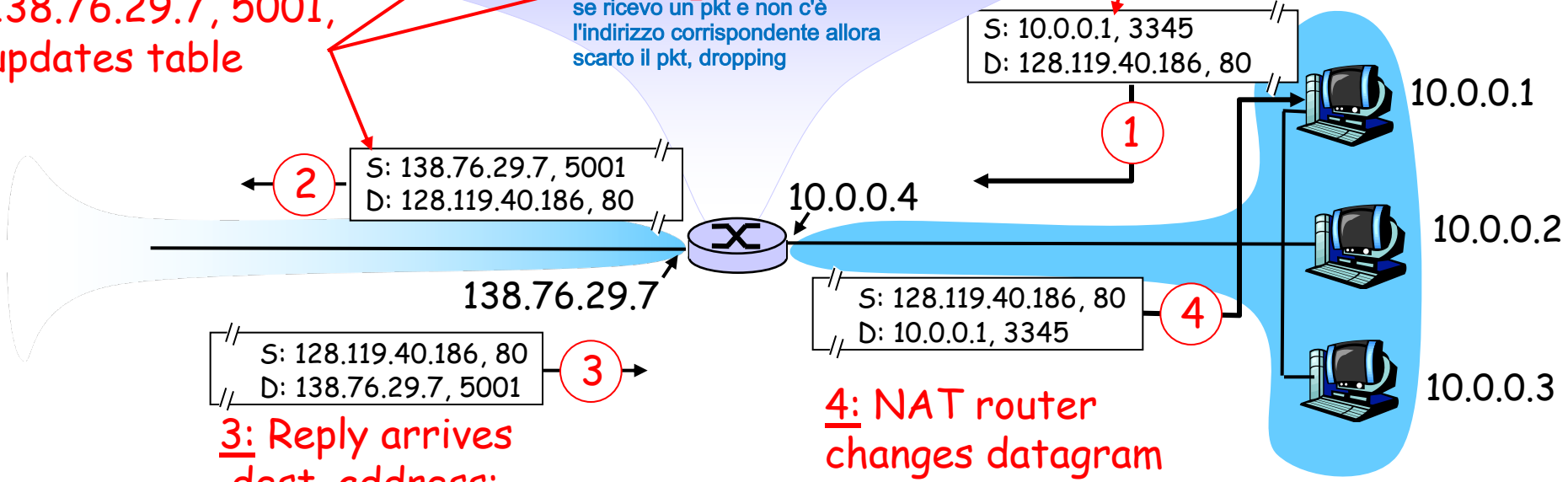
riassegno sia l'indirizzo ip che il numero di porta.

NAT translation table	
WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
.....	.....

2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table

se ricevo un pkt e non c'è l'indirizzo corrispondente allora scarto il pkt, dropping

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80



3: Reply arrives dest. address: 138.76.29.7, 5001

4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345

se il destinatario invia più file nella stessa sottorete ma a host diversi, vengono distinti tramite il numero di porta.

La nat ha dei timer per mantenere un collegamento in tabella, poiché le porte sono in numero finito. Alcune app inviano pacchetti periodici che servono per non far scadere timer

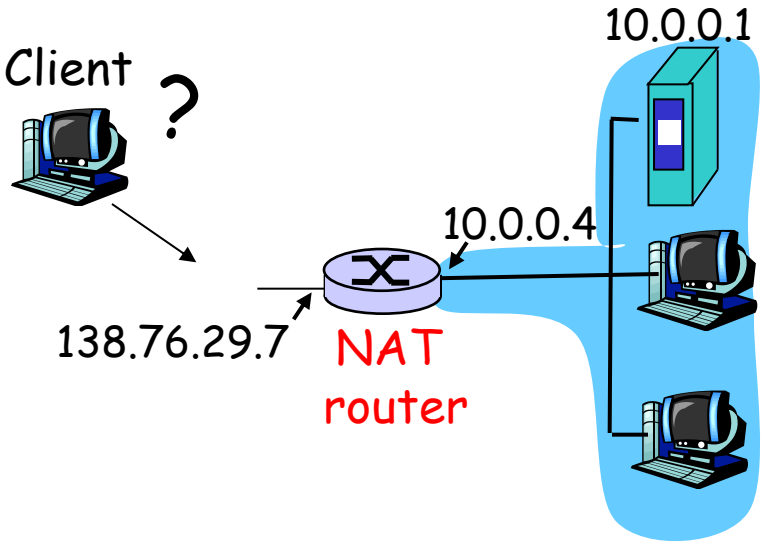
NAT: Network Address Translation

- 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
  - address shortage should instead be solved by IPv6

ma poiché modificano il num di porta, guardano anche l'intestazione del livello 4.

NAT traversal problem

- client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - only one externally visible NATted address: 138.76.29.7
- solution 1: statically configure NAT to forward incoming connection requests at given port to server
  - e.g., (138.76.29.7, port 80) always forwarded to 10.0.0.1 port 80 aggiungo a mano nella tabella nat -> Limite posso avere soltanto un server (un 80 soltanto)

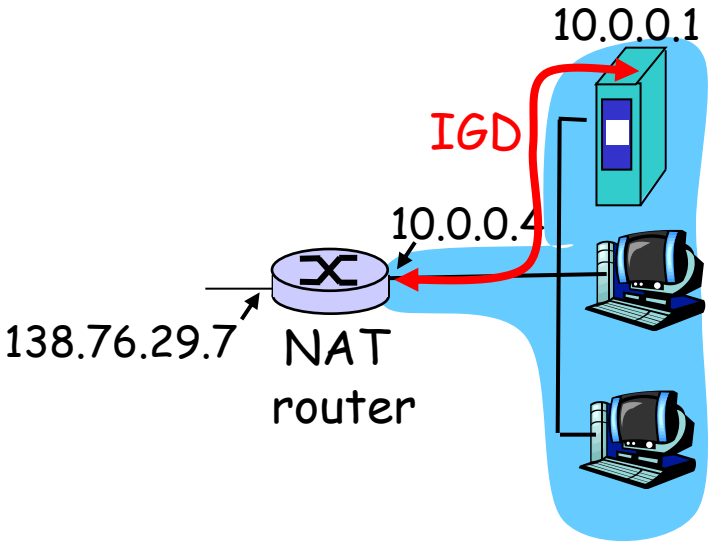


chiedo al nat se quel nodo si trova dietro di lui, esistono dei protocolli.

NAT traversal problem

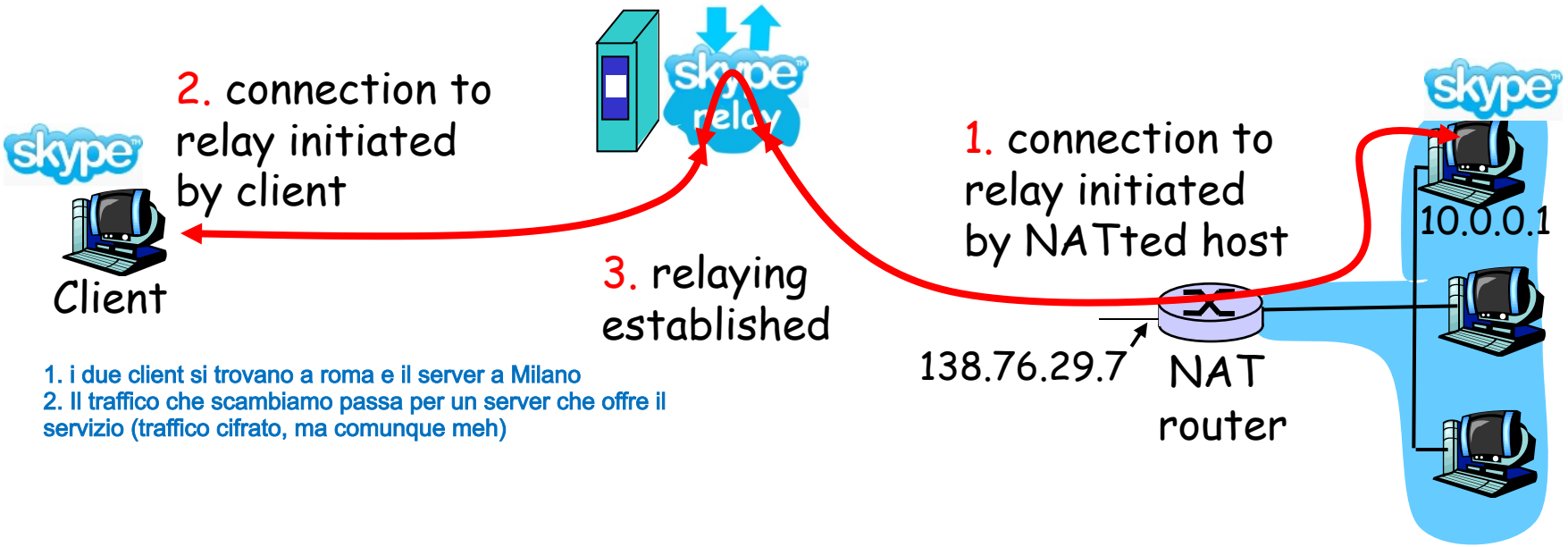
- solution 2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATted host to:
  - learn public IP address (138.76.29.7)
  - add/remove port mappings (with lease times)

  i.e., automate static NAT port map configuration



NAT traversal problem

- solution 3: relaying (used in Skype)
  - NATed client establishes connection to relay
  - External client connects to relay
  - relay bridges packets between to connections



1. i due client si trovano a roma e il server a Milano
2. Il traffico che scambiamo passa per un server che offre il servizio (traffico cifrato, ma comunque meh)

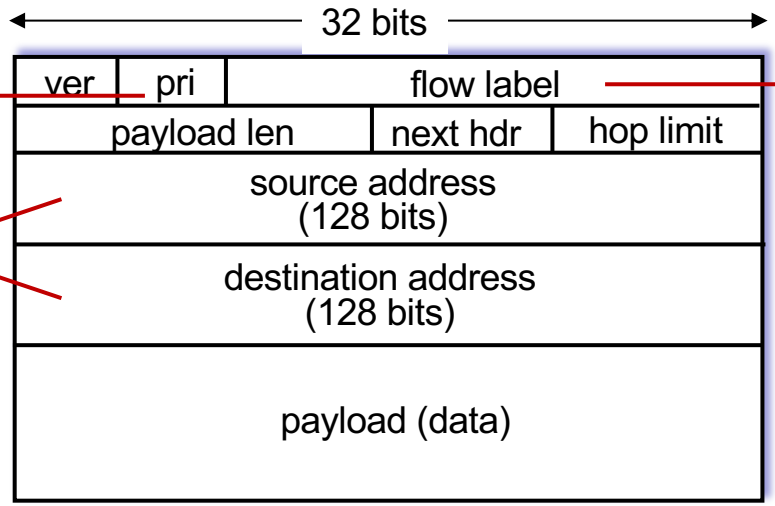
IPv6: motivation

- initial motivation: 32-bit IPv4 address space would be completely allocated
- additional motivation:
  - speed processing/forwarding: 40-byte fixed length header
  - enable different network-layer treatment of “flows”

IPv6 datagram format

priority: identify priority among datagrams in flow

128-bit IPv6 addresses



flow label: identify datagrams in same "flow." (concept of "flow" not well defined).

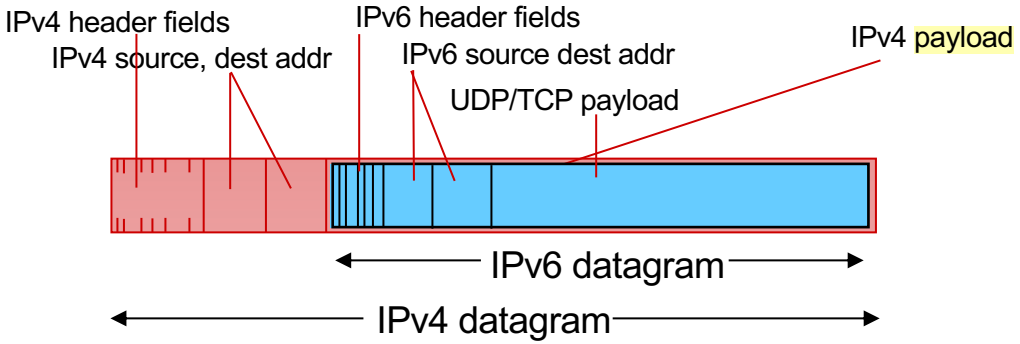
What's missing (compared with IPv4):

- no checksum (to speed processing at routers) non lo gestisco al livello 3 ma al livello 4 si
- no fragmentation/reassembly
- no options (available as upper-layer, next-header protocol at router)

Transition from IPv4 to IPv6

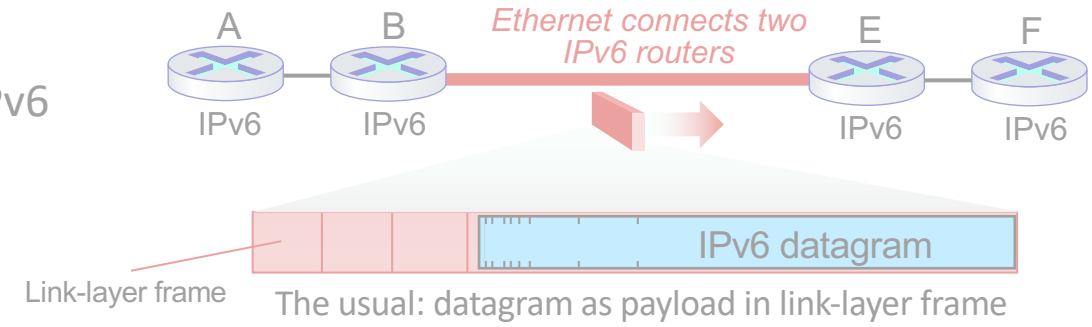
- not all routers can be upgraded simultaneously
  - no "flag days"
  - how will network operate with mixed IPv4 and IPv6 routers?
  - tunneling: IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers ("packet within a packet")
    - tunneling used extensively in other contexts (4G/5G)

utilizzo un protocollo di rete in un protocollo di rete

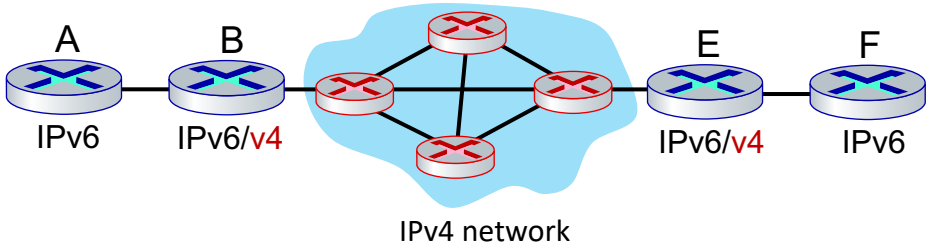


Tunneling and encapsulation

Ethernet connecting two IPv6 routers:

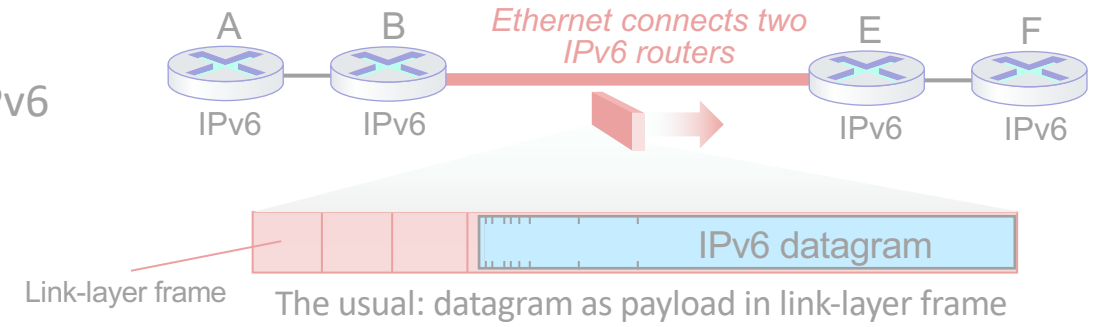


IPv4 network connecting two IPv6 routers

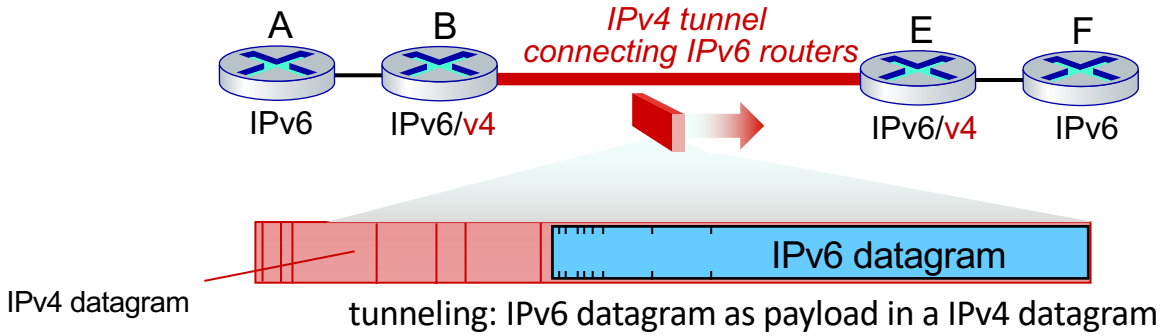


Tunneling and encapsulation

Ethernet connecting two IPv6 routers:

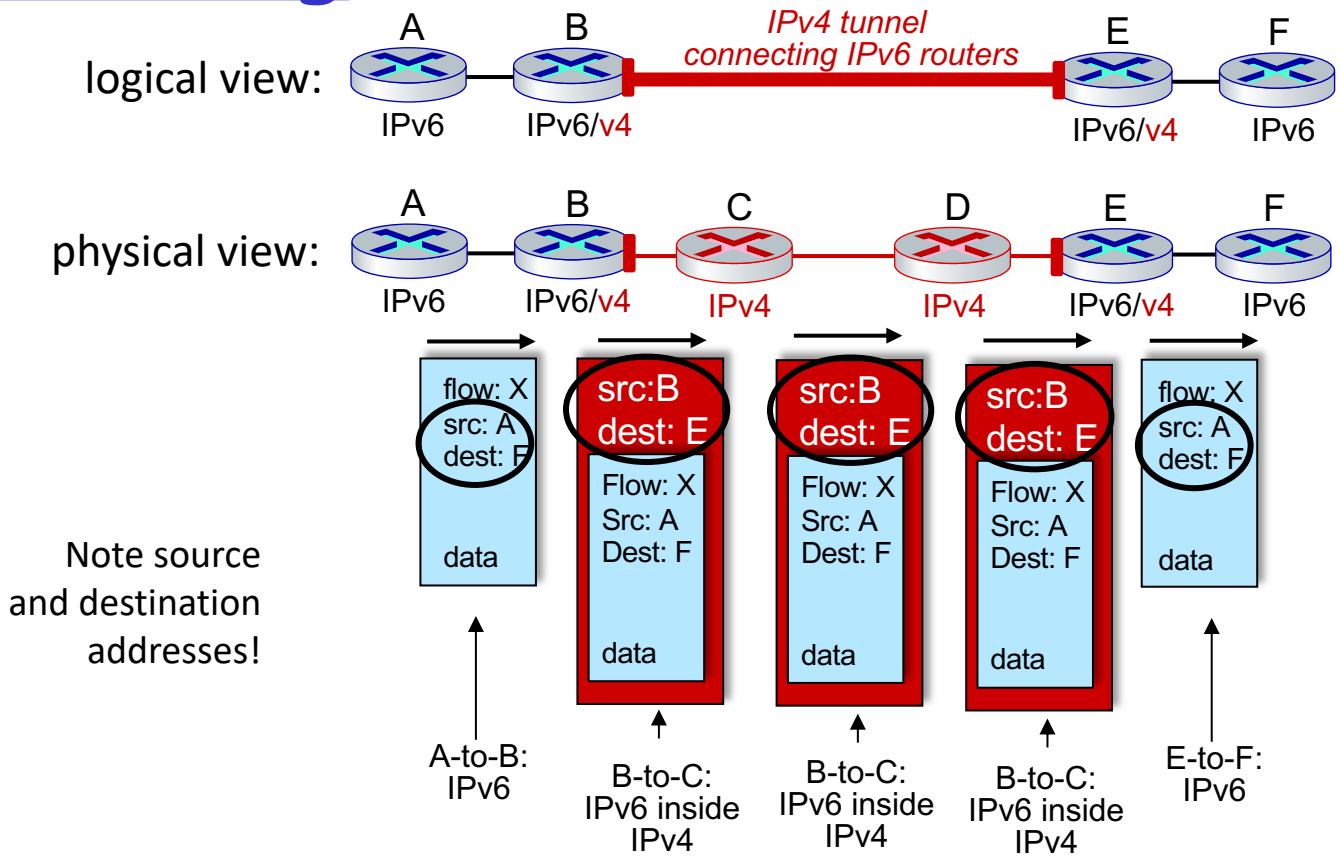


IPv4 tunnel connecting two IPv6 routers



i pkt ipv4 ha un intestazione upper layer, dove può venire specificato ipv6/ipv4/ip

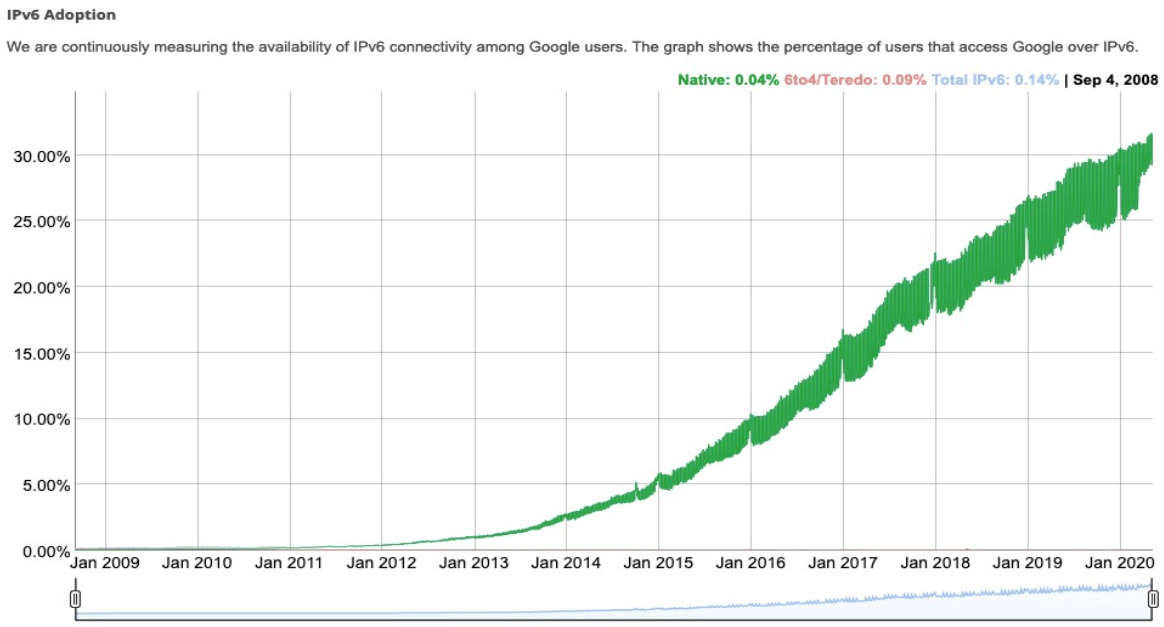
Tunneling



Note source and destination addresses!

IPv6: adoption

- Google[1]: ~ 30% of clients access services via IPv6
- NIST: 1/3 of all US government domains are IPv6 capable



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

IPv6: adoption

- Google[1]: ~ 30% of clients access services via IPv6
- NIST: 1/3 of all US government domains are IPv6 capable
- Long (long!) time for deployment, use
  - 25 years and counting!
  - think of application-level changes in last 25 years: WWW, social media, streaming media, gaming, telepresence, …
  - *Why?*

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

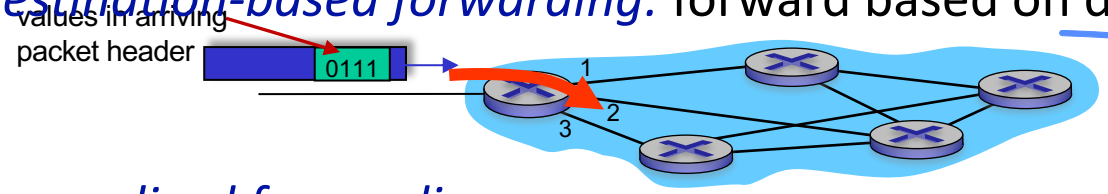
Generalized forwarding: match plus action

In base alla configurazione del mio pkt il router fa qualcosa, non effettua più soltanto una lettura di leggere e inviare.

Se il pkt corrisponde a questo profilo esegui questa azione

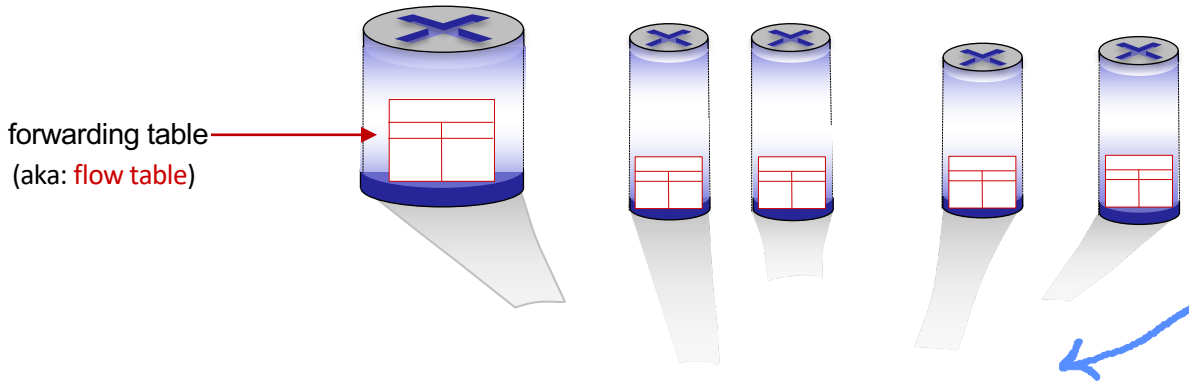
Review: each router contains a forwarding table (aka: flow table)

- "match plus action" abstraction: match bits in arriving packet, take action
  - *destination-based forwarding:* forward based on dest. IP address



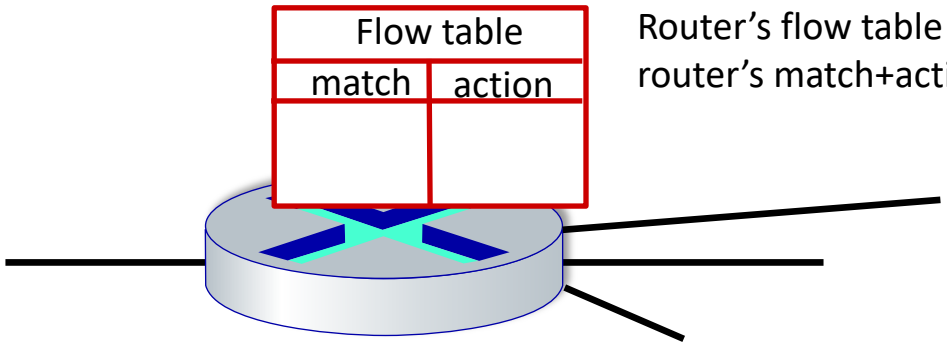
ogni pacchetto scatena un azione.

  - *generalized forwarding*:
    - many header fields can determine action
    - many action possible: drop/copy/modify/log packet



Flow table abstraction

- flow: defined by header field values (in link-, network-, transport-layer fields)
- generalized forwarding: simple packet-handling rules
  - match: pattern values in packet header fields — verifica di un pattern all'interno dell'intestazione (di tutti i livelli)
  - actions: for matched packet: drop, forward, modify, matched packet or send matched packet to controller
  - priority: disambiguate overlapping patterns — se ho più pattern che corrispondono a diverse azioni, eseguo quella con priorità più alta
  - counters: #bytes and #packets

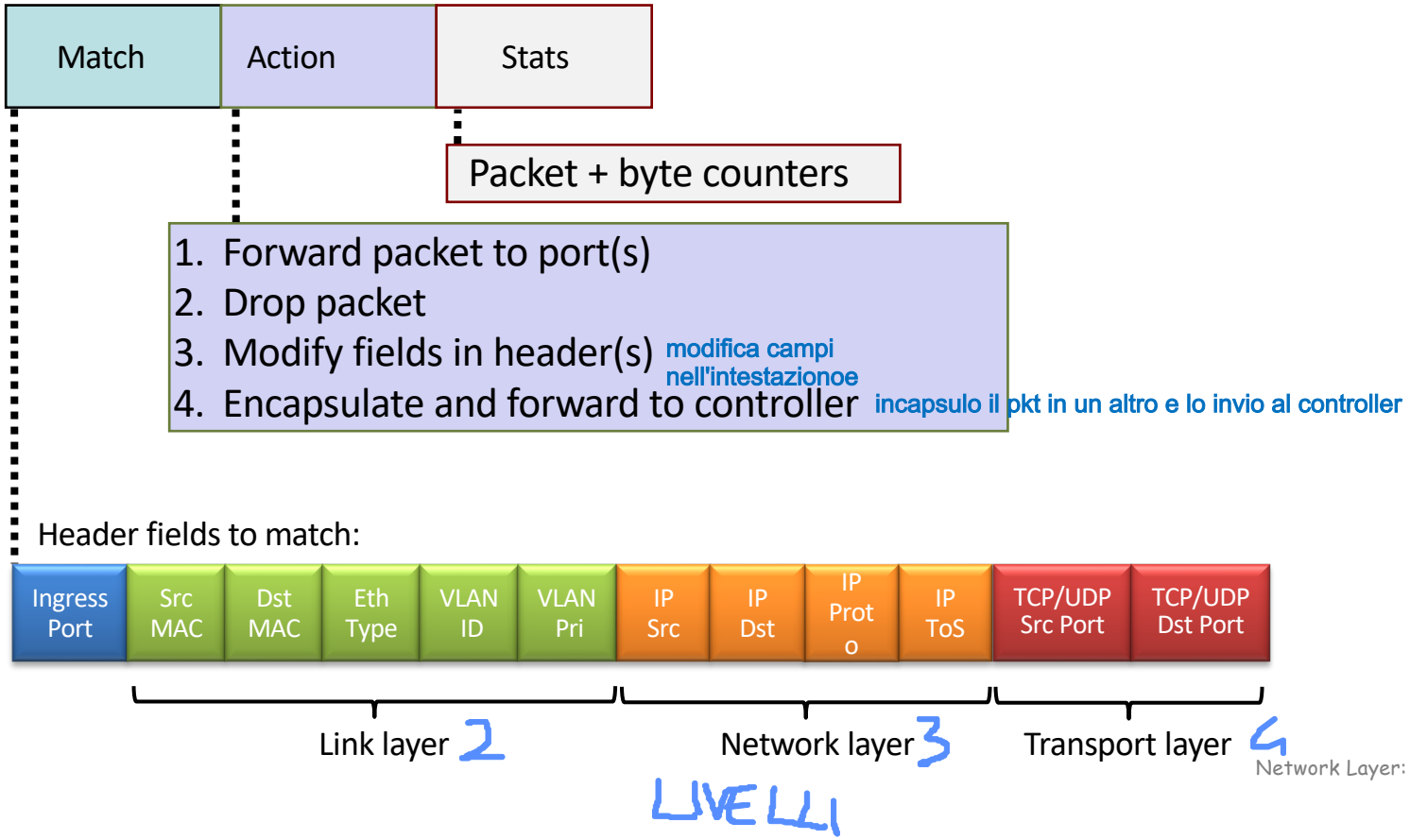


Router's flow table define router's match+action rules

Flow table abstraction

- flow: defined by header fields
- generalized forwarding: simple packet-handling rules
  - match: pattern values in packet header fields
  - actions: for matched packet: drop, forward, modify, matched packet or send matched packet to controller
  - priority: disambiguate overlapping patterns
  - counters: #bytes and #packets

OpenFlow: flow table entries



proxy web, blocca i siti

si può evitare con un firewall.
in genere un firewall è un apparato diverso dal router, ma il router può eseguire, se programmato, può svolgere azioni di firewall

OpenFlow: examples

Destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	*	51.6.0.8	*	*	*	*	port6

IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6

Firewall:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	*	*	*	*	*	22	drop

Block (do not forward) all datagrams destined to TCP port 22 (ssh port #)

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	128.119.1.1	*	*	*	*	*	drop

Block (do not forward) all datagrams sent by host 128.119.1.1

OpenFlow: examples

Layer 2 destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	22:A7:23: 11:E1:02	*	*	*	*	*	*	*	*	*	port3

layer 2 frames with destination MAC address 22:A7:23:11:E1:02 should be forwarded to output port 3

OpenFlow abstraction

- match+action: abstraction unifies different kinds of devices

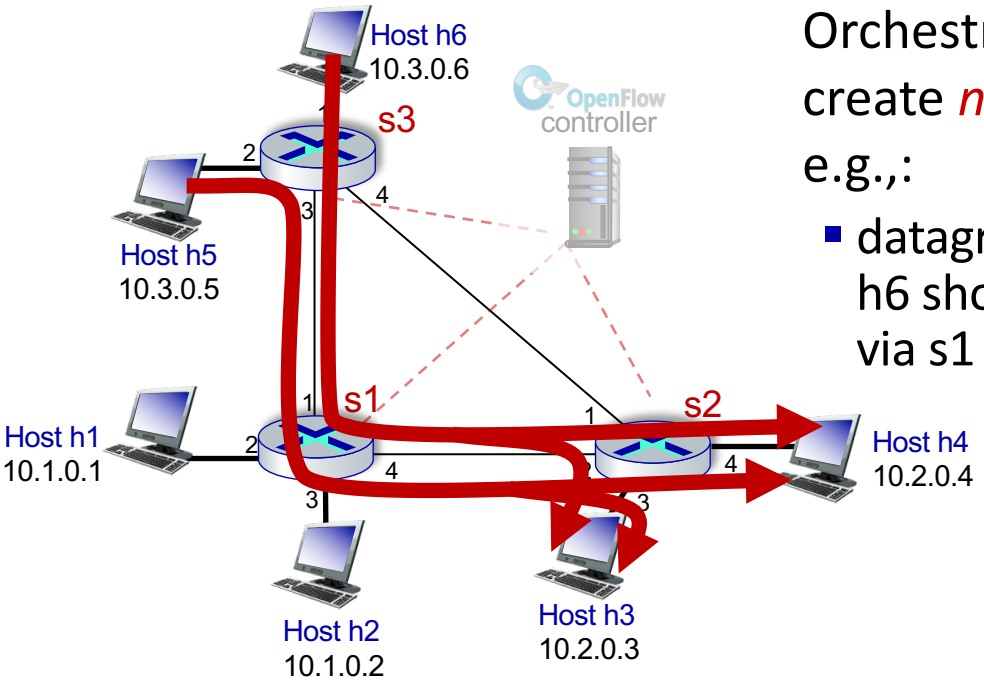
Router
- *match:* longest destination IP prefix
- *action:* forward out a link

Switch
- *match:* destination MAC address
- *action:* forward or flood

Firewall
- *match*: IP addresses and TCP/UDP port numbers
- *action:* permit or deny

NAT
- *match:* IP address and port
- *action:* rewrite address and port

OpenFlow example

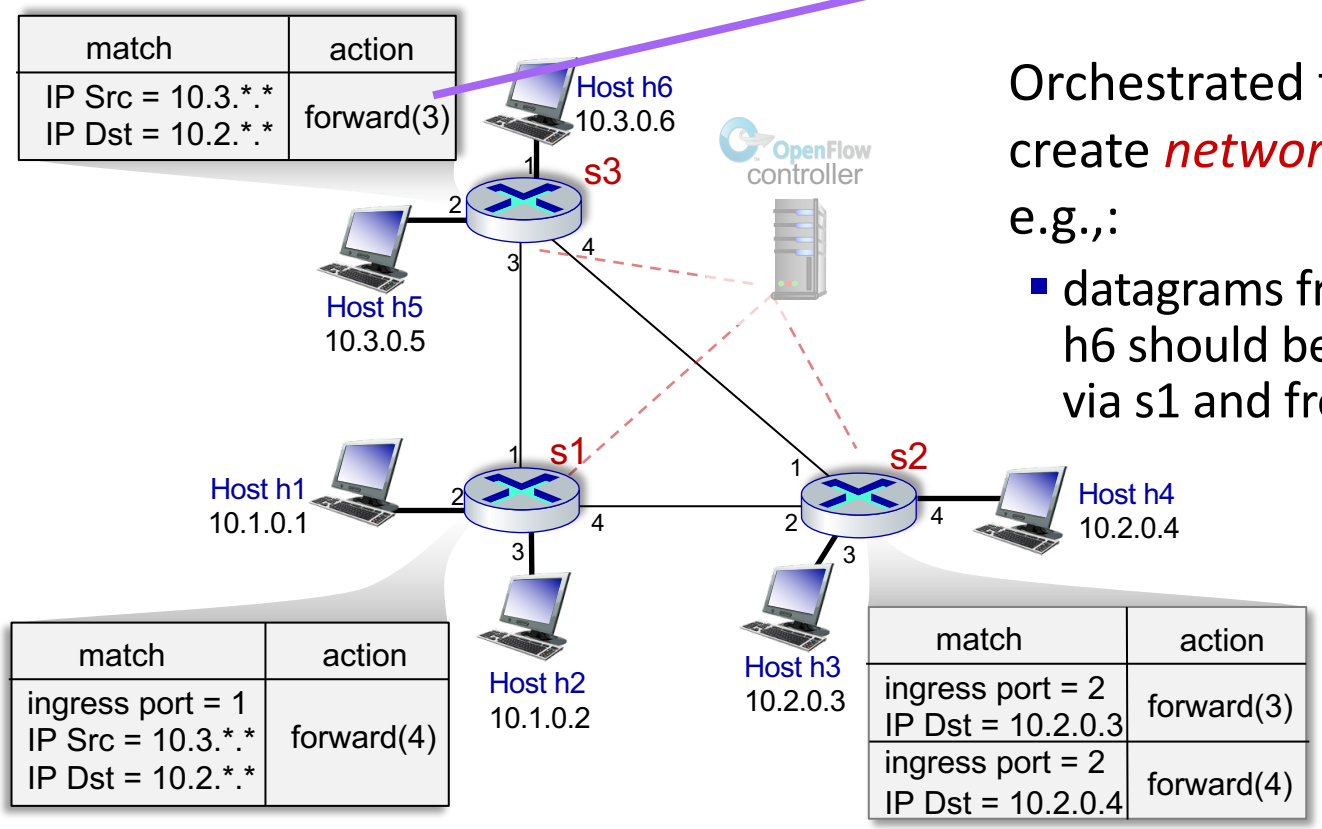


Orchestrated tables can create *network-wide* behavior, e.g.,:

- datagrams from hosts h5 and h6 should be sent to h3 or h4, via s1 and from there to s2

OpenFlow example

falli passare sul link 3



Orchestrated tables can create *network-wide* behavior, e.g.,:

- datagrams from hosts h5 and h6 should be sent to h3 or h4, via s1 and from there to s2

Generalized forwarding: summary

- "match plus action" abstraction: match bits in arriving packet header(s) in any layers, take action
  - matching over many fields (link-, network-, transport-layer)
  - local actions: drop, forward, modify, or send matched packet to controller
  - "program" n*etwork-wide* behaviors
- simple form of "network programmability"
  - programmable, per-packet "processing"
  - *historical roots:* active networking
  - *today:* more generalized programming: P4 (see p4.org).

Middleboxes

scatole di mezzo, apparati aggiuntivi alla rete
(firewall, nat, proxy cache, ecc)

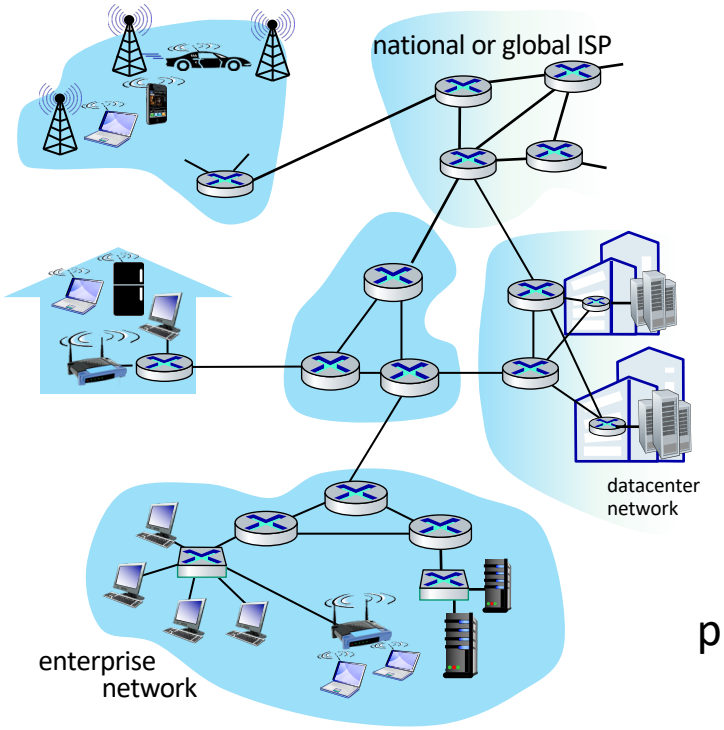
Middlebox (RFC 3234)

"any intermediary box performing functions apart from normal, standard functions of an IP router on the data path between a source host and destination host"

Middleboxes everywhere!

NAT: home, cellular, institutional

Application-specific: service providers, institutional, CDN



Firewalls, IDS: corporate, institutional, service providers, ISPs

Load balancers: corporate, service provider, data center, mobile nets

Caches: service provider, mobile, CDNs

Middleboxes

- initially: proprietary (closed) hardware solutions
- move towards "whitebox" hardware implementing open API
  - move away from proprietary hardware solutions
  - programmable local actions via match+action
  - move towards innovation/differentiation in software
- SDN: (logically) centralized control and configuration management often in private/public cloud
- network functions virtualization (NFV): programmable services over white box networking, computation, storage

componenti non proprietari, programmabili (renderle Generali per tutti)

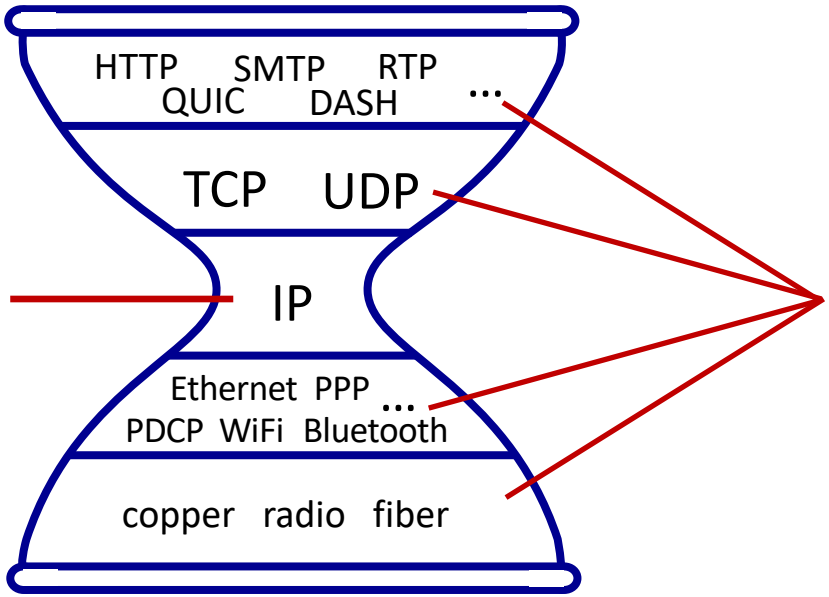
controllore centralizzato su un cloud

i servizi vengono virtualizzati.

The IP hourglass

Internet's "thin waist":

- *one* network layer protocol: IP
- *must* be implemented by every (billions) of Internet-connected devices

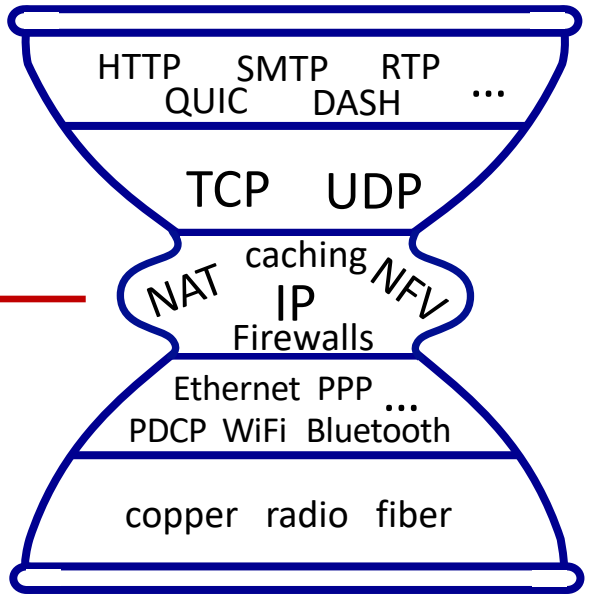


many protocols in physical, link, transport, and application layers

The IP hourglass, at middle age

Internet's middle age "love handles"?

- middleboxes, operating inside the network



Architectural Principles of the Internet

RFC 1958

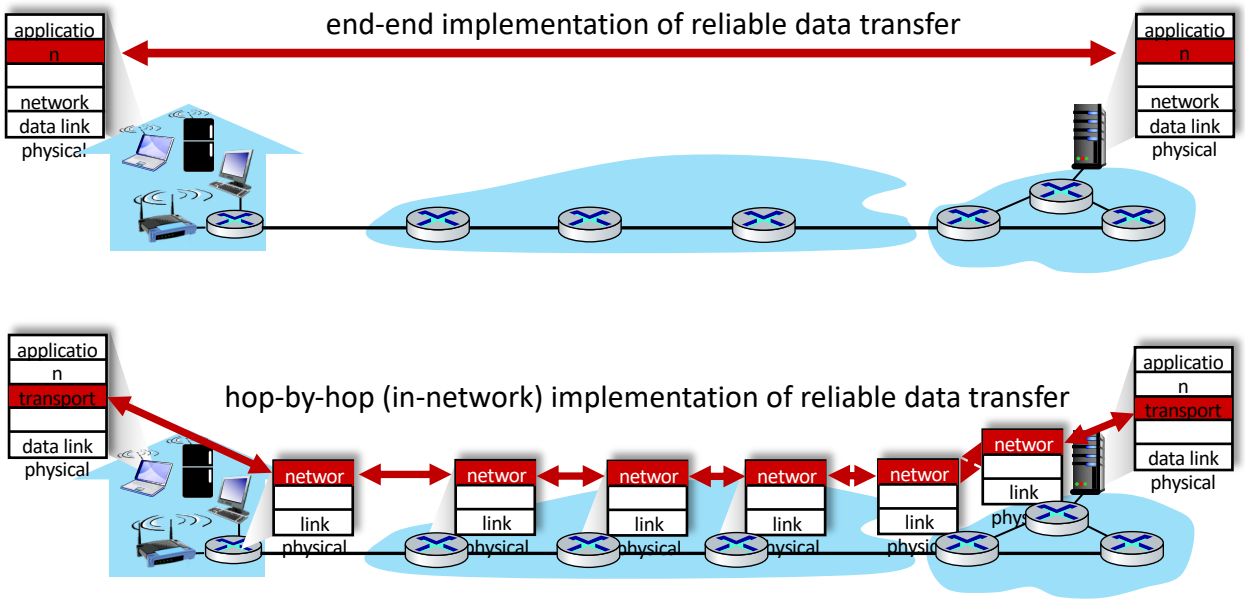
"Many members of the Internet community would argue that there is no architecture, but only a tradition, which was not written down for the first 25 years (or at least not by the IAB). However, in very general terms, the community believes that the goal is connectivity, the tool is the Internet Protocol, and the intelligence is end to end rather than hidden in the network."

Three cornerstone beliefs:

- simple connectivity
- IP protocol: that narrow waist
- intelligence, complexity at network edge

The end-end argument

- some network functionality (e.g., reliable data transfer, congestion) can be implemented in network, or at network edge



The end-end argument

- some network functionality (e.g., reliable data transfer, congestion) can be implemented in network, or at network edge

> "The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible. (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.)
>
> We call this line of reasoning against low-level function implementation the "end-to-end argument."
>
> Saltzer, Reed, Clark 1981