Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- **datagram format**
- fragmentation
- IPv4 addressing
- forwarding
- DHCP
- network address translation
- error messages (ICMP)
- IPv6

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

Internet Protocol

- Uno dei due protocolli principali della suite TCP/IP
- Obiettivi del protocollo
  - Nascondere l'etereogeneità
  - Fornire l'illusione di un'unica rete
  - Virtualizzare gli accessi alla rete

Concetto

IP permette ad un utente di vedere (e percepire) una rete internet come una singola rete virtuale che interconnette tutti gli host che permette la comunicazione tra host. La struttura sottostante è nascosta ed è irrilevante.

Il protocollo IP: Consegna di Datagrammi, Senza Connessione

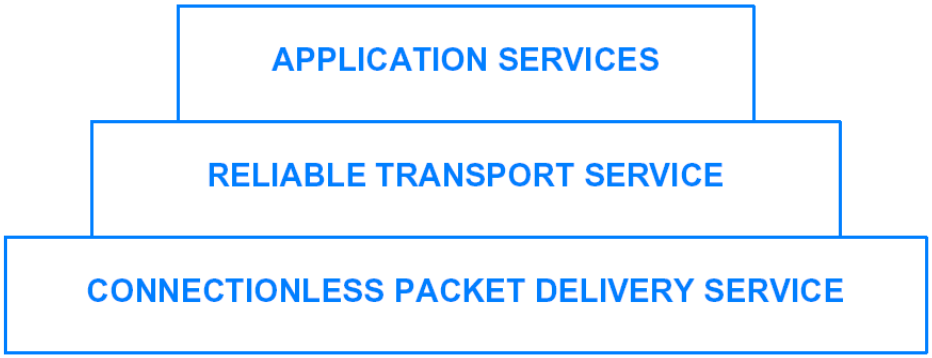
- IP è il protocollo che è responsabile della interconnessione di sottoreti che costituisce l'internet
- È un protocollo di consegna di messaggi senza connessione (**connectionless**)
- **Una rete virtuale**

  Fino ad adesso abbiamo studiato l'architettura di una internet.

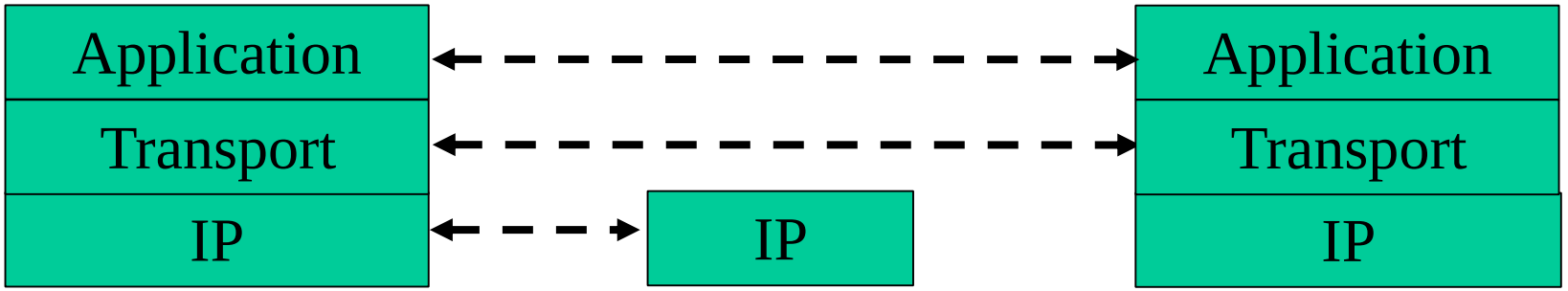
  L'utente non si deve preoccupare delle tecniche necessarie per realizzare l'interconnessione di due qualunque macchine nel mondo

  Ora ci occuperemo proprio di come si realizza questa astrazione di rete virtuale

Architettura e filosofia di internet (I)



Internet è organizzata su tre livelli di servizio
Ognuno di questi livelli di servizio è realizzato mediante opportuni protocolli perché sono *servizi distribuiti*
Ogni livello è costituito da **entità** che si trovano su macchine diverse...



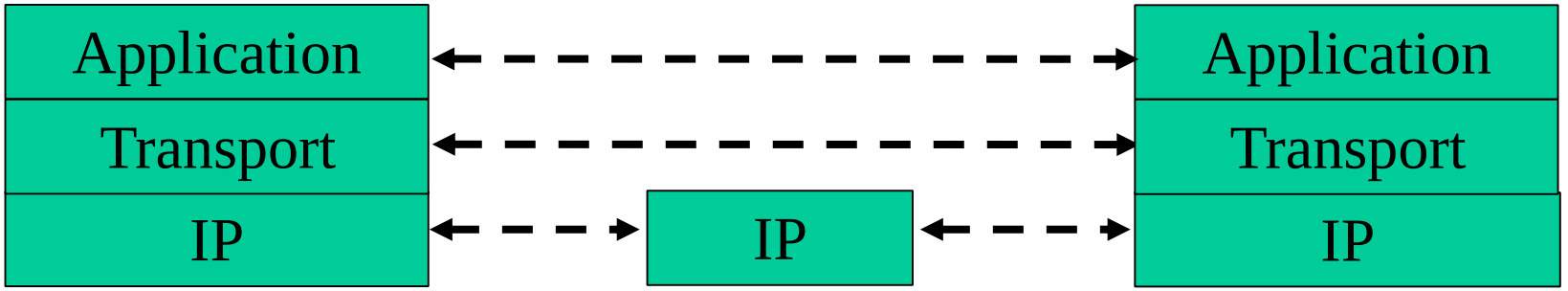
 = protocollo IP, di trasporto o applicativo

Architettura e filosofia di internet (II)

Ogni entità di protocollo comunica con una entità paritaria, dello stesso livello e tipo, che obbedisce allo stesso protocollo (formato dei dati scambiati + regole di comportamento)

La comunicazione avviene interagendo con il livello di servizio sottostante:

- Le applicazioni interagiscono usando il transport
- I transport interagiscono usando IP
- Le entità IP interagiscono usando le reti fisiche che le mettono in comunicazione, e tramite le entità IP dei router




= protocollo IP, di trasporto o applicativo

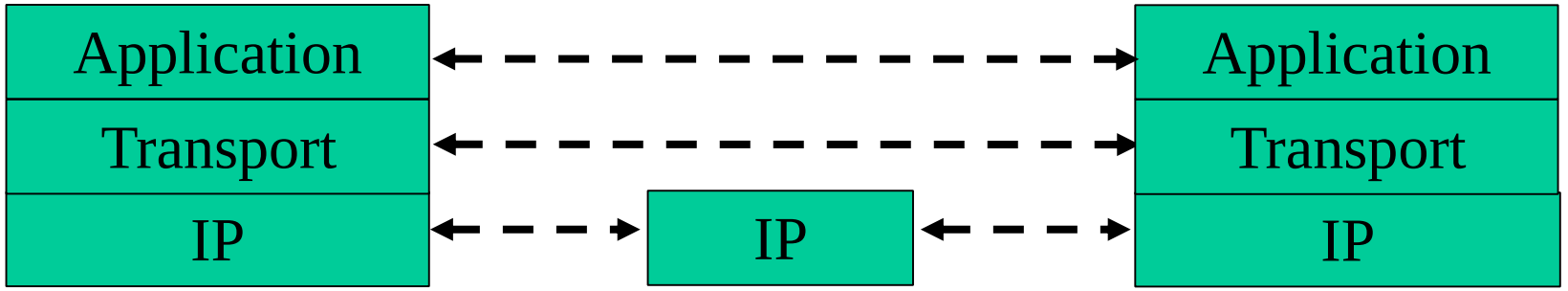
Architettura e filosofia di internet (III)

La comunicazione è "in orizzontale", l'interazione è "in verticale"

In orizzontale comunicano **entità paritarie**

In verticale interagiscono **entità fornitore di servizio** e entità **utilizzatore di servizio**

Quando avrete capito questo … avrete (una del)le chiavi del mondo dei sistemi distribuiti!



= protocollo IP, di trasporto o applicativo

L'organizzazione concettuale del servizio

La suddivisione del sistema in tre livelli di servizi di rete, organizzati in una gerarchia, è la base del successo di Internet

È possibile rimpiazzare o estendere uno dei tre senza disturbare gli altri

Un sistema di consegna senza connessione (connectionless) (I)

Il servizio fondamentale di una internet è un **sistema di consegna di pacchetti**

- **senza** necessità di creare una **connessione** fra mittente e destinatario
- **best-effort**, sfrutta **al meglio** le risorse disponibili
- **non necessariamente affidabile** (non è in grado di **garantire** una data qualità di servizio) ma che offre la **qualità migliore possibile con le risorse a disposizione**

Connectionless: ogni pacchetto viaggia sulla rete "per conto suo", seguendo la strada possibile in quel momento: le risorse sono sfruttate al meglio (***best-effort***) e non sono "prenotate" per una connessione

Non necessariamente affidabile: come conseguenza dell'uso ottimale delle risorse, i pacchetti possono essere

- ritardati,
- consegnati in ordine diverso da quello della spedizione
- persi,
- duplicati

Un sistema di consegna senza connessione (connectionless) (II)

Importante: i pacchetti non sono persi perché i mezzi di comunicazione sono fallaci (anzi, sono in genere molto robusti), né perché IP li butti via a capriccio. Vengono persi perché sono **scartati** quando non si hanno risorse locali per gestirli

Vedremo nel seguito quali risorse

In alcune situazioni, quando IP scarta un pacchetto sa benissimo che lo sta scartando, quindi perdendo. In altre situazioni invece non è a conoscenza di aver scartato un pacchetto

Scopo dell'internet protocol (IP)

Realizzare il servizio di consegna di pacchetti connectionless, best-effort, non necessariamente affidabile

Dobbiamo:

1. Definire il **formato** della unità di base di trasferimento dell'informazione (pacchetti)
2. Realizzare la funzione di **routing** (instradamento) e **forwarding** (inoltro) che interconnette le varie sottoreti
3. Definire un insieme di **regole di protocollo** che incorporano la nozione di consegna connectionless best-effort, non affidabile:
   a) come processare i pacchetti
   b) come e quando generare messaggi di errore
   c) sotto quali condizioni è ammissibile scartare pacchetti

Ora parliamo del formato dei pacchetti

I datagrammi internet

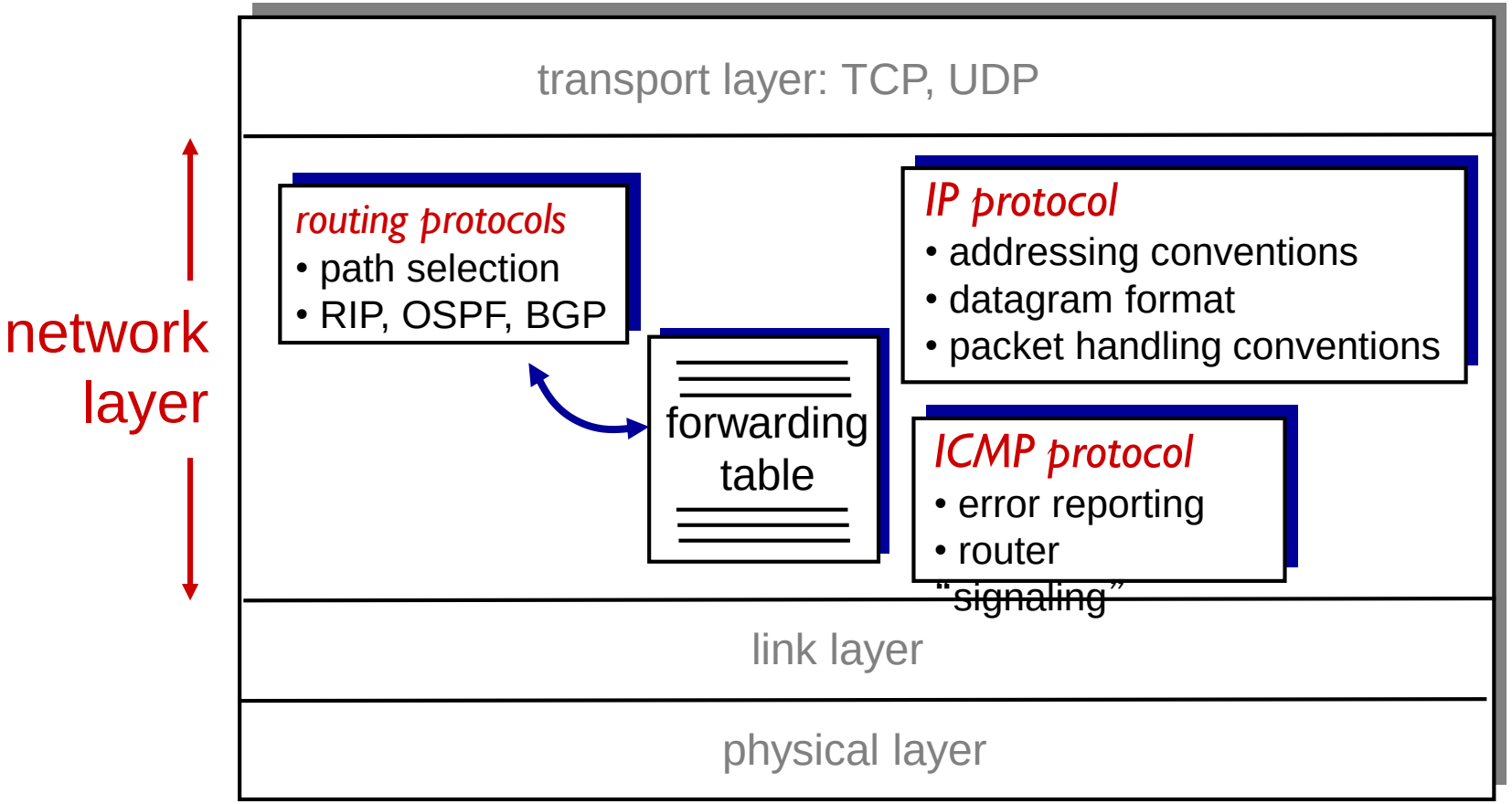
L'unità di trasferimento di dati si chiama *internet datagram*, *IP datagram* oppure semplicemente *datagram*

Si chiamano datagram per sottolineare che non fanno parte di una connessione. Hanno struttura simile a quella dei mezzi fisici

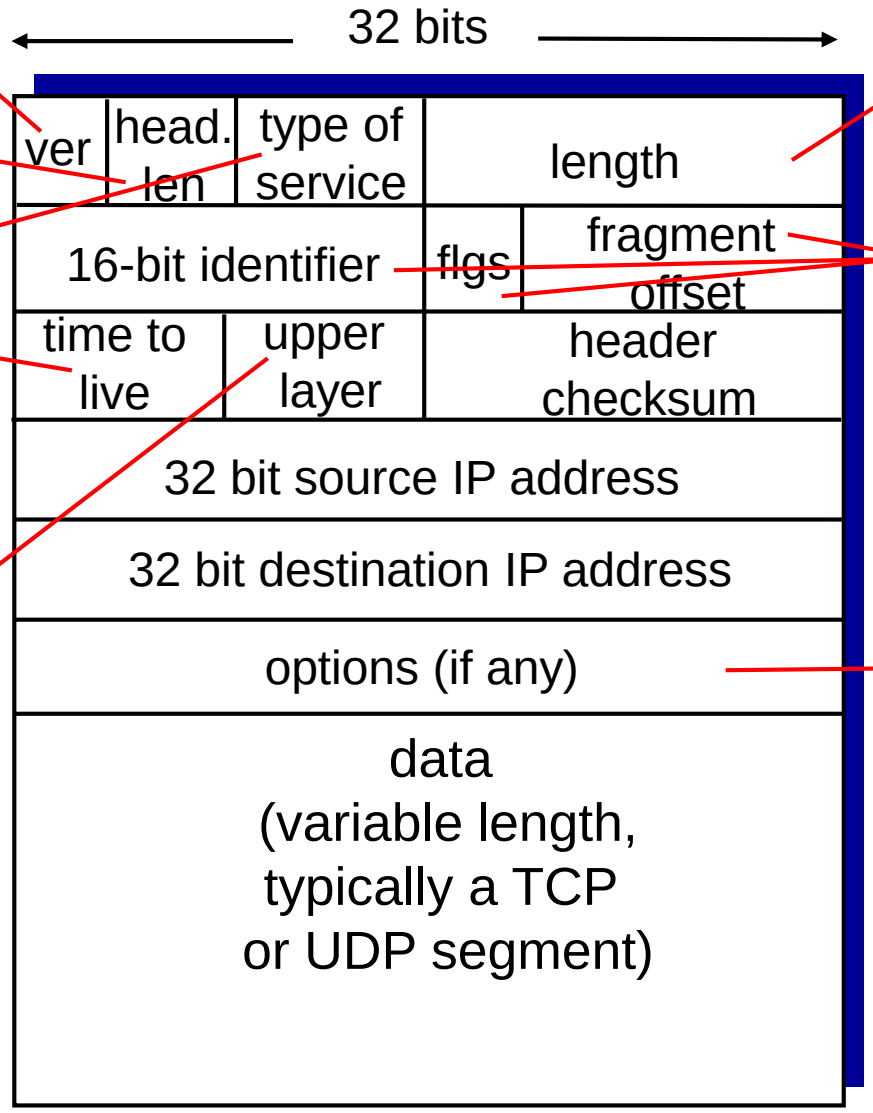
Il formato della **DATA AREA** non è fissato da IP, ma dai protocolli di livello superiore

The Internet network layer

host, router network layer functions:



IP datagram format



IP protocol version number

header length (bytes)

"type" of data

max number remaining hops (decremented at each router)

upper layer protocol to deliver payload to

total datagram length (bytes)

for fragmentation/ reassembly

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

how much overhead?

20 bytes of TCP

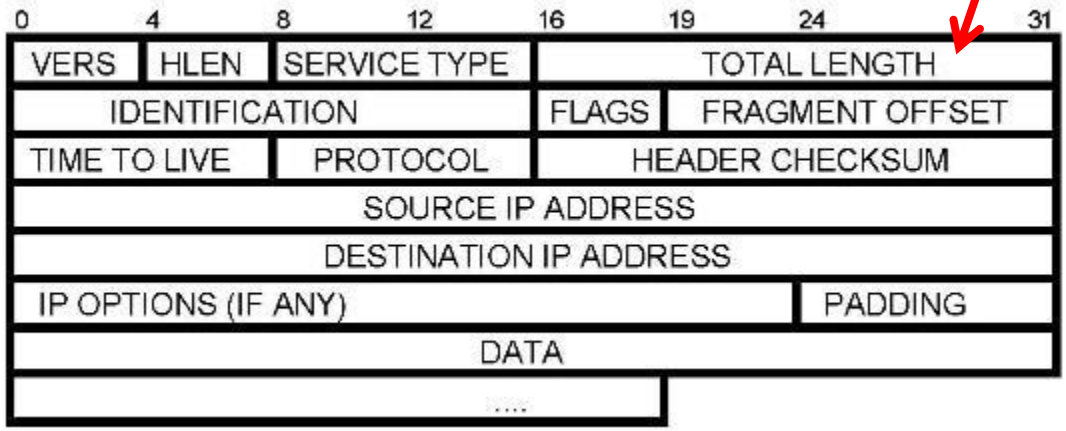
20 bytes of IP

= 40 bytes + app layer overhead

Formato del datagramma IP

Perché occorre esprimere anche la lunghezza totale del datagramma?
Il Data Link mi dice quanto è lunga la sua zona dati, che è occupata dal datagramma.
Perché questa informazione non basta (può non bastare)?
Perché alcuni livelli Data Link hanno una lunghezza minima del frame, e quindi della zona dati

Se il datagramma IP è più corto viene aggiunto padding!!

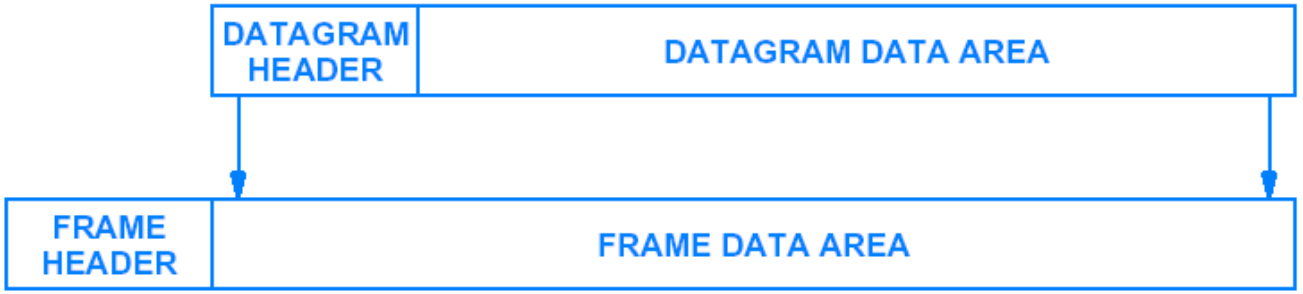


Incapsulamento del datagram

Per comprendere gli altri campi dell'header, occorre considerare la relazione che esiste fra datagram e frame fisici

Dato che i datagram non devono essere trattati dall'hardware dei mezzi di comunicazione, possono avere la lunghezza che il progettista decide. Ci sono solo **16bit** per la lunghezza, quindi a massimo sono **64KB**

Per ragioni di efficienza sarebbe meglio che un datagram viaggiasse in un solo frame fisico (non spezzato in due, se la rete fisica lo può portare): ***incapsulamento***



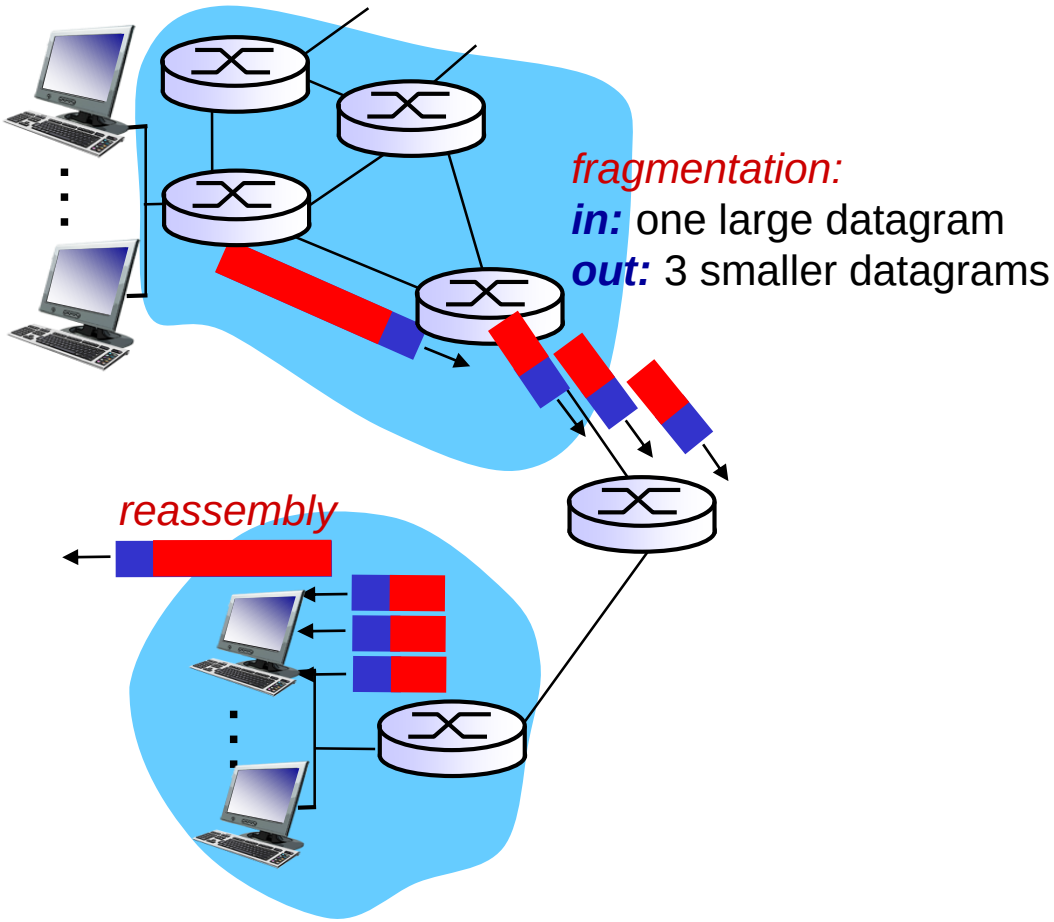
IP fragmentation, reassembly

network links have MTU (max.transfer size) - largest possible link-level frame

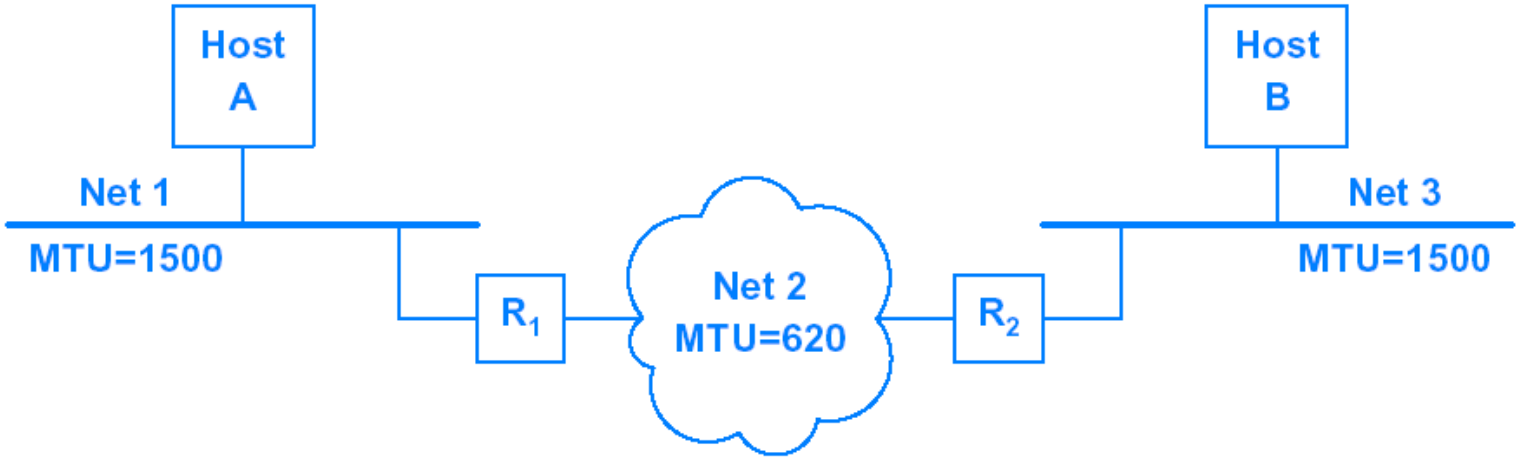
- different link types, different MTUs

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

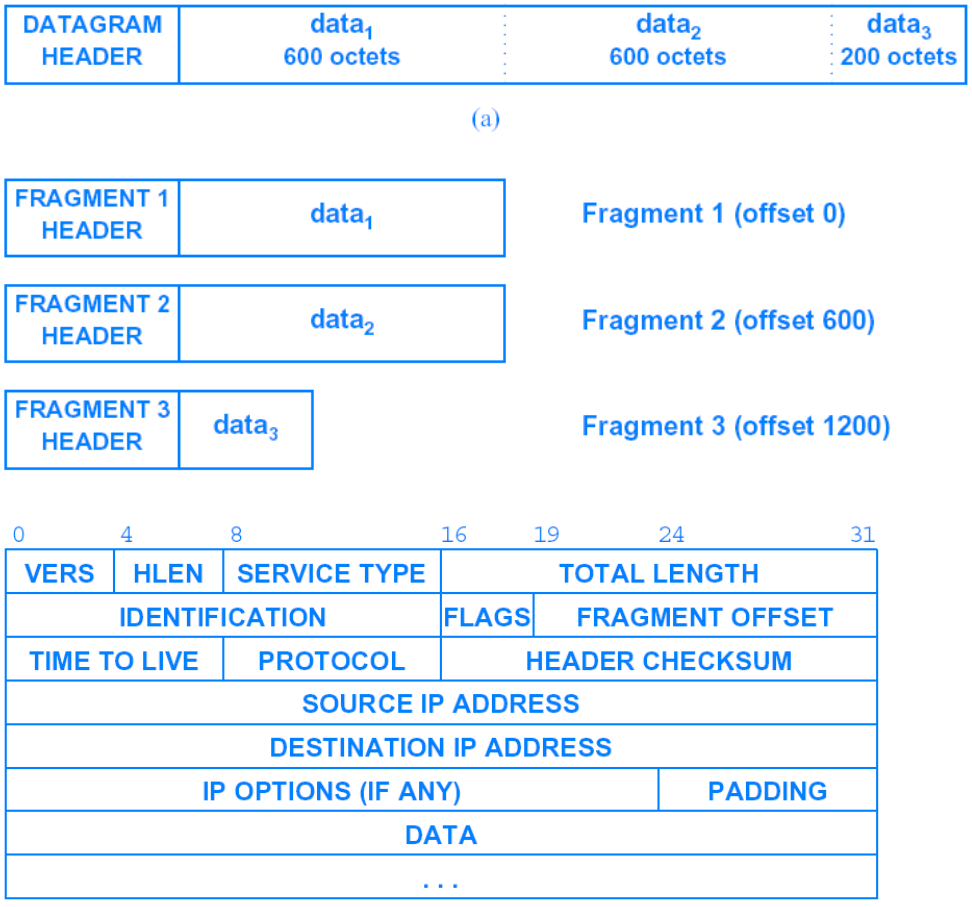


Dimensione del datagram, network MTU e frammentazione



La frammentazione avviene, quando avviene, in un qualche router fra la rete origine e quella destinazione, oppure sull'host mittente.
La dimensione del frammento è quella massima permessa dall'MTU della rete fisica, con il **vincolo che la lunghezza sia multipla di otto ottetti** (vedremo poi perché)
Ciascuno dei frammenti ha la stessa forma del datagram originale!

Dimensione del datagram, network MTU e frammentazione



Frammentazione

- Effettuata dai router
- Dividono i datagram in diversi datagram di dimensioni inferiori chiamati frammenti
- I frammenti utilizzano lo stesso formato per l'header come il datagram
- Ogni frammento viene inoltrato indipendentemente

L'offset specifica la posizione dei dati rispetto al datagram originale

Il bit MORE FRAGMENTS (=1 nei primi due =0 nel terzo)

Dimensione del datagram, network MTU e frammentazione

Gli header dei frammenti 1 e 2 hanno il bit **MoreFragments** settato per indicare che un frammento "segue" logicamente

L'offset indicato nel campo dell'header va moltiplicato per 8

Ogni frammento ha il suo header ***quasi*** uguale a quello originale:

- il valore del campo **FRAGMENT OFFSET** indica la posizione del frammento nel datagram originale: se diverso da zero è sicuramente un frammento
- il terzo bit dei **FLAGS** settato a 1 indica che segue un altro frammento, quindi il datagram è sicuramente un frammento
- il campo **TOTAL LENGTH** indica la lunghezza del frammento

Riassemblaggio dei frammenti (I)

I frammenti viaggiano separati fino all'host destinazione e qui vengono riassemblati da IP

Svantaggi:

- frammenti piccoli viaggiano in sottoreti con MTU grande
- se un frammento viene perso, il riassemblaggio non può essere fatto e quelli arrivati hanno viaggiato invano (**la frammentazione riduce l'affidabilità della rete!**)

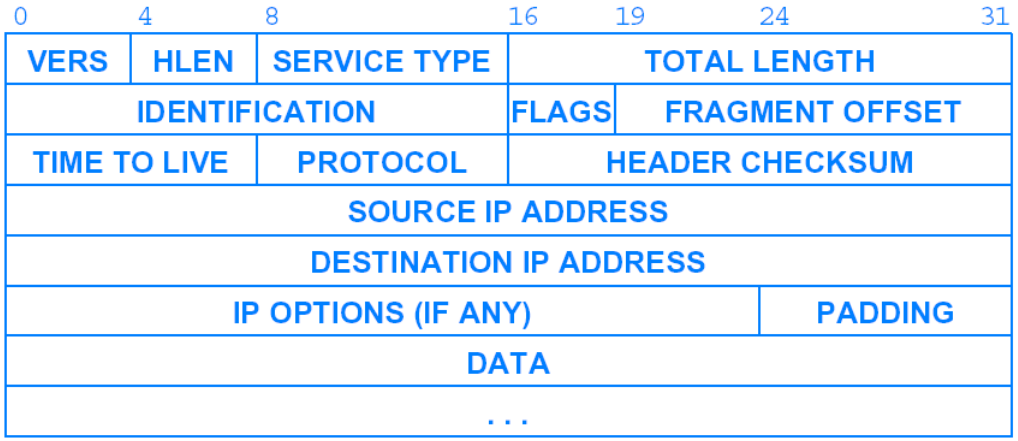
Vantaggi:

- l'operazione di riassemblaggio viene fatta una volta sola anche se avvengono numerose frammentazioni lungo la strada
- gli host partecipano al lavoro
- i router sono scaricati di un lavoro

Riassemblaggio dei frammenti (II)

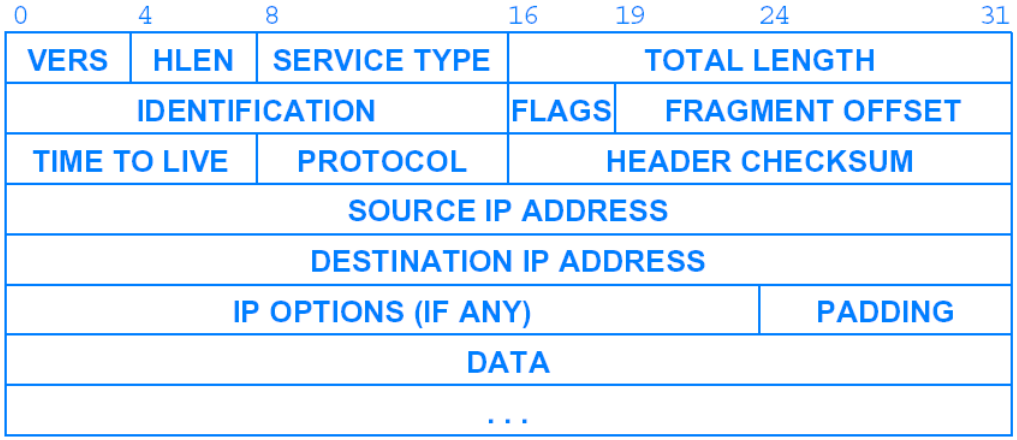
- Il riassemblaggio è basato su un reassembly timer, che il ricevente fa partire quando riceve il primo frammento. Se scade, viene scartato tutto quanto ricevuto
- Perché si **deve** mettere un timer e scartare se scade? Perché i frammenti sono associati dal fatto che hanno il campo **IDENTIFICATION** uguale. Ma questo è lungo solo 16 bit e quindi non può essere unico per lungo tempo
- Anche se ho tantissima memoria sull'host di ricezione, devo scartare i frammenti incompleti quando scatta il timeout fissato dall'RFC

Controllo della frammentazione (I)



- I campi interessati sono **IDENTIFICATION**, **FLAGS**, e **FRAGMENT OFFSET**.
- **IDENTIFICATION** = intero unico, che dovrebbe essere non ripetibile (ma è solo su 16 bit!), che identifica il datagram
- **FRAGMENT OFFSET** = misurato in unità di otto ottetti (per risparmiare spazio nell'header: ecco perché la lunghezza del frammento deve essere multipla di 8). Identifica la posizione del frammento nel datagram originale (notare che **frammentazioni in cascata continuano a indicare l'offset rispetto al datagram originale**). Solo il primo frammento (o un datagram non frammentato) ha offset a 0

Controllo della frammentazione (II)

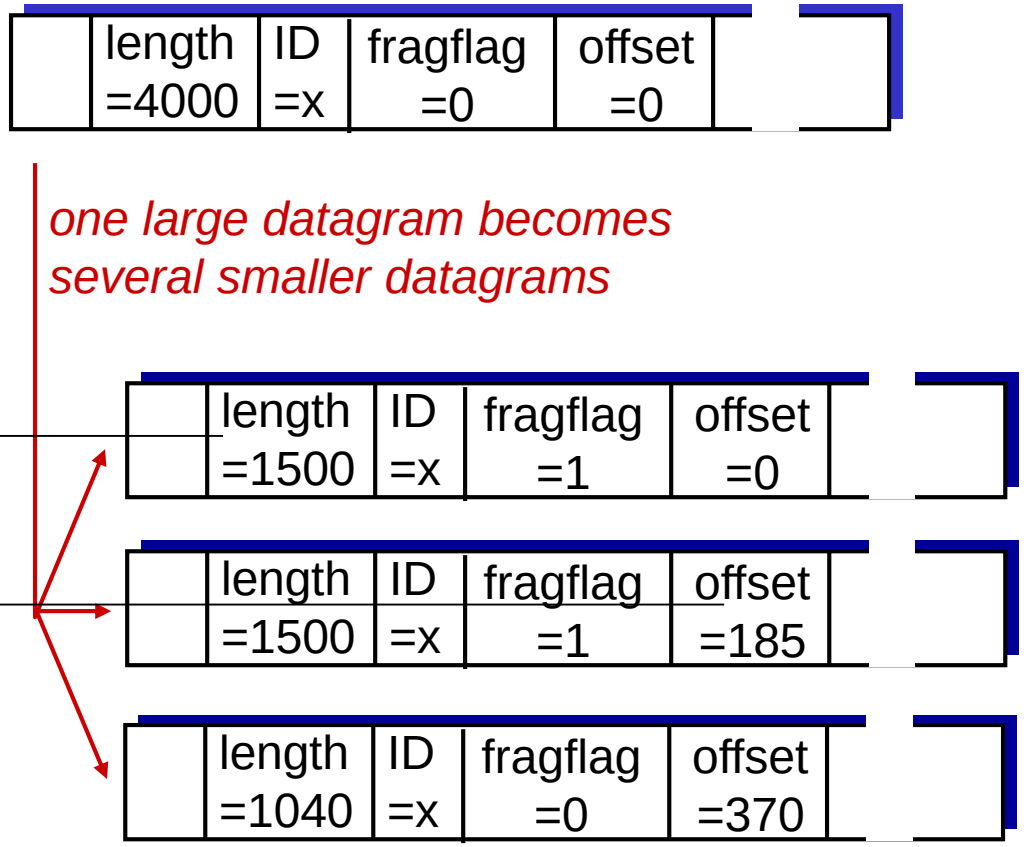


- **FLAGS** = primo bit è riservato (deve essere 0); il secondo se settato ***vieta di eseguire frammentazione*** (se necessaria allora il pacchetto viene scartato con un messaggio di errore al mittente); il terzo se settato indica che seguono ancora frammenti. Notare che è necessario perché la lunghezza del datagram originale non è nota al ricevitore
- Quando abbiamo ricevuto tutto? Quando abbiamo ricevuto l'ultimo (logico; come si identifica?) e la somma dei byte ricevuti è uguale all'offset dell'ultimo frammento + la quantità di dati dell'ultimo frammento (Perché?)

IP fragmentation, reassembly

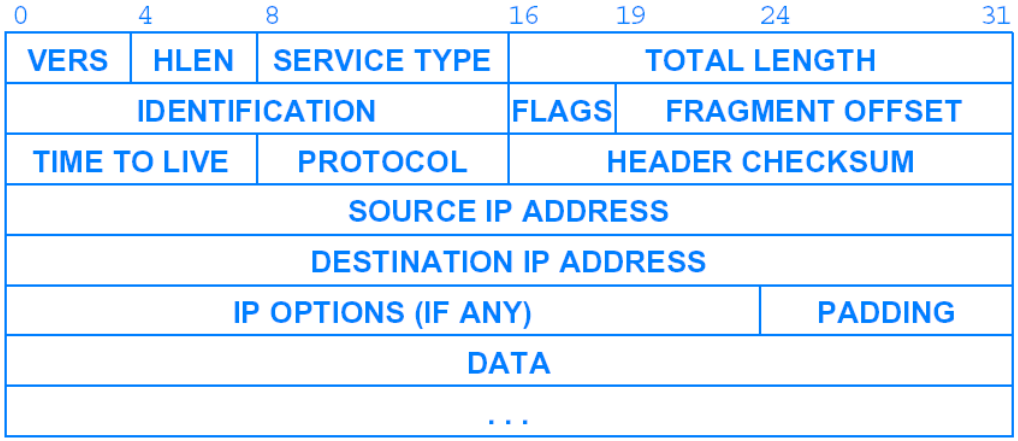
example:

- 4000 byte datagram
- MTU = 1500 bytes



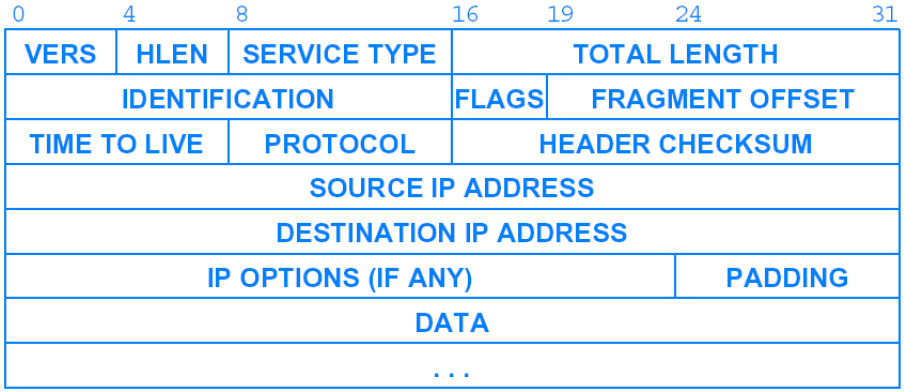
Time to Live (TTL)

Il campo **TIME TO LIVE** specifica il tempo in secondi che il datagram può restare nella rete.
Chi tratta il pacchetto deve decrementare il TTL del tempo impiegato per trattarlo (notare, anche il tempo in coda di spedizione!): quando il TTL non è più positivo il pacchetto deve essere scartato (con messaggio di errore al mittente)
Ogni router deve decrementare almeno di 1 il TTL, e poi dovrebbe considerare il tempo di attesa in coda
La cosa essenziale è che venga sempre decrementato, per evitare che instradamenti sbagliati tengano un pacchetto per sempre nella rete



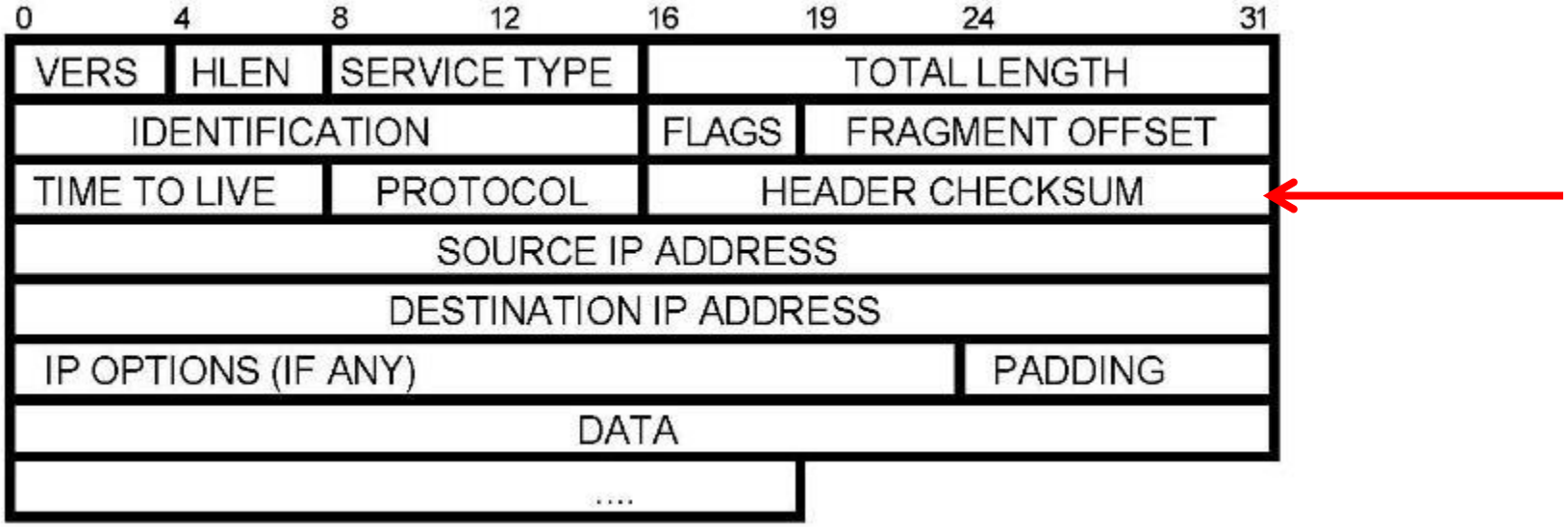
IETF raccomanda come valore iniziale di TTL 255 (al massimo)

Altri campi dell'header



PROTOCOL = analogo al campo frame type di Ethernet: vi possono essere più protocolli utenti di IP, e in questo modo l'IP ricevente sa a chi passare i dati ricevuti (ogni protocollo utente formatta i dati a modo suo!)

IP: Controllo degli errori



Il campo CHECKSUM serve per rilevare errori nella trasmissione dell'header

I dati non sono controllati perché IP non garantisce l'integrità dei dati, in quanto vuole essere un protocollo "leggero"

Dovranno essere controllati dai protocolli di livello superiore

Come funziona questo controllo? Riprendiamo il tema del controllo degli errori che era stato presentato nel capitolo sul Livello di Collegamento

IP header checksum

Obiettivo: rilevare "errori" (e.g., bit scambiati di valore) in un datagramma trasmesso, ma solo relativamente ai campi dell'header

Mittente:

L'header del datagramma viene preparato con il campo checksum, inizializzato a Zero

Tratta il contenuto dell'header come una sequenza di interi a 16 bit

Risultato ← somma cambiata di segno con l'aritmetica in complemento a 1 di tutti gli interi a 16 bit,

Il mittente mette Risultato nel campo checksum dell'header

Ricevitore:

Calcola la somma degli interi dell'header ricevuto, includendo il campo checksum

Controlla che il risultato sia Zero:

NO – rilevati errori

YES – nessun errore rilevato

Ma ciononostante ci potrebbero essere errori?

Parità, checksum e CRC

La tecnica dei bit di parità è molto semplice computazionalmente, ma protegge molto poco

La tecnica del CRC è protegge molto di più, ma richiede molta potenza di calcolo

La tecnica della checksum è più potente della parità, meno potente del CRC, richiede più potenza di calcolo della parità e meno del CRC

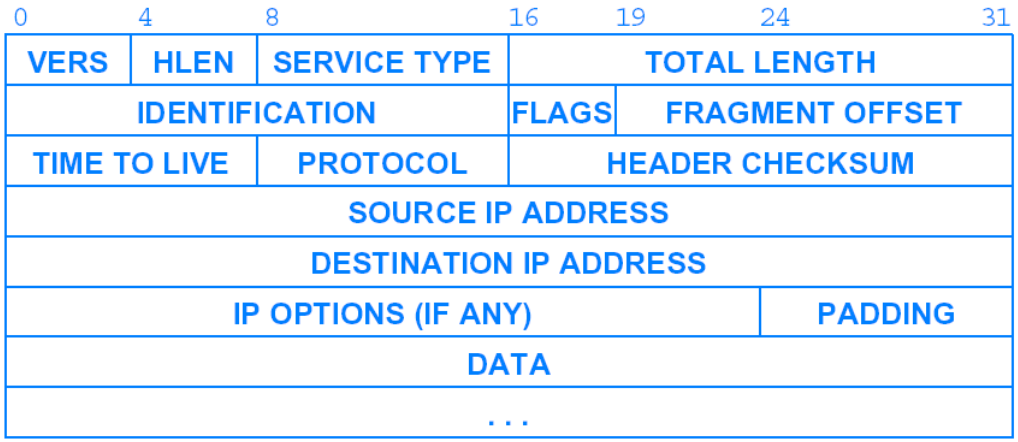
Perché si controllano gli errori sia a livello di collegamento che di IP e di Trasporto?

- A livello collegamento per effettuare subito la ritrasmissione (ed è eseguita sull'adapter/scheda di rete)
- A livello di IP e di Trasporto (anche) perché ci sono anche errori originati sugli host e nel software di comunicazione

Perché in IP e nel Trasporto si usa la checksum che protegge poco?

- Perché non si può consumare troppa CPU (IP e il trasporto sono all'interno del sistema operativo!) e il tasso di errori è basso

Altri campi dell'header



SOURCE IP ADDRESS e
DESTINATION IP ADDRESS ovvii
IP OPTIONS è l'unico a lunghezza variabile; il campo **PADDING** viene aggiunto se necessario per allinearsi a 32bit ...

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- **IPv4 addressing**
- forwarding
- DHCP
- network address translation
- error messages (ICMP)
- IPv6

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

Indirizzi Internet Basati su Classe (vintage)

- L'indirizzamento è una funzionalità essenziale di qualunque sistema di comunicazione, tanto più per un sistema che vuole essere ***universale***, cioè in grado di connettere due qualunque calcolatori al mondo
- In TCP/IP gli indirizzi sono **numeri binari**, compatti, facili da trasmettere, e a partire dai quali il software di rete può calcolare una route, un instradamento
- Perché occorre definire un indirizzo internet e non si possono usare gli indirizzi delle reti fisiche? Perché
  - Nelle reti fisiche gli indirizzi sono di tipo diverso
  - Alcune reti fisiche non hanno indirizzi universali
- Poiché internet è una struttura logica e non fisica, la forma degli indirizzi può essere scelta liberamente
- Ad ogni macchina in una internet TCP/IP viene assegnato un (univoco) indirizzo internet di 32 bit, che viene usato in ogni comunicazione con quella macchina

L'indirizzamento originale basato su classi (II)

- Un indirizzo deve individuare univocamente una macchina (**non ci devono essere due macchine con lo stesso indirizzo**)
- Garantita l'univocità, non ci sono problemi se una macchina possiede più di un indirizzo IP, purché questi indirizzi siano tutti univoci, cioè nessuna altra macchina abbia un indirizzo uguale
  Però abbiamo bisogno che i router basino l'instradamento sulla sola conoscenza della sotto-rete destinazione: allora dobbiamo dare una struttura all'indirizzo!
- **un indirizzo IP è una coppia (netid, hostid)**, dove
  - netid = identifica la rete fisica a cui appartiene la macchina = *indirizzo di rete*
  - hostid = identifica la macchina sulla rete identificata da netid = indirizzo di host relativo alla rete netid

L'indirizzamento originale basato su classi (III)

- In questo modo si "vede" subito quale è la sottorete destinazione e l'instradamento può essere efficiente: ridurre la quantità di informazione da considerare rende più semplice e veloce il processo di decisione
- Il netid deve anche essere **univoco**, cioè non ci possono essere due reti con lo stesso netid. Ad ogni netid corrisponde una unica rete fisica. Anche qui, nulla vieta che ad una rete fisica siano assegnati più di un netid.
- **Il netid è l'indirizzo IP della rete fisica**
  - I valori del netid devono essere amministrati, cioè ci deve essere una autorità centrale che concede l'uso di questi netid, assicurandone così l'univocità.
  - Questa autorità centrale si è organizzata in autorità locali.
  - Oggi nessuno può più ricevere in uso esclusivo un netid: Sono gli ISP (Internet Service Provider) che "affittano" ai loro "clienti" gli indirizzi IP che questi richiedono

L'indirizzamento originale basato su classi (IV)

Proprietà che uno schema di indirizzamento dovrebbe avere

- Indirizzi compatti (dimensioni contenute)
- Meccanismo unversale
- Funziona con tutte le tecnologie di rete
- Permettere di implementare efficientemente
  - Test per controllare se la destinazione può essere raggiunta direttamente
  - Decidere quale router utilizzare per inoltrare i pacchetti
  - Scegliere il prossimo router lungo il cammino verso la destinazione

L'indirizzamento originale basato su classi (V)

- Esigenza di partenza: identificare una macchina con un identificatore univoco, detto **indirizzo** (una macchina **un solo indirizzo)**
- Decisione sulla strada da seguire per raggiungere la destinazione basata solo sulla conoscenza della rete destinazione (non sulla conoscenza di tutti i singoli host che ci sono sull'internet)
- Ridurre l'informazione da esaminare per l'instradamento
- → l'indirizzo dell'host deve contenere al suo interno l'indirizzo della rete fisica su cui l'host si trova: indirizzo diviso in due parti!
- Ma se l'indirizzo dell'host codifica anche l'indirizzo della rete su cui l'host si trova, l'host deve avere tanti indirizzi quante sono le sotto-reti a cui è connesso, quindi: **l'indirizzo dell'host è in realtà l'indirizzo della scheda di rete che esso usa per connettersi all'internet**
- La conoscenza immediata della rete su cui si trova un host **non aiuta solo i router** ma **aiuta anche gli host nella consegna diretta**

L'indirizzamento originale basato su classi (VI)

Quando un host deve consegnare un pacchetto, si deve per prima cosa domandare: la destinazione è sulla mia stessa rete o su un'altra rete fisica?

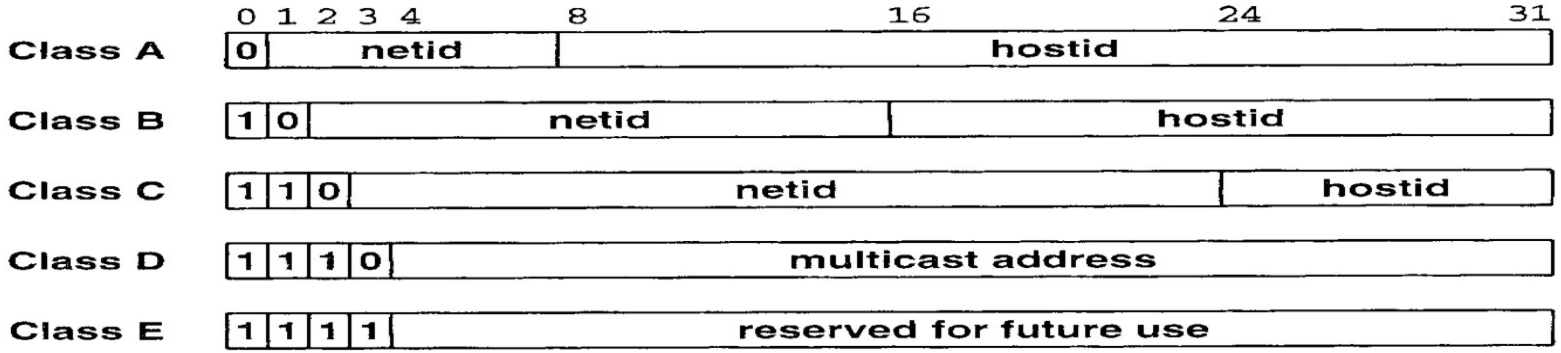
- Calcolare velocemente questo test aiuta anche gli host!
- Senza il netid l'host dovrebbe tenere la lista degli altri host sulla stessa rete -> grandi problemi di aggiornamento di tale lista su tutti gli host della sotto-rete

Come dividere in suffisso e prefisso (dell'indirizzo)

- Un prefisso molto grande (suffisso piccolo) consente di indirizzare molte reti ma ognuna con pochi host
- Con un suffisso grande si hanno reti con molti host ma ci possono essere poche di queste reti
- Indirizzamento basato su classi (proposta originale)

L'indirizzamento originale basato su classi (VII)

- Se ci sono due parti nell'indirizzo che hanno un significato diverso e devono essere usate in modo differente, **occorre distinguerle**, e poter capire quale è il confine fra le due parti: di quanti bit è composto il netid e quindi di quanti l'hostid
- Il numero di bit di cui sono fatte le due parti è definito dai primi bit dell'indirizzo, secondo lo schema seguente



L'indirizzamento originale basato su classi (VIII)

Schema auto-identificante (questo non è più vero per gli indirizzi attuali)

- In pratica, gli indirizzi che si usano sono per la maggior parte delle prime tre forme.
- I primi bit (uno, due o tre) determinano la **classe** a cui appartiene l'indirizzo
- Vi sono quindi reti con diversa lunghezza di netid e di hostid
- Vi sono poche reti con moltissime macchine (A), tantissime con poche macchine (C) e molte reti con molte macchine (B)

	Numero di reti	Numero di macchine
Classe A	126	16.777.214
Classe B	16.384	65.534
Classe C	2.097.152	254

Notate che il numero di bit di *netid* è variabile all'interno di tre varianti soltanto: è uno schema intermedio fra lunghezza variabile e lunghezza fissa

Gli indirizzi specificano le connessioni alla rete (I)

- Quindi non è più vero che ogni macchina può avere un solo indirizzo IP
- I router sono attaccati a più di una sottorete; ma ogni sottorete deve avere il suo netid (diverso), **quindi i router devono avere più di un indirizzo IP**: uno per ogni sottorete a cui sono attaccati.
- Anche le macchine possono avere più di una interfaccia di rete, e allora sono detti ***multihomed***. Vedremo dopo la differenza fra router e macchine multihomed.
- ***Poiché gli indirizzi IP codificano sia una sottorete che una macchina su quella sottorete, essi in realtà specificano non una macchina, ma una connessione/interfaccia ad una sottorete***
- **E questo è uno svantaggio dovuto alla introduzione del netid!**

Gli indirizzi specificano le connessioni alla rete (II)

Ma le cose sono ancora più complesse:

- Ci possono essere più reti logiche diverse sulla stessa rete fisica
- Si possono dare indirizzi multipli ad una stessa interfaccia
  - con stesso netid, quindi sulla stessa rete logica
  - con netid diversi, quindi su reti logiche diverse

Conclusione: Gli host possono avere più indirizzi perché:

- hanno più interfacce, oppure
- hanno più indirizzi sulla stessa interfaccia

Su una rete fisica può essere appoggiata una sola rete logica (un solo netid), oppure più reti logiche, ognuna con il suo netid.

Indirizzi di rete e di broadcast diretto (I)

- Avendo l'identificazione della rete negli indirizzi, si può anche esprimere l'indirizzo di una rete:
  per convenzione, l'hostid=0 non è un indirizzo di host legale come indirizzo destinazione
  ⇒ un indirizzo IP con hostid=0 indica l'intera sottorete.
  Per convenzione, gli indirizzi IP con hostid costituito di tutti i bit a zero sono indirizzi di sottorete
- Analogamente, per convenzione si può esprimere un ***indirizzo di broadcast diretto***, cioè che si riferisce a tutte la macchine di una sottorete. È quell'indirizzo con hostid costituito da tutti *1*

Indirizzi di rete e di broadcast diretto (II)

- Questo indirizzo di broadcast è detto ***indirizzo di broadcast diretto*** perché contiene un indirizzo di sottorete valido e un hostid di broadcast
- Può essere interpretato correttamente in ogni punto dell'internet e instradato, ma per esprimerlo è necessaria la conoscenza dell'indirizzo di sottorete a cui si vuole spedire
- In molte tecnologie di sottorete il broadcast è supportato dal protocollo di trasmissione (ad es. ethernet)
  Gli indirizzi IP possono essere usati per specificare broadcast, e si mappano sul broadcast hardware se esiste. Se la sottorete fisica non ha il broadcast, IP scarta il pacchetto: IP non supporta indirizzi di broadcast su sottoreti senza broadcast fisico

IP non supporta il broadcast a livello di netid!!!!

Broadcast limitato

- È' un indirizzo fatto di 32 bit a uno: significa broadcast sulla sottorete in cui appare. Detto ***indirizzo di broadcast limitato*** o ***indirizzo di broadcast della rete locale***
- L'indirizzo di broadcast limitato non può uscire dalla rete su cui viene inviato, ma non richiede la conoscenza dell'indirizzo di rete: una macchina può inviare un pacchetto indirizzato al broadcast limitato anche senza conoscere l'indirizzo della rete su cui si trova, e quindi neanche il proprio indirizzo.
- Può essere usato quando, all'avviamento della macchina, non è ancora noto l'indirizzo di rete locale: il broadcast limitato serve proprio per poter chiedere il proprio indirizzo IP a server speciali
- Non è supportato su sottoreti fisiche che non hanno broadcast. Per questo PPP ha il broadcast

Usare lo Zero per indicare "Questo”

- Un campo con **tutti Uno** indica "**tutti**". Un campo di **tutti Zero** indica "**questo**".
  - **hostid = 0 indica "questo host” (usato come indirizzo mittente)**
  - **netid = 0 indica "questa rete”**
- Questa convenzione dipende enormemente dal contesto. Se una macchina non conosce il proprio netid, potrebbe inviare pacchetti in cui il proprio hostid è valorizzato, mentre il netid è tutto a Zero.
- Le altre macchine intendono che il sorgente sia sulla loro stessa rete, e il router non inoltra tale pacchetto su altre reti ...

  ... ma sono esempi fittizi

Vantaggi dell'indirizzamento IP

1. **Permette un mixing di reti di grosse/medie/piccole dimensioni**
2. **Lunghezza fissa rende più semplice l'elaborazione**
   - I primi bits specificano le dimensioni del prefix e del suffisso
   - Efficienti operazioni per estrarre netid/hostid
3. **Netid facilita l'instradamento**
   - Minore complessità in memoria e in tempo. Vantaggio soprattutto per router
4. **Netid facilita la consegna diretta**
   - Minore complessità in memoria e in tempo del test di consegna diretta
   - Elimina compiti di gestione di tabelle statiche di host sulla stessa rete
5. **Netid facilita l'amministrazione degli indirizzi**
   - Occorre amministrare centralmente solo i netid: gli hostid vengono assegnati e gestiti dall'amministratore locale

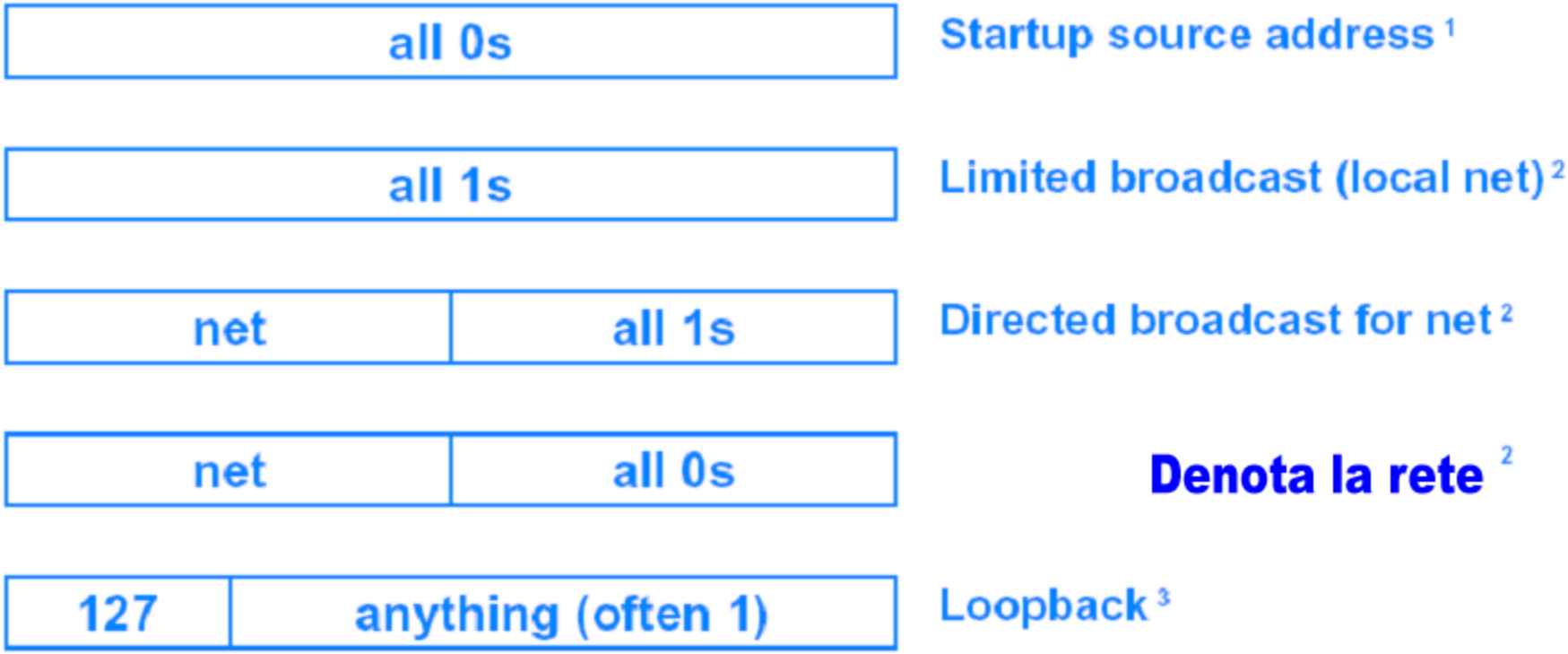
Notazione dotted quad

- Una sequenza di 32 bit è scomoda da ricordare e da scrivere
- Rappresentiamo allora in modo conveniente ciascuno dei quattro byte che costituisce l'indirizzo IP
- Interpretiamo ogni byte dell'indirizzo IP come un numero senza segno scritto in binario in base 10
- Quindi un indirizzo IP si può rappresentare con quattro decimali, e i quattro decimali vengono separati da un "." per non confondere le cifre di uno con quelle dell'altro
  - Esempio:
    10000000 00001010 00000010 00011110
    Ogni byte vale (in decimale)
    128 10 2 30
    Quindi lo scriviamo come:
    128.10.2.30

Indirizzo di loopback

- L'indirizzo 127.0.0.1 è riservato per il ***loopback***, cioè per indicare che la destinazione è la macchina stessa, quindi **il pacchetto non viene nemmeno spedito sulla rete**
- Viene usato per testing degli strati superiori di protocollo (perché il loopback è interpretato dalla parte bassa di IP), e per interprocess communication all'interno della macchina: le applicazioni si parlano come se fossero su macchine diverse
- Che differenza c'è rispetto a usare come destinazione un indirizzo della macchina stessa?
  - Non c'è bisogno di conoscere un indirizzo della macchina
  - Usando un indirizzo della macchina, si spedisce davvero il messaggio nella rete:
    - la rete deve essere in grado di "riflettere" il pacchetto sulla stessa interfaccia: in questi casi serve per verificare il corretto funzionamento della rete
    - oppure ci deve pensare l'interfaccia di rete: in questi casi serve per verificare il funzionamento della interfaccia di rete
- Su certi sistemi operativi funziona solo 127.0.0.1, su altri qualunque 127.x.y.z, su altri tutti però non 127.0.0.0

Convenzioni su indirizzi speciali



Notes:
[1] Never a valid destination
[2] Never a valid source
[3] Should not appear on a network

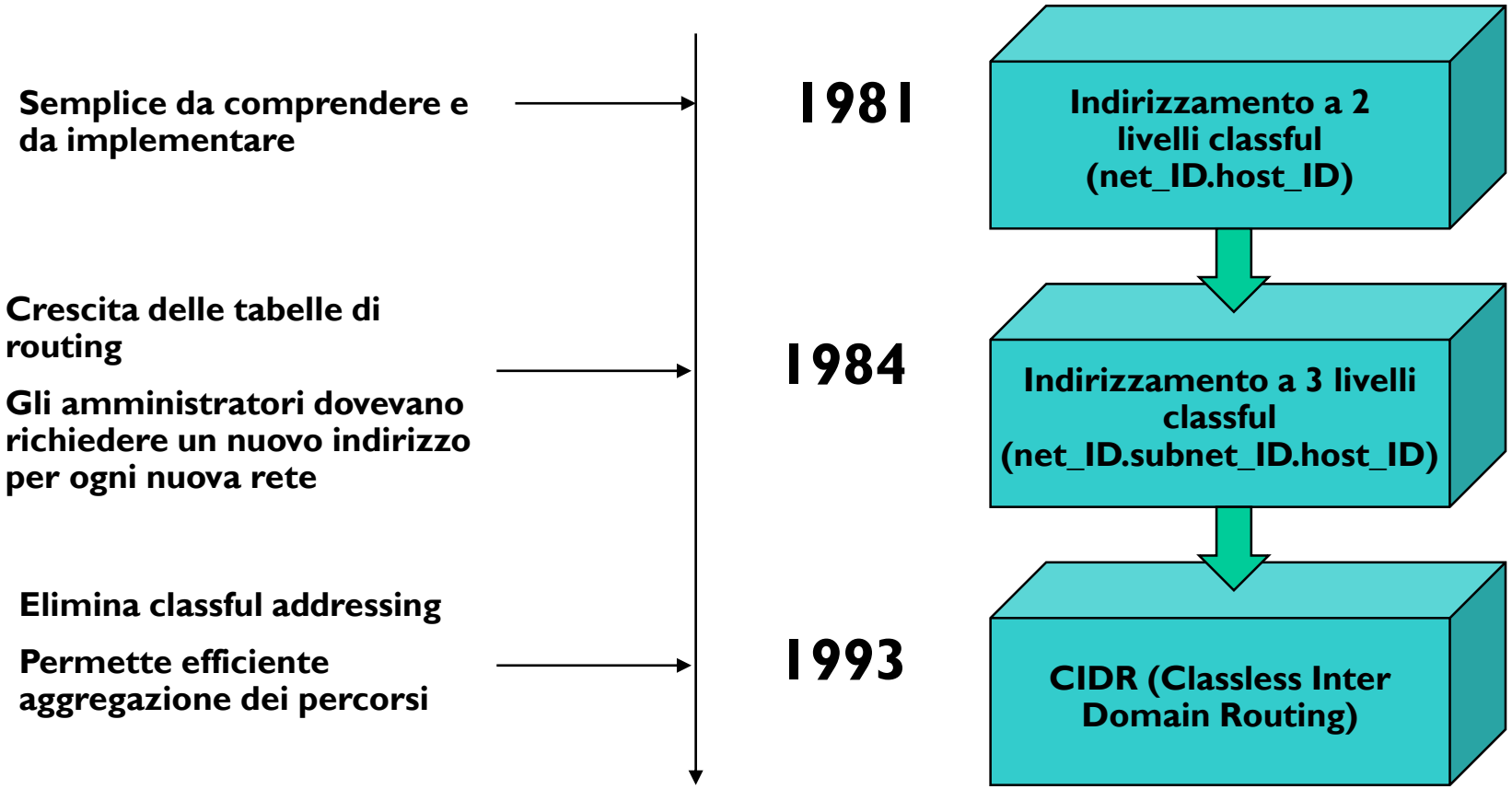
Network byte order

- Realizzare una internet con tecnologie diverse di sottorete e connettere macchine basate su processori diversi, con diverse rappresentazioni dei dati
- Definire una rappresentazione standard dei dati scambiati fra le macchine. Come vengono scambiati gli interi di 32 bit? Ognuno li può rappresentare in forma diversa
- ***Little Endian***: sono macchine nelle quali l'indirizzo di memoria più piccolo contiene il byte meno significativo dell'intero
- ***Big Endian***: sono macchine nelle quali l'indirizzo di memoria più piccolo contiene il byte più significativo dell'intero
- Indirizzi IP e anche interi: **il network standard byte order è essenziale già a livello di IP**

Estensioni di Subnet e Supernet

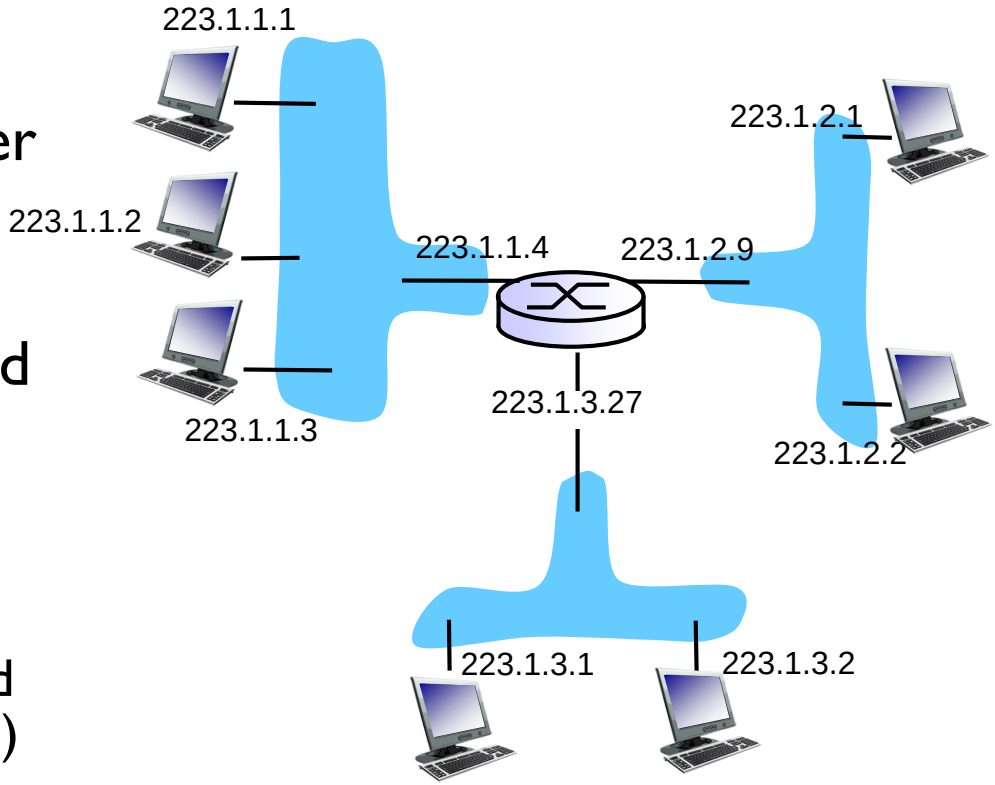
- Un netid deve corrispondere ad una sola rete fisica: in questo modo si consumano molti indirizzi di rete (netid) perché le reti fisiche non possono essere "piene" al massimo di host
- Lo sviluppo delle reti locali ha reso questo fenomeno devastante
- Nel 1990 sono state definite estensioni al meccanismo di indirizzamento che permettono che il confine fra netid e hostid sia posizionato in un punto arbitrario dei 32bit.
- Si è così definito l'indirizzamento senza classi (*classless addressing*) o supernetting.
- Allo stesso modo diverse reti fisiche possono condividere lo stesso prefisso di classe A, B o C, permettendo così di utilizzare meglio lo spazio di indirizzi. Vedremo…

Evoluzione dell'indirizzamento



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 or two interfaces (e.g., wired Ethernet, wireless 802.11)
- *IP addresses associated with each interface*



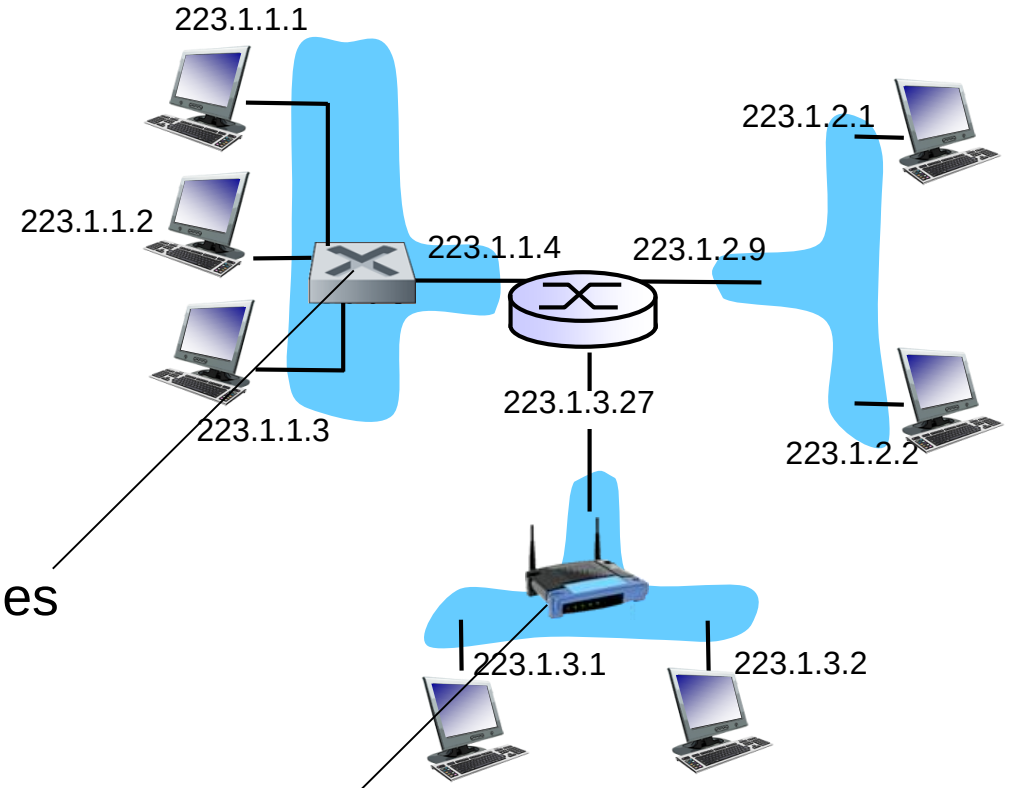
223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

IP addressing: introduction

Q: how are interfaces actually connected?

A: we'll learn about that in chapter 5, 6.



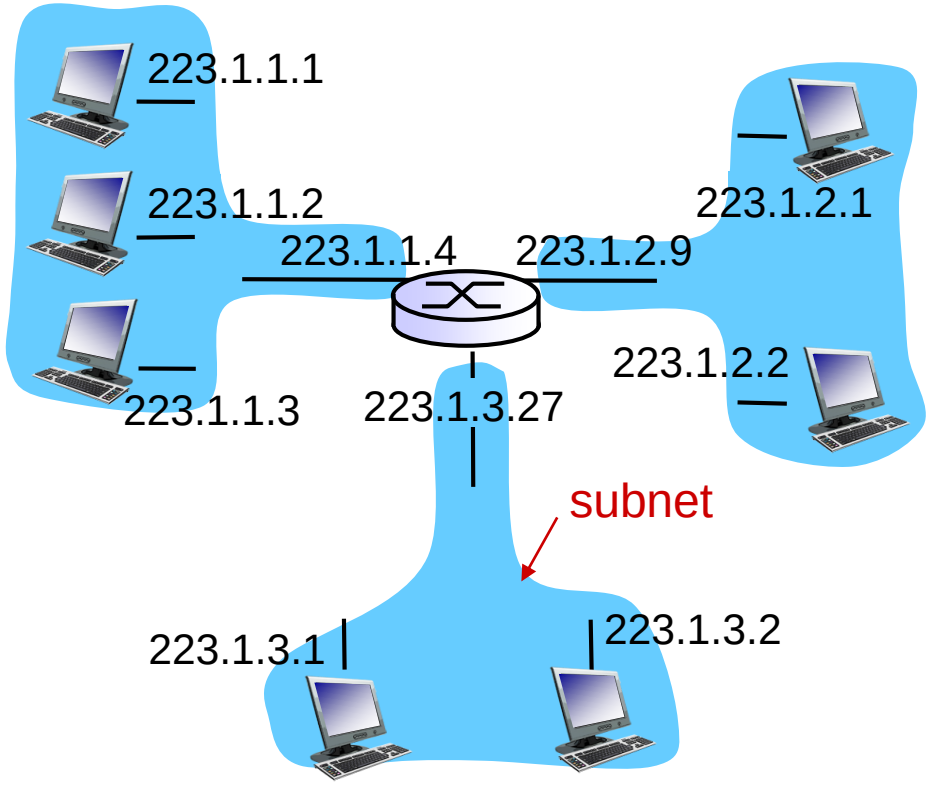
A: wired Ethernet interfaces connected by Ethernet switches

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

A: wireless WiFi interfaces connected by WiFi base station

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*

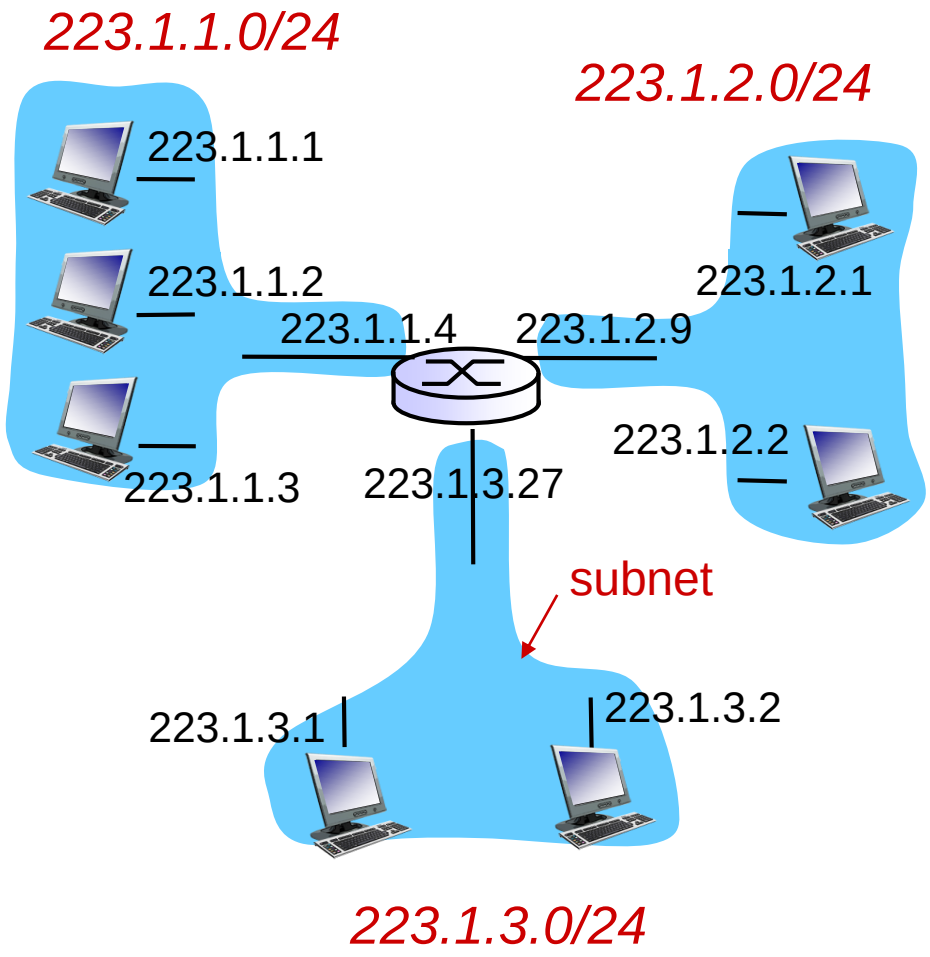


network consisting of 3 subnets

Subnets

recipe

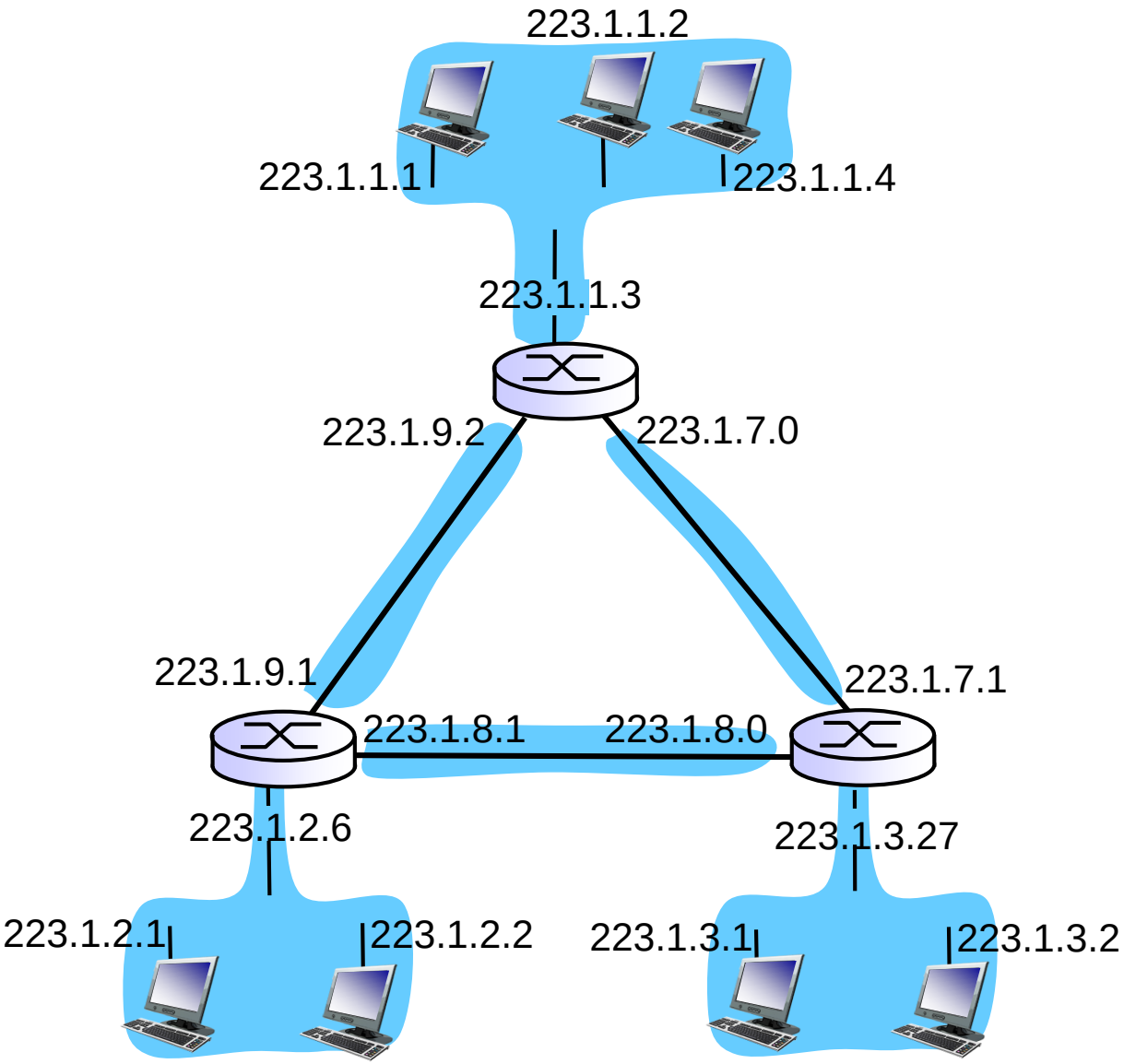
- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- each isolated network is called a *subnet*



subnet mask: /24

Subnets

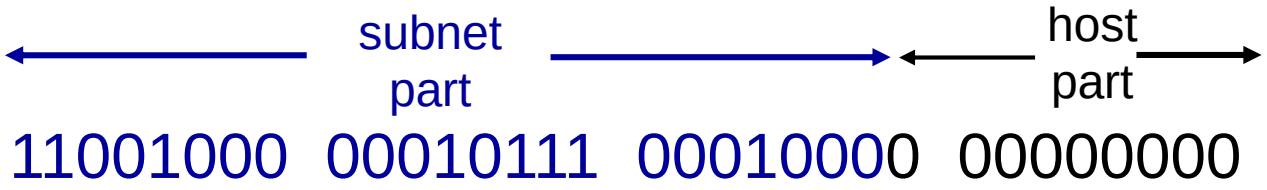
how many?



IP addressing: CIDR

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



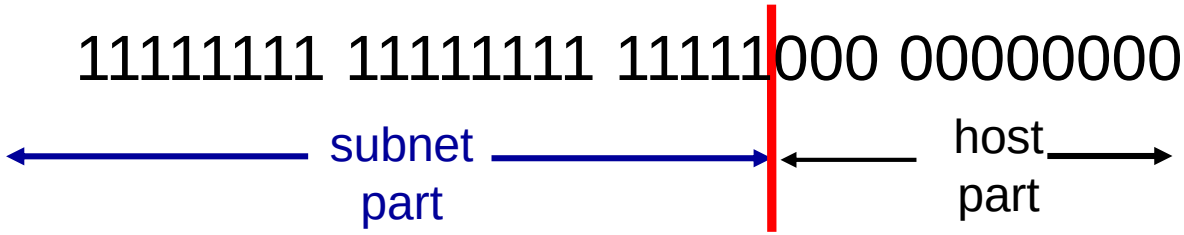
200.23.16.0/23

IP addressing: CIDR

Esempio: ISP a cui viene assegnato un blocco di 2048 indirizzi contigui, iniziando da 128.211.168.0

	Dotted Decimal	32-bit Binary Equivalent
lowest	128.211.168.0	10000000 11010011 10101000 00000000
highest	128.211.175.255	10000000 11010011 10101111 11111111

La maschera di indirizzi CIDR ha 21 bit uguali ad 1 (la divisione tra suffisso e prefisso avviene dopo il ventunesimo bit)



L'operatore potrebbe decidere di dividerlo in blocchi da 256 indirizzi

11111111 11111111 11111000 xxxxxxxx
11111111 11111111 11111001 xxxxxxxx
11111111 11111111 11111010 xxxxxxxx
- - - - - - - - - - - -
11111111 11111111 11111111 xxxxxxxx

8 blocchi da 256 indirizzi ciascuno

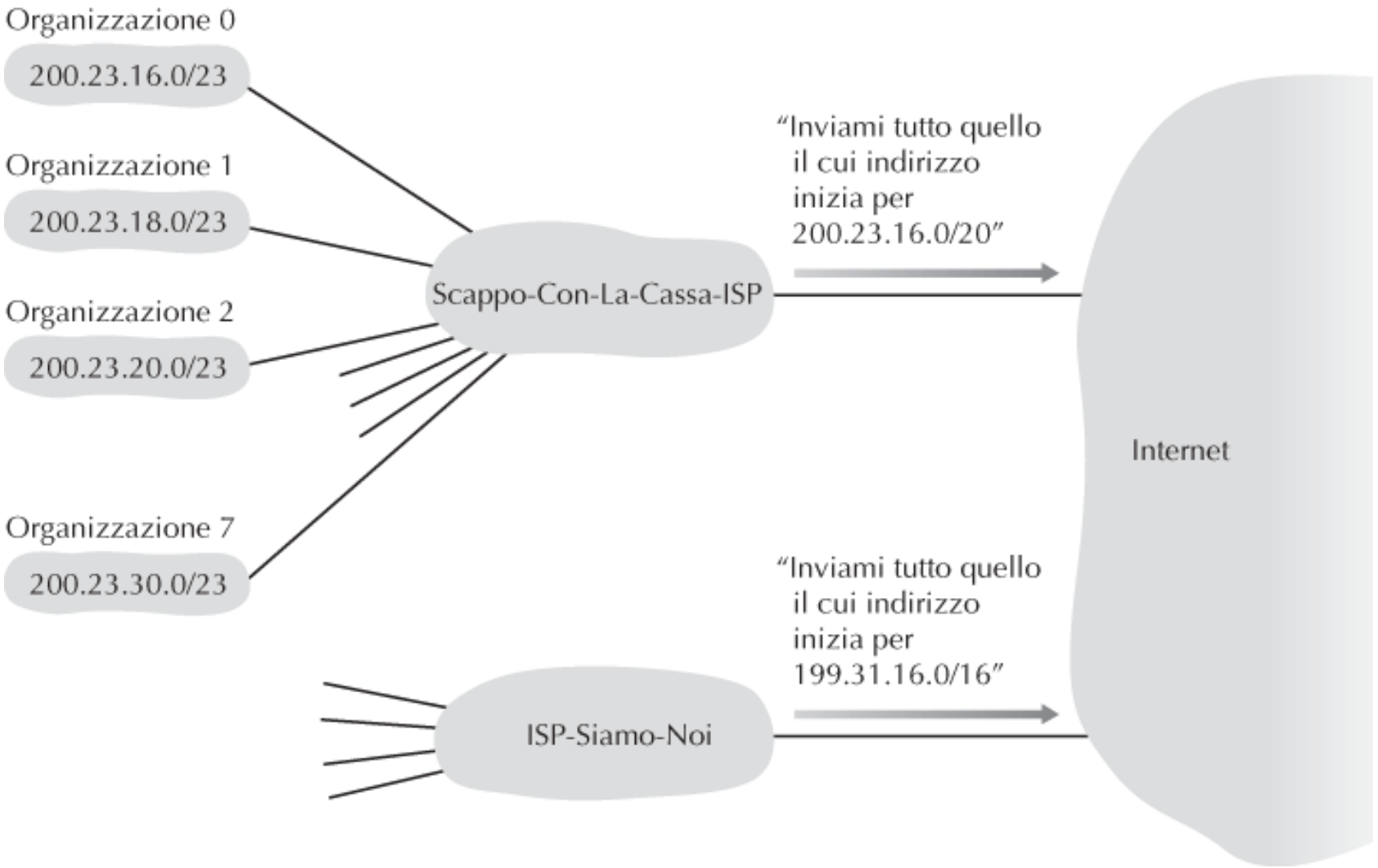
IP addresses: how to get one?

- Ottenere un blocco di indirizzi IP da usare in una sottorete un amministratore di rete deve contattare il proprio ISP
- L'ISP fornisce il blocco richiesto prelevandolo dal blocco (più grande) che gli è stato allocato (vedere esempio precedente)
- L'ISP ottiene il suo blocco da un'autorità globale

ISP's block	11001000 00010111 00010000	00000000	200.23.16.0/20
Organization 0	11001000 00010111 0001000**0**	00000000	200.23.16.0/23
Organization 1	11001000 00010111 0001001**0**	00000000	200.23.18.0/23
Organization 2	11001000 00010111 0001010**0**	00000000	200.23.20.0/23
...	.....	....	....
Organization 7	11001000 00010111 0001111**0**	00000000	200.23.30.0/23

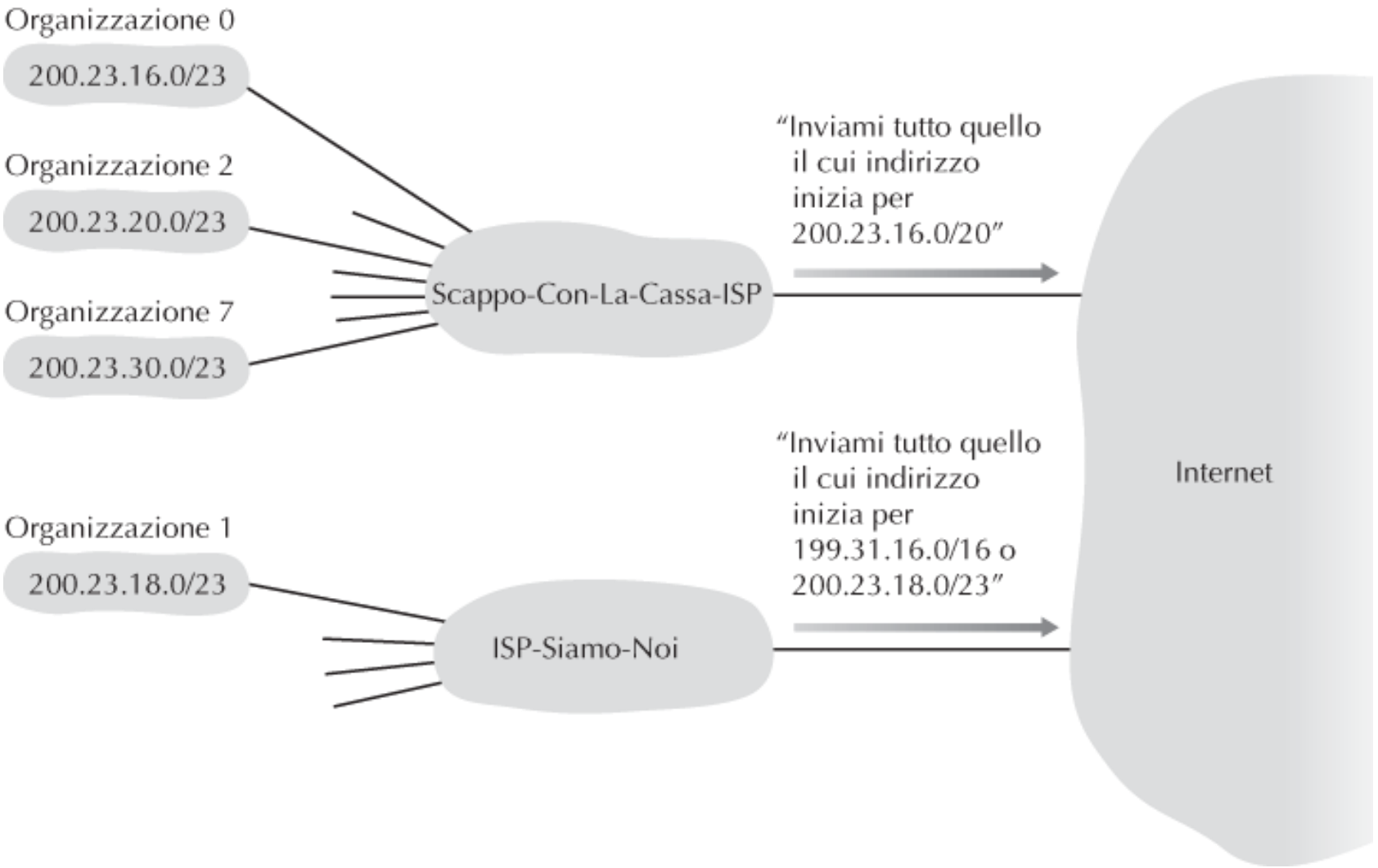
IP addresses: how to get one?

hierarchical addressing allows efficient advertisement of routing information:



IP addresses: more specific routes

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



IP addressing: the last word...

Q: how does an ISP get block of addresses?

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

- allocates addresses
- manages DNS
- assigns domain names, resolves disputes

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- **forwarding**
- DHCP
- network address translation
- error messages (ICMP)
- IPv6

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

IP: Instradamento dei datagrammi IP

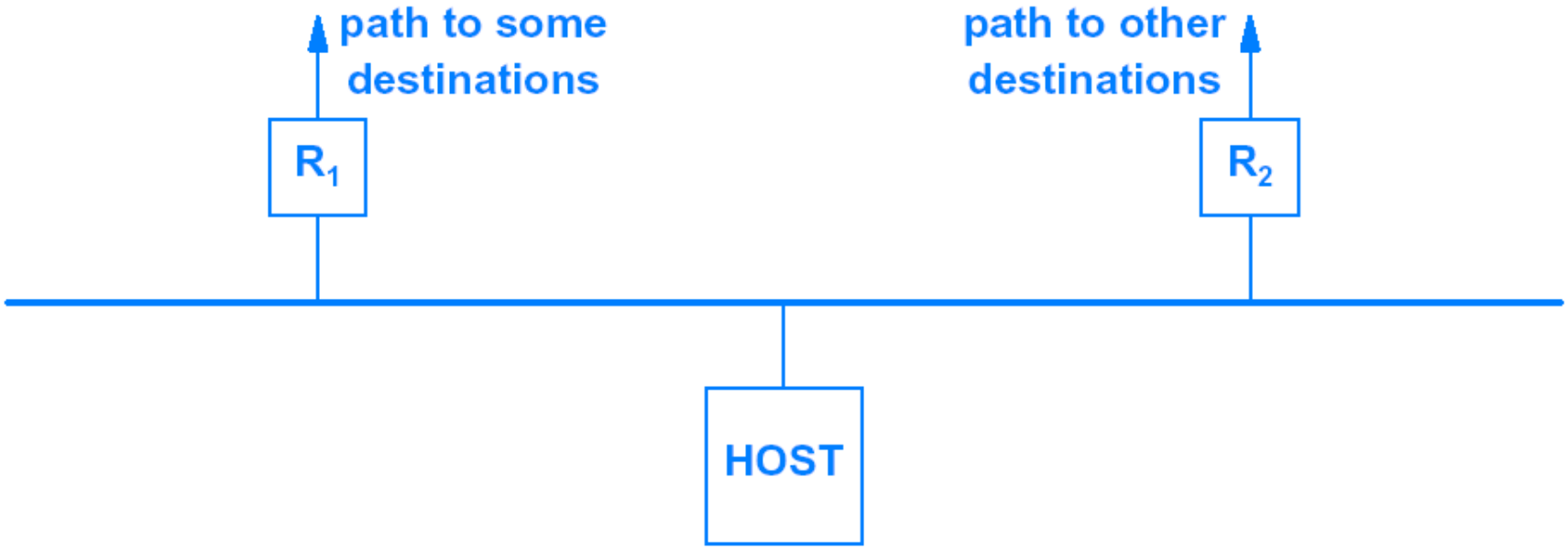
- L'instradamento è la terza funzione fondamentale di un servizio di comunicazione di livello rete
- Il formato del datagramma descrive gli aspetti statici di IP: l'instradamento riguarda gli aspetti **dinamici**
- Gli host inoltrano (consegna diretta) ad altri host connessi alla stessa rete (fisica)
- Gli host inviano al router i datagram che non possono consegnare direttamente
- I router inoltrano datagram verso altri router
- Il router finale consegna direttamente il datagram al host destinatario (consegna diretta)

L'instradamento in una internet (I)

- ***Instradamento (forwarding)***
- ***Router (gateway IP)*** = una macchina situata lungo il cammino, che prende questa scelta (non confondere con host multihomed)
- Alcune sottoreti (per esempio quelle geografiche) possono aver bisogno al loro interno di una funzione analoga
- Difficoltà:
  - nel prendere la decisione occorrerebbe considerare
  - macchine con più di una interfaccia (perché vorrei raggiungere la macchina, attraverso una qualunque delle sue interfacce)
  - carico della rete
  - diversità della lunghezza dei datagrammi
  - tipo di servizio richiesto dai datagrammi (→ TOS/QOS)
  - In genere il software di IP è molto semplice e segue delle semplici regole di instradamento configurate sulla macchina che cercano di seguire il cammino più corto

L'instradamento in una internet (II)

Anche le macchine che non sono router partecipano, in modo semplificato, al processo di instradamento



Consegna diretta e indiretta

- ***Consegna diretta*** = consegna di un datagramma ***direttamente*** da una macchina (host o router) ad un'altra (host o router): quindi sono ***connesse ad una stessa sottorete***
- ***Consegna indiretta*** = consegna di un datagramma da una macchina ad un'altra che ***non*** è ***connessa da una sottorete comune alle due***. Quindi occorre passare attraverso almeno un router

Consegna diretta (attraverso una singola sottorete comune)

- *NON coinvolge router. L'host mittente incapsula il datagramma in una trama fisica, lega l'indirizzo IP all'indirizzo hardware del destinatario, invia la trama direttamente all'host destinazione*
- **Come si scopre che si è sulla stessa rete fisica? Dal prefisso di rete (netid) dell'indirizzo della destinazione, che è uguale al prefisso dell'indirizzo di una delle sottoreti a cui il mittente è connesso. Notare che:**
  - Tutte le interfacce su una sottorete logica IP hanno lo stesso prefisso di rete (netid), e due sottoreti fisiche distinte hanno prefissi di rete diversi
  - Se su una stessa sottorete fisica ci sono più di una sottorete logica, **solo le interfacce che appartengono alla stessa sottorete logica si parlano direttamente**, anche se dal punto di vista fisico potrebbero parlare anche con le altre. Il problema è che … non lo sanno!

Il netid aiuta anche gli host, non solo i router!

Aspetto importante

- La trasmissione di un datagram IP tra due host su una singola rete fisica non coinvolge i router. Il mittente incapsula il datagram in un frame fisico, traduce l'indirizzo IP di destinazione in un indirizzo hardware fisico e invia il frame risultante direttamente alla destinazione

Test per decidere se un host destinatario è sulla stessa rete fisica del host mittente

- Poichè gli indirizzi di internet di tutti i computer posti su una singola rete hanno un prefisso comune di rete e per estrarlo bastano solo poche istruzioni macchina, verificare se un computer può essere raggiunto direttamente è estremamente efficiente

Indirizzi di sottorete: consegna diretta

- Ad ogni sottorete viene assegnato un range di indirizzi definito da un prefisso. Ad es. 200.23.16.0/23 -> indirizzo di sottorete
- Nota bene: non ci possono essere "buchi" all'interno di questo intervallo!
- La prima cosa che IP (sia di router che di host) cerca di fare è la **consegna diretta**:
  - L'indirizzo destinazione è contenuto all'interno del range di una delle mie interfacce?
  - Si -> traduco IP address in MAC address su quella interfaccia, e spedisco con l'interfaccia
  - No -> **consegna indiretta** tramite **tabella di inoltro**

Indirizzi di sottorete: consegna indiretta

Se la consegna diretta fallisce, cioè l'indirizzo IP destinazione non appartiene a nessuna delle sottoreti a cui la macchina è direttamente connessa?

Consegna indiretta

Ricerco l'indirizzo destinazione dentro la tabella di inoltro:

- **Trovo un router associato all'indirizzo**
  - Invio il datagramma a questo router
- Non trovo alcun router associato
  - Segnalo errore di Destinazione Sconosciuta

Come si regola l'host mittente?
Chi gli fornisce la conoscenza del router di default?

Consegna indiretta (I)

- Il mittente deve **scegliere** (**instradamento**) un router a cui affidare il resto della consegna. Dopo aver fatto questa scelta,
  1. incapsula il datagramma nella trama fisica per la sottorete su cui si trova il router,
  2. risolve l'indirizzo IP del router, e
  3. invia la trama al router usando l'indirizzo fisico.
- ***Notare: il pacchetto IP inviato al router contiene l'indirizzo IP del destinatario, non quello del router!!!***

I router di una internet TCP/IP formano una struttura cooperativa e interconnessa!
I datagram passano da un router a router finché non raggiungono uno che li può consegnare direttamente

Domande importanti:

- Come fa il mittente a sapere quale router scegliere?
- Come fa un router a scegliere un altro router per inoltrare il pacchetto?
- Come fa un router a "imparare" nuove strade, o a correggere quelle che conosce se diventano errate o migliorabili?

Table-driven IP routing

- Ogni implementazione di IP è **concettualmente** guidata da una ***tabella di instradamento IP*** che memorizza informazioni sulle **possibili** destinazioni **note**, e sul modo di raggiungerle. Cosa ci deve essere in questa tabella?
  - Se dovesse contenere tutte le possibili destinazioni,
    - Sarebbe pesante tenerla aggiornata
    - Tutti i router dovrebbero avere molta memoria per contenere la tabella
  - Vorremmo che le macchine prendessero decisioni sulla base di informazione minimale

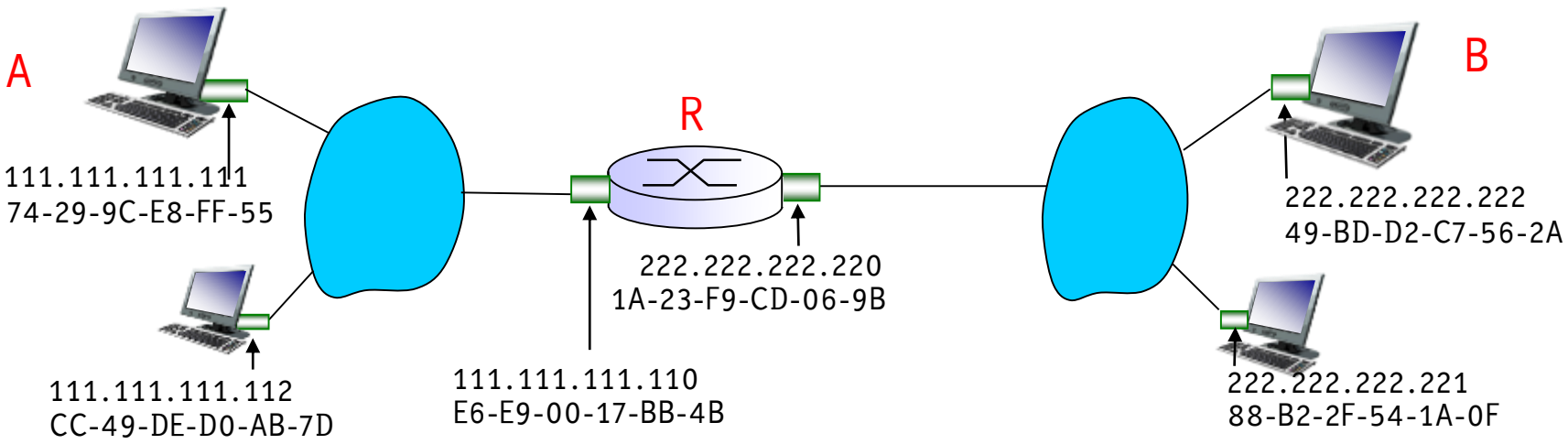
Come già detto, usiamo solo il prefisso (netid) degli indirizzi IP

L'implementazione usa davvero una tabella? Non è detto. Certamente non nei router ad alte prestazioni, che usano strutture dati sofisticate per ridurre i tempi di ricerca
Quindi parliamo di "tabella" in senso logico…

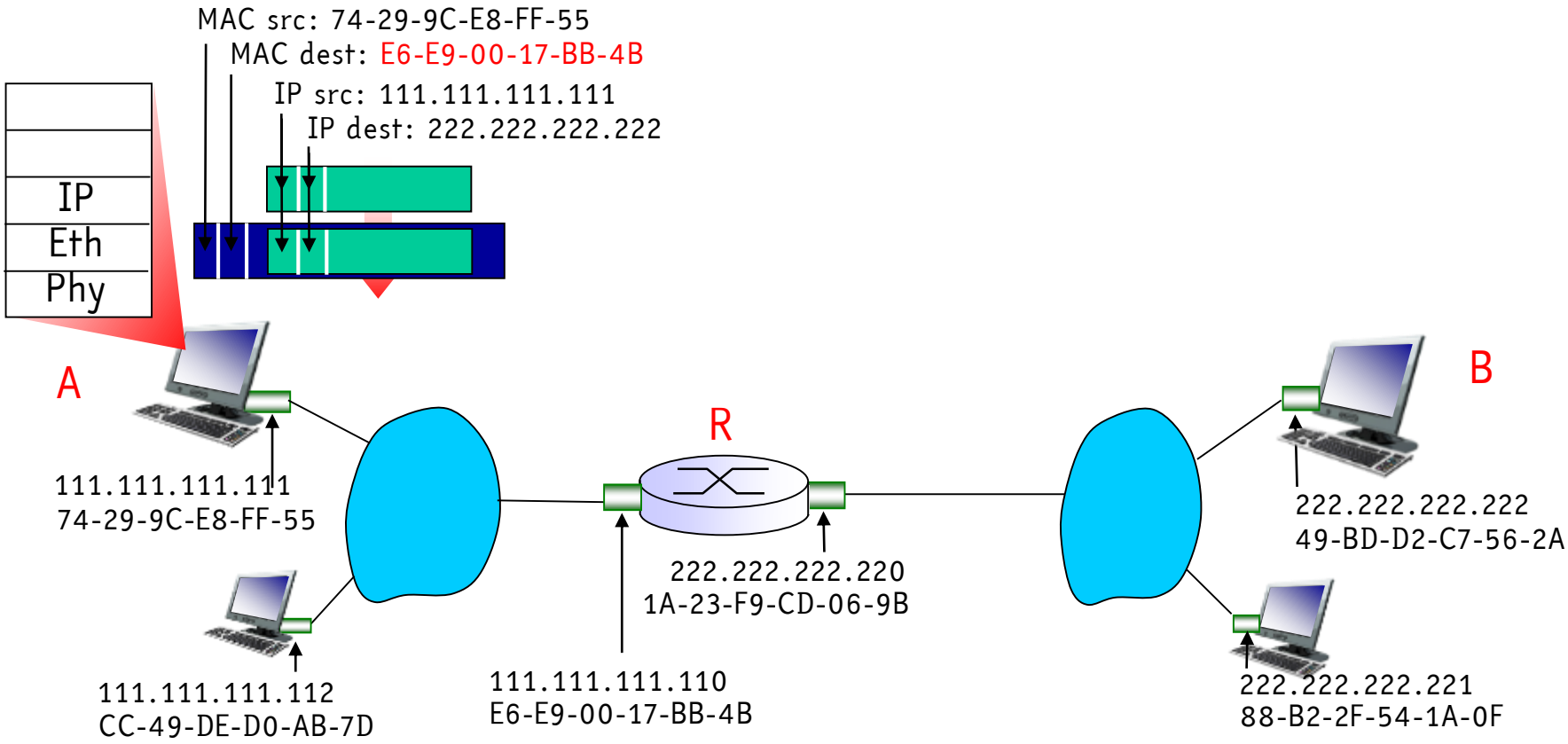
Instradare verso un'altra LAN

Panoramica: inviare un datagram da **A** a **B** via **R** (il router)

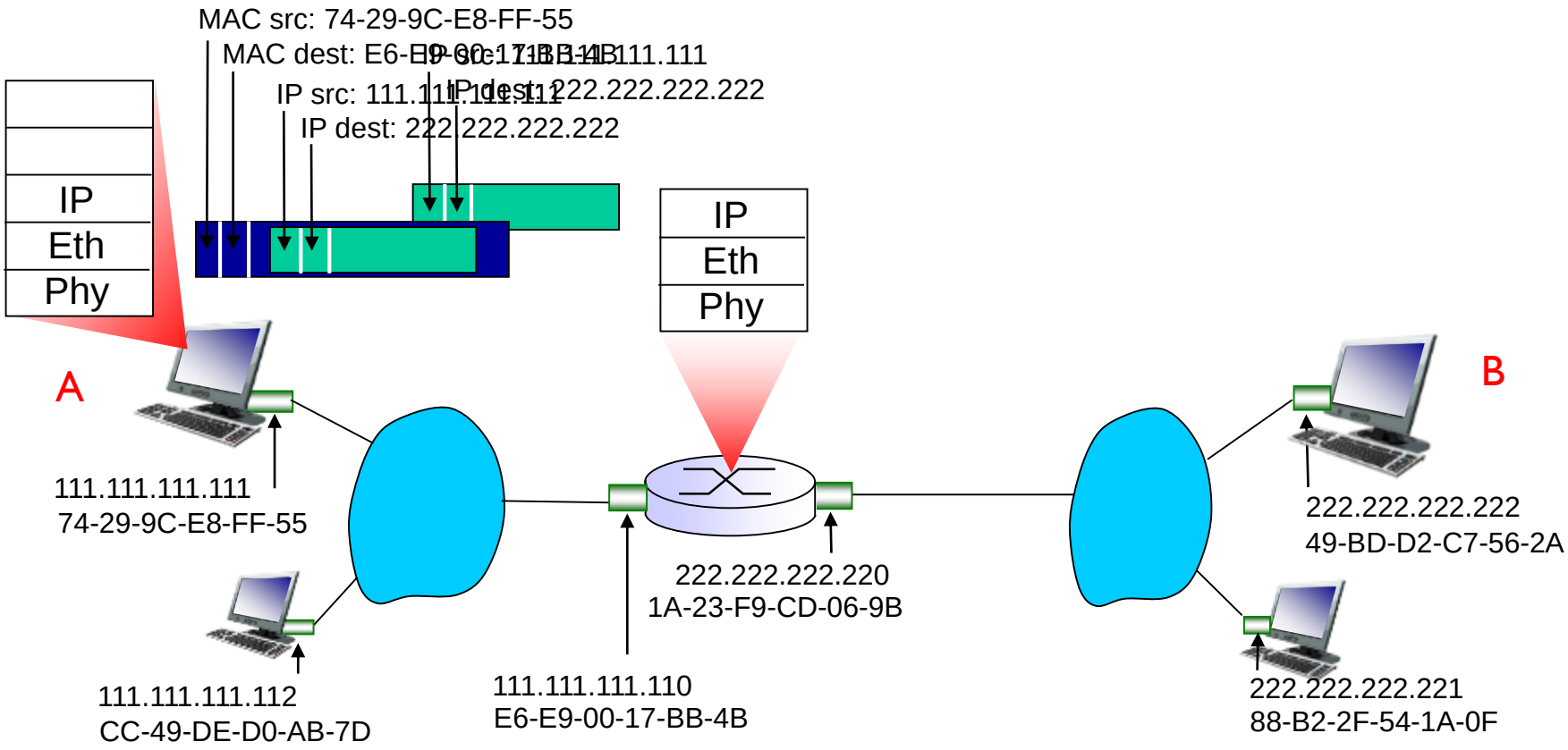
- focus sull'indirizzamento – a livello IP (datagram) e al livello MAC (frame)
- A conosce l'indirizzo IP di B
- A conosce l'indirizzo IP del primo router, R (come? Dalla tabella di inoltro)
- A conosce l'indirizzo MAC di R (come? Da ARP)



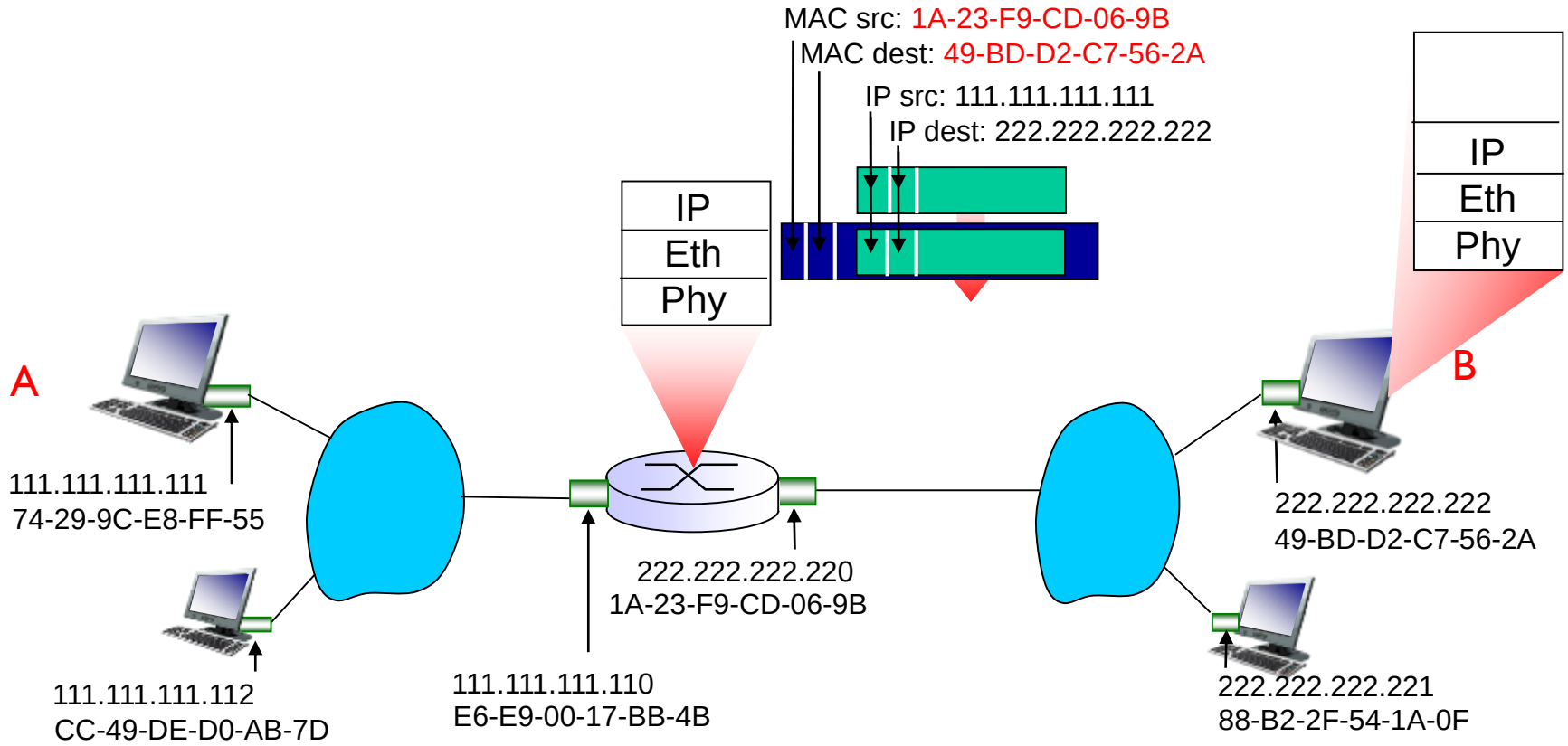
Instradare verso un'altra LAN



Instradare verso un'altra LAN

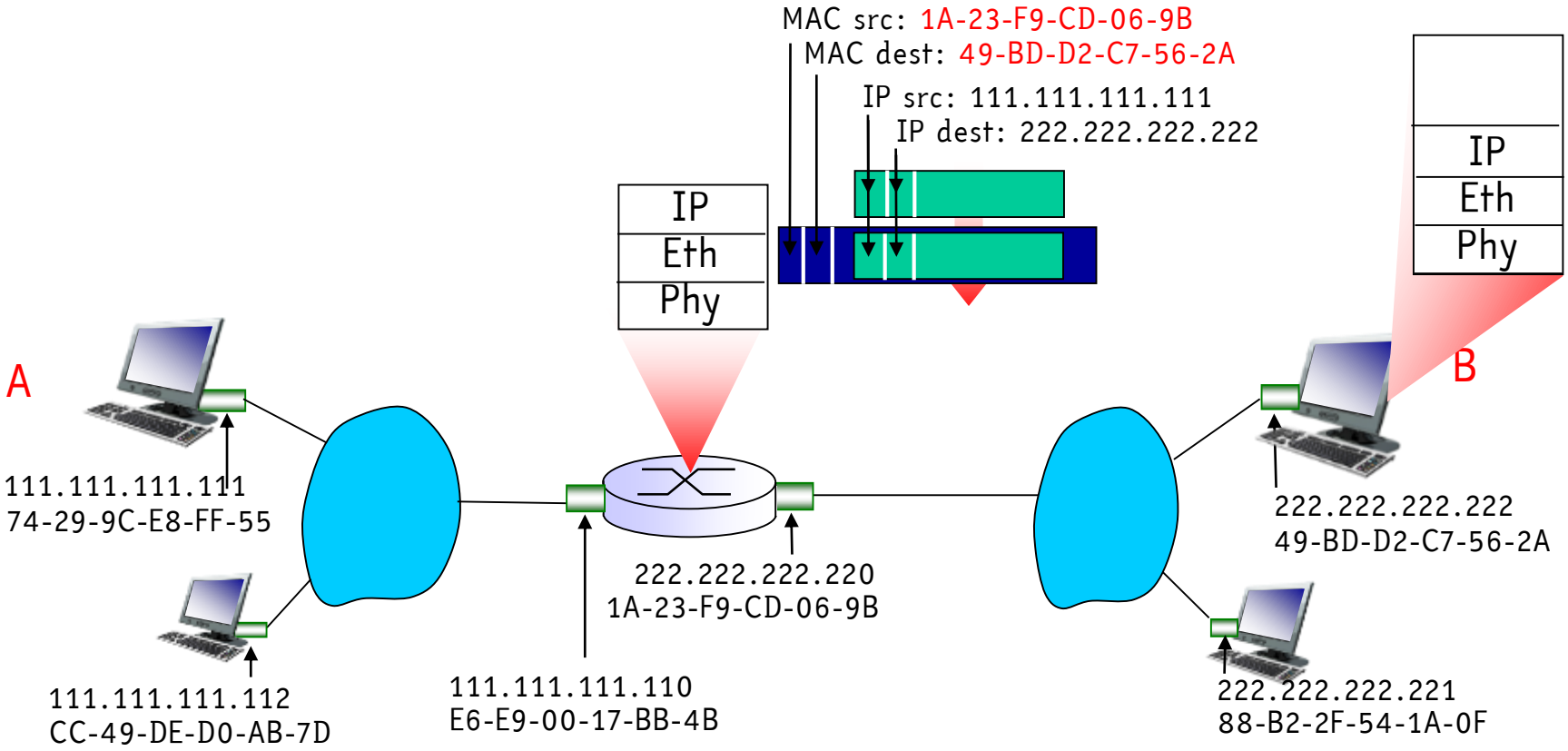


Instradare verso un'altra LAN

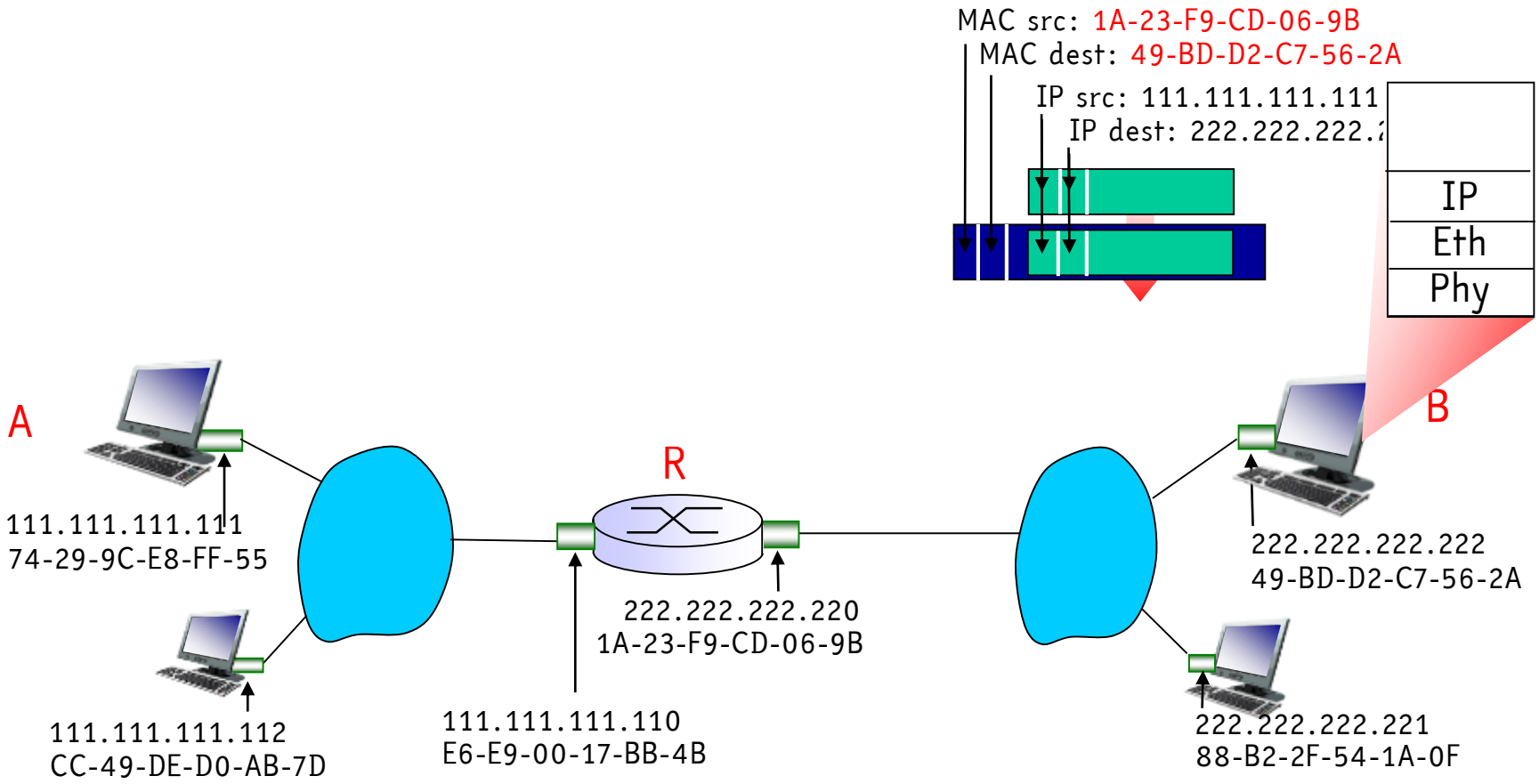


Instradare verso un'altra LAN

- R inoltra il datagramma con sorgente IP A, destinazione IP B
- R crea link-layer frame con il MAC address di B come dest, frame contiene il datagramma IP A-to-B



Instradare verso un'altra LAN



Instradamento basato sul prossimo salto (I)

- La tabella di instradamento contiene informazioni (N, R) dove
  - N = è un prefisso (≤ 32 bit)
  - R = indirizzo IP (32 bit) del prossimo router nel cammino verso tale destinazione. Contiene solo indirizzi di router **direttamente raggiungibili**, **direttamente connessi** (da una sottorete) alla macchina che li "punta" (host o router)
- ***Notare:***
  - *non è descritto l'intero cammino, ma solo il prossimo passo!*
  - *Contiene solo indirizzi IP e non indirizzi fisici di router*
- Quindi è un array a due colonne (N e R) con tante righe quante sono le reti destinazione conosciute

Ricerca nella tabella di inoltro

La chiave di ricerca nella tabella è di lunghezza variabile

La ricerca seguendo il principio del longest prefix matching aggiunge complessità alla ricerca

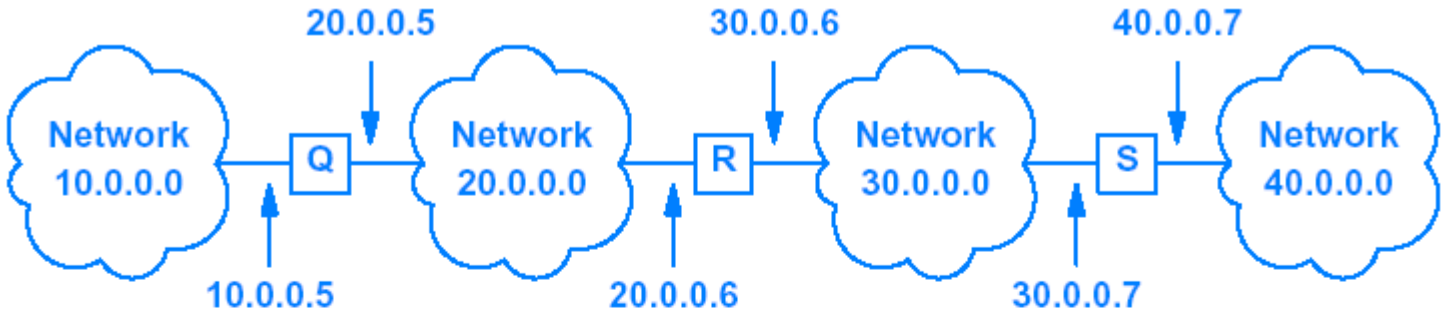
L'indirizzo del next hop individua la interfaccia di uscita, perché tutti gli indirizzi sulla sottorete hanno lo stesso prefisso

Si potrebbe rendere più semplice e quindi più veloce?

MPLS!

Destination Address Range	Link interface
11001000 00010111 00010*** *********	IP_{R0}
11001000 00010111 00011000 *********	IP_{R1}
11001000 00010111 00011*** *********	IP_{R2}
altrimenti	IP_{R3}

Instradamento basato sul prossimo salto (II)



(a)

TO REACH HOSTS ON NETWORK	ROUTE TO THIS ADDRESS
20.0.0.0	DELIVER DIRECTLY
30.0.0.0	DELIVER DIRECTLY
10.0.0.0	20.0.0.5
40.0.0.0	30.0.0.7

(b)

Instradamento basato sul prossimo salto (III)

- **+** Le tabelle di routing hanno una dimensione che dipende dal numero delle reti e non degli host
- **+** Crescono solo quando si aggiunge una rete all'internet
- **-** Solo il router finale può scoprire se la macchina destinazione esiste oppure no (occorre definire metodi di segnalazione degli errori!)
- **-** Il traffico verso una rete destinazione segue sempre la stessa strada, indipendentemente dalle condizioni della internet
- **-** Se esistono più cammini verso una destinazione, ne viene usato uno solo. Se vi è un guasto su questo cammino, cade la connettività (anche se ci sarebbero cammini alternativi) a meno che si modifichi la tabella di routing
- **-** I cammini fra due reti nelle due direzioni possono essere diversi (i router devono cooperare perché la comunicazione bidirezionale sia sempre possibile)

Gli ultimi quattro sono svantaggi!

Instradamento basato sul prossimo salto (IV)

- Questa semplice impostazione (per lungo tempo adottata) non permette
  - di instradare sulla base del TOS del pacchetto
  - dividere il traffico su più strade egualmente convenienti (load sharing)
- Ma è la semplice impostazione iniziale che ha permesso i primi grandi sviluppi di Internet
- Ora si cerca di fare di più sia sul **TOS** che sul **load sharing**

Strade di default

- Per contenere il numero di righe delle tabelle è comodo il concetto di ***default router:***

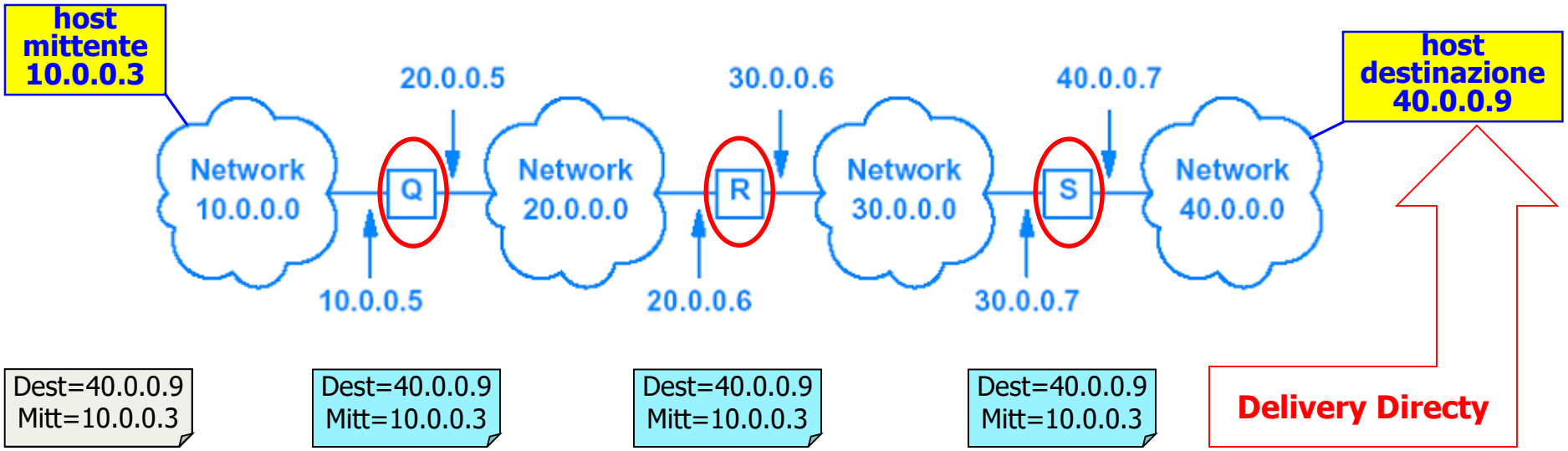
  se non si trova nessun cammino nella tabella di routing, il pacchetto viene inviato ad un particolare router

  Particolarmente utile per gli host!!

Strade specifiche per host

- L'instradamento *per un particolare host destinazione*, invece che per la rete destinazione,
  - questa possibilità è utile per trattare in modo particolare certi host per i quali vale la pena di seguire una strada diversa da quella seguita per gli altri host della rete a cui appartengono (e che vengono instradati per altra via)

L'algoritmo di routing: Esempio



Default router 10.0.0.5

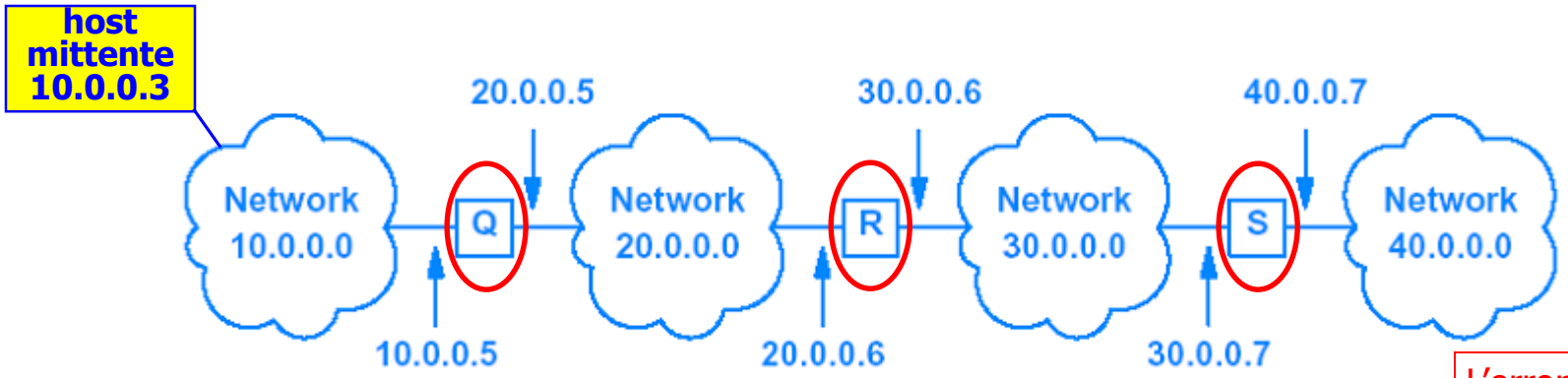
to r. h. on	Route to

to r. h. on	Route to
20.0.0.0	D.D.
10.0.0.0	D.D
30.0.0.0	20.0.0.6
40.0.0.0	20.0.0.6

to r. h. on	Route to
20.0.0.0	D.D.
30.0.0.0	D.D
10.0.0.0	20.0.0.5
40.0.0.0	30.0.0.7

To r. h. on	Route to
30.0.0.0	D.D.
40.0.0.0	D.D.
10.0.0.0	30.0.0.6
20.0.0.0	30.0.0.6

L'algoritmo di routing: host destinatario inesistente (netid "conosciuto")



Dest=40.0.0.19
Mitt=10.0.0.3

Default router 10.0.0.5

to r. h. on	Route to

Dest=40.0.0.19
Mitt=10.0.0.3

to r. h. on	Route to
20.0.0.0	D.D.
10.0.0.0	D.D
30.0.0.0	20.0.0.6
40.0.0.0	20.0.0.6

Dest=40.0.0.19
Mitt=10.0.0.3

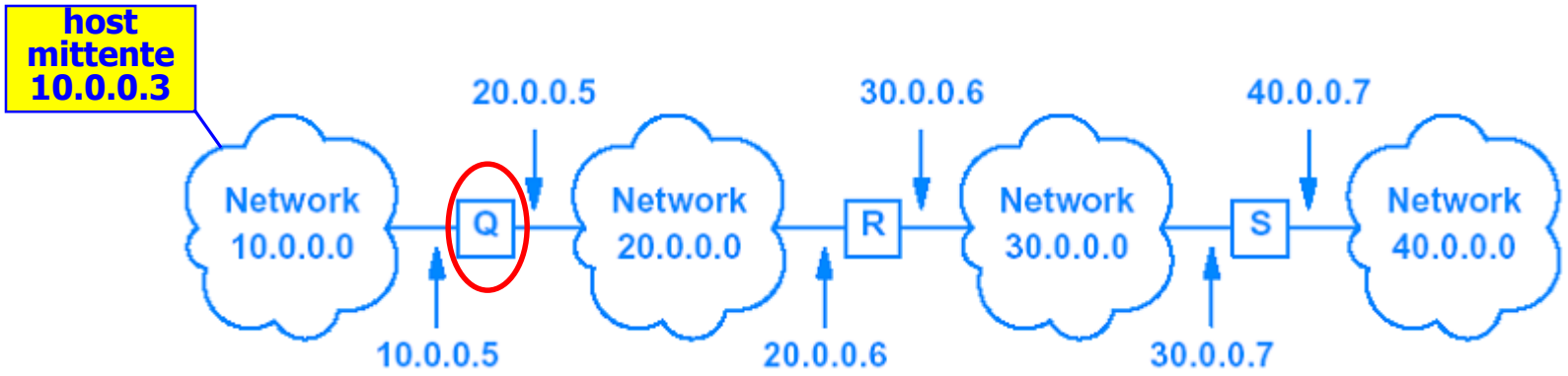
to r. h. on	Route to
20.0.0.0	D.D.
30.0.0.0	D.D
10.0.0.0	20.0.0.5
40.0.0.0	30.0.0.7

Dest=40.0.0.19
Mitt=10.0.0.3

To r. h. on	Route to
30.0.0.0	D.D.
40.0.0.0	D.D.
10.0.0.0	30.0.0.6
20.0.0.0	30.0.0.6

L'errore viene riscontrato dall'ultimo router che non trova l'host (netdid valido) sulla sua sottorete

L'algoritmo di routing: host destinatario inesistente (netid "conosciuto")



Dest=50.0.0.19
Mitt=10.0.0.3

Default router 10.0.0.5

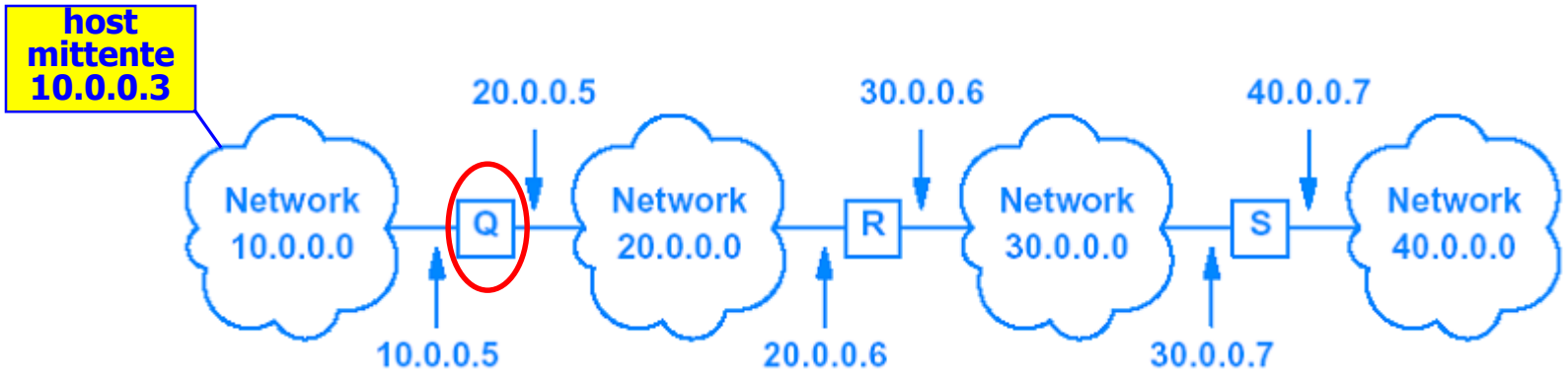
to r. h. on	Route to

to r. h. on	Route to
20.0.0.0	D.D.
10.0.0.0	D.D
30.0.0.0	20.0.0.6
40.0.0.0	20.0.0.6

L'errore viene riscontrato dal router Q che non trova una route per il netid=50.0.0.0

e il datagram viene scartato

L'algoritmo di routing: netid destinatario inesistente



Dest=50.0.0.19
Mitt=10.0.0.3

Default router 10.0.0.5

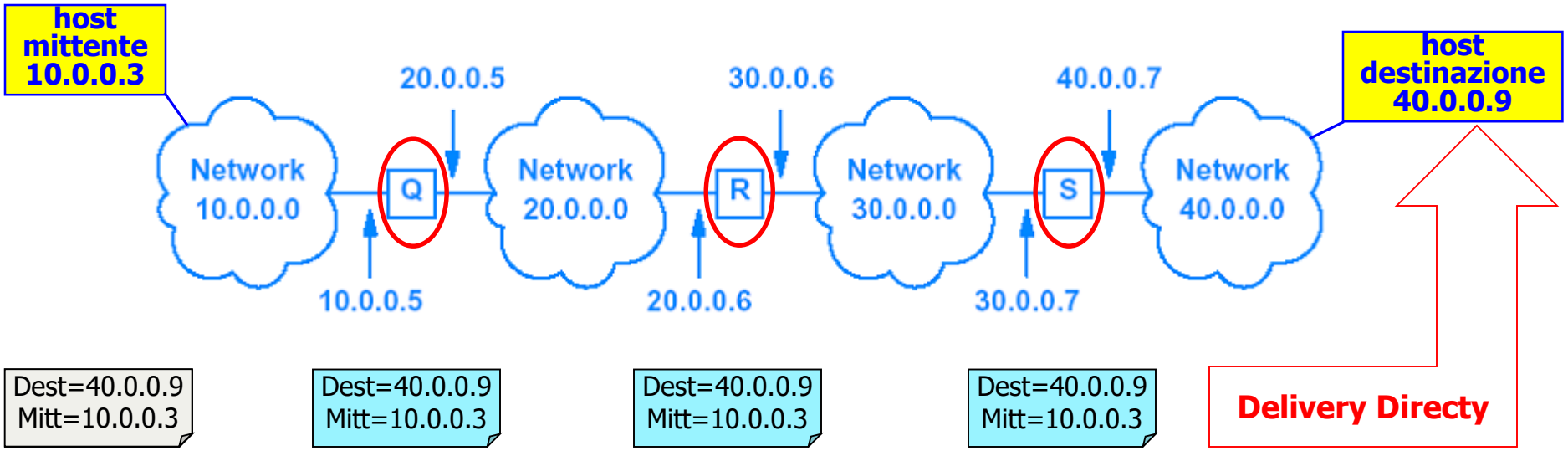
to r. h. on	Route to

to r. h. on	Route to
20.0.0.0	D.D.
10.0.0.0	D.D
30.0.0.0	20.0.0.6
40.0.0.0	20.0.0.6

L'errore viene riscontrato dal router Q che non trova una route per il netid=50.0.0.0

e il datagram viene scartato

L'algoritmo di routing: uso del default routing



Default router 10.0.0.5

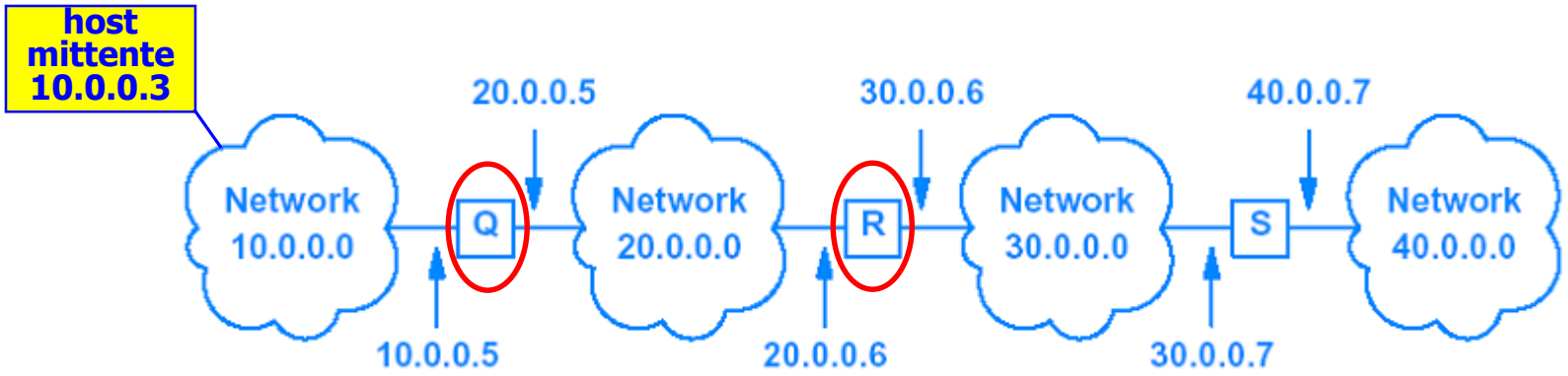
to r. h. on	Route to

to r. h. on	Route to
20.0.0.0	D.D.
10.0.0.0	D.D
30.0.0.0	20.0.0.6
Altre dir.	20.0.0.6

to r. h. on	Route to
20.0.0.0	D.D.
30.0.0.0	D.D
10.0.0.0	20.0.0.5
40.0.0.0	30.0.0.7

To r. h. on	Route to
30.0.0.0	D.D.
40.0.0.0	D.D.
10.0.0.0	30.0.0.6
Altre dir.	30.0.0.6

L'algoritmo di routing: default routing ed errori



Dest=50.0.0.9
Mitt=10.0.0.3

Default router 10.0.0.5

to r. h. on	Route to

Dest=50.0.0.9
Mitt=10.0.0.3

to r. h. on	Route to
20.0.0.0	D.D.
10.0.0.0	D.D
30.0.0.0	20.0.0.6
Altre dir.	20.0.0.6

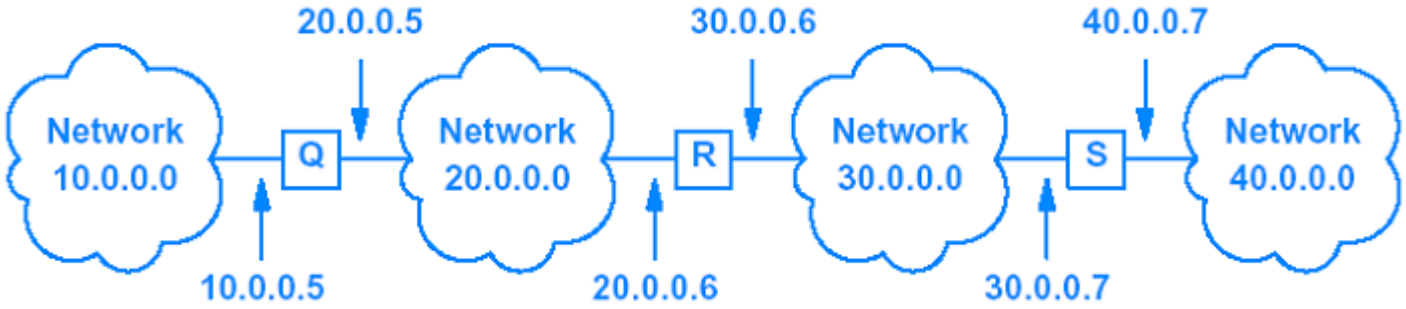
~~Dest=50.0.0.9 Mitt=10.0.0.3~~

to r. h. on	Route to
20.0.0.0	D.D.
30.0.0.0	D.D
10.0.0.0	20.0.0.5
40.0.0.0	30.0.0.7

L'errore viene riscontrato dal router R che non trova una route per il netid=50.0.0.0

e il datagram viene scartato

L'algoritmo di routing



to r. h. on	Route to
20.0.0.0	D.D.
10.0.0.0	D.D
30.0.0.0	20.0.0.6
40.0.0.0	20.0.0.6

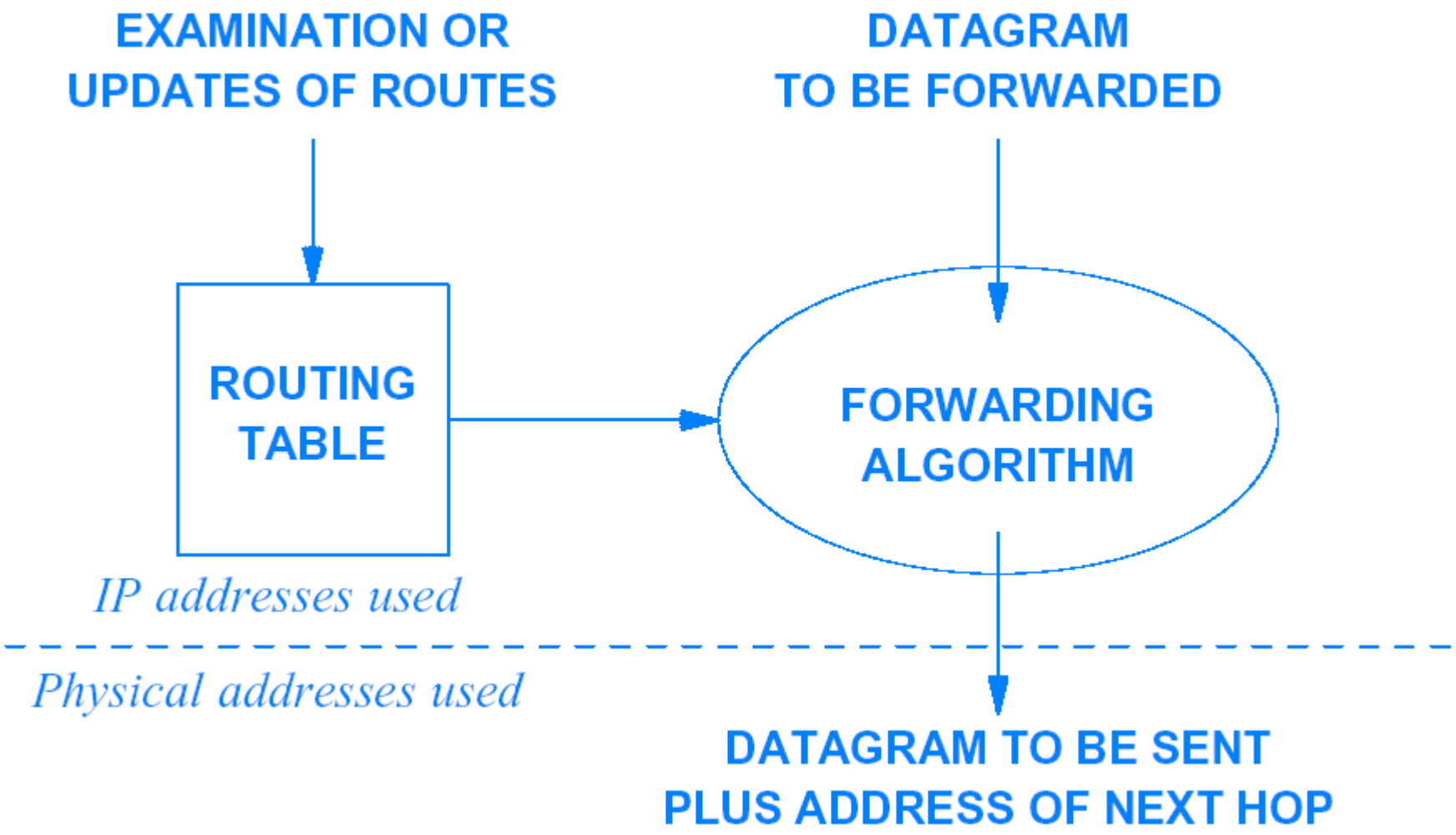
to r. h. on	Route to
20.0.0.0	D.D.
30.0.0.0	D.D
10.0.0.0	20.0.0.5
40.0.0.0	30.0.0.7

To r. h. on	Route to
30.0.0.0	D.D.
40.0.0.0	D.D.
10.0.0.0	30.0.0.6
20.0.0.0	30.0.0.6

Instradamento con router IP address (I)

- L'instradamento IP non modifica il datagramma originale, e in particolare non cambia gli indirizzi IP destinazione e sorgente
- Ogni volta che è stato calcolato il ***next hop address***, cioè l'indirizzo del prossimo router, il software dell'interfaccia prescelta lega/traduce l'indirizzo IP del router all'indirizzo fisico. Non è inefficiente? Non sarebbe utile memorizzare nelle routing table indirizzi fisici? Non si fa per le seguenti ragioni:
  - La routing table fornisce un confine pulito fra il software IP che instrada e il software di livello superiore che manipola/scopre le strade
  - Il software IP deve creare una astrazione al di sopra delle reti fisiche
  - L'indirizzo fisico di un router può cambiare
  - Tutto il software delle parti alte deve lavorare solo su indirizzi IP
  - Ri-effettuare il binding ogni volta può permettere di scoprire che il router vicino è down (ad es. se si usa ARP...)

Instradamento con router IP address (II)



Indirizzi di sottorete: ricezione

Quando IP riceve un datagramma da una certa interfaccia:

L'indirizzo destinazione è uguale ad uno degli indirizzi associati alla interfaccia che lo ha ricevuto?

- Sì: E' un datagramma intero o un frammento?
  - Frammento -> ri-assembla e alla fine tratta come datagramma intero
  - Intero -> passa al protocollo superiore indirizzato da PROTOCOL
- No: **la macchina è host o router?**
  - Host -> errore, scarta il datagramma senza segnalare errori
  - Router -> inoltra tramite tabella di inoltro

Notare che è solo qui la differenza fra host e router!

Gestione dei datagram entranti (I)

- Instradamento non riguarda solo i pacchetti uscenti

  DOPO aver controllato la checksum...
- *Negli **host** si deve:*
  1. Controllare se il datagramma arrivato è destinato a questa interfaccia
  2. Se è destinato a questa interfaccia, si consegna al livello superiore giusto (Quale è? Come fa a scoprirlo?)
  3. Se non è destinato a questa interfaccia, scartare il datagramma (gli host *non devono* inoltrare!)

Gestione dei datagram entranti (II)

- *Nei **router** si deve:*
  1. Controllare se il datagramma arrivato è destinato alla interfaccia su cui è arrivato
  2. Se è destinato alla interfaccia su cui è arrivato, si consegna al livello superiore giusto che è quello indicato dal campo PROTOCOL del datagramma (in genere sono pacchetti di test o di gestione del router)
  3. Se non è destinato a questo router, decrementare time-to-live (se non positivo scartare!), instradare con l'algoritmo di instradamento, e calcolare il nuovo checksum prima di spedire se necessario
- **Determinare se il pacchetto è indirizzato ad una interfaccia non è così semplice:**
  - La interfaccia può avere più indirizzi
  - Deve accettare i broadcast con limited broadcast o direct broadcast
  - Come vedremo, subnetting e multicast rendono il riconoscimento ancora più pesante

Perché scartare i pacchetti non destinati alla macchina se non si è router? (I)

- Se la macchina ha ricevuto un pacchetto non destinato alla sua interfaccia, c'è un errore da qualche parte → meglio non cercare di tamponarlo (in modo che venga corretto da un amministratore)
- L'inoltro provoca traffico addizionale e ruba CPU alla macchina (che non è destinata a questa funzione)
- Errori semplici possono causare un caos: ad es. se ogni host instradasse un pacchetto spedito via broadcast...???
- I router, oltre a inoltrare, segnalano errori: se ogni host mandasse errori il mittente, questi sarebbe bombardato!

Perché scartare i pacchetti non destinati alla macchina se non si è router? (II)

- Inoltre, sui router eseguono applicazioni che si scambiano messaggi che verificano la correttezza delle tabelle di routing: se gli host inoltrassero ma non partecipassero a questa cooperazione, si potrebbero creare anomalie
- Si noti che se su una interfaccia arriva un datagramma diretto ad una diversa interfaccia della stessa macchina multihomed, questo datagramma viene scartato. Su un router viene accettato

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- forwarding
- **DHCP**
- network address translation
- error messages (ICMP)
- IPv6

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

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"

DHCP: Dynamic Host Configuration Protocol

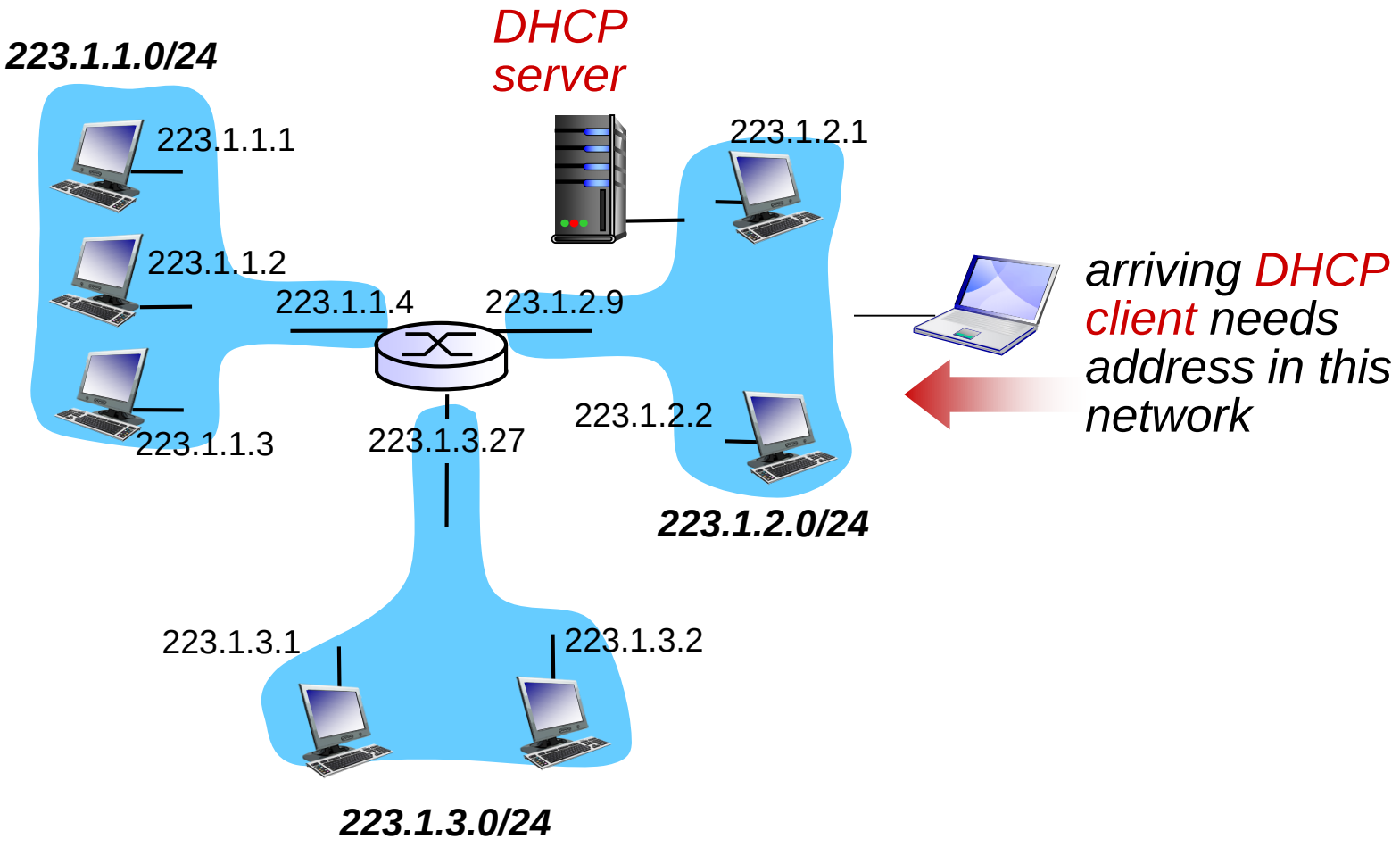
goal: allow host to *dynamically* obtain its IP address from network server when it joins network

- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/"on")
- support for mobile users who want to join network (more shortly)

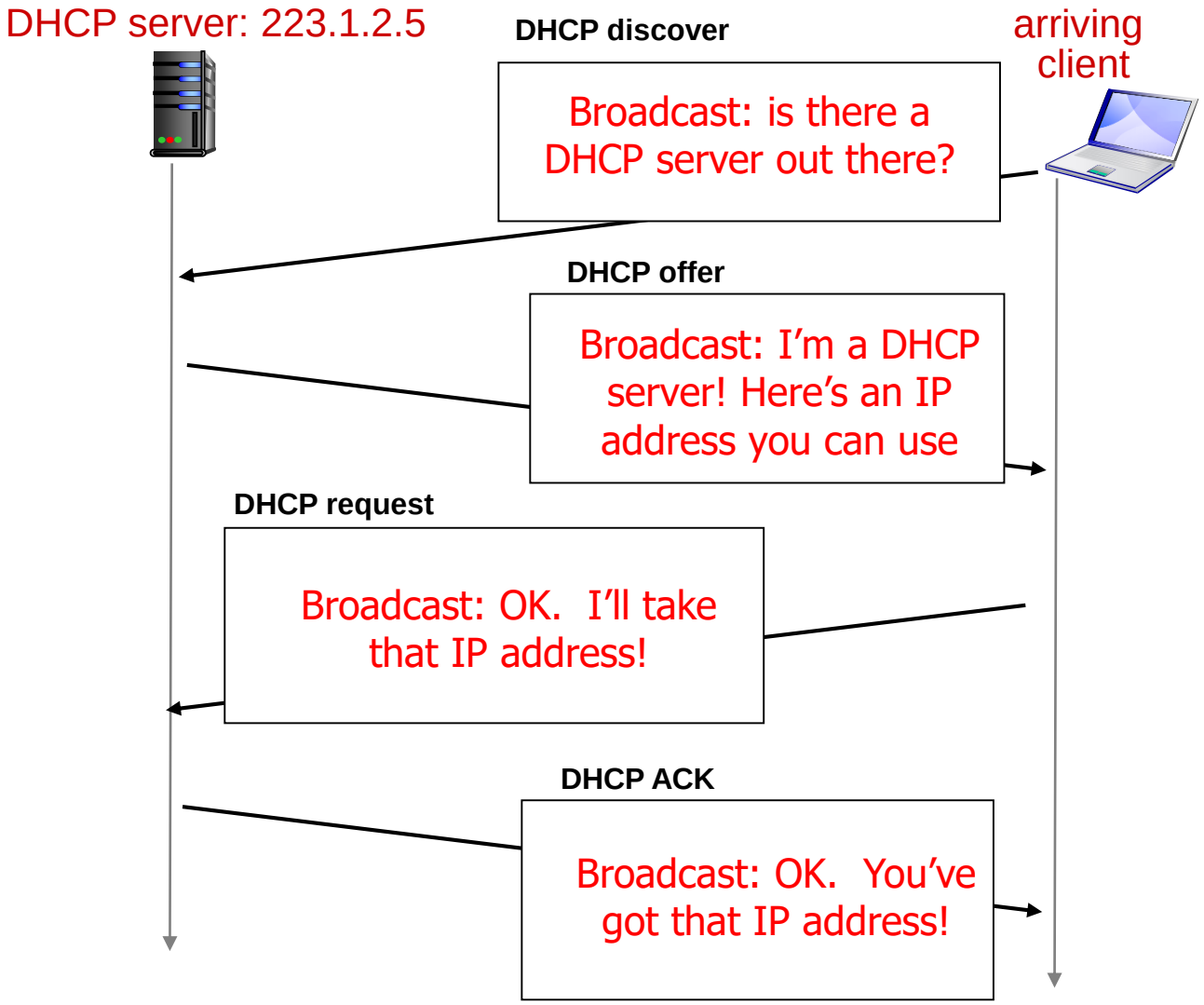
DHCP overview:

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

DHCP client-server scenario



DHCP client-server scenario



Perché il server DHCP non risponde in unicast?

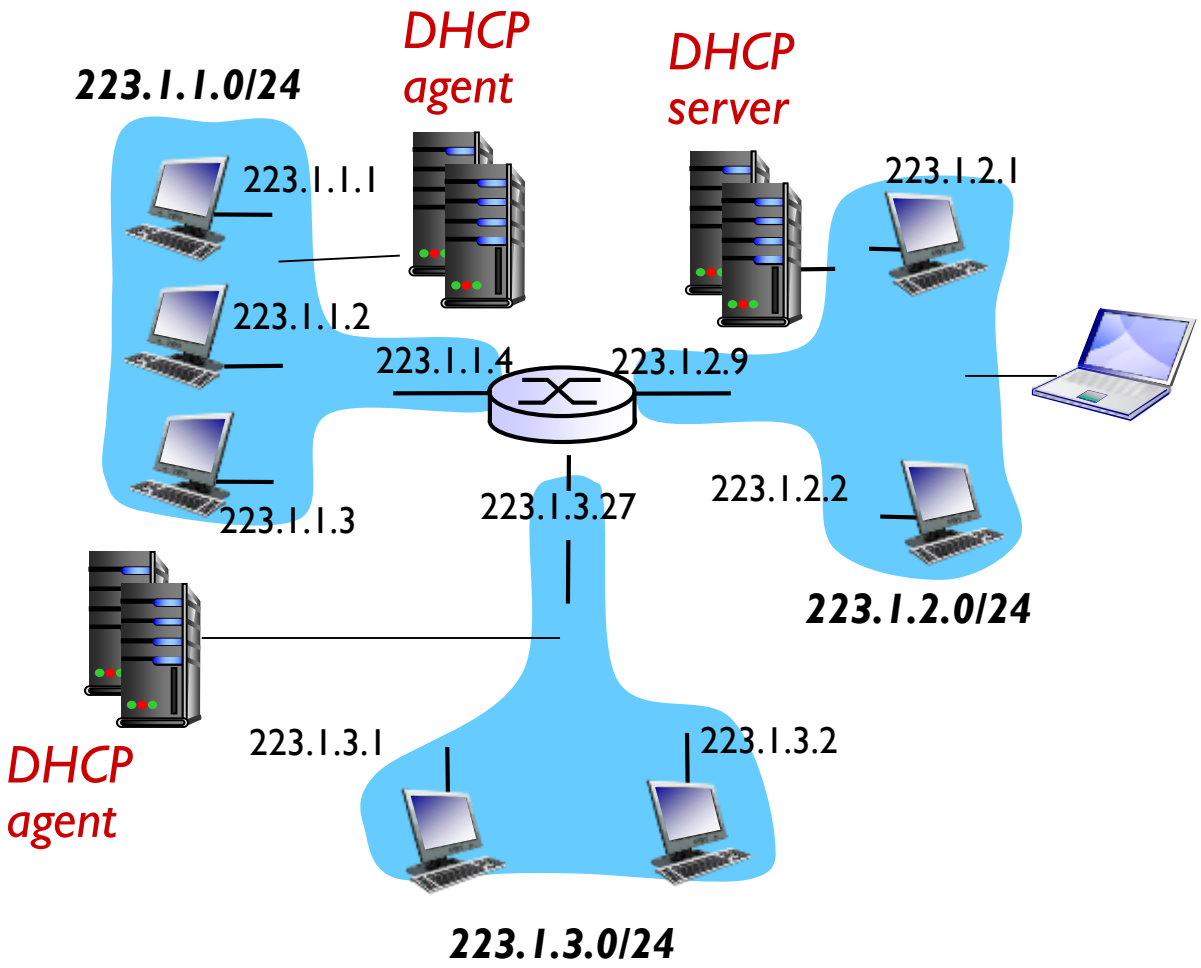
Rispondere in broadcast provoca elaborazione sui client, meglio rispondere in unicast!
Il server conosce l'indirizzo IP che vuole assegnare al client, e l'indirizzo MAC del client: perché non risponde in unicast?

1. Il client non accetterebbe il datagramma perché non diretto all'indirizzo IP della interfaccia (al momento 0.0.0.0)
   - Basta che l'implementazione IP del client sia "tollerante" e accetti il datagramma in questa situazione
2. Il software IP del server tenterebbe di scoprire il MAC address destinatario usando ARP Request: ma il client non conosce il suo indirizzo IP e quindi non risponderebbe!
3. Allora? ./. Allora come e quando il server risponde in unicast?
4. Se la cache ARP del server contenesse la coppia <IPClient, MACClient> la spedizione avrebbe successo
   - La cache ARP è parte del Sistema Operativo
   - Il Sistema Operativo può offrire una system call che permette ad un processo di aggiungere una riga alla cache di ARP
5. Il processo server DHCP può usare questa system call per aggiungere la riga alla cache, … e poi spedire il pacchetto UDP che contiene il messaggio "DHCP offer" (e "DHCP ack")
6. Il software IP del client è "tollerante"
7. Allora se le condizioni di cui sopra sono verificate, il server risponde in unicast!

Quanti server DHCP e dove?

- Se la rete (aziendale o meno) è importante, non posso usare un solo server (Single Point of Failure disastroso!)
- Il client spedisce le sue domande all'indirizzo 255.255.255.255 (broadcast limitato) che i router non devono inoltrare
- Servirebbero due (o più server DHCP) per sottorete IP: costosi da gestire e fonte di errori
- Allora, su tutte le sottoreti IP (meno una) si installano dei DHCP Agent, che funzionano da "router a livello applicativo" e smistano le richieste ai server DHCP e le risposte ai client appropriati
- I DHCP client non richiedono grande lavoro di gestione
- Quindi uno scenario realistico è …

Scenario DHCP "realistico"

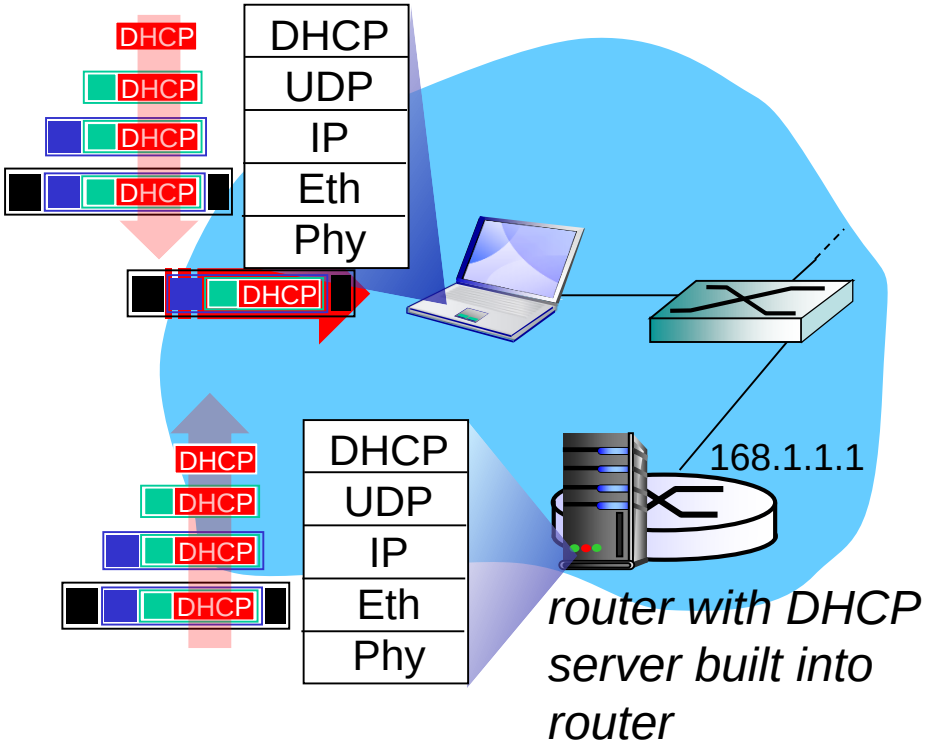


DHCP: more than IP addresses

DHCP can return more than just allocated IP address on subnet:

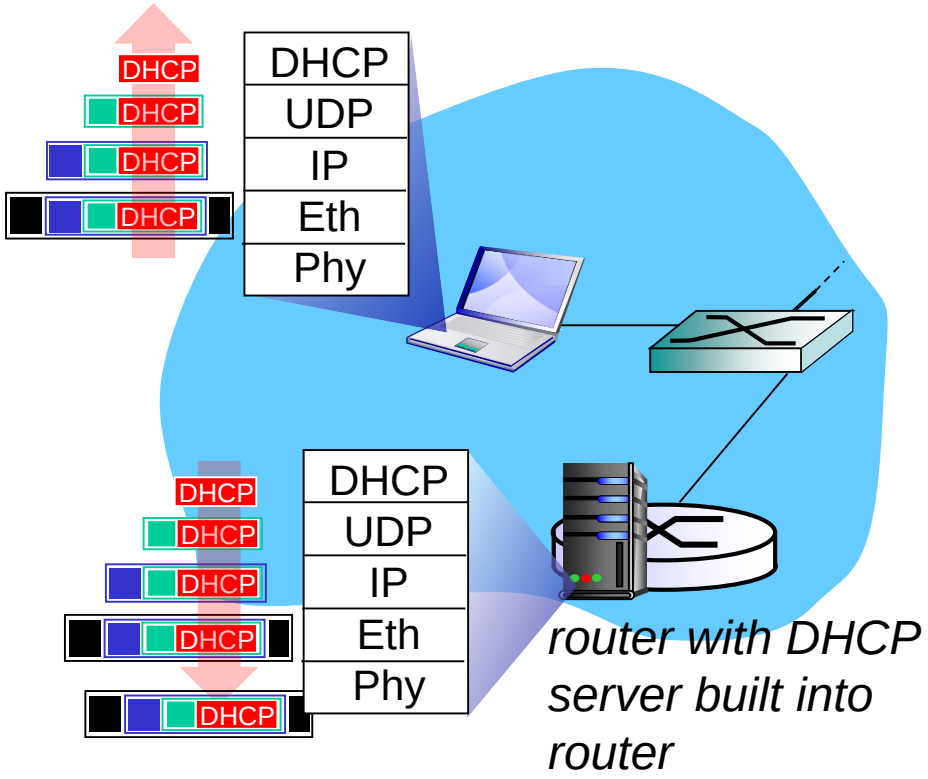
- address of first-hop router for client
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)

DHCP: example



- connecting laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP
- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet
- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demuxed to IP demuxed, UDP demuxed to DHCP

DHCP: example



- DCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulation of DHCP server, frame forwarded to client, demuxing up to DHCP at client
- client now knows its IP address, name and IP address of DSN server, IP address of its first-hop router

DHCP: Wireshark output (home LAN)

request

Message type: **Boot Request (1)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 0.0.0.0 (0.0.0.0)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 0.0.0.0 (0.0.0.0)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) **DHCP Message Type = DHCP Request**
Option: (61) Client identifier
    Length: 7; Value: 010016D323688A;
    Hardware type: Ethernet
    Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Option: (t=50,l=4) Requested IP Address = 192.168.1.101
Option: (t=12,l=5) Host Name = "nomad"
Option: (55) Parameter Request List
    Length: 11; Value: 010F03062C2E2F1F21F92B
    1 = Subnet Mask; 15 = Domain Name
    3 = Router; 6 = Domain Name Server
    44 = NetBIOS over TCP/IP Name Server
    ……

reply

Message type: **Boot Reply (2)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 192.168.1.101 (192.168.1.101)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 192.168.1.1 (192.168.1.1)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) DHCP Message Type = DHCP ACK
Option: (t=54,l=4) Server Identifier = 192.168.1.1
Option: (t=1,l=4) Subnet Mask = 255.255.255.0
Option: (t=3,l=4) Router = 192.168.1.1
Option: (6) Domain Name Server
    Length: 12; Value: 445747E2445749F244574092;
    IP Address: 68.87.71.226;
    IP Address: 68.87.73.242;
    IP Address: 68.87.64.146
Option: (t=15,l=20) Domain Name = "hsd1.ma.comcast.net."

Chapter 4: outline

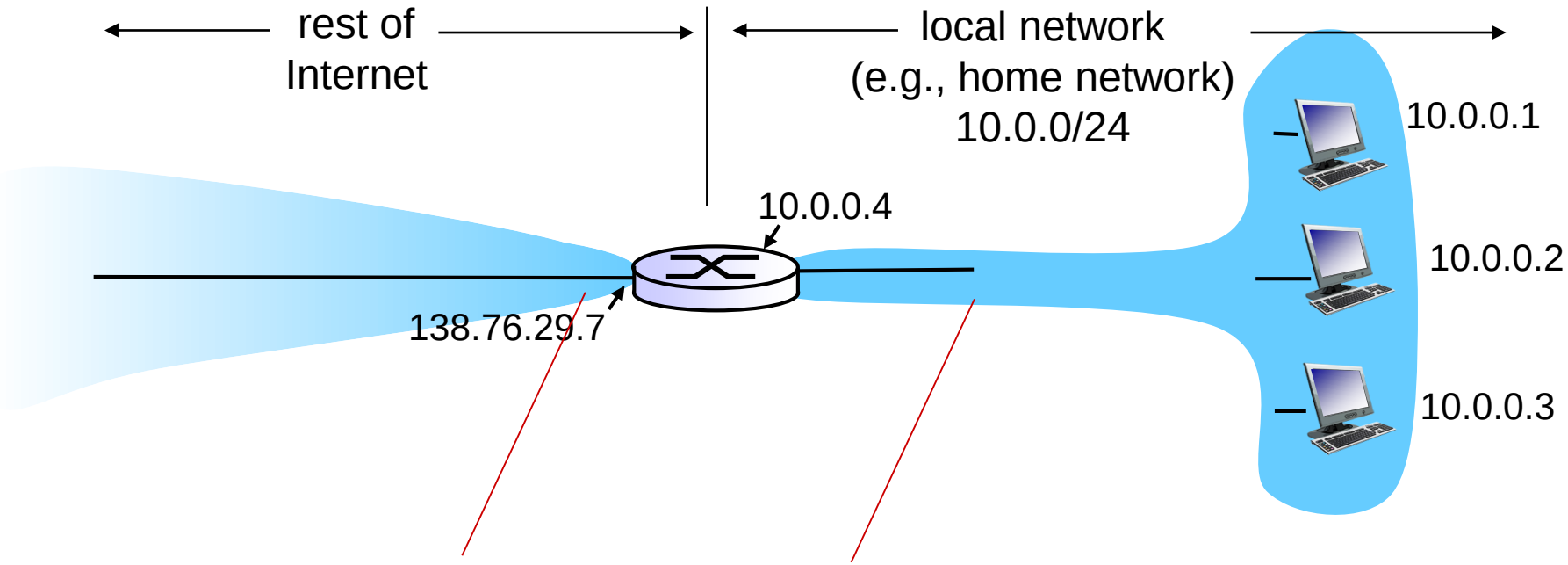
4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- forwarding
- DHCP
- **network address translation**
- error messages (ICMP)
- IPv6

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

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)

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:

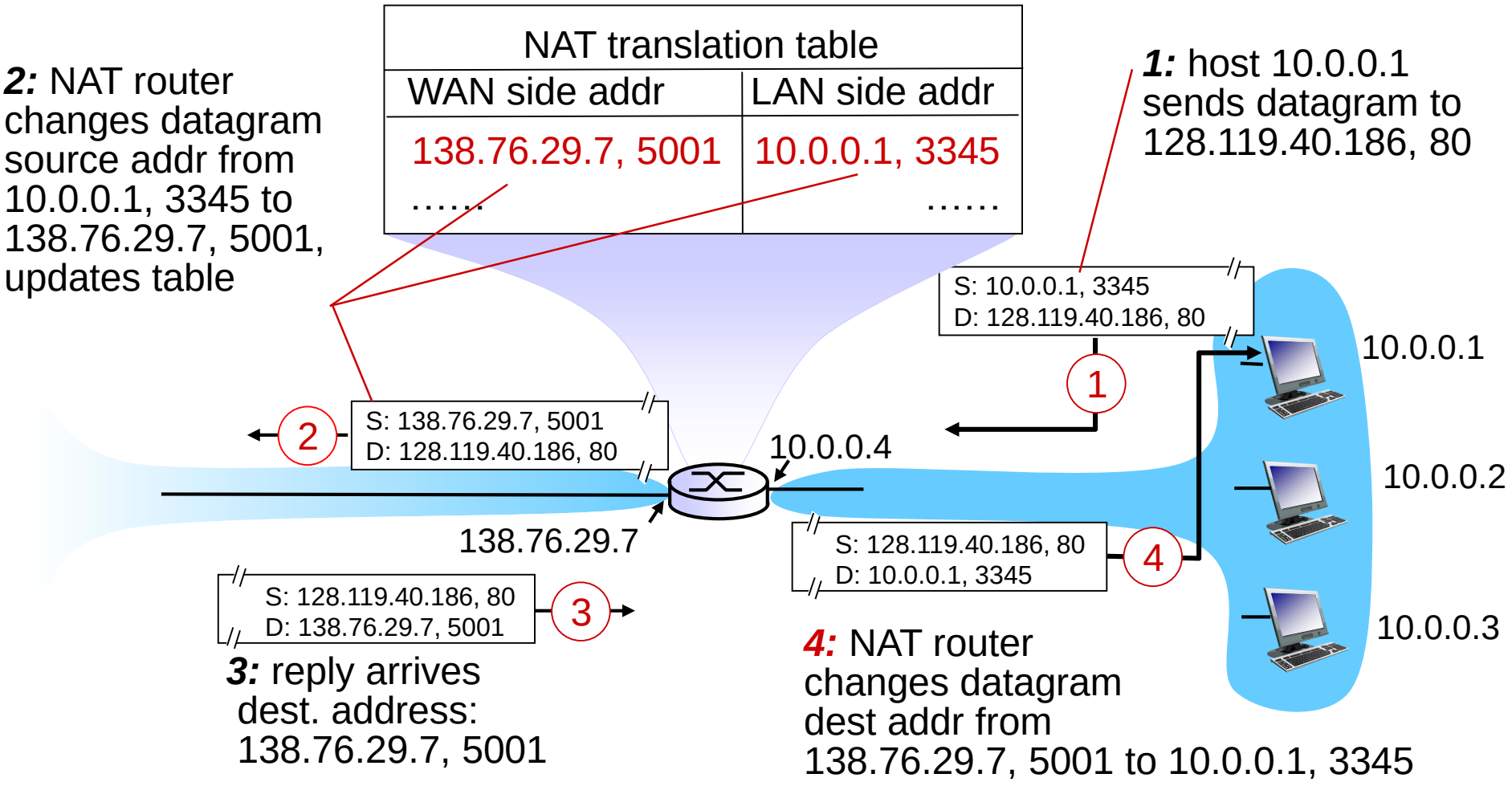
- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)

  . . . 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

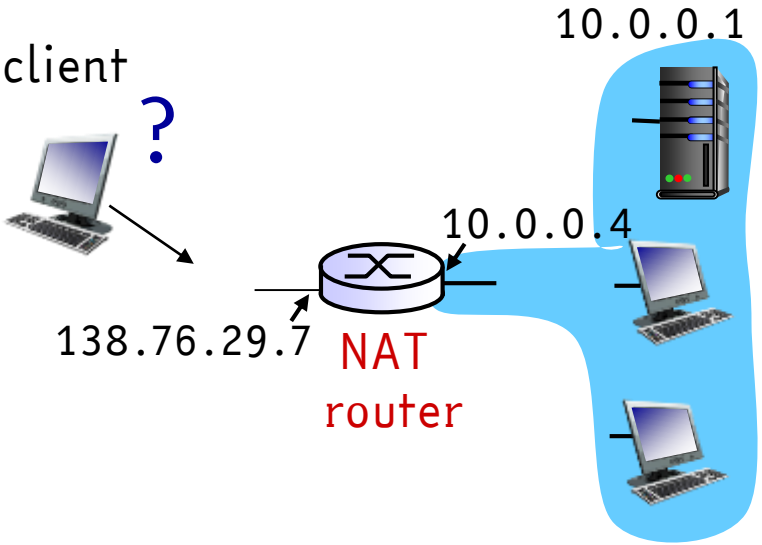
NAT: network address translation



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

Server interno al NAT: problema dell'attraversamento del NAT

- Client vuole connettersi al server con indirizzo 10.0.0.1
  - L'indirizzo del server 10.0.0.1 è solo locale alla LAN (non può essere usato fuori!)
  - L'unico indirizzo esterno visibile è quello del NAT: 138.76.29.7
- *Soluzione 1:* Configurare manualmente staticamente il NAT per inoltrare tutte le richieste di connessione su una porta esterna data verso server interno dato con porta data
  - e.g., (123.76.29.7, porta 80) sempre inoltrata a (10.0.0.1, porta 25000)



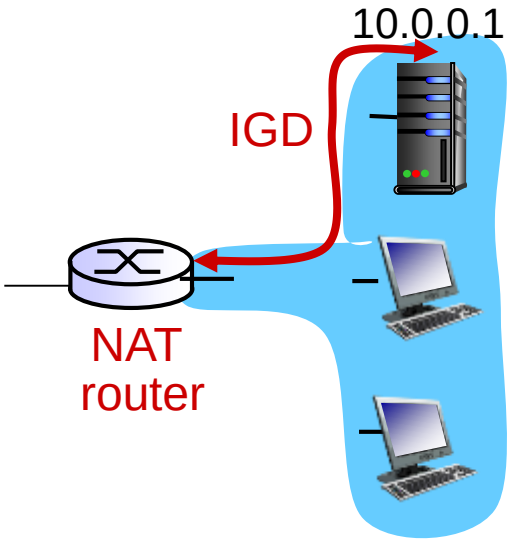
Server interno al NAT: problema…

Soluzione 2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol.

Permette agli host NATtati di:

- Apprendere l'indirizzo IP pubblico (138.76.29.7)
- Aggiungere/rimuovere port mapping nel NAT (con tempo di affitto)

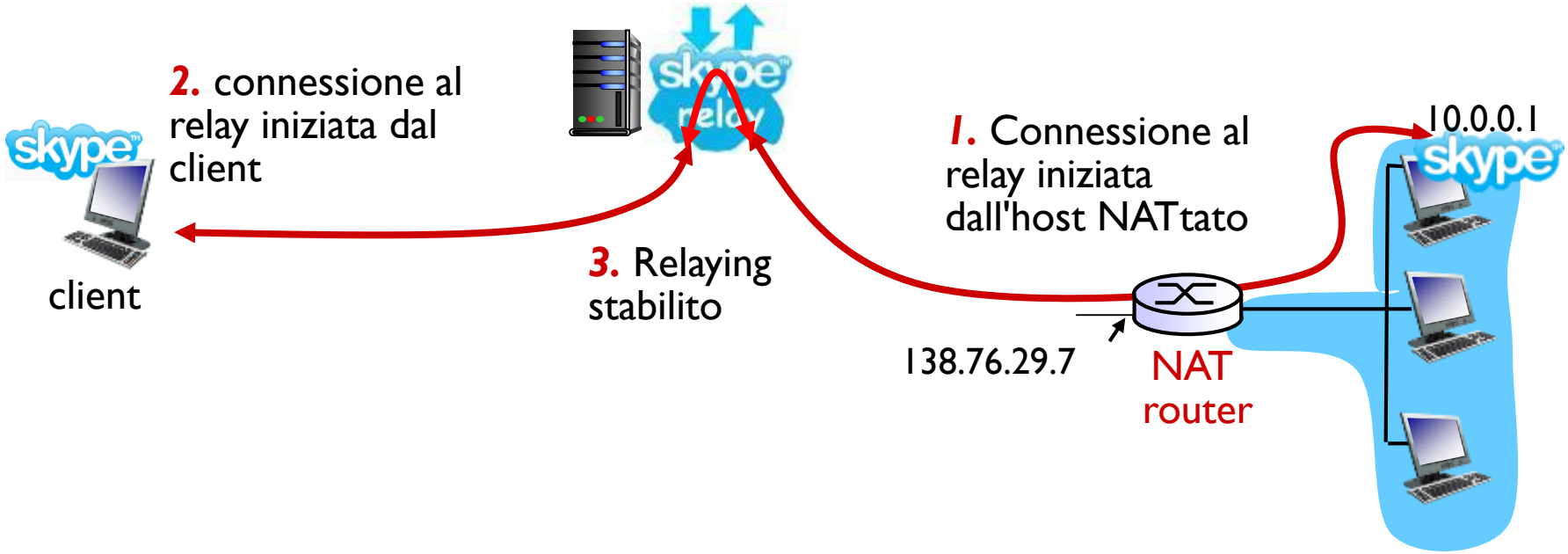
Permette di automatizzare la configurazione (statica) di mapping delle porte NAT



Server interno al NAT: problema…

Soluzione 3: fare relay (usata in Skype)

- Gli host NATtati stabiliscono una connessione con un relay
- Il client esterno si connette al relay
- Il relay fa bridge dei pacchetti fra le due connessione



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
  - address shortage should be solved by IPv6
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, e.g., P2P applications
  - NAT traversal: what if client wants to connect to server behind NAT?

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- forwarding
- DHCP
- network address translation
- **error messages (ICMP)**
- IPv6

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

IP: Messaggi di errore e di controllo

- IP fornisce un servizio non necessariamente affidabile e non fornisce nessuna conferma che il datagramma sia stato ricevuto correttamente
- La **quarta** funzione fondamentale di un servizio di comunicazione di livello di rete è **la segnalazione, alla sorgente, di situazioni di errore o di anomalia**, in modo che possano essere evitate e corrette
- Una funzione correlata è quella di fornire **strumenti di controllo e valutazione del funzionamento dell'internet**

Obiettivi di ICMP

ICMP consente ai router di inviare messaggi di errore e di controllo ad altri router o host; inoltre fornisce la comunicazione tra il software IP su una macchina e il software IP su un'altra

Internet Control Message Protocol (ICMP)

- **ICMP** è un protocollo a se stante
- È ***obbligatorio*** implementare ICMP assieme a IP
- I messaggi di ICMP **viaggiano nella parte dati di IP**, ma sono interpretati dallo stesso IP
- Un utente di IP viene però informato se **è all'origine** del malfunzionamento o **ne può essere interessato** in quanto ha inviato il messaggio coinvolto nel malfunzionamento
- I messaggi ICMP vengono nella maggior parte dei casi generati dai router, ma ICMP non è ristretto ai router: anche gli host, con qualche restrizione, possono/devono inviare messaggi ICMP, nonché i protocolli superiori
- Perché gli host segnalano meno errori dei router?
  - Perché gli host sono molti più numerosi dei router e sono meno "controllati" e meno "gestiti": occorre proteggersi da messaggi inutili generati da host mal gestiti o da attacchi alla rete

Segnalazione e correzione di errori (I)

- ICMP **non**
  - Fornisce dei meccamismi che permettono l'interazione tra un router ed una sorgente (che ha provocato con il suo comportamento dei malfunzionamenti)
  - Consente di mantenere informazioni di stato (ogni pacchetto viene trattato indipendentemente)
- Conseguenza

Quando un datagram causa un errore ICMP può solo indicare la condizione di errore alla sorgente del datagram, la quale deve mettere in relazione l'errore con un singolo programma applicativo oppure eseguire un'altra operazione per risolvere il problema

Segnalazione e correzione di errori (II)

- ICMP è un **meccanismo di segnalazione di errori**: non specifica l'azione da intraprendere a fronte degli errori e non intraprende nessuna azione
- ICMP riporta l'errore alla sorgente del datagramma, e quindi i router intermedi non ne vengono informati (e la strada a ritorno potrebbe essere diversa da quella all'andata!). La sorgente del datagramma a volte non è responsabile dell'errore (ad es. errore di routing) e non è in grado di correggerlo

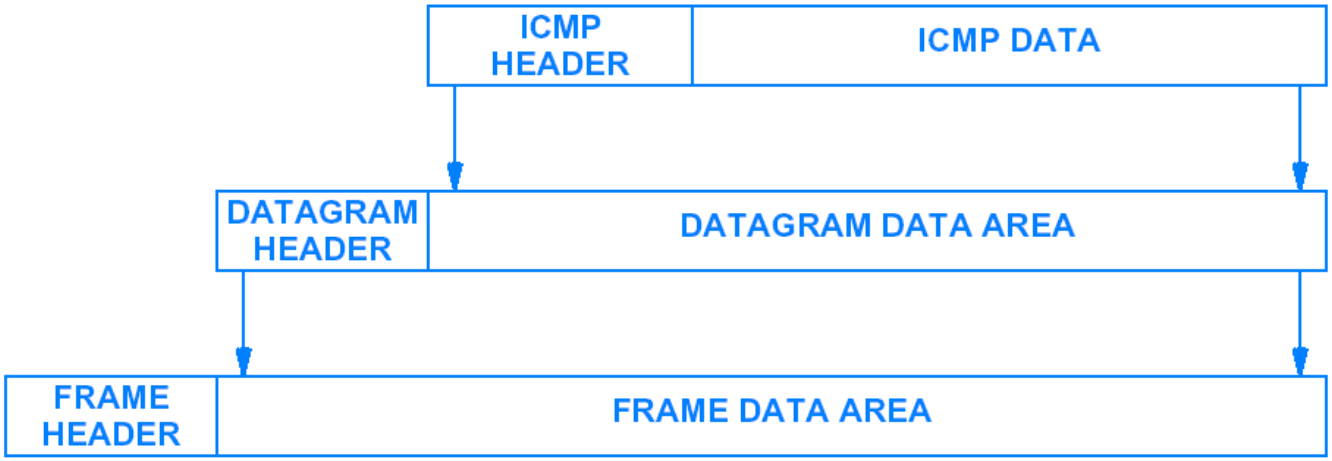
Segnalazione e correzione di errori (III)

- Perché riportare gli errori alla sorgente?
  - il datagramma contiene solo informazioni sulla sorgente e la destinazione
  - i router definiscono, e *possono cambiare*, le proprie tabelle di routing ⇒ non esiste un punto centrale in cui ci sia la conoscenza di tutta la internet *a cui riportare l'errore*
  - un errore nell'instradamento viene individuato da un router, che non può conoscere la strada seguita dal datagramma

Occorre cooperazione fra gli amministratori degli host e dei router per localizzare e riparare il problema: *molto difficile!*

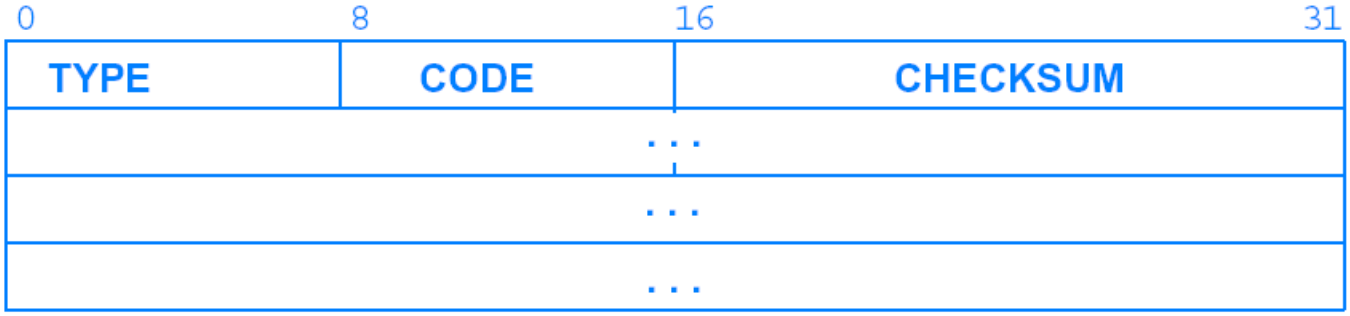
Consegna dei messaggi ICMP

Poiché ICMP è costruito sopra IP, i messaggi ICMP subiscono due incapsulamenti



- Attenzione!!! *Anche* se i messaggi sono mandati usando IP, ICMP *non* è considerato un protocollo di livello superiore
  - … questo è brutto!. La ragione di questa "bruttura" è che si è voluto usare il routing, la frammentazione e il riassemblaggio di IP
- Ma allora come si distinguono i messaggi ICMP, che sono diretti a IP stesso?
  - Per identificare un messaggio ICMP, il campo PROTOCOL vale 1. Da questo punto di vista è proprio trattato come se fosse un protocollo di livello superiore

Formato dei messaggi ICMP (I)

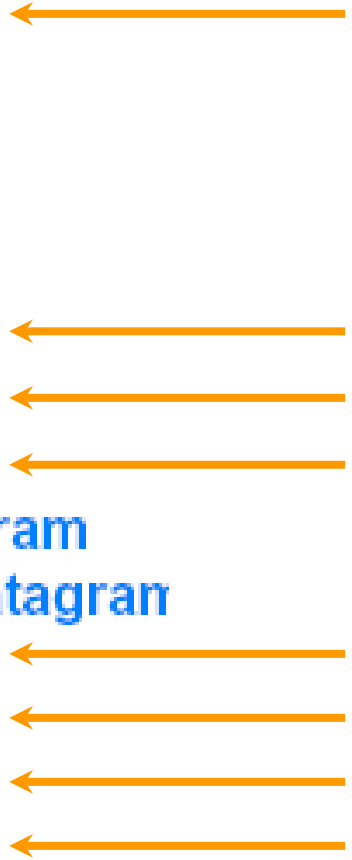


- TYPE 8bit
  CODE 8bit ulteriore identificazione
  CHECKSUM 16bit del solo messaggio ICMP
  - TYPE specifica che la destinazione non è raggiungibile
  - CODE specifica se si tratta di host o network (non raggiungibile)
- Quei messaggi ICMP che riportano errori di un datagramma IP, portano anche l'header IP e i primi 64 bit (8 byte, 2 parole di 32 bit) del datagramma che ha dato (che è coinvolto nel)l'errore

Classificazione dei messaggi ICMP
- Richiesta
- Errore

Formato dei messaggi ICMP (II)

Type Field	ICMP Message Type
0	Echo Reply
3	Destination Unreachable
4	Source Quench
5	Redirect (change a route)
6	Alternate Host Address
8	Echo Request
9	Router Advertisement
10	Router Solicitation
11	Time Exceeded for a Datagram
12	Parameter Problem on a Datagram
13	Timestamp Request
14	Timestamp Reply
15	Information Request
16	Information Reply



Formato dei messaggi ICMP (III)

Type Field	ICMP Message Type
17	Address Mask Request
18	Address Mask Reply
30	Traceroute
31	Datagram Conversion Error
32	Mobile Host Redirect
33	IPv6 Where-Are-You
34	IPv6 I-Am-Here
35	Mobile Registration Request
36	Mobile Registration Reply
37	Domain Name Request
38	Domain Name Reply
39	SKIP
40	Photuris

Messaggi di errore ICMP

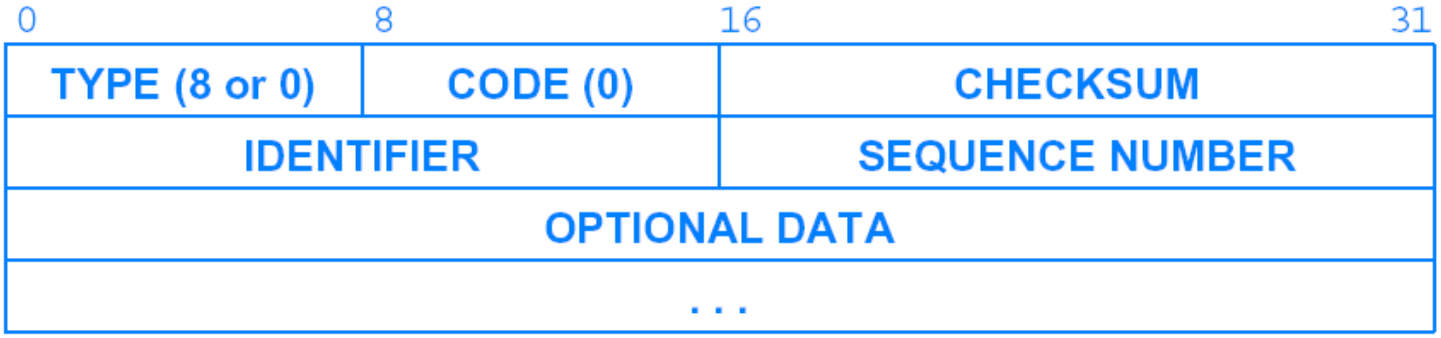
Per non causare un aumento eccessivo di messaggi di errore ICMP questi non vengono generati in risposta a:

- Un messaggio di errore ICMP
- Un pacchetto IP multicast o broadcast
- Un frammento di un pacchetto diverso dal primo

Controllo della raggiungibilità e dello stato di una destinazione

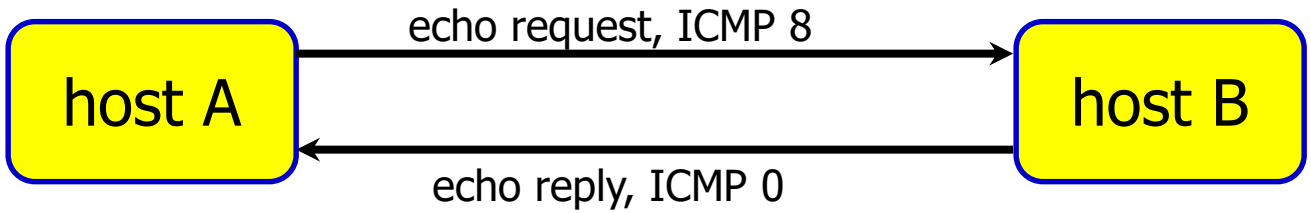
- Gli *Echo Request* e *Echo Reply* sono fra i più usati. La *Reply*, inviata alla sorgente del datagramma a cui si risponde, contiene (opzionalmente) una copia dei dati che il datagramma originale opzionalmente può contenere. Viene controllato così che:
  - il software IP della macchina sorgente è in grado di instradare verso quella destinazione
  - i router intermedi sono in funzione e sanno instradare da sorgente a destinazione
  - Il software IP destinazione è in funzione (sa rispondere agli interrupt) e sa instradare verso la sorgente
  - i router intermedi sono in funzione e sanno instradare da destinazione a sorgente
  - il software IP mittente è anche in grado di ricevere
- Il (famoso) programma applicativo che usa questa funzione si chiama (spesso) *ping*

Formato Echo Request e Reply



- I campi IDENTIFIER, SEQUENCE NUMBER e OPTIONAL DATA permettono al mittente di calcolare statistiche sul *round trip time* e tassi di perdita dei datagrammi di lunghezza diversa
- Nella *Reply* ci sono i dati della *Request* in modo che il datagramma sia lungo quanto richiesto! Con lunghezze diverse del datagramma si misura il comportamento della rete in presenza di frammentazione
- Notate che nonostante che i due messaggi siano molto "parenti", essi in realtà appartengono a "famiglie" diverse perché hanno valori differenti di TYPE

Formato Echo Request e Reply (II)

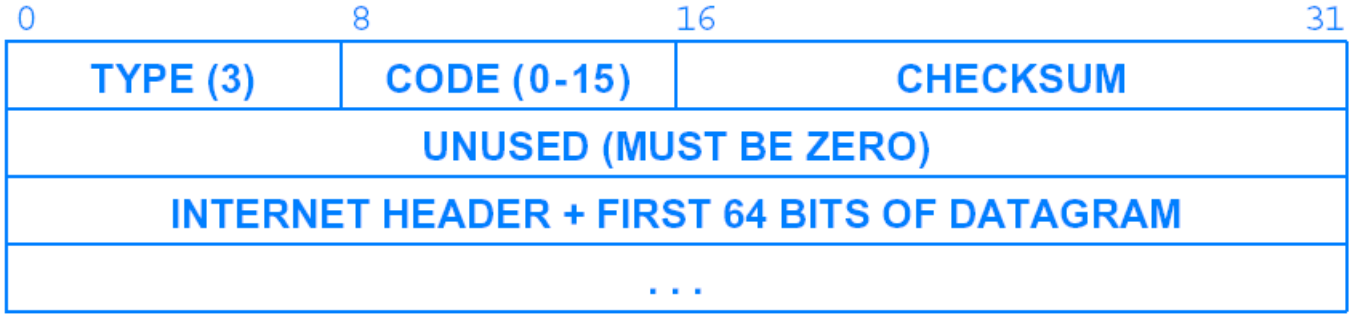


OPZIONI DEL COMANDO

Il comando **ping** ha diverse opzioni. Alcuni esempi:

1. ping -t (ttl) 192.168.0.1
2. ping -l 10000 192.168.0.1
3. ping -w 2000 192.168.0.1

Destinazioni irragiungibili (I)



- Quando un router non può raggiungere una destinazione, invia alla sorgente un messaggio ICMP di destinazione irraggiungibile
- Quante ragioni perché una destinazione sia irraggiungibile: **studiarle**! E domandarsi il perché di ciascuna ragione!
- Non per tutte le ragioni è un router il componente che individua l'errore e lo segnala: importante capire quale componente individua l'errore. Anche altri componenti individuano errori!

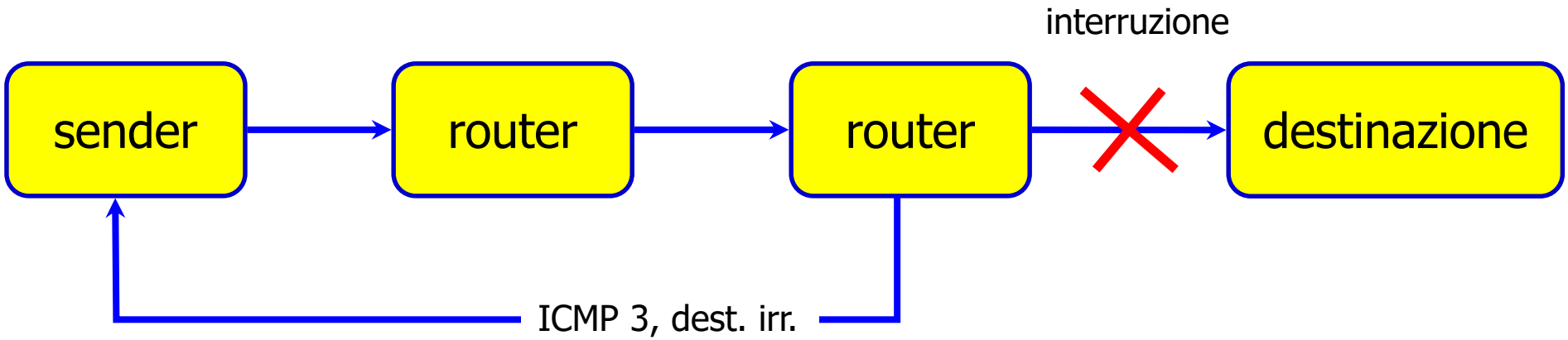
Destinazioni irragiungibili (II)

Code Value	Meaning
0	Network unreachable
1	Host unreachable
2	Protocol unreachable
3	Port unreachable
4	Fragmentation needed and DF set
5	Source route failed
6	Destination network unknown
7	Destination host unknown
8	Source host isolated
9	Communication with destination network administratively prohibited
10	Communication with destination host administratively prohibited
11	Network unreachable for type of service
12	Host unreachable for type of service
13	Communication administratively prohibited
14	Host precedence violation
15	Precedence cutoff in effect

Destinazioni irragiungibili (III)

- Un router *può/deve* scartare datagrammi quando un errore gli impedisce di raggiungere la destinazione
- ***Network/Host Unknown:*** non so come **instradare**
- ***Network Unreachable* =** errore di consegna: non so come **inoltrare/consegnare**
- ***Host Unreachable* =** fallimento nella consegna, rilevato dal router (altri errori di consegna possono non essere visibili, vedi es. in ethernet):
  - hardware (scheda) fuori servizio
  - indirizzo non esistente
- ***Protocol Unreachable***: non so come consegnare/Instradare a un protocollo superiore a IP ⇒ lo segnala l'IP di un host!
- ***Port unreachable*** sarà chiaro più avanti; non lo segnala un router, e neanche l'IP di un host, ma un altro componente: un transport

Destinazioni irragiungibili (IV)



Un router invia un messaggio di destinazione irraggiungibile quando incontra un datagram che non può essere inoltrato o consegnato; comunque il router non può rilevare tutti gli errori di questo tipo

Controllo della congestione e del flusso di datagrammi (I)

- IP è un connectionless service => i router non possono riservare
  - memoria o
  - risorse di rete o
  - di CPU
- per garantire che i datagrammi vengano trattati adeguatamente (ecco le risorse che servono per offrire il servizio IP!)
- Parliamo di CONGESTIONE quando mancano le risorse necessarie!
- I router possono essersi sommersi da traffico che non riescono a smaltire => **congestione**: si può verificare per **tre** ragioni:
  1. una macchina può generare traffico maggiore di quanto una sottorete può assorbire
  2. molte macchine generano un traffico aggregato superiore a quanto una sottorete può assorbire
  3. il traffico che arriva ad un router è superiore a quello che il router riesce ad elaborare

Controllo della congestione e del flusso di datagrammi (II)

Abbiamo tre scenari tipici (due "in uscita", e uno "in ingresso"):

1. *Singola sorgente:* una macchina è connessa ad una LAN veloce, e genera traffico verso un host che si trova su una sottorete geografica lenta *al di là di un router* => il router che si affaccia sulla sottorete lenta riceve più traffico di quanto possa smaltire in uscita => **congestione in uscita**
2. *Molte sorgenti:* anche se nessuna delle macchine genera un singolo flusso di dati superiore alla capacità della sottorete destinazione, il traffico aggregato è superiore a quello che si può smaltire; ad es. il router può avere molte interfacce, tutte della stessa velocità, ma il flusso dei dati è tutto verso una sola linea in uscita => **congestione in uscita**
3. *In ingresso:* il router non ha sufficiente capacità elaborativa per trattare tutti i messaggi che arrivano dalla(e) sottorete(i) => arriva in ritardo a leggere i messaggi => vengono scartati dall'hardware della sottorete (ad es. in Ethernet) => **congestione in ingresso**: il router non la può rilevare!

Nei casi 1 e 2 il router "si accorge" di essere congestionato, nel caso 3 non se ne può accorgere!
Quindi non si può pretendere che i router segnalino **sempre** la congestione

Cosa c'è sotto davvero? (I)

- Cosa c'è alla base di questo fenomeno di congestione? E' chiaramente un problema di velocità, ma si potrebbe fare qualcosa? **Le sottoreti fisiche potrebbero fare qualcosa?**
- Le sottoreti possono avere o meno le seguenti funzionalità:
  - **Riscontro:** vi sono elementi di protocollo (messaggi e campi di messaggi) che informano il mittente della corretta (a volta anche della incorretta) ricezione di un messaggio inviato => il mittente ritenta la spedizione se riceve riscontro negativo o non riceve quello positivo entro un certo tempo
    Notare: in assenza di questa funzionalità, messaggi spediti possono essere persi senza che il mittente lo possa scoprire (es. ethernet), e se il ricevente è congestionato (non ce la fa a leggere i pacchetti) il destinatario neanche lo sa!

Cosa c'è sotto davvero? (II)

- **Controllo di flusso:** vi sono elementi di protocollo (messaggi e campi di messaggi) che informano il mittente della disponibilità del ricevente a ricevere nuovi messaggi => il mittente spedisce solo se ha l'assicurazione che il ricevente ha la possibilità di trattare il messaggio (buffer di memoria, tempo di CPU). I ricevente può fare **back-pressure** per rallentare un produttore troppo veloce

- Notare: in assenza di controllo di flusso, messaggi possono essere persi non per errori di comunicazione, ma per mancanza di risorse nel ricevente (es. ethernet: questa è praticamente l'unica ragione per cui si perdono messaggi con ethernet). In genere, se si ha riscontro si ha anche controllo di flusso (per evitare ritrasmissioni inutili), raramente si trova un protocollo con controllo di flusso senza riscontro, o con riscontro ma senza controllo di flusso

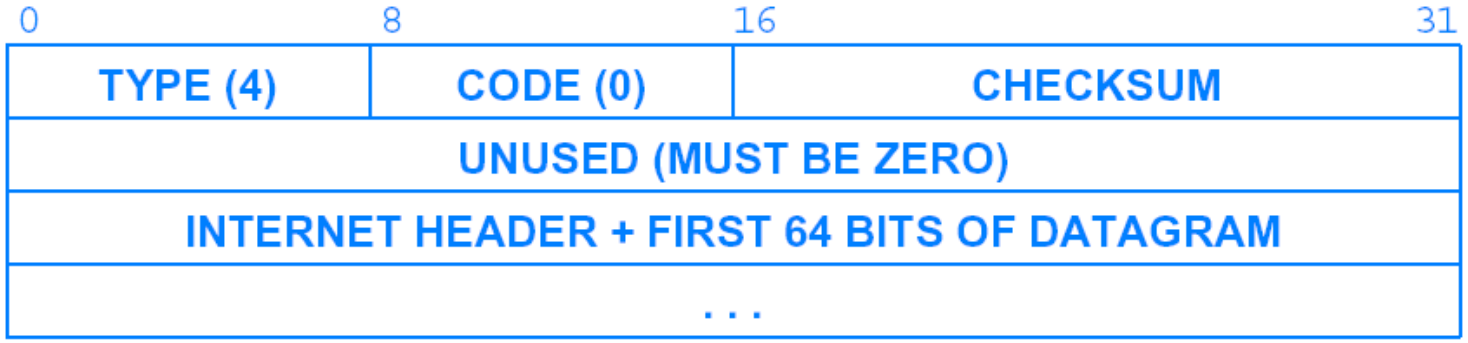
Cosa c'è sotto davvero? (III)

- Da cosa dipende la **quantità** di dati che si riesce ad **inviare** su una sottorete? Quindi la sua **velocità**?
  1. Dalla **velocità di trasmissione** del mezzo fisico: numero di bit al secondo che si riesce a "pompare" dentro al mezzo con il protocollo di accesso (es. ethernet <= 10Mbit/sec)
  2. Dalla **velocità di gestione** hardware/software: i messaggi vanno preparati, prelevati dall'hardware rispondendo alle interruzioni, ... Tutto ciò provoca (può provocare) ritardi e la velocità netta è più bassa di quella del mezzo fisico
  3. **Protocollo di riscontro**: la mancanza di riscontri provoca ritrasmissioni e quindi il traffico realmente smaltito si abbassa
  4. **Protocollo di controllo di flusso**: Se il ricevente è più lento a trattare messaggi del mittente, il mittente viene rallentato, velocità ancora abbassata

Cosa c'è sotto davvero? (IV)

- Cosa succede se il traffico da inviare è più alto di quello che viene assorbito dalla sottorete? IP accoda i messaggi in uscita. Se la situazione di sbilancio di velocità perdura, finiscono i buffer assegnati a quella sottorete (non si possono assegnare tutti i buffer ad una sottorete! Perché?)
- Quando i buffer sono finiti, **si può fare back-pressure sulla sottorete che origina il traffico**? In uno schema connectionless (come è IP) **non si può fermare una sottorete di origine, perché da quella sottorete giungono datagrammi per molte destinazioni**, non solo quella congestionata => quando arriva un datagramma che deve essere instradato verso la sottorete congestionata ***viene scartato***! Solo se il datagramma è stato scartato per congestione in uscita si ha ***ICMP source quench (smorzare la sorgente)***
- **Quando i datagrammi sono persi in lettura ... chi li conosce?**

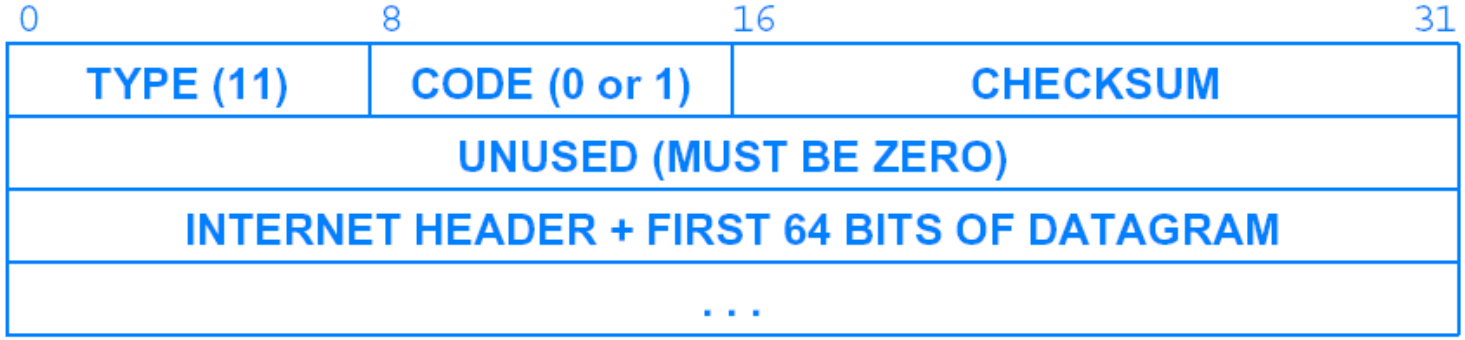
Messaggio di Source Quench



- Normalmente, un messaggio come questo viene inviato alla sorgente ogni volta che si scarta un datagramma per congestione, oppure quando le code cominciano a essere lunghe (**Random Early Discard** = **RED**)
- La sorgente rallenta il tasso di invio dei datagrammi, fino a che non riceve più Source Quench. Poiché non esiste un messaggio che chieda di accelerare, la sorgente dopo un po' rialza il tasso di spedizione "alla cieca"
- Notare che la sorgente smorzata non è necessariamente la causa della congestione! E notare che **la "sorgente" non è IP**: è chi decide di chiedere a IP di spedire: TCP, o l'applicazione che usa UDP (come vedremo)
- **Il messaggio di Source Quench non permette di trattare con precisione il problema del controllo di flusso**

Tempo scaduto

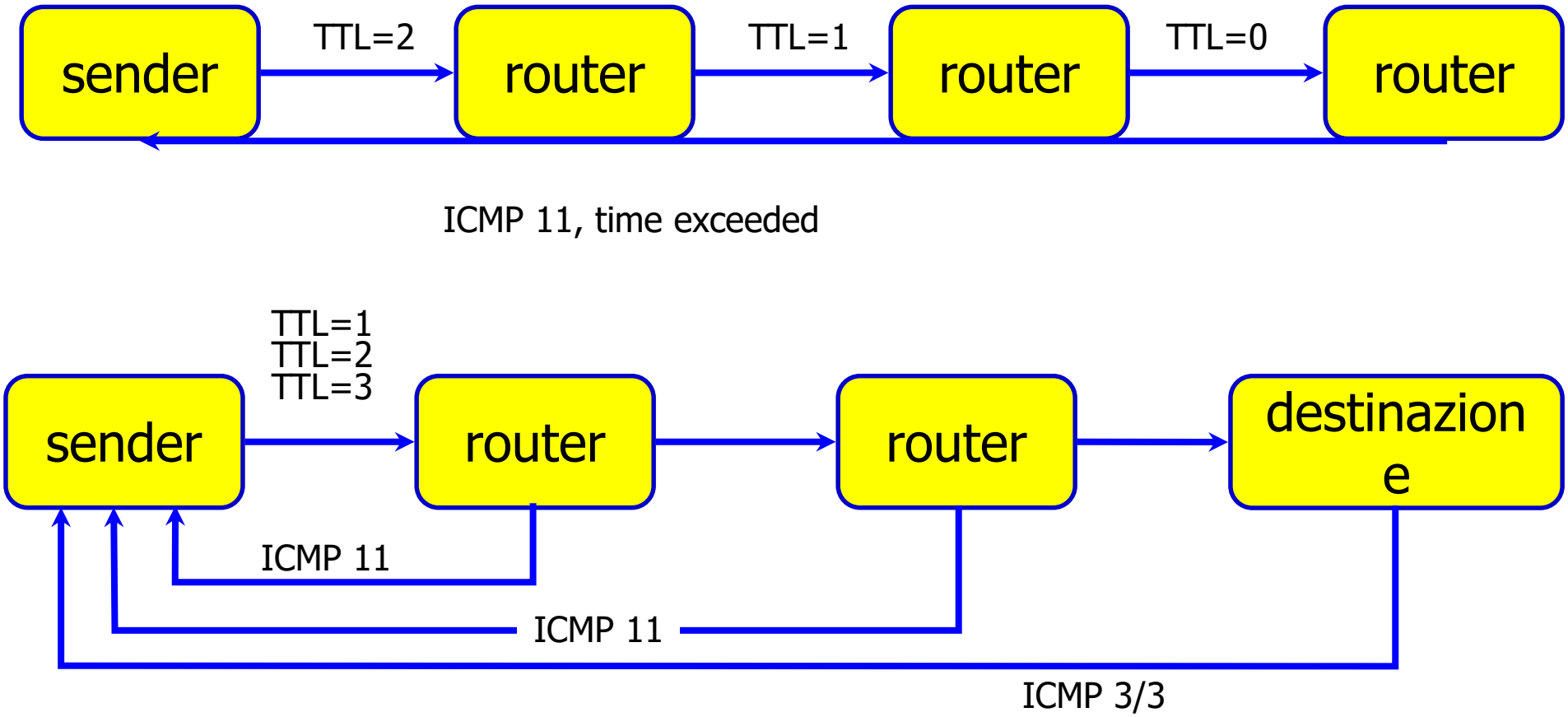
- La protezione da strade eccessivamente lunghe è data dal Time-To-Live. Quando un datagramma viene scartato per Time-To-Live diventato zero, il router invia un messaggio ICMP di ***time exceeded***.
- Un host invia un ***time exceeded*** quando non riesce a riassemblare un datagramma



Code Value	Meaning
0	Time-to-live count exceeded
1	Fragmetnt reassembly time exceeded

Il programma ***traceroute usa TTL crescenti per scoprire i router sul cammino verso una destinazione (vedere traccia sul sito della didattica)***

Time Exceeded e traceroute



Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- forwarding
- DHCP
- network address translation
- error messages (ICMP)
- **IPv6**

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

IPv6: motivation

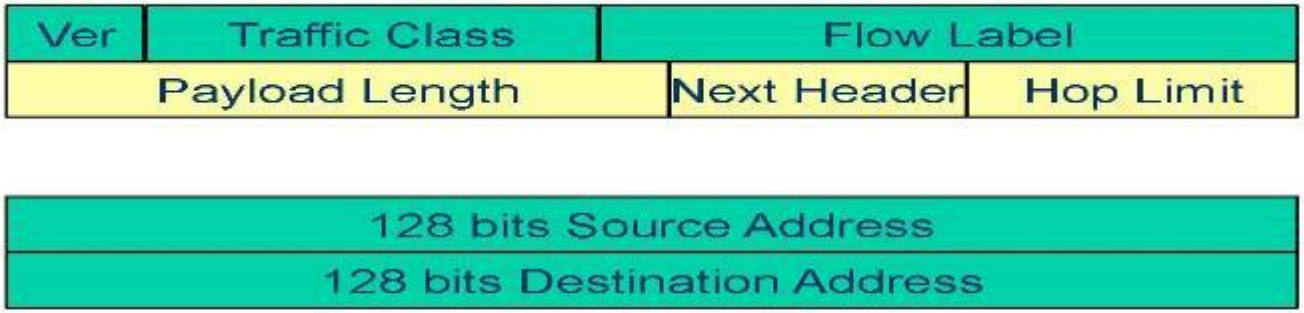
- *initial motivation:* 32-bit address space soon to be completely allocated.
- additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

IPv6 datagram format:

- fixed-length 40 byte header
- no fragmentation allowed

Formato generale di un datagramma

Figura 31.1 La forma generale di un datagramma IPv6 con più intestazioni. Solo l'intestazione di base è obbligatoria, mentre le intestazioni di estensione sono facoltative.



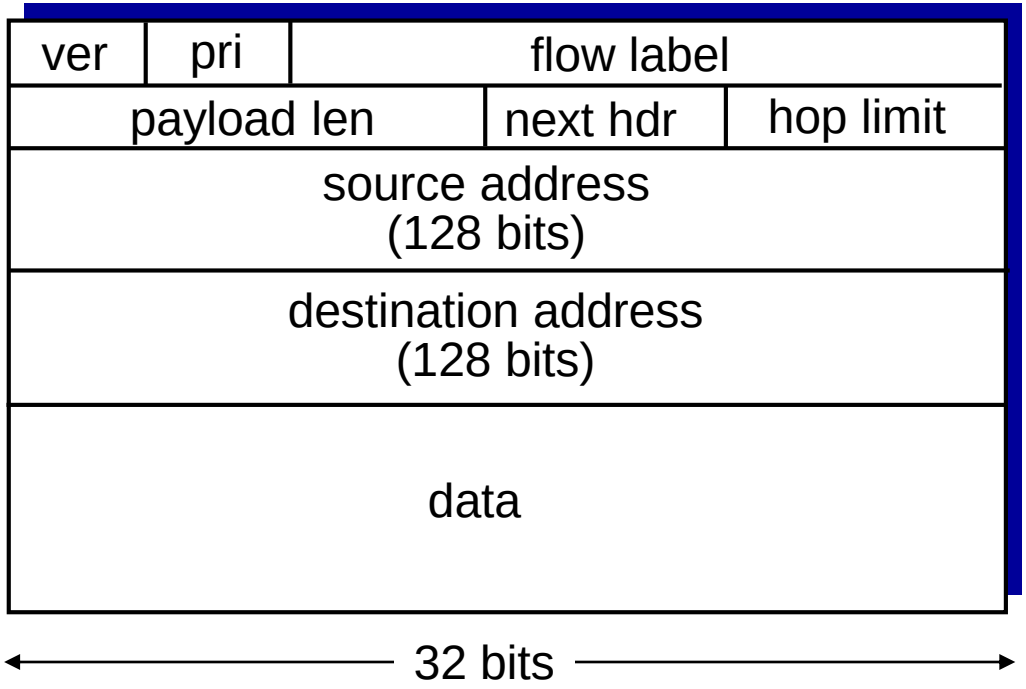
In giallo i campi ereditati da IPv4 ma rinominati

IPv6 datagram format

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



Parsing di un datagramma IPv6

Le intestazioni estensione, che devono essere trattate dai router intermedi (*intestazioni salto-per-salto*), devono precedere quelle che devono essere trattate solo dalla destinazione finale

Nel primo datagramma il campo NEXT HEADER contiene l'identificatore del protocollo TCP: quindi a seguire vi è un segmento TCP

I router già ora con IPv4 devono "capire" almeno parte della intestazione di TCP per applicare le ACL di controllo del traffico

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

Lo spazio di indirizzamento

Gli indirizzi occupano 16 ottetti, quattro volte quelli di IPv4

128 bits → 2^{128} indirizzi possibili (~ 3,4 × 10^{38}).

Sono tanti che ciascun abitante del pianeta di oggi potrebbe indirizzare una sua rete "personale" grande circa quanto l'Internet attuale

Sono tanti che per ogni chilometro quadrato della superficie terrestre (circa $5,1*10^8$) ci sono più di 10^{24} indirizzi (circa una Internet attuale!)

Sono tanti che se assegnassi un **milione** di indirizzi ogni **microsecondo** impiegherei **10^{14}** anni per assegnarli tutti

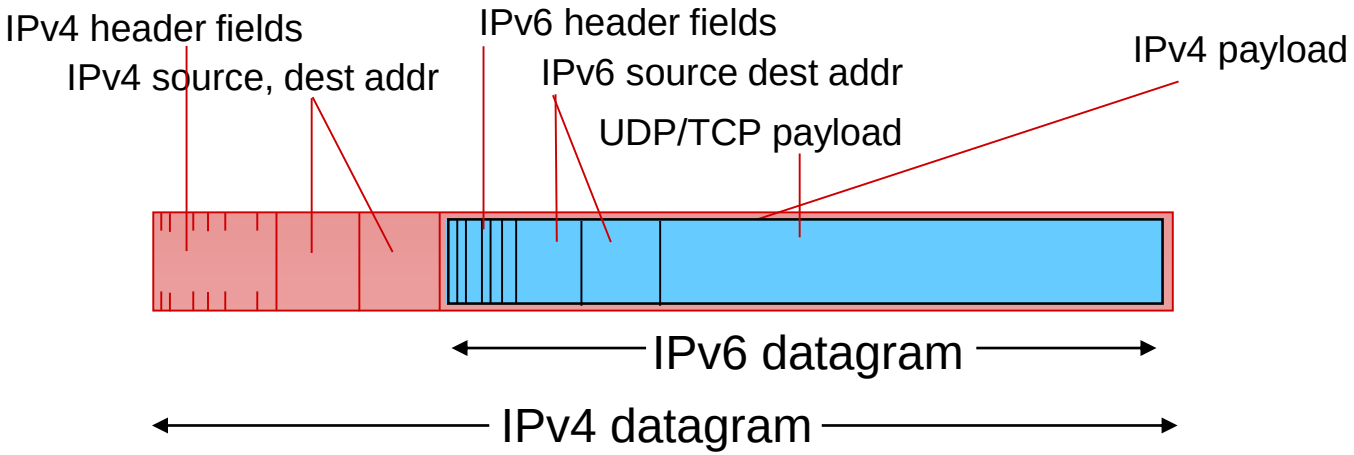
Alcuni sostengono che c'è un indirizzo per ogni atomo dell'universo …

Questo enorme spazio è stato motivato dalla necessità di avere molti schemi di allocazione questi indirizzi e di lasciarsi di riserva ulteriori numerosi schemi di allocazione ancora non inventati

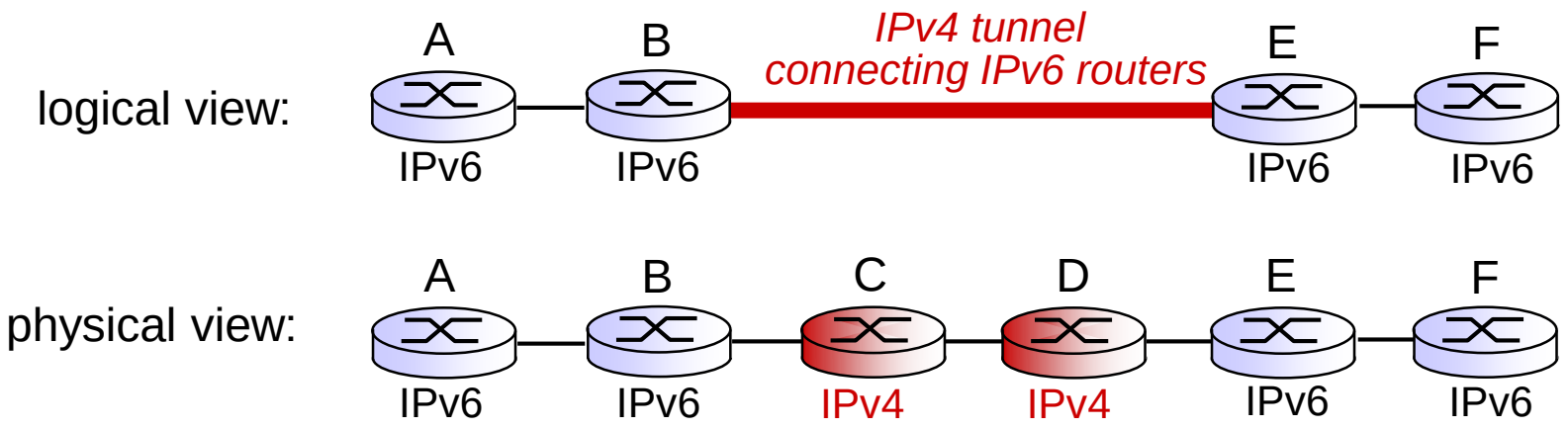
Tutto lo spazio di indirizzi di IPv4 è contenuto in una porzione dello spazio di IPv6: si può esprimere come fosse un indirizzo IPv6

Transition from IPv4 to IPv6

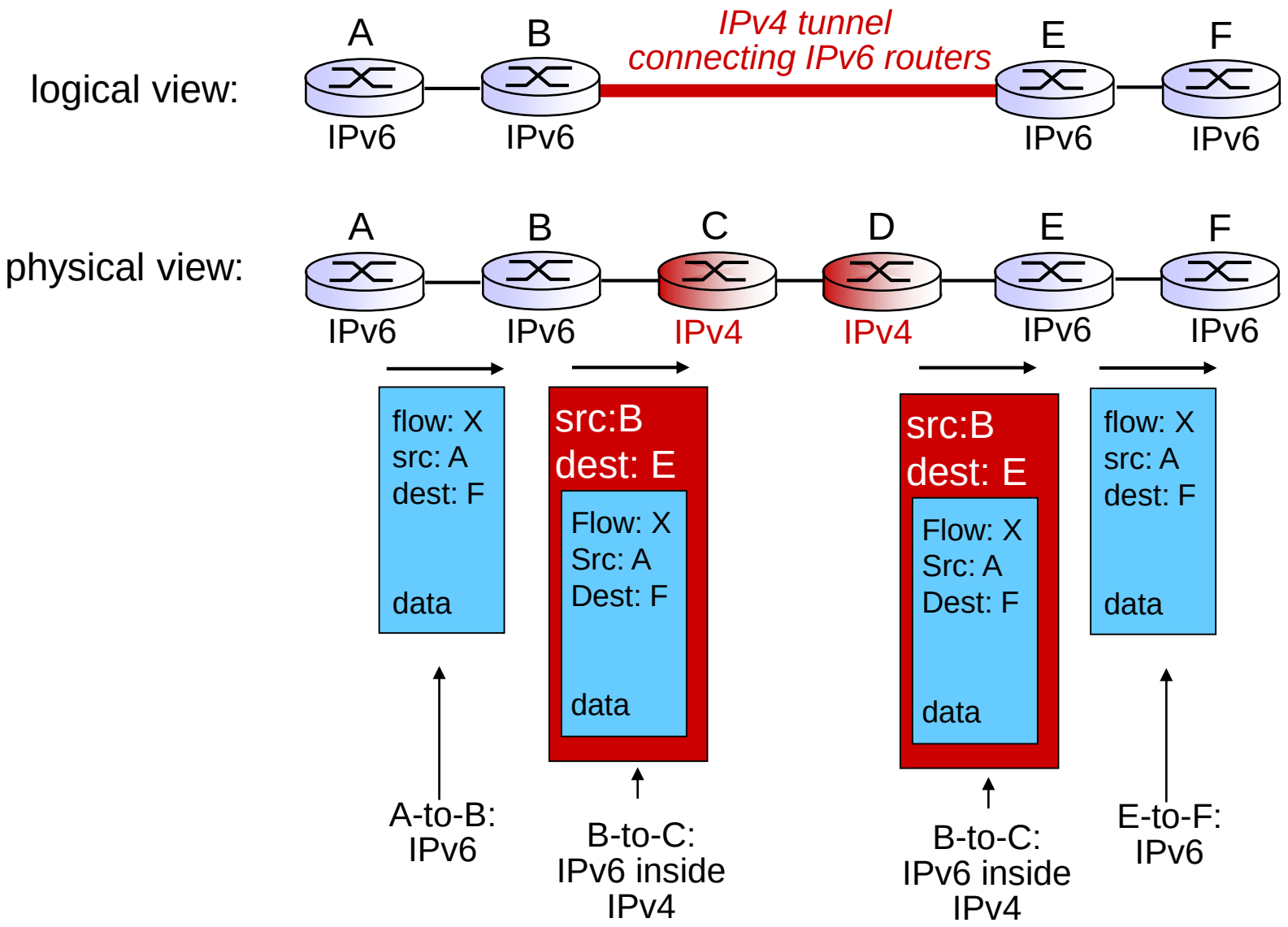
- not all routers can be upgraded simultaneously
  - no "flag days"
  - how will network operate with mixed IPv4 and IPv6 routers?
- *tunneling:* IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers



Tunneling


logical view:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
IPv6
IPv6
IPv6
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6

Tunneling


logical view:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
IPv6
IPv6
IPv6
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6
flow: X
src: A
dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
flow: X
src: A
dest: F
data
A-to-B:
IPv6
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
E-to-F:
IPv6

IPv6: adoption

Google: 8% of clients access services via IPv6

NIST: 1/3 of all US government domains are IPv6 capable

Long (long!) time for deployment, use

- 20 years and counting!
- think of application-level changes in last 20 years: WWW, Facebook, streaming media, Skype, …
- *Why?*

Chapter 4: outline

4.1 4.1 Overview of Network layer
- data plane
- control plane

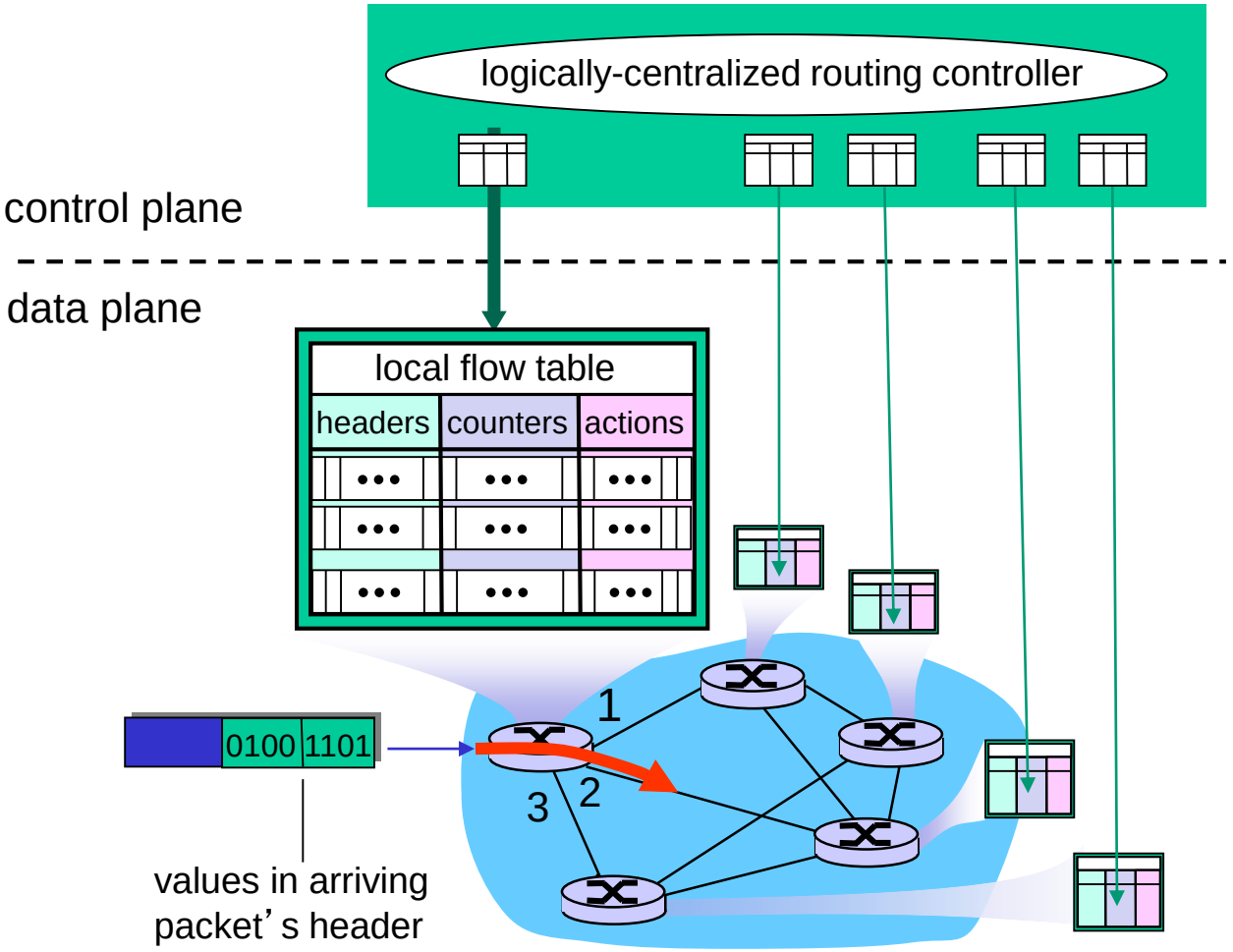
4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- forwarding
- DHCP
- network address translation
- error messages (ICMP)
- IPv6

4.4 Generalized Forward and SDN
- **match**
- **action**
- **OpenFlow examples of match-plus-action in action**

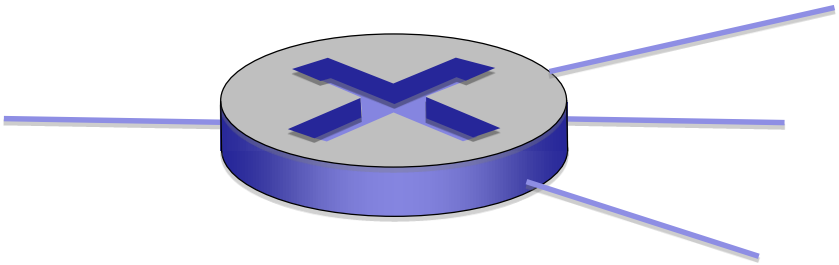
Generalized Forwarding and SDN

Each router contains a *flow table* that is computed and distributed by a *logically centralized* routing controller



OpenFlow data plane abstraction

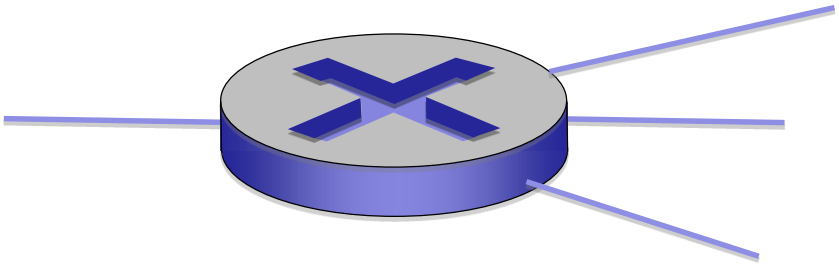
- *flow*: defined by header fields
- generalized forwarding: simple packet-handling rules
  - *Pattern:* match values in packet header fields
  - *Actions: for matched packet:* drop, forward, modify, matched packet or send matched packet to controller
  - *Priority*: disambiguate overlapping patterns
  - *Counters:* #bytes and #packets



Flow table in a router (computed and distributed by controller) define router's match+action rules

OpenFlow data plane abstraction

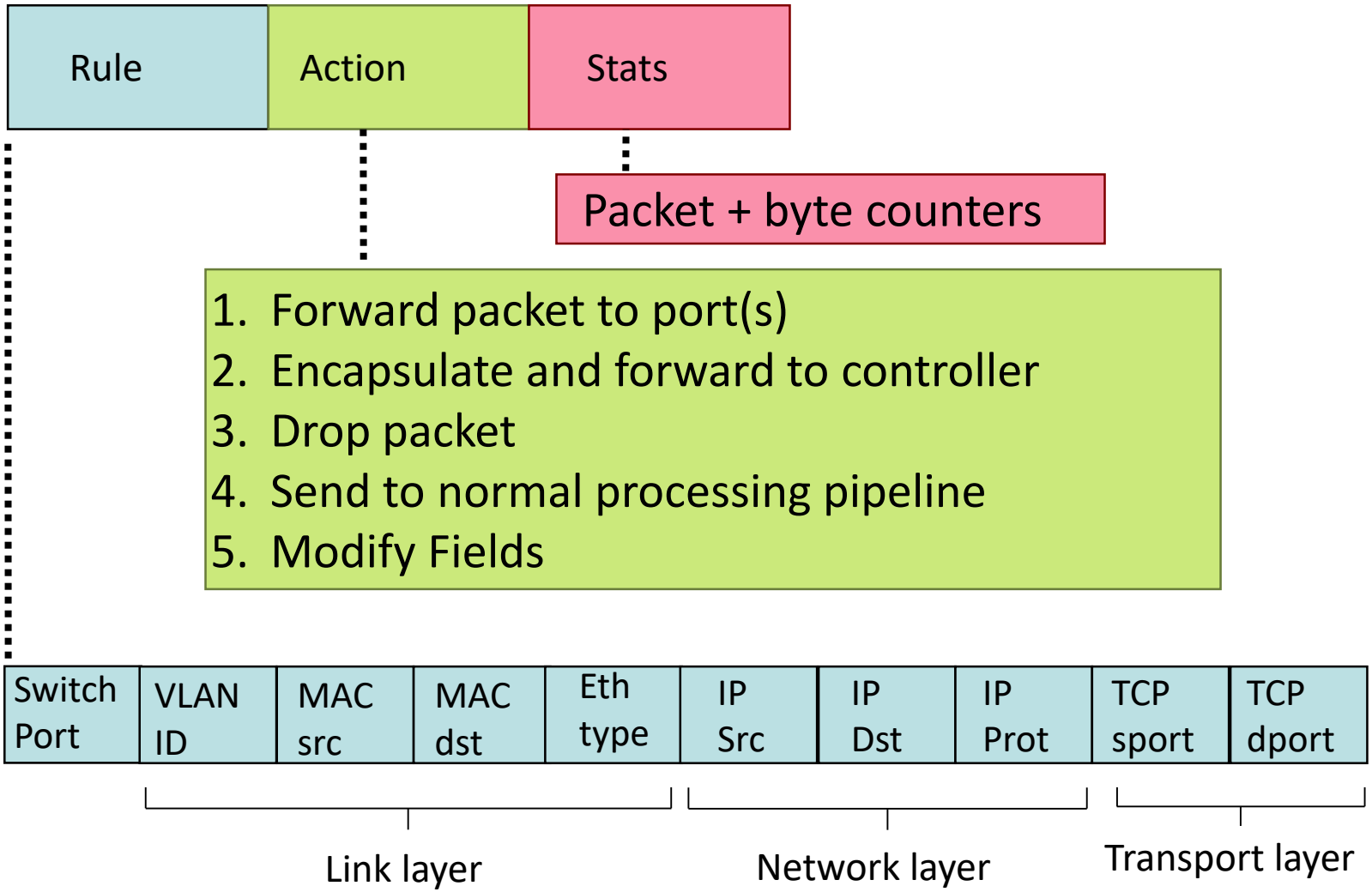
- *flow*: defined by header fields
- generalized forwarding: simple packet-handling rules
  - *Pattern:* match values in packet header fields
  - *Actions: for matched packet:* drop, forward, modify, matched packet or send matched packet to controller
  - *Priority*: disambiguate overlapping patterns
  - *Counters:* #bytes and #packets



* : wildcard

1. src=1.2.*.*, dest=3.4.5.* → drop
2. src = *.*.*.*, dest=3.4.*.* → forward(2)
3. src=10.1.2.3, dest=*.*.*.* → send to controller

OpenFlow: Flow Table Entries



Examples

Destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	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	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	*	*	*	*	22	drop

do not forward (block) all datagrams destined to TCP port 22

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	128.119.1.1	*	*	*	*	drop

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

Examples

Destination-based layer 2 (switch) forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	22:A7:23: 11:E1:02	*	*	*	*	*	*	*	*	port3

layer 2 frames from MAC address 22:A7:23:11:E1:02 should be forwarded to output port 6

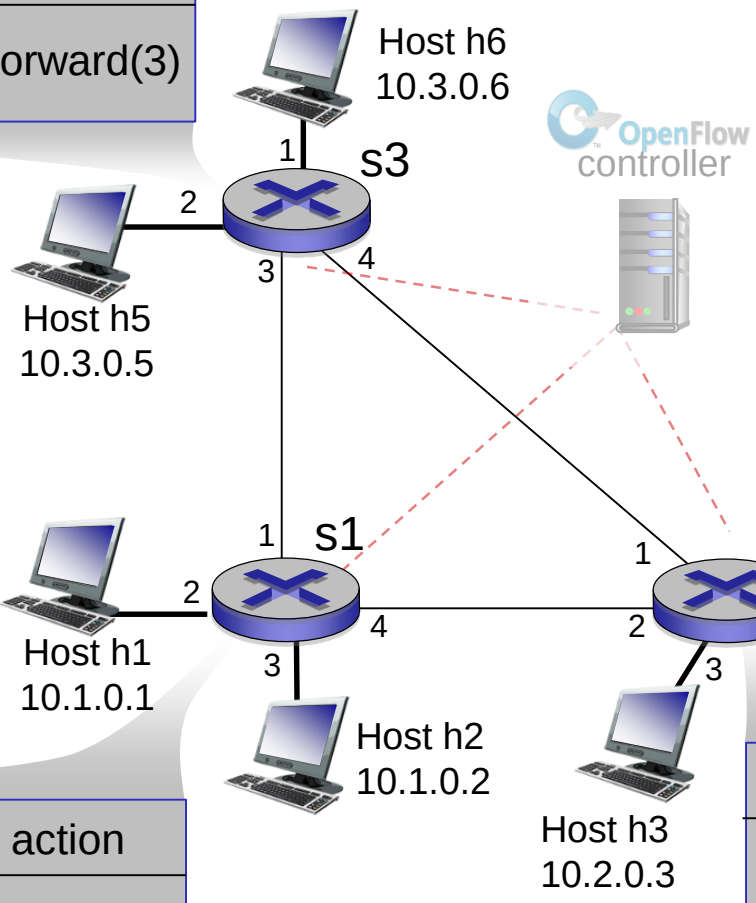
OpenFlow abstraction

match+action: 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

match	action
IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(3)



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

match	action
ingress port = 1 IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(4)

match	action
ingress port = 2 IP Dst = 10.2.0.3	forward(3)
ingress port = 2 IP Dst = 10.2.0.4	forward(4)

Chapter 4: outline

4.1 4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- forwarding
- DHCP
- network address translation
- error messages (ICMP)
- IPv6

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

Question: how do forwarding tables (destination-based forwarding) or flow tables (generalized forwarding) computed?

Answer: by the control plane (next chapter)