

# **Reti di Calcolatori**

Ahmad Shatti

2021-2022

# Indice

<b>1 Reti e modelli stratificati</b>	<b>3</b>
1.1 Reti . . . . .	3
1.2 Commutazione . . . . .	4
1.3 Internet . . . . .	4
1.4 Metriche . . . . .	5
1.5 Modelli stratificati . . . . .	6
1.5.1 Open System Interconnection . . . . .	6
1.6 Stack protocolare TCP/IP . . . . .	7
1.6.1 Incapsulamento e decapsulamento . . . . .	7
1.7 Esercizi . . . . .	8
<b>2 Livello applicazione</b>	<b>11</b>
2.1 Applicazioni di rete . . . . .	11
2.2 API e protocolli TCP/UDP . . . . .	11
2.3 Web . . . . .	12
2.4 Protocollo HTTP . . . . .	13
2.4.1 Messaggi HTTP . . . . .	13
2.4.2 Metodi HTTP . . . . .	15
2.5 Web caching . . . . .	15
2.6 TELNET . . . . .	16
2.7 Posta elettronica . . . . .	16
2.7.1 Protocollo SMTP . . . . .	17
2.7.2 Format dei messaggi . . . . .	18
2.7.3 Protocolli POP e IMAP . . . . .	18
2.8 DNS . . . . .	19
2.8.1 Spazio dei nomi . . . . .	19
2.8.2 Record e messaggi DNS . . . . .	21
2.9 Protocollo FTP . . . . .	22
2.10 Esercizi . . . . .	23
<b>3 Livello trasporto</b>	<b>30</b>
3.1 Introduzione . . . . .	30
3.2 I servizi del livello trasporto . . . . .	30
3.3 TCP . . . . .	30
3.3.1 Formato segmento TCP . . . . .	31
3.3.2 La connessione TCP . . . . .	32
3.3.3 Trasferimento dati . . . . .	33
3.3.4 Controllo di flusso . . . . .	35
3.3.5 Controllo di congestione . . . . .	35
3.3.6 TCP Reno . . . . .	35
3.3.7 TCP Tahoe . . . . .	36
3.3.8 TCP: throughput . . . . .	36
3.4 UDP . . . . .	36
3.4.1 Checksum UDP . . . . .	36
3.5 TCP vs UDP . . . . .	37
3.6 Esercizi . . . . .	37

<b>4 Livello di rete</b>	<b>42</b>
4.1 Comunicazione e servizi . . . . .	42
4.2 Internet Protocol . . . . .	42
4.2.1 Formato datagramma IP . . . . .	42
4.2.2 Frammentazione . . . . .	43
4.2.3 Indirizzamento IP . . . . .	44
4.2.4 Assegnamento dei blocchi di indirizzi . . . . .	45
4.2.5 Aggregazione degli indirizzi . . . . .	46
4.2.6 Dynamic Host Configuration Protocol . . . . .	46
4.2.7 Inoltro dei datagrammi IP . . . . .	47
4.2.8 Network Address Translation . . . . .	49
4.3 Internet Control Message Protocol . . . . .	50
4.3.1 Tracerout . . . . .	50
4.4 Data plane e control plane . . . . .	51
4.4.1 Componenti router . . . . .	52
4.5 Routing . . . . .	52
4.5.1 Routing gerarchico . . . . .	54
4.6 IPv6 . . . . .	56
4.7 Esercizi . . . . .	56
<b>5 Livello collegamento</b>	<b>61</b>
5.1 Introduzione . . . . .	61
5.2 Data-link control . . . . .	61
5.3 Protocollo ad accesso multiplo . . . . .	61
5.4 Indirizzi a livello collegamento . . . . .	62
5.4.1 Address Resolution Protocol . . . . .	62
5.5 Ethernet . . . . .	63
5.6 Dispositivi di interconnessione . . . . .	64
5.7 Esercizi . . . . .	65
<b>6 VLAN, e cenni di applicazioni P2P e sicurezza</b>	<b>66</b>
6.1 VLAN . . . . .	66
6.2 Paradigma peer-to-peer . . . . .	67
6.2.1 BitTorrent . . . . .	67
6.3 Sicurezza di rete . . . . .	68
6.3.1 Secure Sockets Layer . . . . .	69
6.3.2 Sicurezza a livello rete: IPSec . . . . .	69

# Chapter 1

## Reti e modelli stratificati

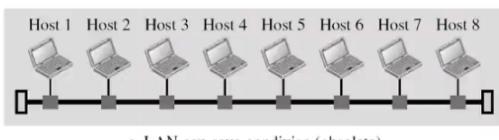
### 1.1 Reti

Una rete è una interconnessione di dispositivi in grado di scambiarsi informazioni. Si indica con **hosts** i sistemi terminali, cioè i sistemi che risiedono alla periferia della rete e che usano la rete per scopi applicativi, trasmissione di contenuti, streaming di file, ... Per far comunicare gli hosts tra di loro occorrono le tecnologie di comunicazione che prendono il nome di **link** o **collegamenti** (fibra ottica, rame, onde radio, satellite). Non si ha un collegamento fisico per ogni coppia di hosts, ma si ha bisogno di **dispositivi di interconnessione** che permettono a più flussi di traffico, da una estremità all'altra, di condividere i collegamenti di comunicazione. Essi si distinguono in:

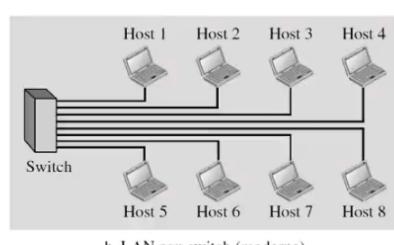
- **Router**, sono dispositivi che collegano una rete ad altre reti
- **Switch**, collegano fra di loro più sistemi terminali a livello locale

Una rete può essere classificata in base all'estensione geografica:

- **Local Area Network (LAN)** o rete locale, sono reti di computer circoscritte ad un'area limitata (ufficio, scuola, edificio ...) con un'estensione che arriva fino a qualche km e sono di proprietà, tipicamente, di una organizzazione privata. Le tecnologie possono essere un cavo di rame, un cavo ethernet, wireless... Qualsiasi sistema terminale in una LAN deve avere un indirizzo che lo identifica univocamente nella rete. Un pacchetto inviato da un sistema terminale ad un altro contiene entrambi gli indirizzi di mittente e destinatario. In passato tutti i dispositivi di una rete locale erano collegati mediante un cavo condiviso per cui il pacchetto inviato da un dispositivo veniva ricevuto da tutti gli altri. Il destinatario elaborava il pacchetto, mentre gli altri lo dovevano ignorare. Oggi la maggior parte delle LAN utilizza uno switch di interconnessione, al quale ogni dispositivo in rete è direttamente connesso. Lo switch è in grado di riconoscere l'indirizzo di destinazione e inviare quindi il pacchetto al solo destinatario senza inviarlo ad altri dispositivi. Quindi lo switch riduce il traffico nella LAN.



a. LAN con cavo condiviso (obsoleta)



b. LAN con switch (moderna)

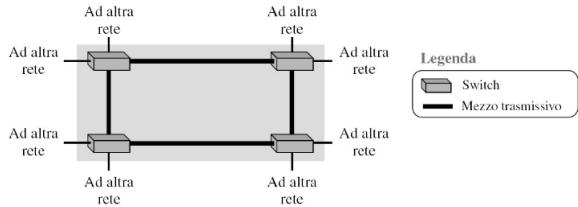


- **Wide Area Network (WAN)** o rete geografica, è un'interconnessione di dispositivi in grado di comunicare, ma a differenza delle LAN, una WAN ha una estensione superiore potendo servire una città, una regione, una nazione o addirittura tutto il globo. Una WAN è generalmente creata e gestita da un operatore di telecomunicazioni che fornisce i suoi servizi alle organizzazioni che ne fanno parte. Oggi esistono due tipi di WAN:

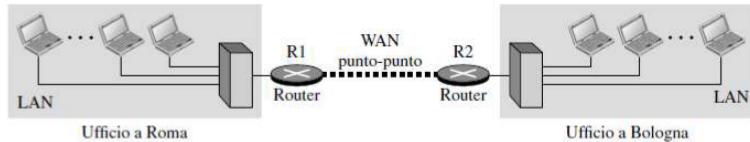
- **WAN punto - punto**, è una rete che collega due dispositivi tramite un mezzo trasmissivo (cavo o wireless)



- **WAN a commutazione**, è una rete con più di due punti di terminazione



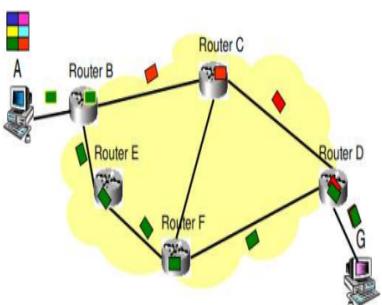
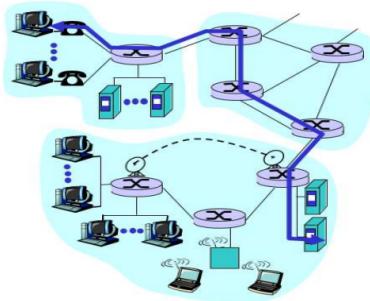
Al giorno d'oggi è raro vedere LAN o WAN isolate: esse sono in genere connesse fra di loro, formando una *internet-work* o *internet*.



## 1.2 Commutazione

I sistemi terminali comunicano tra di loro per mezzo di dispositivi come switch e router che si trovano nel percorso tra il sistema sorgente e destinazione. La **tecnica di commutazione** è la modalità con cui viene determinato e gestito il percorso sorgente-destinazione, e come vengono dedicate a questo percorso le risorse di rete. Esistono due approcci:

- **Reti a commutazione di circuito**, è presente una fase di **setup della connessione** per instaurare un cammino dedicato tra i due dispositivi che vogliono comunicare: una serie di nodi di commutazione che collegano sorgente e destinazione. Sul percorso vengono dedicate risorse alla comunicazione per tutta la durata della stessa. Il vantaggio è che le prestazioni sono garantite. Gli svantaggi sono che è necessaria una fase di setup della comunicazione e le risorse rimangono inattive se non utilizzate (ad esempio silenzi durante una conversazione telefonica).
- **Reti a commutazione di pacchetto**, il flusso di dati viene suddiviso in pacchetti. I pacchetti condividono le risorse di rete e ogni pacchetto è instradato singolarmente e indipendentemente dagli altri pacchetti della stessa comunicazione (possono seguire lo stesso percorso o percorsi diversi). Un router in una rete a commutazione di pacchetto possiede una coda dove possono essere memorizzati i pacchetti prima di essere inoltrati. Se i pacchetti arrivano in un router quando la linea di comunicazione fra i due router è già utilizzata alla sua massima capacità, i pacchetti devono essere memorizzati per essere successivamente inoltrati nell'ordine di arrivo, quindi possono incorrere in qualche ritardo. Si possono verificare situazioni di perdita di pacchetti quando la dimensione della coda del router non è sufficientemente grande a contenere il flusso dei pacchetti entranti. Vi è un utilizzo efficiente delle risorse, perché esse vengono usate a seconda della necessità, ma non c'è garanzia nelle prestazioni (ad esempio i ritardi)



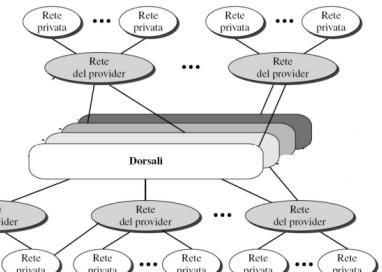
La commutazione di circuito preallocata l'utilizzo del collegamento trasmissivo con collegamenti garantiti, mentre, nella commutazione di pacchetto, pacchetto dopo pacchetto, la capacità trasmissiva dei collegamenti sarà condivisa solo tra gli utenti che devono trasmettere sul collegamento.

## 1.3 Internet

Internet è una rete di reti, ovvero una interconnessioni di reti. È un sistema complesso a commutazione di pacchetto in cui ci possono essere miliardi di dispositivi connessi. Una **internet** è costituita da due o più reti interconnesse. L'internet più famosa è chiamata **Internet** ed è composta da migliaia di reti interconnesse. Ogni rete connessa a

Internet deve usare l'**Internet Protocol (IP)** e rispettare certe convenzioni su nomi e indirizzi. Le reti degli host sono connessi a Internet attraverso una gerarchia di fornitori di servizi Internet (Internet Service Provider):

- **Dorsali** sono al livello più alto e sono reti particolarmente estese di proprietà di qualche compagnia telefonica. Le reti dorsali sono interconnesse tramite sistemi di commutazione notevolmente complessi, chiamati **peering point**. Spesso chiamate *ISP internazionali*;
- **Reti di provider** sono al secondo livello, sono reti più piccole che utilizzano a pagamento i servizi delle dorsali e sono connesse, oltre ai dorsali, a volte anche ad altre reti di provider. Spesso chiamate *ISP nazionali o regionali*;
- **Reti private** sono reti ai confini di Internet che utilizzano i servizi a pagamento forniti dalle reti dei provider.



Internet è un internetwork che consente a qualsiasi utente di farne parte. L'utente, tuttavia, deve essere fisicamente collegato a un ISP, solitamente mediante una WAN punto-punto. Il collegamento che connette l'utente al primo router di internet è detto **rete di accesso**. Gli standard internet e del Web:

- **Internet Engineering Task Force (IETF)** è l'organismo che studia e sviluppa i protocolli in uso su Internet. Si basa su gruppi di lavoro a cui chiunque può accedere
- **Internet Corporation for Assigned Names and Numbers (ICANN)** coordina il sistema dei nomi di dominio (DNS), assegna i gruppi di indirizzo di rete, identificativi di protocollo e ha funzioni di controllo (blando) dello sviluppo di Internet
- **World Wide Web Consortium (W3C)** comunità internazionale che sviluppa standard aperti per favorire lo sviluppo del Web

## 1.4 Metriche

Come misurare le prestazioni della rete:

- **Larghezza di banda**, è una quantità che si misura in *hertz* e rappresenta l'intervallo di frequenze che un mezzo fisico consente di trasmettere senza danneggiare il segnale in maniera irrecuperabile. Maggiore è l'ampiezza di banda, maggiore è la quantità di informazione che può essere veicolata attraverso il mezzo trasmisivo
- **Velocità di trasmissione o bit rate**, quantità di bit al secondo che un certo link garantisce di trasmettere. Dipende dalla tecnica trasmittiva ed è proporzionale alla larghezza di banda
- **Throughput**, indica la velocità con cui si trasferiscono i dati al netto di perdite sulla rete, protocolli,... dipende dalla velocità di trasmissione del collegamento e dalla quantità di dati
- **Latenza**, è il tempo richiesto affinché un messaggio arrivi a destinazione dal momento in cui il primo bit parte dalla sorgente. La latenza, o **ritardo totale**, deriva dalla seguente formula:

$$\text{Latenza} = \text{ritardo propagazione} + \text{ritardo trasmissione} + \text{ritardo accodamento} + \text{ritardo elaborazione}$$

- **ritardo di propagazione**, è il tempo che serve ad un bit per viaggiare dal punto A al punto B nel mezzo di trasmissione. Il ritardo di propagazione dipende dalla velocità di propagazione del mezzo (tipicamente data dalla velocità della luce) e dalla lunghezza del link

$$d = \text{lunghezza (distanza) del collegamento fisico}$$

$$s = \text{velocità di propagazione del collegamento fisico}$$

$$r_{pr} = \frac{d}{s}$$

- **ritardo di trasmissione**, è il tempo impiegato per trasmettere un pacchetto sul link. Chiaramente il ritardo di trasmissione è maggiore per un pacchetto più lungo e minore se il mittente riesce a trasmettere più velocemente.

$$L = \text{lunghezza del pacchetto}$$

$$R = \text{rate di trasmissione del collegamento}$$

$$r_{tr} = \frac{L}{R}$$

- **ritardo di accodamento**, è il tempo che un pacchetto passa nella coda del router. Dipende dalla dimensione del buffer, e dall'intensità e dal tipo di traffico.

- **ritardo di elaborazione**, è il tempo che serve a un sistema terminale per ricevere un pacchetto dalla sua porta di input, rimuovere l'intestazione del pacchetto, eseguire una procedura di rilevamento errori e consegnare il pacchetto alla porta di output
- **Prodotto rate-ritardo**, si può pensare al link tra due punti come a un tubo. La sezione trasversale del tubo rappresenta il rate e la lunghezza rappresenta il ritardo, si può dire che il volume del tubo definisce il prodotto rate-ritardo, ovvero il numero massimo di bit che il link può contenere

## 1.5 Modelli stratificati

Il modello usato nei sistemi di comunicazione è il **modello stratificato** che è un sistema organizzato in strati funzionali. Uno **strato** (o livello) è un modulo interamente definito attraverso servizi, interfacce e protocolli che lo caratterizzano, e svolge una sola e ben definita funzione. Uno strato fornisce servizi allo strato superiore, riceve servizi da quello inferiore, quindi gli strati comunicano solo con altri strati a loro adiacenti attraverso un'interfaccia. Il flusso dati attraverso le interfacce di ogni strato deve essere minimizzato. Il numero degli strati deve essere sufficientemente alto per garantire che nessun livello sia troppo complesso e contenga troppe funzioni, ma anche sufficientemente basso per non rendere troppo onerosa l'integrazione e l'architettura poco flessibile. La stratificazione permette di scomporre il problema in sottoproblemi più semplici da trattare, rende i vari livelli indipendenti, facilita la manutenzione e l'aggiornamento del sistema. I principi base della stratificazione sono:

- **separation of concern**, separazione degli interessi e delle responsabilità, ovvero fare ciò che compete e delegare ad altri tutto ciò che è delegabile
- **information hiding**, nascondere tutte le informazioni che non sono indispensabili per il committente per definire compiutamente un'operazione

Un **servizio** è un insieme di primitive che uno strato fornisce ad uno strato soprastante. Un **interfaccia** è un insieme di regole che governano il formato e il significato delle unità di dati che vengono scambiati tra due strati adiacenti della stessa unità. Un **protocollo** è un insieme di regole che

- permettono a due entità dello stesso strato (omologhe) uno scambio efficace ed efficiente delle informazioni. Efficace: riesce a raggiungere lo scopo prefissato con la maggior frequenza possibile. Efficiente: riesce a raggiungere lo scopo prefissato con il minor sforzo possibile
- definiscono il formato e l'ordine dei messaggi inviati e ricevuti tra entità omologhe della rete e le azioni che vengono fatte per la trasmissione e ricezione dei messaggi

In un protocollo vanno specificati la *sintassi* di un messaggio (formato e campi), la *semantica* (significato) e le azioni da intraprendere dopo la ricezione di un messaggio

### 1.5.1 Open System Interconnection

Le prime reti di calcolatori nacquero come sistemi chiusi in cui tutti i componenti dovevano essere dello stesso costruttore e quindi vi erano problemi di interoperabilità. I **sistemi aperti** nascono con l'obiettivo di realizzare una rete di calcolatori che consente a due terminali qualsiasi di comunicare indipendentemente dalla loro architettura sottostante. Per realizzare un sistema aperto è necessario stabilire degli standard. Un set di protocolli è aperto se:

- i dettagli del protocollo sono disponibili pubblicamente
- i cambiamenti sono gestiti da un'organizzazione la cui partecipazione è aperta al pubblico

Un sistema che implementa protocolli aperti è un sistema aperto. L'International Organization for Standards (**ISO**) ha specificato uno standard per l'interconnessione di sistemi aperti il cui modello di riferimento è l'Open System Interconnection (**OSI**). Il modello ISO/OSI prevede di dividere le funzionalità del protocollo di telecomunicazione in strati ognuno dei quali svolge una parte piccola e indipendente dalle altre. La comunicazione tra i vari livelli è assicurata da chiamate standard in cui ogni livello è tenuto a rispondere in maniera corretta alle chiamate che gli competono e che verranno generate dal livello superiore e inferiore adiacente. È composto da 7 livelli:

7. **Applicazione**: elaborazione dei dati
6. **Presentazione**: unificazione dei dati
5. **Sessione**: controllo del dialogo tra l'host sorgente e destinazione

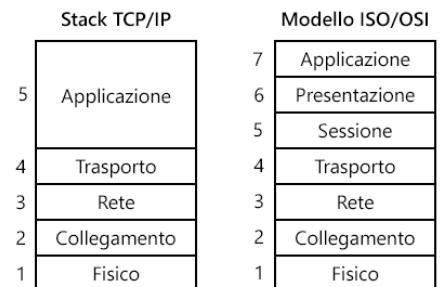
4. **Trasporto:** si occupa di trasferire i dati da un sistema terminale all'altro
3. **Rete:** si occupa dell'instradamento del traffico, ovvero del percorso da sorgente a destinazione attraverso i nodi della rete
2. **Datalink:** si occupa di inviare un unità informativa (frame) da un nodo al nodo successivo
1. **Fisico:** per trasmettere i bit che compongono un certo messaggio su un collegamento

L'informazione ha origine al livello applicativo e discende i vari livelli fino alla trasmissione sul canale fisico. Ogni livello aggiunge all'informazione del livello superiore una propria sezione informativa ed esegue una operazione di incapsulamento su dati già incapsulati dal livello precedente. Per i dati ricevuti si segue il cammino inverso. La definizione dell'incapsulamento è tale da garantire la possibilità di estrarre i dati precedentemente incapsulati.

## 1.6 Stack protocollare TCP/IP

TCP/IP è una famiglia di protocolli attualmente utilizzata in Internet. Si tratta di una gerarchia di protocolli, ciascuno dei quali fornisce funzionalità specifiche. Il termine *gerarchia* significa che ciascun protocollo di livello superiore è supportato dai servizi forniti dai protocolli di livello inferiore. Se un protocollo viene eliminato da un livello, è necessario apportare le modifiche necessarie ai protocolli del livello immediatamente superiore, che utilizzavano i servizi del protocollo rimosso. La pila TCP/IP è oggi intesa come composta di cinque livelli:

5. **Applicazione:** supporta le applicazioni di rete
4. **Trasporto:** supporta il trasferimento di dati da un host all'altro
3. **Rete:** instrada i dati da sorgente alla destinazione attraverso un certo percorso
2. **Link:** trasferimento dei dati in frame attraverso il collegamento tra elementi di rete vicini
1. **Fisico:** trasferimenti dei bit di un frame sul mezzo trasmissivo



*Differenze con ISO/OSI:* il modello ISO/OSI è più *generale*, cioè mette d'accordo più aziende, ed essendo uno standard è più rigoroso. Alcuni livelli del modello ISO/OSI, come presentazione e sessione, non sono mai stati completamente specificati: il software corrispondente non è dunque mai stato completamente sviluppato. L'OSI venne pubblicato quando il TCP/IP era già ampiamente diffuso e gli si erano già dedicate notevoli risorse: un'eventuale sostituzione avrebbe rappresentato un costo notevole. Quando alcune organizzazioni implementarono l'OSI in qualche applicazione, non riuscirono a dimostrare delle prestazioni tali da convincere le autorità di Internet a sostituire il TCP/IP con il modello OSI. Il TCP/IP non è un modello generale ma è uno *standard de facto*, cioè è ampiamente utilizzato anche se non è ufficialmente uno standard, ed è guidata dall'implementazione.

### 1.6.1 Incapsulamento e decapsulamento

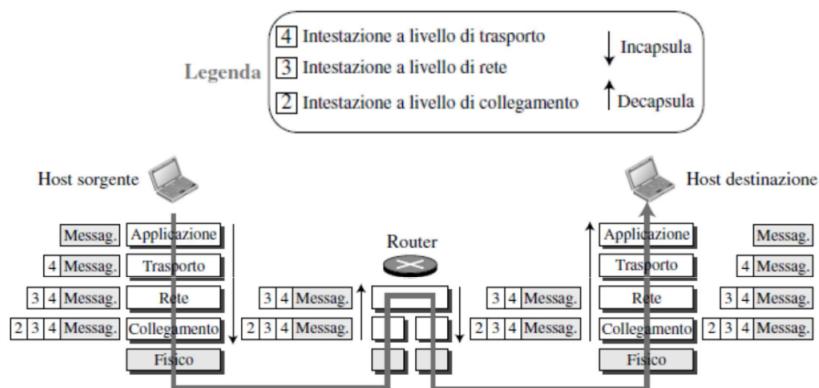
**Incapsulamento nell'host sorgente:** la sorgente effettua solo l'incapsulamento

1. A livello applicazione i dati da scambiare vengono chiamati **messaggi**. Un messaggio solitamente non contiene alcuna intestazione (**header**) o informazioni di controllo (**trailer**). Il messaggio viene passato al livello trasporto
2. Il livello trasporto considera il messaggio ricevuto come **payload**, ovvero come carico dati, e gli aggiunge l'intestazione di livello trasporto che contiene varie informazioni per la gestione del pacchetto. Questo procedimento viene detto incapsulamento. Al livello trasporto il risultato dell'incapsulamento è un **segmento**. Il livello trasporto passa quindi il pacchetto al livello rete
3. Il livello rete aggiunge un'intestazione al pacchetto passatogli dal livello trasporto. L'intestazione contiene gli indirizzi degli host sorgente e destinazione e altre informazioni per il controllo degli errori dell'intestazione stessa. Il risultato è un pacchetto di livello rete, chiamato **datagramma**, che viene dunque passato al livello di collegamento
4. Il livello di collegamento considera il pacchetto di livello rete come un payload e gli aggiunge la propria intestazione, contiene l'indirizzo del livello di collegamento dell'host o del router. Il risultato è un pacchetto di livello di collegamento chiamato **frame** che viene quindi passato al livello fisico per la trasmissione

**Incapsulamento e decapsulamento nel router:** si effettua sia l'incapsulamento che il decapsulamento poiché questi è collegato a due o più link.

1. L'insieme di bit ricevuti dal livello fisico viene passato al livello collegamento, che decapsula il frame (ovvero rimuove l'header) e passa il datagramma al livello di rete
2. Il livello di rete consulta solamente gli indirizzi di sorgente e destinatario nell'header del datagramma e consulta la propria tabella di instradamento per decidere il router al quale consegnare il datagramma. Il contenuto del datagramma non deve essere alterato dal livello rete nel router, a meno che non si renda necessario frammettarlo nel caso sia di dimensioni eccessive per essere trasmesso attraverso il link individuato. Il datagramma viene quindi passato al livello di collegamento del link interessato
3. Il livello di collegamento incapsula quindi il datagramma in un frame e lo passa al livello fisico per la trasmissione

**Decapsulamento nell'host destinatario:** ciascun livello nell'host destinatario rimuove la propria intestazione del pacchetto ricevuto estraendone il payload. Questo viene passato al protocollo del livello sovrastante, fino a quando il messaggio raggiunge il livello applicazione. Il decapsulamento nell'host comporta anche il controllo degli errori.



## 1.7 Esercizi

**Esercizio 1.** Una rete che collega i computer in un ufficio viene tipicamente chiamata

- nessuna delle opzioni
- LAN
- WAN
- MAN

**Esercizio 2.** L'interconnessione di due o più reti si chiama

- nessuna delle opzioni
- LAN
- MAN
- internet

**Esercizio 3.** Il ritardo di propagazione dipende da:

- capacità dei buffer dei router
- lunghezza del collegamento
- larghezza di banda
- velocità di trasmissione

**Esercizio 4.** Il ritardo di trasmissione dipende da:

- lunghezza del collegamento
- capacità dei buffer dei router

- lunghezza in bit del pacchetto
- velocità di trasmissione

**Esercizio 5.** Quando in un router la velocità dei pacchetti in arrivo supera la velocità di trasmissione sul collegamento in uscita si può verificare un ritardo di:

- propagazione
- accodamento, poiché il router non fa in tempo a smaltire i pacchetti che arrivano
- trasmissione
- nessuno dei tre

**Esercizio 6.** Si supponga che l'host A voglia inviare un file voluminoso all'host B. Il cammino tra l'host A e l'host ha tre collegamenti con velocità di trasmissione R1=500 kbps, R2=3 Mbps, R3=1 Mbps, rispettivamente. Assumendo che non ci sia altro traffico nella rete qual è il throughput stimato per il trasferimento dei file?

- nessuno dei tre
- R2
- R1
- R3

**Esercizio 7.** Si consideri l'invio di un file di 1 Mbit su un datalink di lunghezza di 4800 km. Calcolare il ritardo di propagazione

$$r_{pr} = \frac{d[m]}{s[m/sec]} = \frac{4800 \cdot 10^3 m}{3 \cdot 10^8 m/sec} = 0.016[sec]$$

Sia la velocità di trasmissione pari a 64 kbps, calcolare il ritardo di trasmissione

$$r_{tr} = \frac{L[bits]}{R[bps]} = \frac{10^6 bits}{64 \cdot 10^3 bps} = 15.625[sec]$$

**Esercizio 8.** Ci sono 10 auto in carovana e due caselli. Le automobili viaggiano alla velocità di 100 km/h. Il casello fa transitare un'auto ogni 12 secondi. Quanto tempo occorre affinché le 10 auto in carovana si trovino di fronte al secondo casello? *Ipotesi semplificativa:* la prima auto che arriva al casello attende le altre nove

$$r_{tr} = 12 \cdot 10 = 120 \text{ sec}$$

tempo richiesto al casello per trasmettere l'intera carovana

$$r_{pr} = \frac{100 \text{ km}}{(100 \text{ km/h})} = 1 \text{ h}$$

tempo richiesto a un'auto per viaggiare dall'uscita di un casello fino al casello successivo

**Esercizio 9.** Si calcoli il ritardo end-to-end di un pacchetto sul percorso con i router A e B. Sia trascurabile il ritardo di congestione, il ritardo di elaborazione e si suppongono uguali su tutti i link il ritardo di propagazione e il ritardo di trasmissione

$$r_{tot} = 3 \cdot r_{pr} + 3 \cdot r_{tr} + 3 \cdot r_{elab}$$

essendoci due router intermedi, bisogna attraversare tre link

**Esercizio 10.** Vi sono due hosts, A e B, collegati attraverso un link con velocità di trasmissione R bps e lunghezza m metri e si ipotizzi con la velocità di propagazione sul mezzo sia  $v \text{ m/s}$ . L'host A sta per inviare un pacchetto di L bit all'host B.

- a) L'host A inizia a trasmettere il pacchetto all'istante  $t=0$ . Dove si trova l'ultimo bit del pacchetto all'istante  $d_{trasm}$ ?

*all'istante  $t = d_{trasm}$  host A ha appena finito di trasmettere i bit del pacchetto nel link, quindi, l'ultimo bit è all'inizio del collegamento*

- b) Si supponga che valga  $d_{prop} > d_{trasm}$ . Dove si trova il primo bit del pacchetto all'istante  $t = d_{trasm}$ ?

*ci vuole più tempo a trasmettere un bit da A e B che trasmettere tutto il messaggio di A, quindi, quando l'host A ha finito di trasmettere il pacchetto, il primo bit non è ancora arrivato, ovvero, il primo bit è lungo il collegamento (in mezzo)*

- c) Si supponga che valga  $d_{prop} < d_{trasm}$ . Dove si trova il primo bit del pacchetto all'istante  $t = d_{trasm}$ ?  
 viceversa rispetto al punto precedente, quindi, quando l'host A ha finito di trasmettere il pacchetto, il primo bit è arrivato all'host B. Il primo bit è all'host B
- d) Si supponga che  $v = 2.5 \cdot 10^8 m/s$ ,  $L = 120 \text{ bit}$  e  $R = 56 \text{ kbps}$ . Si determini la distanza m per cui  $d_{prop} = d_{trasm}$
- $$m = \frac{120 \cdot 2.5 \cdot 10^8}{56000} = 535.7 \text{ km} \simeq 536 \text{ km}$$

**Esercizio 11.** Quali dei seguenti sono vantaggi del modello stratificato?

- tutte le opzioni
- è possibile modificare l'implementazione di uno strato senza dover cambiare gli altri strati, a patto che l'interfaccia non cambi
- un livello può essere implementato da soggetti diversi
- scomponere il problema in sottoproblemi più semplici da trattare

**Esercizio 12.** L'espressione "insieme di regole che governano il formato e il significato dei frame, dei pacchetti o dei messaggi che vengono scambiati tra due o più entità omologhe (ovvero allo stesso strato della pila protocollare) si intende la definizione di?

- rete
- protocollo
- interfaccia
- strato

**Esercizio 13.** Indicare quale dei livelli della pila di protocollo TCP/IP ha la responsabilità delle seguenti azioni

- a) Incapsulare dati in segmenti  
*livello trasporto*
- b) Inviare/ricevere frame tra nodi adiacenti  
*livello collegamento*
- c) Trasformare bit in segnali elettromagnetici  
*livello fisico*
- d) Instradare i datagrammi dalla sorgente alla destinazione  
*livello rete*

**Esercizio 14.** Quale delle seguenti unità dati viene incapsulata in un frame

- messaggio
- nessuna delle opzioni
- datagramma
- segmento

**Esercizio 15.** Quale delle seguenti coppie di livelli dello stack TCP/IP e funzioni non è corretta?

- livello applicativo e trasferimento in frame
- nessuna delle opzioni
- livello di trasporto e trasferimento dati end-to-end
- livello di rete e instradamento

**Esecizio 16.** In quale livello dello stack TCP/IP sono generati i dati?

- collegamento
- rete
- fisico
- applicativo

# Chapter 2

## Livello applicazione

### 2.1 Applicazioni di rete

I protocolli al livello applicazione non forniscono servizi ad altri protocolli di livello superiore, ma utilizzano solamente i servizi offerti dai protocolli di livello inferiore (trasporto). Questo significa che eliminare o aggiungere protocolli da questo livello, a patto che siano in grado di utilizzare i servizi di uno dei protocolli di livello trasporto, è un'operazione facile. Dello strato applicativo fanno parte le applicazioni di rete che sono applicazioni distribuite, ovvero, formate da processi in esecuzione su host diversi. All'interno dello stesso host, due processi possono comunicare attraverso la **comunicazione inter-processo** definita dal sistema operativo. L'astrazione che il livello applicazione fornisce è quello di un collegamento diretto dove processi applicativi facenti parte di due host diversi possono inviare e ricevere messaggi. Il protocollo dello strato di applicazione:

- definisce i *tipi di messaggi* scambiati a livello applicativo
- la *sintassi* dei vari tipi di messaggio
- la *semantica* dei campi
- le *regole* per determinare quando e come un processo invia o risponde ai messaggi

I paradigmi del livello applicazione:

- **Client - server**, è formato da due componenti: il client e il server. Il server è sempre attivo per svolgere le operazioni necessarie per realizzare un servizio ed è in attesa delle richieste del client. Il client invia al server le richieste ed attende una risposta
- **Peer-to-peer**, i client e i server si scambiano i ruoli continuamente, quindi vi è uno scambio di servizi alla pari
- **Misto**

### 2.2 API e protocolli TCP/UDP

Application Program Interface è un insieme di regole che un programmatore deve rispettare per utilizzare delle risorse e per realizzare l'interazione tra due entità. L'API che funge da interfaccia tra lo strato di applicazione e di trasporto è chiamata **socket** ed è usata dai processi dello strato applicativo per inviare e ricevere dati dallo strato di trasporto. La socket è un'astrazione, e consiste in una struttura dati utilizzata dal programma applicativo. La comunicazione tra un processo client e server è realizzata attraverso la comunicazione tra le due socket: il client considera la socket come l'entità che riceve le richieste e fornisce le risposte; per il server la socket è l'entità che gli sottopone le richieste e alla quale inviare le risposte. Un **socket address** identifica l'host sul quale il processo client o server è in esecuzione, ed è composto dalla combinazione di un indirizzo IP e un numero di porta.



Nel livello trasporto della pila di protocolli TCP/IP sono previsti due protocolli principali:

## TCP

- Connection oriented: setup richiesto tra client e server
- Trasporto affidabile tra processo mittente e destinatario
- Controllo del flusso: il mittente non "inonderà" di dati il destinatario
- Controllo di congestione: "strangola" il mittente quando la rete è sovraccarica
- Non offre garanzie di timing né di ampiezza minima di banda

## UDP

- Connection less
- Trasporto non affidabile
- No controllo di flusso
- No controllo di congestione
- Non offre garanzie di timing né di ampiezza minima di banda

Quando si sviluppa una nuova applicazione si decide quale protocollo utilizzare. La scelta ha un notevole impatto sulle funzionalità dei processi applicativi. I requisiti più importanti che permettono di distinguere qual è il protocollo di trasporto più adatto per un certo servizio applicativo sono:

- **Throughput**, tasso al quale il processo mittente può inviare i bit al processo ricevente. Alcune applicazioni richiedono un certo throughput per essere efficaci come per esempio le applicazioni multimediali
- **Perdita dei dati**, alcune applicazioni possono tollerare alcune perdite (una chiamata telefonica) mentre altre richiedono un trasferimento dati affidabile al 100% (browser)
- **Sensibilità ai ritardi**, alcune applicazioni come i giochi interattivi richiedono un basso ritardo per essere efficaci

## 2.3 Web

Il **Web** è un'enorme collezione di informazioni nella quale i documenti, chiamati **pagine web**, sono distribuiti in tutto il mondo e collegati l'uno all'altro come in una ragnatela. È un sistema aperto, ovvero, qualsiasi server Web nel mondo può aggiungere una nuova pagina alla collezione e renderla disponibile a tutti gli utenti di Internet. Il collegamento delle pagine web è ottenuto tramite una tecnica chiamata **hypertext**. L'idea alla base dell'ipertesto è quella di utilizzare una macchina in grado di recuperare automaticamente un documento memorizzato nel sistema ogni volta che viene incontrato un link al documento. Oggi l'espressione hypertext, coniata per indicare un insieme di documenti testuali collegati, è stata cambiata in **hypermedia**, per indicare che una pagina può contenere documenti testuali così come immagini, clip audio o video. Il Web è un servizio client/server distribuito, nel quale un client che utilizza un browser può accedere a un servizio offerto da un server. Il servizio fornito è distribuito su numerose locazioni fisiche chiamate **siti**. Ogni sito contiene uno o più pagine web, ciascuna delle quali può contenere collegamenti ad altre pagine web. Una pagina web, essendo un file, deve avere un identificatore univoco per distinguerla dalle altre pagine. Uniform Resource Identifier (**URI**) è una forma generale per identificare una risorsa presente sulla rete e ne esistono di due tipi:

- Uniform Resource Locator (**URL**): sottoinsieme di URI che identifica le risorse attraverso il loro meccanismo di accesso (per esempio protocollo HTTP). Sintassi:

*< scheme >://< user >:< password >@< host >:< port >/< path >*

- *< scheme >* protocollo di accesso alla risorsa, ad esempio HTTP
- *< user >* e *< password >* sono opzionali e generalmente sono deprecati
- *< host >* nome del dominio o l'indirizzo IP
- *< port >* numero di porta del server, se non specificato si usano quelle di default di quel protocollo
- *< path >* contiene dati specifici per l'host o scheme, e identifica la risorsa nel contesto di quello schema

Ogni protocollo ha delle piccole modifiche, come per esempio **HTTP URL**:

*http://< host >:< port >/< path >?< query >*

dove *< query >* è una stringa di informazioni che devono essere interpretate dal server. Le URL possono essere assolute o relative:

- **URL assoluta**, identifica una risorsa indipendentemente dal contesto in cui è usata
- **URL relativa**, informazioni per identificare una risorsa in relazione ad un'altra URL. Non viaggiano sulla rete e sono interpretate dal browser in relazione al documento di partenza
- Uniform Resource Name (**URN**): sottoinsieme di URI che devono rimanere globalmente unici e persistente anche quando la risorsa cessa di esistere e diventa non disponibili

Quindi per identificare una pagina web sono necessari tre identificatori: host, porta e percorso, ma prima di definire la pagina web è necessario indicare al browser quale applicazione client/server si desidera utilizzare, specificando il protocollo.

## 2.4 Protocollo HTTP

HyperText Transfer Protocol è usato come protocollo di trasferimento per il World Wide Web. È un protocollo che definisce come devono essere scritti i programmi client/server per accedere alle pagine web. Inoltre è un **protocollo generico**, poiché non dipende dal formato delle risorse; è un protocollo di tipo **richiesta/risposta** e infine **stateless** poiché le coppie richiesta/risposta sono indipendenti l'una dall'altra, cioè una richiesta viene eseguita indipendentemente da quelle che l'hanno preceduta. Il Web non si può permettere perdita di dati e quindi HTTP utilizza il TCP: il client richiede l'instaurazione di una connessione TCP con il server e i processi si scambiano i messaggi attraverso le rispettive socket. Una **connessione HTTP** è un circuito logico di livello trasporto stabilito tra due programmi applicativi per comunicare tra loro. HTTP nelle versioni precedenti alla 1.1 utilizzava connessioni **non persistenti** dove per ogni richiesta che il client invia al server viene stabilita una nuova connessione TCP. Questo è cambiato in HTTP 1.1 dove le connessioni sono **persistenti**, ovvero, se non diversamente indicato, il client può assumere che il server manterrà una connessione persistente (aperta). Lo standard specifica un meccanismo con cui client e server possono indicare la chiusura della connessione TCP. Dopo la chiusura, il client non deve più inviare richieste su quella connessione.

**Esempio.** Si suppone ci sia una connessione *non persistente* e l'utente digiti il seguente URL:

```
www.someSchool.edu/someDepartment/home.index
```

1. Il client http inizia la connessione TCP verso il server http al `www.someSchool.edu` (la porta 80 viene usata di default per il server http)
2. Il server http dell'host `www.someSchool.edu` aspetta le richieste di connessione TCP alla porta 80. Accetta la connessione, e lo notifica al client
3. Il client http invia un **messaggio di richiesta http** (che contiene la URL) alla socket di connessione con lo strato di trasporto (TCP)
4. Il server http riceve il messaggio di richiesta, compila il **messaggio di risposta** che contiene l'oggetto richiesto (`someDepartment/home.index`) manda il messaggio alla socket e in seguito chiude la connessione
5. Il client http riceve il messaggio di risposta che contiene l'oggetto richiesto

Si immagina ora che l'URL contiene il riferimento a  $n$  file diversi (tutti sullo stesso server): per ogni riferimento si devono ripetere i passaggi definiti sopra, quindi aprire e chiudere la connessione  $n+1$  volte. La strategia non persistente impone un sovraccarico elevato sul server che deve utilizzare  $n+1$  buffer differenti, uno per ogni connessione aperta. Con una *connessione persistente* invece il server lascerebbe aperta la connessione TCP dopo aver spedito la prima risposta e vi riceverebbe quindi le richieste successive. Il server HTTP chiude la connessione su richiesta del client (specificato nell'header del messaggio) o allo scadere di un timeout.

**Pipelining**, serve per migliorare ulteriormente le prestazioni. Consiste nell'invio da parte dei client di molteplici richieste senza aspettare la ricezione di ciascuna risposta. Il server deve inviare le risposte nello stesso ordine in cui sono state ricevute le richieste. Quindi se una richiesta richiede tempo per essere processata le risposte alle richieste successive sono bloccate. Il client non può inviare in pipeline richieste che usano metodi HTTP non *idempotenti*. Per questi motivi è implementato in pochi browser

### 2.4.1 Messaggi HTTP

Un messaggio HTTP può essere di due tipi: *request* o *response*. La riga iniziale distingue la richiesta dalla risposta, seguono una serie di header e il corpo del messaggio (non sempre presente).

- **HTTP request:**

```
Request = Request-Line
         *( general-header
           | request-header
           | entity-header
         CRLF
         [ message-body ]
```

```
Request-Line = Method SP
Request-URI SP
HTTP-Version CRLF
```

- *Method*: operazione che il client richiede venga effettuata sulla risorsa identificata dalla Request-URI
- *Request-URI*: identificativo della risorsa
- *HTTP-Version*: indica la versione del protocollo

Esempio:

```
GET /pub/WWW/TheProject.html HTTP/1.1
Host: www.w3.org
Connection: close
User Agent: Mozilla/4.0
Accept-language: it
```

- **HTTP response**

```
Response = Status-Line
         *( general-header
           | response-header
           | entity-header
         CRLF
         [ message-body ]
```

```
Status-Line
HTTP-Version SP
Status-Code SP
Reason-Phrase CRLF
```

- *Status-Code*, intero a tre cifre che rappresenta un codice di risultato dell'operazione richiesta
  - \* **1xx: Informational**, la richiesta è stata ricevuta ed è in corso
  - \* **2xx: Success**
  - \* **3xx: Redirection**, affinchè la richiesta venga soddisfatta si deve compiere un'altra azione
  - \* **4xx: Client Error**, la richiesta contiene errori di sintassi o non può essere soddisfatta
  - \* **5xx: Server Error**, il server non riesce a soddisfare la richiesta del client
- *Reason-Phrase*, descrizione testuale dello Status code (pensata per l'utente umano)

Esempio:

```
HTTP/1.1 200 OK
Date: Sun, 14 May 2000 23.49.39 GMT
Server: Apache/1.3.9 (Unix) (Red Hat/Linux)
Last-Modified: Tue, 21 Sep 1999 14:46:36 GMT
```

Gli **header** sono un insieme di coppie nome - valore che specificano alcuni parametri del messaggio trasmesso o ricevuto:

- **General Header**, relativi alla trasmissione, per esempio la data del messaggio, se il client richiede di chiudere o di tenere attiva la connessione, la codifica usata, ...

```
Date: Tue, 15 Nov 1994 08:12:31 GMT
Connection: close
Transfer-Encoding: chunked
```

- **Request Header**, relativi alla richiesta ovvero chi fa la richiesta, a chi viene fatta la richiesta, che tipo di caratteristiche il client è in grado accettare, autorizzazioni, ...

```
Accept: text/plain; q=0.5, text/html,
        image/png, application/json
Accept-Charset: iso-8859-5,
                unicode-1-1; q=0.8
Accept-Encoding: compress, gzip
```

- **Accept**: specifica il formato del corpo del messaggio accettabile per una risposta: "q" indica una sorta di preferenza, default = 1
- **Accept-Charset**: set di caratteri accettabile per una risposta
- **Accept-Encoding**: tipo di codifica del contenuto accettabile per la risposta
- **Response Header**, relativi al messaggio risposta ovvero manda delle informazioni sul server
- **Entity Header**, relativi all'entità trasmessa, per esempio il tipo e lunghezza del contenuto, la data di scadenza del contenuto, ...

## 2.4.2 Metodi HTTP

I **metodi di una richiesta HTTP** definiscono i modi in cui il client può interagire con le risorse del server:

- **GET**, richiede il trasferimento di una risorsa identificata da un URL. In questo caso il corpo del messaggio rimane vuoto. Sono possibili **conditional get**, per esempio se si vuole la risorsa solo se è stata modificata da una certa data in poi
- **HEAD**, simile al GET ma il server non trasferisce il corpo del messaggio ma solo l'header. Utile per controllare lo stato dei documenti per esempio la data di ultima modifica o può essere utilizzato anche per verificare se un'URL è valida. Il messaggio di risposta in questo caso prevede solo l'intestazione, il corpo del messaggio rimane vuoto.
- **PUT**, il client chiede al server di creare/modificare una risorsa, solo nel caso in cui questa operazione è consentita
- **POST**, serve per inviare al server alcune informazioni aggiuntive come parte di una richiesta, ad esempio aggiungere informazioni a una risorsa
- **TRACE**, è usato per debug: il client chiede al server di restituirci (echo) la richiesta per verificare se è stata ricevuta correttamente
- **DELETE**, il client chiede di cancellare una risorsa identificata dalla Request URI, ovviamente solo se il client ha i permessi necessari
- **OPTIONS**, consente al client di sapere quali sono i metodi che sono permessi sulla URI

I metodi GET, HEAD, OPTIONS, TRACE sono **safe methods** cioè non hanno effetti "collaterali" come per esempio non modificano la risorsa. I metodi GET, HEAD, OPTIONS, TRACE, PUT, DELETE sono **metodi idempotenti** che invece possono avere degli effetti collaterali ma hanno la particolarità che se si effettua  $N$  volte quella determinata richiesta l'effetto non cambia

## 2.5 Web caching

Il **web caching** serve per soddisfare la richiesta del client senza contattare il server, per fare ciò si memorizzano copie temporanee di risorse Web (esempio pagine HTML, immagini) e si servono al client per ridurre l'uso di risorse, e diminuire il tempo di risposta al client. Esistono due modelli:

- **User Agent Cache**, il browser mantiene una copia delle risorse visitate dall'utente
- **Proxy Cache**, ovvero dei computer che conservano una copia delle risposte alle richieste recenti. Il client http invia una richiesta al proxy, il quale controlla nella propria memoria cache. Se la risposta non è già presente nella cache, il proxy invia la richiesta al server corrispondente. Quando in seguito il proxy riceve la risposta, oltre a inviarla al client che l'aveva richiesta, la memorizza anche nella propria cache per renderla disponibile ad altri client che dovessero effettuare la stessa richiesta. Il proxy riduce il carico sul server originale, spesso riduce il traffico in rete e la latenza. Naturalmente il client, per poter utilizzare il proxy, deve essere configurato in modo da accedere al proxy anziché direttamente al server Web.

HTTP è un protocollo senza stato ovvero i server http non mantengono informazioni sui clients e tipicamente un utente si connette ogni volta con un indirizzo IP e porta diverso. Quindi come riconoscere un client di un'applicazione Web? Come realizzare applicazioni Web con stato? Per risolvere questo problema si usano i **cookie** che funzionano in questo modo:

1. Un client invia a un server una richiesta HTTP
2. Il server invia al client la risposta HTTP + linea **set-cookie**, per esempio *set cookie: 1678453*
3. Il client memorizza il cookie in un file associandolo al server. A tutte le successive richieste a quel sito, il client aggiunge alla richiesta la linea *cookie: 1678453*
4. Alla successiva richiesta il server confronterà il cookie presentato con l'informazione che ha associato a quel cookie

In questo modo quando un client invia una richiesta a un server, controlla nella directory dei cookie l'eventuale presenza di un cookie precedentemente inviato dal server. Il cookie eventualmente trovato viene incluso nella richiesta; il server che la riceve in questo modo capisce che si tratta di un vecchio client già visto in precedenza. I contenuti del cookie non sono mai interpretati dal browser o resi disponibili all'utente: il cookie è "preparato" e "consumato" dal server. I cookie vengono utilizzate per autenticazione, ricordare profile utente, scelte precedenti e in generale creare sessioni sopra un protocollo stateless

## 2.6 TELNET

TERminal NETwork è un protocollo di terminale remoto il cui scopo è quello di permettere l'uso interattivo di macchine remote. TELNET permette ad un utente su una macchina di stabilire una connessione effettuando una sessione di login in una macchina remota. In seguito passa i comandi dalla macchina locale alla macchina remota, e l'output della macchina remota viene trasportata al terminale dell'utente. I comandi vengono eseguiti come se fossero stati battuti al terminale della macchina remota. Il modello TELNET include:

- Un programma server che accetta le richieste
- Un programma client che effettua le richieste:
  - interagisce con il terminale utente sull'host locale
  - scambia i messaggi con il TELNET server

Il client per default si connette alla porta 23 del server e la connessione TCP persiste per la durata della sessione di login. TELNET opera con sistemi operativi eterogenei, ovvero i terminali possono differire gli uni dagli altri per set e codifica di caratteri, diversa combinazione di tasti per interrompere un processo, larghezza della linea e lunghezza della pagina, ... quindi TELNET definisce uno standard comune: **Network Virtual Terminal**, ovvero un terminale virtuale con delle regole per la codifica dei caratteri e dei comandi che permette di trasformare il set di caratteri in uso locale in un set di caratteri universale. L'accesso al server richiede nome utente e password ma il protocollo è estremamente vulnerabile agli attacchi di sicurezza in quanto tutte le informazioni, comprese le credenziali sono trasmesse in chiaro. Questo significa che un attaccante potrebbe semplicemente intercettare le credenziali ed ottenere l'accesso al sistema remoto. Per questo motivo si usa prevalentemente un altro protocollo chiamato **SSH**

## 2.7 Posta elettronica

La posta elettronica è considerata una transazione unidirezionale. Quando il mittente invia un email al destinatario, quest'ultimo potrebbe non essere in quel momento disponibile ad accettare messaggi di posta perché magari è impegnato o il computer è spento. Quindi servono dei computer intermedi (server) per trasferire i messaggi. I componenti che permettono di realizzare il servizio di posta elettronica sono:

- **User Agent**, per la composizione, editing, lettura di messaggi di posta (esempio *Thunderbird, Outlook...*)
- **Mail server**, ovvero i server intermedi. I messaggi in entrata ed uscita vengono archiviati sul server, in particolare una mailbox contiene i messaggi in ingresso diretti all'utente che devono essere ancora letti, e una coda di messaggi in uscita che devono essere inviati
- **Protocollo SMTP**, definisce come deve avvenire il dialogo tra mail server.

I sistemi che si occupano della consegna dei messaggi di posta elettronica necessitano di un sistema di indirizzamento che permetta di individuare in modo univoco gli utenti. In Internet, gli **indirizzi di posta elettronica** consistono di due parti: una parte locale e un nome di dominio, separati dal simbolo @. La parte locale specifica la mailbox nel mail server. Il nome di dominio, invece, specifica un mail server. I server di posta adottano una tecnica denominata **spooling**:

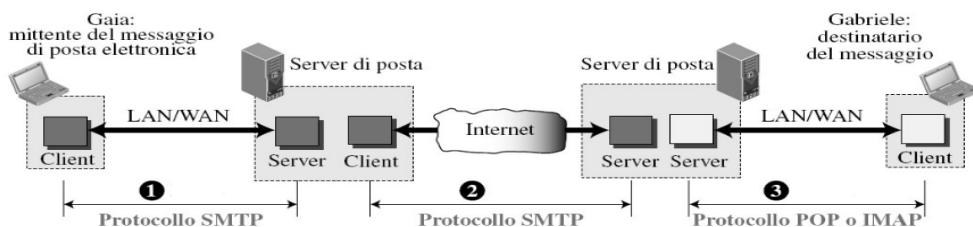
- L'utente invia un messaggio. Il sistema ne pone una copia in memoria (**spool**) insieme con ID mittente, ID destinatario, ID macchina di destinazione e tempo di deposito
- Il sistema avvia il trasferimento alla macchina remota stabilendo una connessione TCP con la macchina di destinazione
- Se la connessione viene aperta inizia il trasferimento del messaggio. Se il trasferimento va a buon fine il client cancella la copia locale della mail. Se il trasferimento non va a buon fine, il processo di trasferimento scandisce periodicamente l'area di spooling, e tenta il trasferimento dei messaggi non consegnati. Oltre un certo intervallo di tempo (definito dall'amministratore del server) se il messaggio non è stato consegnato, viene inviata una notifica all'utente mittente

Non sempre esiste una corrispondenza uno a uno tra utente e indirizzo di email. L'alias è una **cassetta postale virtuale** che serve a ridistribuire i messaggi verso uno o più indirizzi di posta elettronica personali.

- *molti - uno*: il sistema permette ad un singolo utente di avere identificatori di mail multipli. Un utente, più indirizzi postali
- *uno - molti*: il sistema permette di associare un gruppo di destinatari ad un singolo identificatore. Un indirizzo postale, più utenti destinatari (esempio: mailing list)

### 2.7.1 Protocollo SMTP

L'obiettivo di SMTP è il trasferimento affidabile e efficiente di mail. SMTP è indipendente dal sistema di trasmissione, richiede solo il trasferimento di stream di byte ordinato e affidabile. Una caratteristica di SMTP è la capacità di trasportare mail attraverso più reti. Quando un client SMTP vuole trasferire un messaggio, stabilisce un **canale di trasmissione bidirezionale con un server SMTP**. La responsabilità di un client è di trasferire la mail a un server SMTP, o comunicare un eventuale insuccesso: **scambio formale di responsabilità**.



Quando non si riceve un messaggio, i possibili problemi sono:

- connessione con mail server del mittente, ovvero server inesistente o irraggiungibile
- connessione con mail server destinatario, ovvero server inesistente o irraggiungibile
- inserimento in mailbox destinatario, quindi utente non conosciuto o mailbox piena

Comunque in tutti questi casi il mittente riceve una notifica. Il destinatario può non ricevere il messaggio, senza che il mittente sia avvisato, solo se qualcuno (intruso o filtro antispam) rimuove il messaggio. SMTP utilizza il protocollo TCP (**porta 25**) per consegnare in modo affidabile i messaggi dal client al server, e il funzionamento avviene in tre fasi distinte:

- **Handshaking**, il client stabilisce la connessione e attende che il server invii **220 READY FOR MAIL**. Il client risponde con il comando **HELO**, seguito dal suo nome di dominio e il server risponde identificandosi. A questo punto il client può trasmettere i messaggi
- **Trasferimento del messaggio**
- **Chiusura della connessione**

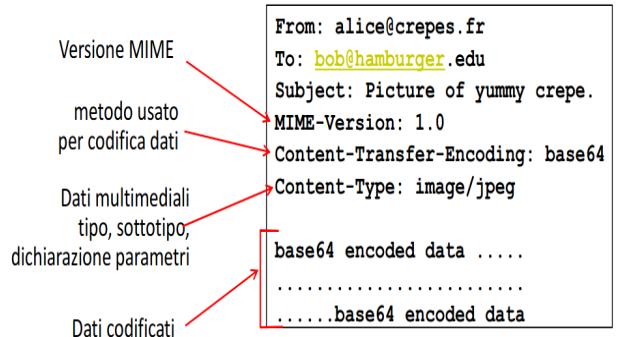
L'interazione avviene con uno scambio di comandi (testo ASCII) da parte del client e di risposte (codice di stato e descrizione) da parte del server. I comandi lato client più importanti sono:

Keyword	Argomento	Descrizione
<b>HELO</b>	nome host mittente	host mittente si identifica
<b>MAIL FROM</b>	mittente messaggio	identifica mittente messaggio
<b>RCPT TO</b>	destinatario	identifica destinatario messaggio
<b>DATA</b>	corpo messaggio	invia contenuto messaggio
<b>QUIT</b>		termina la sessione SMTP

## 2.7.2 Format dei messaggi

Il messaggio può avere delle **linee di intestazione** (esempio *To*, *From*, *Subject*), che sono diversi dai comandi SMTP, e poi il corpo del messaggio. La specifica del protocollo SMTP permette di inviare solo messaggi di testo in ASCII 7 bit, quindi, è stata aggiunta una estensione **MIME** per permettere agli utenti di internet di inviare/ricevere testo in set di caratteri diversi da US-ASCII, allegati in formato non testuale, corpo del messaggio con più parti e header in set di caratteri non-ASCII. MIME continua a usare il formato del messaggio specificato in SMTP ma aggiunge una struttura al messaggio e definisce un insieme di regole di codifica per il trasferimento di testo non-ASCII. MIME agisce lato mittente convertendo tutti i dati in formato non ASCII e poi il messaggio viene decodificato in formato originale presso il destinatario. Questo ha permesso di inviare messaggi MIME usando protocolli e mail server esistenti. Necessario invece cambiare gli user agents. MIME definisce cinque tipi di linee di intestazioni aggiuntive:

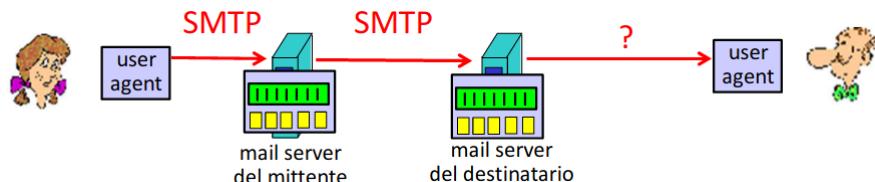
- **MIME-version**: la versione MIME usata
- **Content-Type**: definisce il tipo di dati nel corpo del messaggio
- **Content-Transfer-Encoding**: definisce il metodo utilizzato per la codifica del messaggio
- **Content-Id**: individua univocamente una parte del messaggio in messaggi composti multipli
- **Content-Description**: indica se il corpo del messaggio contiene un'immagine, audio o video



MIME fornisce vari schemi di *transfer encoding*, tra cui:

- **ASCII encoding of binary data**, gruppi di 24 bit sono divisi in 4 unità da 6 bit e ciascuna unità viene inviata come un carattere ASCII
- **Quoted-printable encoding**, per messaggi testuali con pochi caratteri non-ASCII, più efficiente

Oggiorno i mail server possono negoziare l'invio di dati codifica binaria, se la negoziazione non ha successo si usano i caratteri ASCII



## 2.7.3 Protocolli POP e IMAP

Il primo e il secondo stadio della trasmissione della posta elettronica utilizzano SMTP, che non è invece impiegato nel terzo stadio perché essendo un protocollo di tipo *push*: esso "spinge" il messaggio dal client al server. Il terzo stadio richiede invece un protocollo di tipo *pull*: il client deve "tirare" i messaggi dal server, in questo caso la direzione della maggior parte dei dati è dal server al client. I protocolli di tipo *pull* sono:

### • POP3

Io user agent apre una connessione TCP (**porta 110**) verso il server di posta. Durante la fase di autorizzazione con i comandi *user* e *pass* il client invia il proprio nome utente e relativa password, e il server può rispondere con *OK* o *ERR* nel caso di errore. Durante la fase di scambio, i possibili comandi del client sono:

- *list*: visualizza la lista dei messaggi
- *retr*: preleva il messaggio per numero
- *dele*: elimina il messaggio dal server
- *quit*: chiude la sessione

Nella fase finale di aggiornamento, dopo *quit*, il server cancella i messaggi marcati per la rimozione

### • IMAP

è simile al POP3 ma ha più funzionalità. POP3 non consente di organizzare la propria posta, di gestire più cartelle sul server, e inoltre non consente all'utente di controllare parte del contenuto del messaggio prima del suo prelievo completo. IMAP4 fornisce varie funzionalità aggiuntive quali:

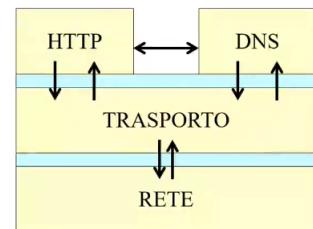
- controllare l'intestazione dei messaggi prima del prelievo
- ricercare una stringa specifica nei messaggi prima del prelievo
- prelevare i messaggi in modo parziale
- creare, cancellare o rinominare le mailbox sul server di posta
- creare una gerarchia di cartelle all'interno della mailbox sul server a scopo di archiviazione

- **HTTP**

se l'user agent è un browser

## 2.8 DNS

I dispositivi connessi in rete vengono individuati dai protocolli TCP/IP mediante il loro indirizzo IP; gli utenti, però, preferiscono usare nomi piuttosto che indirizzi numerici. Per questo motivo è necessario un sistema che associa un indirizzo IP ad ogni nome. Un nome identifica un oggetto mentre un indirizzo specifica dove l'oggetto è situato. Inizialmente l'associazione tra nomi logici e indirizzi IP era statica, tutti i nomi logici e relativi IP erano contenuti in un host file e periodicamente tutti gli host prelevavano una versione aggiornata del file da un server ufficiale. Date le dimensioni attuali di Internet, questo approccio è impraticabile, si utilizza pertanto il DNS. Il DNS è posizionato nel livello applicativo, si basa sul paradigma client-server e si affida al sottostante protocollo di trasporto punto-punto per trasferire messaggi tra sistemi terminali. Funzionamento risoluzione:



1. L'utente digita un URL, per esempio, `www.sss.com/contact.html`
2. Il browser ha bisogno di tradurre l'URL in un indirizzo IP, quindi userà il servizio DNS
3. Una volta avuto la traduzione potrà chiedere al livello di trasporto di aprire la connessione TCP con il server Web (indirizzo IP e porta), a questo punto sulla connessione manderà un messaggio di richiesta e si aspetterà il messaggio di risposta

Il DNS è un meccanismo che deve specificare la sintassi dei nomi e le regole per gestirli, e consentire la conversione da nomi a indirizzi e viceversa. Esso è costituito essenzialmente da:

- uno **schema di assegnazione dei nomi** gerarchico e basato su domini
- un **database distribuito** contenente i nomi e le corrispondenze con gli indirizzi IP implementato con una gerarchia di name server
- un protocollo per la distribuzione delle informazioni sui nomi tra name server

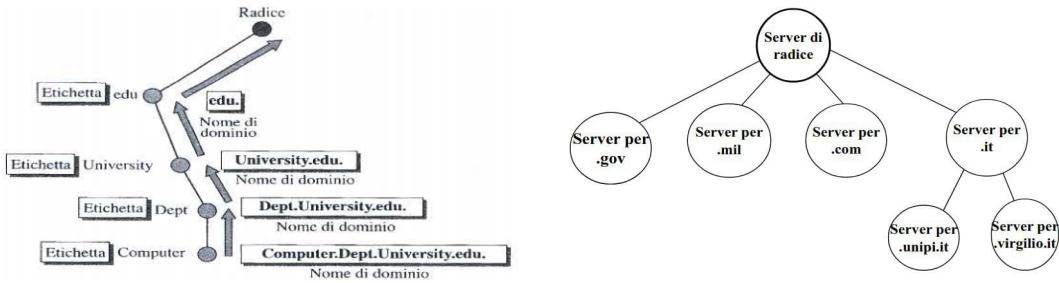
I servizi DNS sono:

- **Risoluzione** di nomi di alto livello (hostname) in indirizzi IP
- **Host aliasing**, un host può avere più nomi
- **Mail server aliasing**, sinonimi per mail server
- **Distribuzione carico**, associare a un hostname canonico più indirizzi IP in modo tale che possa distribuire il carico

### 2.8.1 Spazio dei nomi

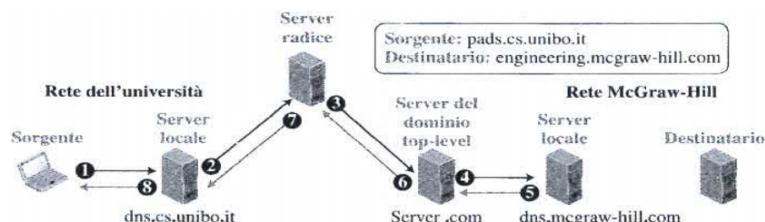
Al fine di evitare ogni ambiguità è necessario definire uno spazio dei possibili nomi da assegnare ai calcolatori connessi in rete. In questo spazio si deve poter assegnare in modo univoco un nome a un indirizzo, in modo tale che anche il nome individui senza ambiguità un calcolatore. Si ha bisogno di una **struttura gerarchica** dove ogni nome è composto da diverse parti: la prima parte può definire la natura dell'organizzazione, la seconda il suo nome, la terza i vari dipartimenti all'interno dell'organizzazione e così via. La struttura gerarchica permette autonomia nella scelta dei nomi all'interno di un dominio. Anche se due organizzazioni scegliersero il medesimo prefisso per alcuni dei loro host, la cosa non creerebbe problemi perché gli indirizzi sarebbero comunque diversi nella parte riguardante il tipo e la denominazione dell'organizzazione. Esempio: `server1.di.unipi.it` e `server1.cs.cornell.edu`. I

nomi hanno una struttura ad albero con un numero di livelli variabile. Ogni nodo è individuato da un'etichetta e alla radice è associata un'etichetta vuota. Ogni nodo dell'albero ha un nome di dominio, ovvero una sequenza di etichette separati da punti. **Dominio**: sottoalbero nello spazio dei nomi di dominio che viene identificato dal nome di dominio del nodo in cima al sottoalbero. Un dominio può essere suddiviso in ulteriori domini, detti sottodomini.

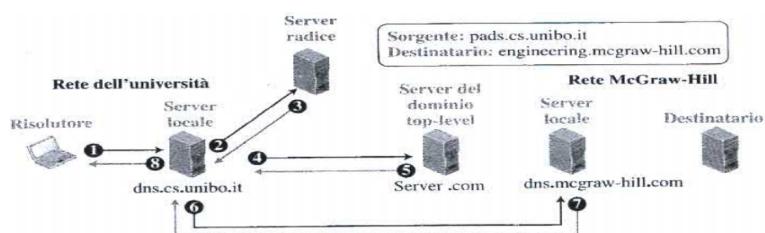


Alla gerarchia dei nomi di dominio corrisponde una gerarchia dei server. Con la gerarchia dei name server si possono distribuire le informazioni sui domini su più name server. Una **zona** è tutto ciò di cui è responsabile un server. Zona e dominio non coincidono necessariamente. Il server immagazzina le informazioni relative alla propria zona inclusi i riferimenti ai server dei domini di livello inferiore. Gerarchia dei server:

- **Server radice**, è un server che ha per zona l'intero albero e restituisce le informazioni (nome, indirizzo IP) sui name server di server top-level domain
- **Server top-level domain**, mantengono le informazioni dei nomi di dominio e restituiscono le informazioni sui name server di competenza dei sottodomini.
- **Server di competenza**, sono l'autorità per una certa zona, memorizza nome e indirizzo IP di un insieme di host e può effettuare traduzioni nome/indirizzo per quegli host. Per una certa zona ci possono essere server di competenza:
  - **Primari**, mantengono il file di zona
  - **Secondari**, ricevono il file di zona e offrono il servizio di risoluzione
- **Local name server**, non appartengono strettamente alla gerarchia dei server. Ciascun ISP, come un'università, ha il suo default name server locale. Quando un host si connette a un ISP, quest'ultimo gli fornisce un indirizzo IP tratto da uno o più dei suoi local name server. Quando un programma, come ad esempio un browser, deve trasformare un nome in un indirizzo IP chiama un programma detto **resolver** passando il nome come parametro di ingresso. Il resolver interroga il local name server che cerca il nome nelle sue tabelle e restituisce l'indirizzo al resolver oppure inoltra la query alla gerarchia DNS. La modalità di svolgimento della risoluzione può essere:
  - **query ricorsiva**, la query DNS viaggia dal programma applicativo che ne ha fatto richiesta fino a un host che conosce l'indirizzo IP richiesto, eventualmente scalando, poi discendendo e infine percorrendo all'indietro l'intera gerarchia dei server DNS.



- **query iterativa**, ogni server che non conosce la risposta alla domanda del client risponde con l'indirizzo di un altro server in grado di risolvere il problema



Una volta che un name server ha appreso una associazione, la inserisce nella **cache**. I record nella cache vengono cancellati dopo un certo tempo per il rischio di contenere informazioni obsolete. Il caching migliora il ritardo e riduce il numero di messaggi DNS.

## 2.8.2 Record e messaggi DNS

Il database di un name server è una collezione di **record risorsa** che vengono inviati al client che ne fa richiesta. Un record risorsa è formato da quattro campi:

(Nome di dominio, Valore, Tipo, TTL)

dove: *nome di dominio* identifica il record risorsa; *valore* contiene l'informazione memorizzata relativa al nome di dominio; *tipo* definisce come interpretare i campi nome e valore e *TTL* indica il numero di secondi per cui l'informazione deve essere ritenuta valida. I significati di nome e valore dipendono dal tipo:

<i>Tipo</i>	<i>Nome</i>	<i>Valore</i>
<b>A</b>	hostname	indirizzo IP
<b>CNAME</b>	hostname (sinonimo)	nome canonico dell'host
<b>NS</b>	nome di dominio	hostname dell'autoritative name server per quel dominio
<b>MX</b>	nome di dominio	nome canonico del server di posta associato a name

Esempi di record DNS:

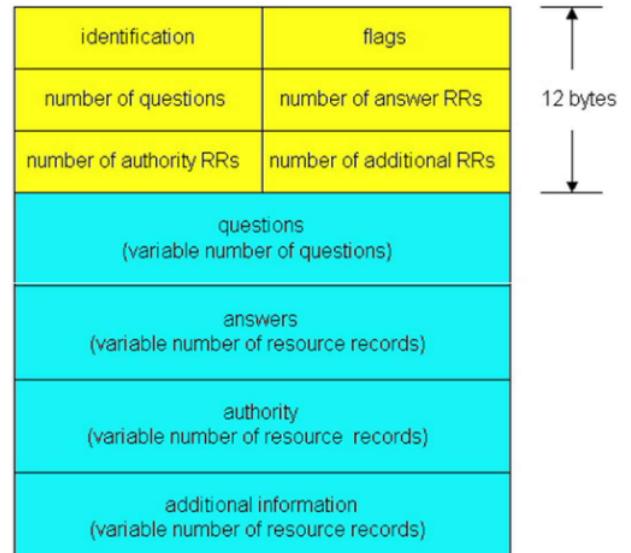
- (HostName, indirizzoIPdiHostName, A, ...) ad un nome corrispondono due indirizzi IP:  
(www.ccn.com, 157.166.224.25, A, ...) e (www.ccn.com, 157.166.224.26, A, ...)
- (Dominio, NomediAuthoritativeServerPerDominio, NS, ...)  
(cli.di.unipi.it, nameserver.cli.di.unipi.it\*, NS, ...)
- (Alias, HostNameMailServerConTaleAlias, MX, ...)  
(cli.di.unipi.it, mailserver.cli.di.unipi.it\*, MX, ...)

I **messaggi DNS** sono di due tipi *query* e *reply*, entrambi con lo stesso formato di messaggio. Si descrivono brevemente i campi di un messaggio DNS:

- **Identificazione**, campo da 16 bit ed è utilizzato dal client per associare la risposta all'interrogazione
- **flags** indica se si tratta di un messaggio di richiesta o di risposta. È utilizzato anche per segnalare eventuali errori

I quattro campi successivi nell'intestazione definiscono il numero di ciascun tipo di record nel messaggio. Sezioni:

- **Domande**, è inclusa nell'interrogazione ed è ripetuta nel messaggio di risposta, consiste di uno o più record di richiesta
- **Risposta**, presente esclusivamente nei messaggi di risposta, consiste di uno o più record di risorsa
- **Competenza**, fornisce informazioni di uno o più server autorevoli per l'interrogazione
- **Supplementare**, fornisce informazioni addizionali che potrebbero essere utili al risolutore

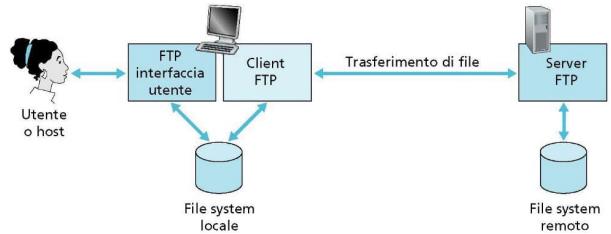


Il sistema DNS può usare sia il protocollo UDP (porta 53) che quello TCP. Il protocollo UDP viene usato quando la dimensione del messaggio di risposta è inferiore a 512 byte. In caso contrario viene usato il protocollo TCP. Il DNS è altamente decentralizzato: nessun singolo server DNS contiene tutti gli indirizzi IP e i rispettivi domini. Una query viaggia lungo una catena di server DNS prima di ottenere il risultato. Il **DNS Hijacking** è la pratica di restituire risposte non corrette alle query DNS reindirizzando il client verso siti malevoli.

## 2.9 Protocollo FTP

Il File Transfer Protocol è il protocollo standard per il trasferimento di file in reti TCP/IP, ma fornisce anche funzionalità aggiuntive come:

- **Accesso interattivo**, l'utente può navigare e cambiare/modificare l'albero di directory nel file system remoto
- **Specifiche del formato dei dati da trasferire**
- **Autenticazione**, il client può specificare login e password



FTP adotta il modello client/server dove il client ha tre componenti: interfaccia utente, processo di controllo e processo di trasferimento dati, mentre, il server ha invece due sole componenti: processo di controllo e processo di trasferimento dati. Il protocollo FTP usa due connessioni TCP distinte:

- **Connessione di controllo**, scambio di comandi e risposte tra client e server. Segue il protocollo TELNET. La connessione rimane aperta per l'intera durata della sessione interattiva
  - Il client FTP contatta il server FTP alla porta 21
  - Il client ottiene l'autorizzazione sulla connessione di controllo
  - Il client invia i comandi sulla connessione di controllo (invio file, cambio directory, ...)
- **Connessione dati**, connessione su cui i dati sono trasferiti con modi e tipi specificati. I dati trasferiti possono essere parte di un file o un set di file. Viene aperta e chiusa la connessione per ogni singolo scambio. Quando il server riceve un comando per trasferire un file sulla connessione di controllo:
  1. Il server apre una connessione TCP con il client
  2. Trasferisce il file sulla connessione dati
  3. Dopo il trasferimento di un file, il server chiude la connessione

Per mantenere simultaneamente, aperte tra loro, una connessione di controllo e una connessione dati sono necessarie due porte, una per connessione. FTP usa la connessione di controllo per permettere a client e server di coordinare l'uso delle porte assegnate dinamicamente per il trasferimento dati. La comunicazione sulla connessione di controllo avviene per mezzo di caratteri con una codifica standard NVT ASCII, sia per i comandi che per le risposte. FTP è un protocollo **STATEFUL** ovvero il server deve tener traccia dello stato dell'utente. Per creare la connessione TCP per il trasferimento dati sono possibili due modalità:

- **Active mode**, modalità descritta precedentemente
  - Il server apre una connessione dati TCP con il client
  - Il server deve conoscere il numero di porta lato client, ovvero, il client gliela comunica sulla connessione di controllo
- **Passive mode**, il client chiede al server di mettersi in ascolto su una porta per una connessione dati, ottiene questo numero di porta dal server e lo usa per aprire la connessione con il server

I dispositivi dove risiedono client e server FTP sono diversi: sistema operativo, strutture per gestire i file, diversi formati dei file. Per effettuare il trasferimento file, il client deve definire il tipo di file, la struttura dati e la modalità di trasmissione al fine di risolvere i problemi di eterogeneità tra client e server. Il trasferimento file viene preparato attraverso uno scambio di informazioni lungo la connessione di controllo. La modalità di trasmissione può essere:

- **Stream mode**: FTP invia i dati a TCP con un flusso continuo di bit
- **Block mode**: FTP invia i dati a TCP suddivisi in blocchi. Ogni blocco è preceduto da un header
- **Compressed mode**: si trasmette il file compresso

Il collegamento FTP con un server può essere anonimo, tipicamente consentono di accedere solo ad una parte del file system e permettono solo un subset di operazioni

## 2.10 Esercizi

**Esercizio 1.** In una comunicazione client/server che usa HTTP su TCP, se più oggetti vengono inviati sulla stessa connessione TCP, allora la connessione è

- non persistente*
- affidabile*
- stateful*
- persistente*

**Esercizio 2.** In una URL HTTP dove si trova la componente "port"

- non è una componente di una URL HTTP*
- a destra della componente "path", dopo il separatore ":"*
- a destra della componente "http", prima del separatore ":"*
- a destra della componente "host", dopo il separatore ":"*

**Esercizio 3.** Quale dei seguenti header HTTP può essere usato per creare richieste GET condizionali?

- nessuna delle opzioni*
- Allow*
- Content-type*
- If-Modified-Since*

**Esercizio 4.** DELETE è un metodo:

- idempotente*
- nessuna delle opzioni*
- safe*
- safe e idempotente*

**Esercizio 5.** Un messaggio di risposta HTTP deve obbligatoriamente contenere

- status line*
- status line e almeno un header*
- status line e message body*

**Esercizio 6.** Quale delle seguenti informazioni è presente sia nella Request Line che nella Status line?

- URL*
- status code*
- nessuna delle opzioni*
- versione HTTP*

**Esercizio 7.** Quale metodo HTTP può essere usato per recuperare informazioni su una risorsa (es: file html) senza che venga trasferita la risorsa stessa?

- POST*
- COPY*
- HEAD*
- nessuna delle opzioni*

**Esercizio 8.** Dire quale delle seguenti affermazioni NON è corretta?

- lo scopo principale di TELNET è trasferire file*
- TELNET permette di effettuare sessioni di login su macchine remote*
- TELNET è un programma client-server general purpose (non specializzato per tipo applicativo)*

nessuna delle opzioni

**Esercizio 9.** Dato il seguente messaggio di richiesta HTTP, rispondere alle domande sotto elencate

```
GET /examples/interactive/quotations.htm HTTP/1.1
Host: gaia.cs.umass.edu
Accept: text/plain, text/html, image/jpeg, image/png, audio/basic,
audio/vnf.wave, video/mp4, video/mpeg
Accept-Language: en-us, en-gb;q=0.4, en;q=0.3, fr, fr-ch
If-Modified-Since: Tue, 29 Sep 2020 00:00:29 GMT
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/85.0.4183.121 Safari/537.36
```

1. Qual è il nome del file che si vuole recuperare con questo messaggio?

*quotations.htm*

2. Quale versione di HTTP è utilizzato dal client?

*HTTP/1.1*

3. Vero o falso: il client accetta file html

*Vero, lo si vede nel campo Accept*

4. Vero o falso: il client accetta immagini jpeg

*Vero, lo si vede nel campo Accept*

5. Qual è la versione inglese preferita dal client?

*en-us: statunitense*

6. Qual è la versione inglese meno preferita dal client?

*en: english, perché ha il valore q più basso*

7. Vero o falso: il client accetta la lingua tedesca

*Falso, il client non ha incluso German nel campo Accepted-Language*

8. Vero o falso: il client ha già una copia in cache del file

*Vero, il client ha una copia del file che è stata aggiornata in data Tue, 29 Sep 2020 00:00:29 GMT*

**Esercizio 10.** Dato il seguente messaggio di risposta HTTP (il messaggio body è omesso per ragioni di spazio) rispondere alle domande in basso

```
HTTP/1.1 200 OK
Date: Tue, 29 Sep 2020 07:19:05 +0000
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 29 Sep 2020 07:29:05 GMT
Content-Length: 94641
Keep-Alive: timeout=60, max=72
Connection: Keep-alive
Content-type: text/html
```

1. Il server usa HTTP 1.0 o HTTP 1.1?

*HTTP 1.1*

2. Il server ha mandato il documento richiesto con successo?

*Sì, lo si vede dall'OK*

3. Qual è la dimensione del documento in bytes?

*94641*

4. La connessione è persistente o non persistente?

*Persistente, lo si vede da Connection*

5. Quale tipo di file ha inviato il server in risposta?

*File html, lo si vede da Content-type*

**Esercizio 11.** Un Web Browser deve recuperare una risorsa alla URL `http://www.example.com/index.html`. Il browser instaura una connessione con il server `www.example.com` per ottenere la risorsa. Il browser ha già una copia in cache della pagina con data `Mon, 03 Sep 2018 12:30:00 GMT`, che è ancora la versione più aggiornata presente nel server di origine. Descrivere lo scambio di messaggi HTTP tra client e server commentando brevemente e indicando per ciascun messaggio request/response line e eventuali header opportuni

*Il browser invia una richiesta di tipo GET condizionale indicato in un header (vedi sotto) che richiede la risorsa solo se è stata aggiornata dopo una certa data. Come riportato nel testo dell'esercizio, il server di origine non ha una versione più aggiornata e quindi invia una risposta al client (vedi response line sotto) senza includere nel body della risposta la risorsa*

```
GET http://www.example.com/index.html HTTP/1.1
If-Modified-Since: Mon, 03 Sep 2018 12:30:00 GMT
...
HTTP/1.1 304 Not Modified
...
```

**Esercizio 12.** Data la stringa `http://demo.di.unipi.it/service/sum?a=3&b=5`, indicare le parti di cui è composta e il loro significato

*La stringa è una http URL. È composta dalle seguenti parti:*

- *schema (`http://`) - indica lo schema per identificare la risorsa, in una http URL indica il meccanismo di accesso i.e protocollo;*
- *Authority o host (`demo.di.unipi.it`) - in questo caso è un nome di dominio di un host, in alternativa potrebbe essere indicato un indirizzo IP;*
- *Path (`service/sum`) - identifica la risorsa nel contesto dello schema e dell'authority, rappresenta il percorso logico con cui sono organizzate le risorse offerte dal servizio Web;*
- *query (`?a=3&b=5`) - rappresenta una query e i relativi parametri di input e deve essere interpretata dal server*

**Esercizio 13.** Considera le seguenti azioni legate alla mail:

- a1: uno user agent invia una mail ad un mail server
- a2: uno user agent recupera una mail da un mail server
- a3: un utente controlla le email con un browser web

Quale protocollo di livello applicativo viene usato per ciascuna attività?

- a1: POP3, a2: SMTP, a3: IMAP
- a1: SMTP, a2: POP3, a3: HTTP
- a1: SMTP, a2: FTP, a3: HTTP
- a1: HTTP, a2: SMTP, a3: POP3

**Esercizio 14.** Quale protocollo di trasporto è usato per mail elettronica

- TCP
- SMTP
- UDP
- IP

**Esercizio 15.** Quando un mail server trasferisce una mail ad un altro mail server si comporta da

- client SMTP
- nessuna delle opzioni
- server POP
- server SMTP

**Esercizio 16.** In SMTP, l'indirizzo mail del destinatario è indicato tramite il comando

- DATA
- nessuna delle opzioni
- MAIL FROM
- RCPT TO
- HELLO

**Esercizio 17.** Quale, tra quelli elencati, è il numero di porta di default di un server SMTP

- 20
- 25
- 80
- 33

**Esercizio 18.** Il DNS si avvale dei servizi di livello trasporto con porta di default 53. Indicare quale opzione è corretta

- nessuna delle opzioni
- il DNS usa solo il protocollo UDP
- il DNS usa il protocollo UDP e in alcuni casi TCP
- il DNS usa solo il protocollo TCP

**Esercizio 19.** Un server secondario di competenza per un certo dominio riceve il file di zona da:

- il server primario di competenza per la stessa zona
- il local name server
- un server di competenza di una qualsiasi zona
- nessuna delle opzioni

**Esercizio 20.** Nel nome di dominio *www.unipi.it* l'etichetta più specifica è

- it
- unipi
- nessuna delle opzioni
- www

**Esercizio 21.** Si consideri un client HTTP che desidera recuperare una risorsa Web ad una certa URL, di cui, all'inizio, non conosce l'indirizzo IP. Quali protocolli di trasporto sono richiesti in questo scenario?

- DNS e HTTP
- nessuna delle opzioni
- TCP per HTTP e UDP per DNS
- TCP per HTTP e DNS

**Esercizio 22.** L'host *host1.tintin.fr* vuole visitare il sito web *www.example.it*, di cui non conosce l'indirizzo IP. Si assuma che sul TLD server siano presenti i seguenti record:

- (*www.example.it*, *dns.example.it*, NS)
- (*dns.example.it*, 146.54.128.216, A)

Si assuma che sul server DNS di competenza per il dominio *example.it* siano presenti i seguenti record:

- (*www.example.it*, *host3.example.it*, CNAME)
- (*host3.example.it*, 142.81.17.206, A)
- (*example.it*, *mail.example.it*, MX)
- (*mail.example.it*, 247.29.41.120, A)

Nell'ipotesi di query iterativa e assumendo che non ci siano record in cache, quali Resource record sono inviati nel messaggio di risposta dal TLD server?

- entrambi i record:*
  - (www.example.it, dns.example.it, NS)
  - (dns.example.it, 146.54.128.216, A)
- entrambi i record:*
  - (www.example.it, host3.example.it, CNAME)
  - (host3.example.it, 142.81.17.206, A)
- il record: (www.example.it, host3.example.it, CNAME)*

Invece, nell'ipotesi di query iterativa e assumendo che non ci siano record in cache, quali Resource record sono inviati nel messaggio di risposta dal server DNS di competenza per il dominio example.it?

- entrambi i record:*
  - (www.example.it, mail.example.it, MX)
  - (mail.example.it, 247.29.41.120, A)
- entrambi i record:*
  - (www.example.it, host3.example.it, CNAME)
  - (host3.example.it, 142.81.17.206, A)
- entrambi i record:*
  - (www.example.it, dns.example.it, NS)
  - (dns.example.it, 146.54.128.216, A)

**Esercizio 23.** Alice invia un messaggio a Bob con l'ipotesi che il mail agent di Bob usa il protocollo POP3. Guardando il grafico sotto rispondere alle seguenti domande:

1. Quale protocollo viene usato al punto 2 del grafico?

*Il mail agent di Alice usa SMTP per trasmettere la mail al server mail di Alice*

2. Quale protocollo viene usato al punto 4 del grafico?

*Il server mail di Alice usa SMTP per trasmettere la mail al server mail di BOB*

3. Quale protocollo viene usato al punto 6 del grafico?

*Il mail agent di Bob usa POP3 per recuperare le mail destinate a Bob dal server mail di Bob*

4. SMTP usa TCP o UDP?

*SMTP usa TCP*

5. SMTP è un protocollo push o pull?

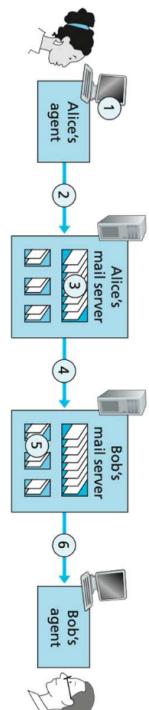
*SMTP è un protocollo push*

6. POP3 è un protocollo push o pull?

*POP3 è un protocollo pull*

7. Quale porta usa SMTP? Invece POP3?

*SMTP usa la porta 25, POP3 usa la porta 110*



**Esercizio 24.** I comandi HELO e MAIL FROM sono entrambi necessari in SMTP? Perchè? Se nell'intestazione del messaggio di email compare il campo to: con la specifica di un indirizzo email non è necessario inviare il comando SMTP RCPT TO. Vero o falso?

*Il comando HELO è necessario affinché il client possa identificarsi utilizzando il proprio nome di dominio, durante la creazione del collegamento. Il comando MAIL FROM è necessario per indicare il mittente del messaggio e fornire al server un indirizzo e-mail per eventuali messaggi di segnalazione o di errore. Falso, il comando SMTP RCPT TO è necessario per indicare il destinatario al mail server a cui viene trasferita la email (se ci sono più destinatari viene inviato un comando RCPT TO per ciascun destinatario)*

**Esercizio 25.** L'utente *mickey@disney.com* invia dal suo PC una email a *donald@disney.com*. Indicare la sequenza di comandi SMTP inviati e ricevuti dal PC di *mickey@disney.com* se: (a) il mailserver di disney.com non è raggiungibile, oppure (b) il mailserver di disney.com è raggiungibile

- (a) *Il PC di mickey@disney.com non riesce a stabilire una connessione TCP con il mailserver di disney.com, quindi nessun messaggio SMTP viene inviato o ricevuto*
- (b) *Il PC di mickey@disney.com stabilisce una connessione TCP con il mailserver di disney.com su cui scambia i seguenti comandi SMTP:*

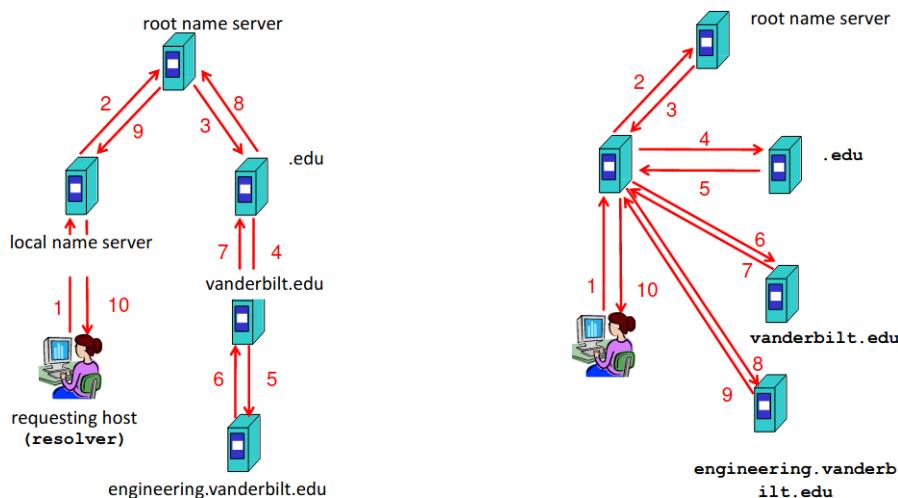
```

<- 220 service ready -
- HELO ... ->
<- 250 OK -
- MAIL FROM: mickey@disney.com ->
<- 250 OK -
- RCPT TO: donald@disney.com ->
<- 250 OK -
DATA -->
<- 354 End data with <CR><LF>.<CR><LF>
(testo dell'email)
.
<- 250 OK -
QUIT ->
<- 221 service closed -

```

**Esercizio 26.** Un host deve risolvere il nome simbolico *host.engineering.vanderbilt.edu*, il cui indirizzo IP non è noto al suo resolver (i.e servizio DNS dell'host) né al local name server. Supponendo che la gerarchia dei name server abbia 4 livelli, indicare - giustificando la risposta - il numero di messaggi DNS che, nel caso peggiore, circoleranno in internet per risolvere tale nome simbolico, se: (a) si utilizza ad ogni livello una risoluzione ricorsiva, oppure, (b) una risoluzione iterativa

- (a) *Ricorsiva: bisogna percorrere tutto l'albero dei name server, dal name server locale fino ad un root server, e poi da questo al name server di competenza (authoritative), che nel caso peggiore è il name server per engineering.vanderbilt.edu, per poi tornare indietro: in totale sono 4+4 messaggi a cui vanno aggiunti i 2 messaggi dal resolver al name server locale del client, e viceversa: quindi 10*
- (b) *Iterativa: anche ora i messaggi saranno 10, perché i name server coinvolti saranno 4, nel caso pessimo, più i 2 messaggi dal resolver al name server locale del client, e viceversa*



**Esercizio 27.** Quale delle seguenti associazioni metodi HTTP e operazione NON è corretta?

- DELETE: cancellazione di una risorsa*
- PUT: creazione o modifica di una risorsa*
- POST: restituisce una risposta*

**Esercizio 28.** Data la sequenza di caratteri *http://www.unipi.it/demo/corsi.html* indicare quale affermazione NON è corretta:

- E' una URL HTTP e la componente path è: *demo/corsi.html*
- E' una URL HTTP e la componente host è: *www.unipi.it*
- E' una URL HTTP e la componente schema è: *www.unipi.it*

**Esercizio 29.** Durante una sessione FTP la connessione DATI viene aperta

- una volta
- ad ogni trasferimento di file
- due volte

**Esercizio 30.** Un record DNS che contiene le seguenti informazioni

- Name: *di.unipi.it*
- Value: *ns1.unipi.it*

è un record di tipo:

- NS
- A
- MX

**Esercizio 31.** Per trasferire una mail ad un server SMTP quali comandi vanno inviati e in quale ordine?

- DATA, EHLO, RCPT TO, MAIL FROM
- EHLO, MAIL FROM, RCPT TO, DATA
- RCPT TO, MAIL FROM, EHLO

# Chapter 3

## Livello trasporto

### 3.1 Introduzione

Il livello trasporto si colloca fra il livello rete e il livello applicazione, e fornisce una **comunicazione logica** tra processi residenti in host diversi, ovvero, i processi si comportano come se gli host fossero direttamente collegati. Quindi non si preoccupano dei dettagli dell'infrastruttura usata. I protocolli di trasporto vengono eseguiti nei sistemi terminali. Offre servizi allo strato di applicazione: un'applicazione interagisce con i protocolli di trasporto per trasmettere o ricevere dati; l'applicazione sceglie lo stile di trasporto tra sequenza di messaggi singoli o sequenza continua di byte e poi il programma applicativo passa i dati nella forma richiesta al livello di trasporto per la consegna. Utilizza servizi dello strato di rete: il livello di rete si occupa della comunicazione tra host e il protocollo di rete consegna il datagramma all'host destinatario.

### 3.2 I servizi del livello trasporto

Lo strato di trasporto offre due tipi di servizi di connessione:

- **Servizio privo di connessione - UDP**, il processo mittente consegna i messaggi al livello trasporto uno per uno. Il livello trasporto tratta ogni messaggio come entità singola senza mantenere alcuna relazione fra di essi. I segmenti possono non essere consegnati o non arrivare in ordine
- **Servizio orientato alla connessione - TCP**, il client e il server stabiliscono una connessione logica

Il livello trasporto effettua un **controllo degli errori**, e il servizio di **multiplexing/demultiplexing**. Il termine *multiplexing* fa riferimento al caso in cui un'entità riceve informazioni da più di una sorgente, viceversa, il termine *demultiplexing* fa riferimento al caso in cui un'entità trasmette informazioni a più di un destinatario. Il livello trasporto effettua il multiplexing nel sito mittente e il demultiplexing nel sito destinatario. Le operazioni di multiplexing e demultiplexing si basano su socket address dei processi. Ogni comunicazione di trasporto (TCP o UDP) è identificata in maniera univoca grazie alle coppie:

- **Indirizzo IP**, indirizzo di 32 bit presente nello stack TCP/IP
- **Porta**, numero che viene assegnato a un processo, o più precisamente a un punto di demultiplexing dei protocolli TCP o UDP.

Il sistema operativo assegna dinamicamente le porte ai processi che ne fanno richiesta. In un *demultiplexing senza connessione* lo strato di trasporto dell'host ricevente consegna il segmento UDP alla socket identificata da IP e porta destinazione. I datagrammi con IP e/o porta mittenti differenti ma stessi IP e porta destinatari vengono consegnati alla stessa socket. Nel *demultiplexing orientato alla connessione*, la socket TCP connessa è identificata da 4 parametri: indirizzo IP di origine, numero di porta di origine, indirizzo IP di destinazione, numero di porta di destinazione. L'host ricevente usa i 4 parametri per inviare il segmento alla socket appropriata. Quindi un host server può supportare più socket contemporanee.

### 3.3 TCP

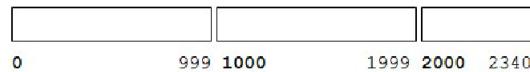
Il protocollo TCP è un protocollo orientato alla connessione: i processi effettuano un *handshake* prima dello scambio di dati. La connessione è **full-duplex**: il flusso dati tra due host può avvenire contemporaneamente nelle due direzioni

e le due direzioni sono slegate. Inoltre **TCP è un protocollo orientato allo stream** (lunghezza di byte indefinita a priori). TCP vede i dati come un flusso di byte ordinati, ma non strutturati. Quindi le proprietà del servizio TCP sono:

- **Funzioni base per il trasferimento di dati:** capacità di trasferire un flusso continuo di byte e trasferimento bidirezionale (full-duplex)
- **Multiplexing/demultiplexing:** consente di assegnare una data connessione ad un particolare processo
- **Controllo della connessione:** meccanismi di inizio e fine trasmissione
- **Trasferimento dati ordinato e affidabile,** corregge tutti i tipi di errore come per esempio dati corrotti, segmenti persi, segmenti duplicati e segmenti fuori sequenza
- **Controllo di flusso,** evita di spedire più dati di quanti il ricevitore sia in grado di trattare
- **Controllo di congestione,** ha lo scopo di recuperare situazioni di sovraccarico nella rete
- **Trasferimento bufferizzato,** il software del protocollo TCP è libero di suddividere il flusso di byte in **segmenti** in modo indipendente dal **programma applicativo** che li ha generati. Per fare questo è necessario disporre di un buffer dove immagazzinare la sequenza di byte. Appena i dati sono sufficienti per riempire un segmento ragionevolmente grande, questo viene trasmesso attraverso la rete. La bufferizzazione consente una riduzione del traffico sulla rete "ottimizzando" in qualche modo il numero di segmenti da trasmettere.

TCP per realizzare un servizio di trasporto affidabile usa un meccanismo di numerazione che prevede due campi, chiamati **numero di sequenza** e **numero di riscontro (ACK)**. Questi due campi sono contenuti nell'intestazione dei segmenti TCP e fanno riferimento a un numero di byte (e non a un numero di segmento). Il *numero di sequenza* associato a un segmento rappresenta il numero, all'interno nel flusso di byte, del primo byte contenuto nel segmento.

**Esempio.** Si immagina di avere una sequenza di 2341 byte e ogni segmento contiene al massimo 1000 byte. Si divide il flusso di byte in 3 gruppi come in figura. Nel primo segmento, nel campo numero di sequenza, si avrà 0, nel secondo segmento 1000 e nell'ultimo segmento 2000

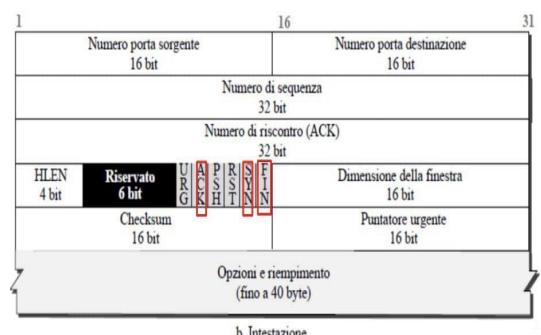
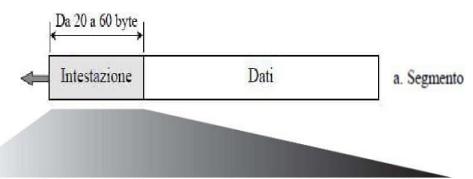


Questa informazione servirà per ricostruire il flusso di segmenti a destinazione. In generale si parte da un numero di sequenza generati in modo casuale (e quindi non 0). Il *numero di riscontro* indica il numero del prossimo byte che l'entità si aspetta di ricevere. Nell'ultimo esempio, il destinatario riceve il primo segmento, nell'ipotesi in cui possa mandare subito riscontro, manderà un segmento con numero di riscontro pari a  $999+1$  in modo tale da avere la sequenza ininterrotta.

### 3.3.1 Formato segmento TCP

Campi dell'intestazione del segmento:

- **Numero di porta sorgente**
- **Numero di porta destinazione**
- **Numero di sequenza**, contenente il numero attribuito al primo byte di dati contenuto nel segmento
- **Numero di riscontro**, contenente il numero di sequenza del byte che il destinatario si aspetta di ricevere
- **HLEN**, lunghezza dell'intestazione TCP espressa in parole da 4 byte
- **Dimensione finestra**, indica la dimensione della finestra di cui l'altro host coinvolto nella trasmissione deve disporre
- **Checksum**, dell'intero pacchetto per rilevare errori
- **Puntatore urgente**, se URG=1, punta al primo byte di dati *non* urgenti a partire dal numero di sequenza, e consente di far passare i dati urgenti in testa alla coda di ricezione
- **Opzioni**, campo facoltativo, negoziazione di vari parametri



- **Flags**, sono 6 e servono per:

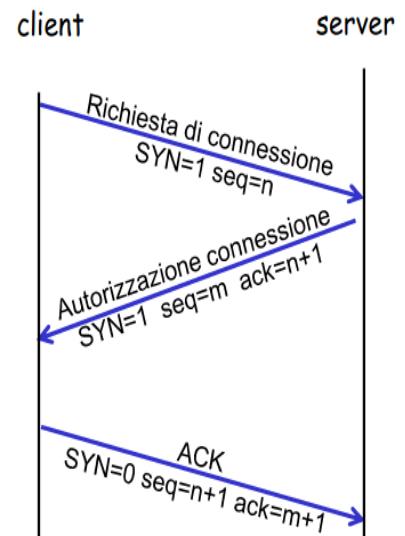
- **URG**, il campo puntatore urgente contiene dati significativi da trasferire in via prioritaria
- **ACK**, il numero di riscontro contiene dati significativi
- **PSH**, funzione push, trasferimento immediato dei dati in un segmento dal trasporto al livello applicativo
- **RST**, reset della connessione
- **SYN**, sincronizza il numero di sequenza
- **FIN**, non ci sono altri dati dal mittente, chiusura della connessione

### 3.3.2 La connessione TCP

Il protocollo TCP è orientato alla connessione, ovvero stabilisce un percorso virtuale fra il mittente e il destinatario. Utilizzare un percorso virtuale per l'intero messaggio semplifica il processo di conferma e di ritrasmissione dei segmenti smarriti o danneggiati. La trasmissione nel protocollo TCP richiede tre fasi: apertura della connessione, trasferimento dati e chiusura della connessione.

- **Apertura della connessione**, chiamata anche **three way handshake**, ovvero servono tre messaggi affinché i due capi della connessione si mettano d'accordo per l'instaurazione:

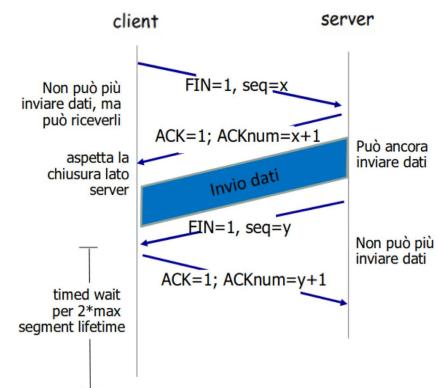
1. Il client invia un **segmento SYN** di richiesta apertura a un server TCP con flag SYN settato a uno e un numero di sequenza iniziale generato casualmente
2. Il server estrae il segmento, alloca i buffer e le variabili TCP per la connessione. In seguito invia il secondo **segmento**, chiamato **SYNACK**, di autorizzazione connessione con i due flag SYN e ACK attivati. Il server utilizza questo segmento per inizializzare il numero di sequenza necessario per numerare i byte inviati dal server al client, e per riscontrare la ricezione del segmento SYN dal client impostando il flag ACK e indicando il prossimo numero di sequenza che si aspetta di ricevere dal client.
3. Il client alloca buffer, variabili di connessione, e invia il terzo segmento, un **segmento ACK**, che conferma l'avvenuta ricezione del secondo segmento mediante il flag ACK. SYN è inattivo, inizia lo scambio dati. Alcune implementazioni consentono a questo terzo segmento di trasportare i primi dati dal client: in questo caso si deve utilizzare un numero di sequenza per indicare il numero del primo byte nei dati



Dopo l'handshaking a livello di trasporto non c'è più distinzione tra client e server. I segmenti SYN e SYNACK non contengono dati utente. Il fatto di avere tre messaggi serve per assicurarsi che il client e il server abbiano capito che l'altro ha ricevuto il messaggio e risponde che è pronto per andare avanti con la connessione

- **Chiusura della connessione**, client e server chiudono ciascun il loro lato della connessione. Il processo applicativo quando non ha più dati da mandare chiede al livello di trasporto di chiudere la connessione da quel lato. Questo si traduce in un segmento TCP con il bit FIN settato a uno. Implementazione per la chiusura della connessione:

1. Il client invia un **segmento FIN** con il flag FIN settato a 1 e un numero di sequenza  $x$ . Non trasporta dati
2. Il server notifica la situazione al suo processo applicativo e invia il **segmento FINACK** per riscontrare la ricezione del segmento FIN al client e per annunciare la chiusura della connessione nella direzione opposta. Il numero di riscontro è pari a  $x + 1$ . Questo segmento può anche contenere gli ultimi dati da parte del server; se non trasporta dati consuma un numero di sequenza  $y$
3. Il client invia l'ultimo segmento con ACK = 1 per riscontrare la ricezione del segmento dal server, e un numero di sequenza  $y+1$



Quindi ciascuno risponde al FIN ricevuto con un ACK. È possibile anche lo scambio simultaneo di FIN. La connessione tra i due processi applicativi non viene chiusa subito ma esiste uno stato finale **TIME-WAIT** dove si attende per un certo intervallo di tempo:  $2 \times \text{max segment lifetime}$ . È lo stato finale in cui il capo di una connessione che esegue la chiusura attiva resta prima di passare alla chiusura definitiva della connessione. La MSL è la stima del massimo periodo di tempo che un pacchetto IP può vivere sulla rete. Lo stato TIME-WAIT serve per implementare in maniera affidabile la terminazione della connessione in entrambe le direzioni e consentire l'eliminazione dei segmenti in rete. Uno dei due processi può smettere di inviare dati mentre ne sta ancora ricevendo, si tratta della cosiddetta **half close**. Questo può avvenire quando il server ha bisogno di tutti i dati prima di poter procedere alla loro elaborazione

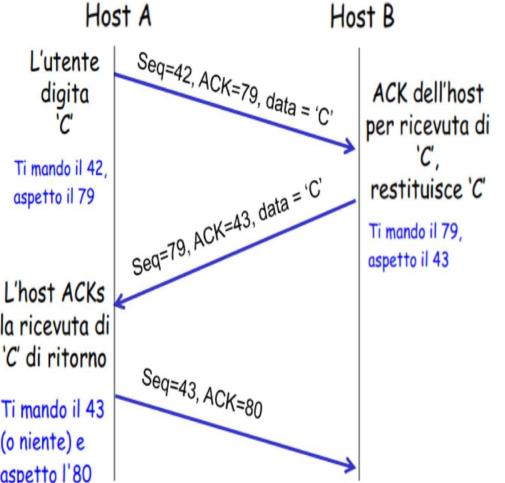
### 3.3.3 Trasferimento dati

Dopo aver stabilito la connessione è possibile trasferire i dati. TCP crea un servizio di trasferimento dati affidabile sul servizio inaffidabile di IP. Vi è il meccanismo di checksum per effettuare controlli in cui i segmenti corrotti vengono scartati. Quindi si ha una perdita in rete o a destinazione. Per gestire gli eventi di perdita vi è l'utilizzo del numero di sequenza, numero di riscontro, e inoltre:

- **Riscontro cumulativo**, si effettua il riscontro dei byte fino al primo byte mancante nel flusso
- **Timer**, intervallo di tempo per verificare se si riceve un riscontro del segmento che si è mandato. Se non si riceve un riscontro, allora si ha una perdita, e quindi si deve inviare di nuovo quel segmento

Esempio di TELNET su TCP con l'invio di 1 byte di dati, ovvero, senza numerazione dei segmenti:

1. Il client invia il primo segmento con numero di sequenza 42, numero di riscontro 79 e nel campo dati la lettera 'C'. Ogni carattere immesso dall'utente nel client verrà spedito all'host remoto; quest'ultimo restituirà una copia di ciascun carattere. Questo "eco" viene utilizzato per assicurarsi che i caratteri visibili all'utente siano già stati ricevuti ed elaborati nel sito remoto.
2. Il server invia il secondo segmento con numero di sequenza 79, ossia il numero iniziale del flusso di dati da server al client, il numero di riscontro 43, il server indica al client di aver ricevuto con successo i primi 42 byte e di attendere i byte da 43 in poi, e infine l'eco del campo dati
3. Il terzo segmento inviato dal client ha come unico scopo di dare un riscontro dei dati inviati dal server. Il segmento ha 80 nel suo campo ACK in quanto il client ha ricevuto il flusso di byte fino al numero di sequenza 79 e ora è in attesa dei byte dall'80 in avanti. Il campo dati è vuoto.



In particolari condizioni il mittente può inviare più segmenti senza attendere il riscontro e quindi permette di aumentare la produttività (*pipeline*). **Eventi lato mittente**: il TCP mittente riceve i dati dall'applicazione, incapsula i dati in uno o più segmenti, e assegna il numero di sequenza. TCP utilizza un **meccanismo di timeout e ritrasmissione RTO** per recuperare i segmenti persi: il TCP mittente ha un segmento da far partire, avvia il timer di ritrasmissione e fa partire il segmento. Se entro lo scadere del timeout non riceve riscontro allora ritrasmette quel segmento e rinvia il timer. Il timer non viene avviato per tutti i segmenti ma solo al primo o a un ristretto gruppo. Se il mittente riceve tre ACK duplicati, il segmento successivo a quello riscontrato è andato perso. In questo caso, il mittente senza aspettare lo scadere del timeout effettua una ritrasmissione. Questo evento si chiama **ritrasmissione veloce**. Il tempo di timeout RTO è fondamentale per il funzionamento di TCP, deve essere maggiore del tempo di **RTT**, ovvero, il tempo trascorso da quando si invia un segmento a quando se ne riceve il riscontro. RTO viene calcolato analizzando gli RTT dei segmenti non trasmessi

$$\text{EstimatedRTT} = (1 - \alpha)\text{EstimatedRTT} + \alpha(\text{SampleRTT})$$

Il *SampleRTT* (campioni) è la quantità di tempo che intercorre tra l'istante di invio del segmento e quello di ricezione dell'ACK del segmento, mentre, *EstimatedRTT* è una media ponderata dei valori di *SampleRTT*. Il valore di  $\alpha$  viene posto a 1/8 in modo da rendere via via meno importanti gli RTT dei pacchetti più vecchi. Oltre al valore stimato è necessario anche una stima della variabilità di RTT data dalla seguente formula:

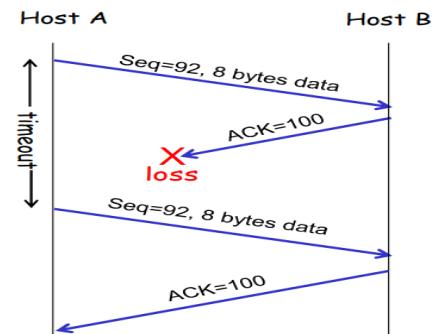
$$\text{RTT}_{\text{DEV}} = (1 - \beta)\text{RTT}_{\text{DEV}} + \beta | \text{RTT}_{\text{SAMPLE}} - \text{RTT}_{\text{ESTIMATED}} |$$

stima di quanto *SampleRTT* si discosta da *EstimatedRTT*. Il valore di  $\beta$  viene posto a 1/4. Una volta ottenuti questi valori, il time-out viene normalmente calcolato come

$$RTO = RTT_{ESTIMATED} + 4RTT_{DEV}$$

In molte implementazioni, dopo un errore, ad esempio un riscontro non ricevuto, si raddoppia il timeout: si tratta di un primo meccanismo di *controllo della congestione*. Esempio di ritrasmissione dovuta a riscontro perso:

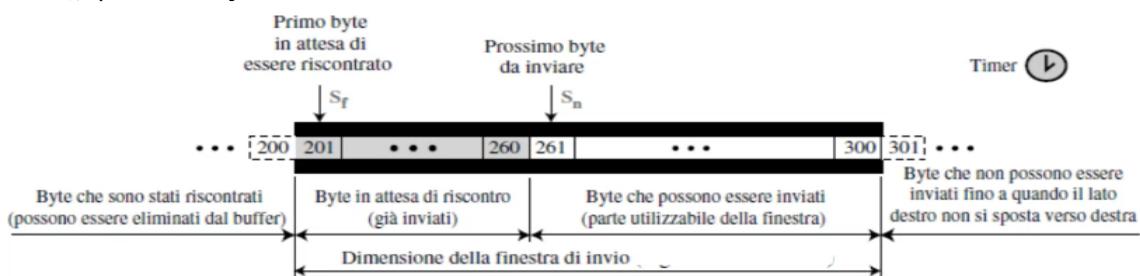
1. L'host A spedisce un segmento con numero di sequenza 92 e 8 byte di dati.
2. Dopo aver inviato il segmento, A attende un segmento dall'host B con un numero di riscontro 100, ma questo viene perso. In questo caso si verifica l'evento di timeout e l'host A ritrasmette lo stesso segmento.
3. L'host B riceve la ritrasmissione, rileva dal numero di sequenza che il segmento contiene dati che sono già stati ricevuti. Quindi B scarta i byte del segmento ritrasmesso.



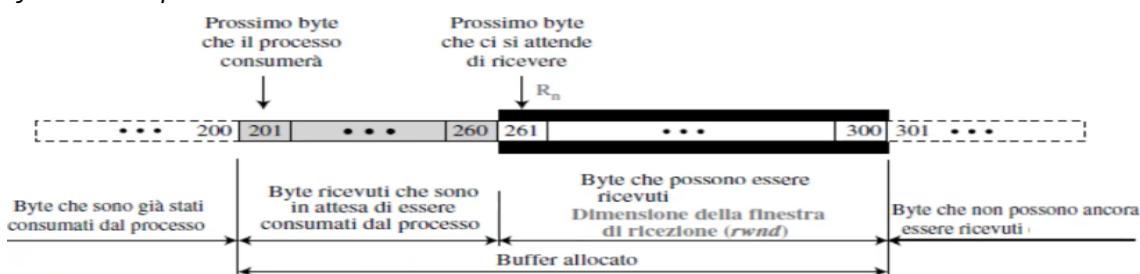
I segmenti possono arrivare fuori sequenza: non è obbligatorio ma l'implementazione del TCP può memorizzare questi segmenti (viene inserito nelle *opzioni*). In questo modo l'entità TCP destinataria avrà in un buffer la sequenza ininterrotta e in un altro buffer si mantengono i segmenti fuori sequenza, in modo tale da poter ricostruire la sequenza ininterrotta quando arriva un segmento mancante. Infatti il TCP non dice come il destinatario deve gestire i pacchetti fuori sequenza, ma dipende dall'implementazione. **Eventi lato destinatario**: esistono tali possibilità:

- Se il destinatario deve riscontrare dei segmenti ricevuti e anche inviare dei dati, allora si possono inviare le due cose insieme.
- Se il destinatario non ha dati da inviare e riceve un segmento "in ordine" ritarda l'invio dell'ACK di 500 ms a meno che non riceva un nuovo segmento. Questo per ottimizzare la ricezione dei dati che sono arrivati in due segmenti.
- Se il destinatario riceve un segmento atteso e il precedente non è stato riscontrato allora invia immediatamente l'ACK.
- Se il destinatario riceve un segmento fuori sequenza, oppure mancante, o duplicato, allora invia immediatamente un ACK

Il flusso di byte che il mittente invia al destinatario si può regolare attraverso la **finestra di invio**, e si distinguono i byte: *trasmessi e riscontrati*, *trasmessi e non riscontrati*, *trasmissibili* e *non trasmissibili*. La somma dei byte trasmessi e non riscontrati e trasmissibili è limitata dalla finestra di invio che varia dinamicamente. Ci sono delle variabili importanti per tenere lo stato di questa finestra: **Send First**  $S_f$ , primo byte in attesa di essere riscontrato e **Send Next**  $S_n$ , prossimo byte da inviare.



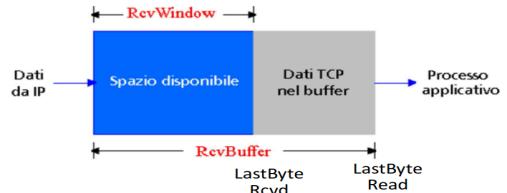
Anche il destinatario possiede un buffer di ricezione, chiamata **finestra di ricezione** composta da: *i byte già stati consumati*, *i byte ricevuti ed in attesa di essere consumati*, uno spazio vuoto per i *byte che possono essere ricevuti*, ed infine *byte che non possono ancora essere ricevuti*.



### 3.3.4 Controllo di flusso

Il **controllo di flusso** permette di evitare di non mandare al destinatario più byte di quelli che possa ricevere, ovvero, è la capacità del mittente di evitare la possibilità di saturare il buffer del ricevitore. Quindi è un meccanismo che mette in relazione la frequenza di invio del mittente con la frequenza di lettura dell'applicazione ricevente. TCP implementa il controllo di flusso tramite una variabile detta **rwnd**: questa variabile fornisce un'idea di quanto spazio è ancora a disposizione nel buffer del ricevitore:

- **RevBuffer** è la capacità del buffer
- **RevWindow** è la misura dello spazio disponibile nel buffer.  
Si calcola sottraendo da RevBuffer i byte che sono presenti nel buffer, ovvero  $\text{RevBuffer} - \text{LastByteReceived} - \text{LastByteRead}$



La variabile rwnd è dinamica. L'host destinatario comunica la dimensione di rwnd al mittente. Il mittente si assicura che  $\text{LastByteSent} - \text{LastByteAcked} \leq \text{rwnd}$ . Se rwnd = 0, il mittente manda segmenti "sonda" di 1 byte per ricevere l'aggiornamento sulla dimensione di rwnd

### 3.3.5 Controllo di congestione

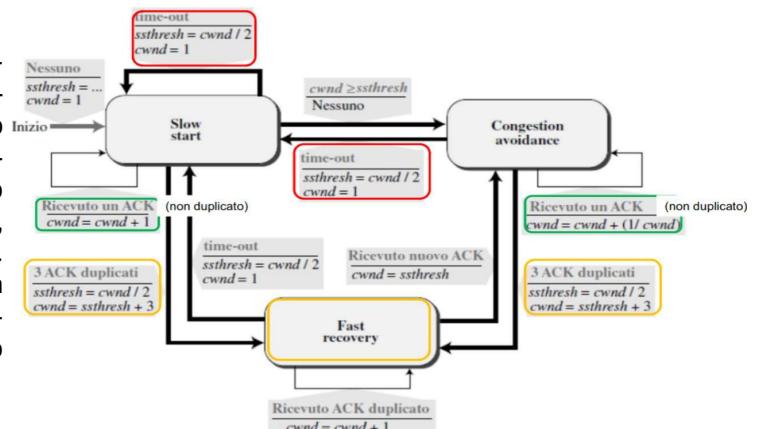
Il fenomeno della congestione è originato dal tentativo delle sorgenti di richiedere più banda di quella disponibile sul percorso, oppure, troppe sorgenti che trasmettono troppi dati a una velocità elevata per cui la rete non può gestirli. Il traffico eccessivo nella rete può provocare: lunghi ritardi (accodamenti nei buffer dei router) e/o perdita di pacchetti. Il protocollo TCP prevede il **controllo della congestione**, imponendo a ciascun mittente di limitare la frequenza di invio di pacchetti sulla connessione, in funzione della congestione percepita, e a quel punto cercare di adattarsi alla velocità della rete. Il controllo della congestione del TCP è principalmente punto-punto, quindi il meccanismo principale è quello di dedurre la congestione dai sistemi terminali, ovvero lo scambio dei segmenti. Nessun supporto esplicito della rete. Si combinano il controllo di flusso e di congestione: il minimo tra la **finestra** di ricezione **rwnd** e di **congestione cwnd** restituisce il limite della finestra di invio. La finestra di congestione **cwnd** si misura in **MSS** che è la quantità massima di dati trasportabili da un segmento. Un MSS è determinato in base alla **MTU** che è l'unità trasmisiva massima. Per trovare MSS si calcola MTU - header IP - header TCP. L'algoritmo che il mittente TCP utilizza per regolare la propria frequenza di invio in funzione della congestione rilevata, è costituita da tre passi:

- **Partenza lenta**, l'idea è quella di partire lentamente, ponendo all'inizio la finestra di congestione a 1 MSS, e poi di incrementare la finestra di congestione di 1 MSS ad ogni riscontro
- **Incremento additivo e decremento moltiplicativo**, TCP del mittente aumenta proporzionalmente la propria finestra di congestione ad ogni ACK ricevuto. Ad ogni riscontro la finestra di congestione viene incrementata in modo che si abbia una crescita pari a 1 MSS per ogni RTT. Ad ogni evento di perdita il TCP del mittente dimezza la finestra di congestione
- **Ripresa veloce**

Questi meccanismi servono per variare il cwnd: TCP non può avere una frequenza di invio dei dati superiore a cwnd/RTT.

### 3.3.6 TCP Reno

Una delle implementazioni per il controllo di congestione di TCP è il **TCP Reno**: si definisce inizialmente una variabile "soglia" alla quale è assegnato un valore alto. La soglia determina quando termina la partenza lenta e inizia la fase di controllo di congestione. Finché cwnd rimane sotto la soglia, cwnd aumenta esponenzialmente (partenza lenta). Se invece la finestra di congestione è maggiore della soglia, cwnd aumenta linearmente (incremento additivo). Nel TCP Reno gli eventi di perdita vengono trattati in questo modo:

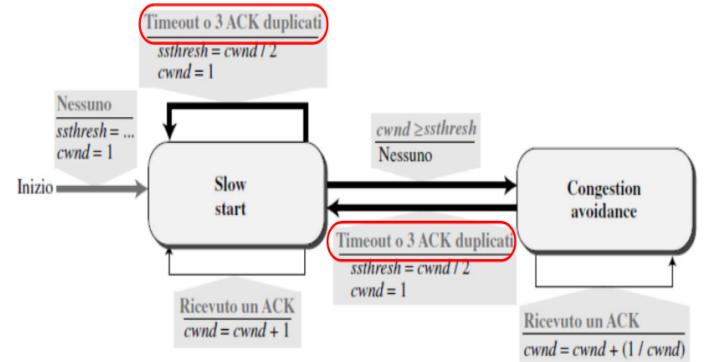


- Se vi sono 3 ACK duplicati, si pone prima la soglia a metà di cwnd e poi cwnd = soglia + 3 MSS (ripresa veloce)
- Se si è perso un ACK per timeout si pone la soglia a metà di cwnd e poi cwnd = 1 MSS (partenza lenta)

Nella ripresa veloce se avviene un timeout si va in partenza lenta, finché continuano ad arrivare riscontri duplicati cwnd++. Se arriva un nuovo riscontro non duplicato, si va in controllo di congestione e cwnd = soglia

### 3.3.7 TCP Tahoe

Si è studiato il TCP Reno ma in realtà ci sono varie implementazioni del controllo di congestione come il **TCP Tahoe** che è una versione precedente a Reno. TCP Tahoe utilizzava solo le prime due fasi: partenza lenta e controllo di congestione. Una volta aperta la connessione, TCP avvia l'algoritmo di partenza lenta e imposta la soglia ad un valore preconcordato e cwnd a 1 MSS. Se viene rilevata congestione (scadenza del timeout o ricezione di tre ACK duplicati), TCP fa ripartire l'algoritmo di partenza lenta con una nuova soglia uguale alla metà del valore corrente di cwnd. Se non viene rilevata congestione fino al raggiungimento della soglia, TCP si sposta nello stato di controllo congestione. Se la congestione viene rilevata in questo stato, TCP reimposta il valore della soglia alla metà del valore corrente di cwnd e si sposta nuovamente nello stato di partenza lenta.



### 3.3.8 TCP: throughput

Per stimare il throughput di una comunicazione TCP si indica con  $W$  il valore massimo in byte della finestra (ovvero quando si verifica un errore), TCP in regime stazionario offre il seguente throughput medio (frequenza di invio media)

$$\text{Throughput} = \frac{0.75 \cdot W}{RTT}$$

Quando la finestra è  $W$ , il throughput è  $W/RTT$ , dopo l'evento di perdita va a  $W/2$ , quindi il throughput è  $W/2RTT$

## 3.4 UDP

UDP è un protocollo del livello di trasporto inaffidabile e privo di connessione (non si introduce ritardo). Il motivo per cui un processo sceglie di utilizzare UDP è perché esso è un protocollo semplice e leggero da gestire con un carico aggiuntivo minimo. Se un processo vuole inviare un piccolo messaggio senza doversi preoccupare troppo dell'affidabilità, può utilizzare UDP. Inviare un messaggio con UDP richiede meno iterazioni fra il mittente e il destinatario rispetto all'impiego del TCP. Non vi è il controllo di congestione e di flusso. UDP viene utilizzato spesso in applicazioni multimediali dove c'è tolleranza per piccole perdite. I pacchetti UDP sono chiamati **datagrammi utente** e data l'assenza di una connessione vengono trasmessi in modo indipendente gli uni dagli altri. L'intestazione dei pacchetti UDP è costituita da 4 campi di 2 byte (16 bit) per un totale di 8 byte. I primi due campi definiscono i numeri di porta di mittente e destinatario. Il terzo campo definisce la lunghezza totale del datagramma compresa l'intestazione. L'ultimo campo può contenere il checksum opzionale.

16 bit	
source port #	dest. port #
length	checksum
<i>application data</i>	

### 3.4.1 Checksum UDP

Il checksum serve per effettuare un controllo end-to-end dell'informazioni presenti sia nell'intestazione che nel campo dati. Se un pacchetto risulta corrotto, viene scartato, ma il mittente non ne riceve notifica. La checksum viene calcolata sull'intero datagramma UDP più lo pseudo-header. Lo pseudo-header raccoglie delle informazioni che poi saranno veicolate nell'intestazione di livello rete, quindi non sono informazioni pienamente appartenenti al livello di trasporto. Se il checksum non includesse lo pseudo-header, il datagramma utente, pur non avendo subito danni durante la trasmissione, potrebbe finire all'host sbagliato per via di errori nel pacchetto IP che lo trasporta. Calcolo del checksum:

#### Mittente

- Campo checksum posto a 0
- Il contenuto del datagramma UDP viene diviso in blocchi da 16 bit
- Somma le parole da 16 bit in complemento a uno
- Effettua il complemento a uno del risultato della somma
- Il mittente pone il valore della checksum nel campo checksum del datagramma UDP

$$\begin{array}{r} 1011101110111011 + \\ 1000111100001111 = \\ \hline 0100101011001010 + \\ \hline \end{array}$$

↓  
1 =

$$\begin{array}{r} 0100101011001011 \\ 1011010100110100 \end{array}$$

#### Destinatario

- calcola la checksum ricevuta
- Se il valore della checksum è zero, il messaggio viene scartato

### 3.5 TCP vs UDP

Nella programmazione di rete si deve ricordare che:

- Il TCP offre un servizio di trasporto a stream, quindi si può leggere da un input di rete quanti byte si desiderano
- L'UDP offre un servizio a messaggi, quindi occorre leggere tutto il messaggio in arrivo

UDP è adeguato per processi che richiedono uno scambio di dati con volume limitato con scarso interesse al controllo di flusso e degli errori; processi che hanno meccanismi interni di controllo di flusso e degli errori; trasmissioni multicast e applicazioni interattive in tempo reale che non tollerano ritardi variabili

#### Servizio TCP

- *connection-oriented*: richiesto handshake tra client e server
- *trasporto affidabile*: tra i processi mittente e destinatario
- *controllo di flusso*: il mittente non saturerà il destinatario
- *controllo della congestione*: limita il mittente quando la rete è sovraccarica
- non fornisce: timing, banda minima garantita

#### Servizio UDP

- trasferimento dati non affidabile tra mittente e destinatario
- non fornisce: setup della connessione, affidabilità, controllo di flusso, controllo della congestione, timing, o garanzia di banda
- richiede minor overhead

### 3.6 Esercizi

**Esercizio 1.** Quale di queste applicazioni può funzionare usando i servizi di trasporto di UDP

- Streaming multimediale*
- Web*
- Email*
- Trasferimento file*

**Esercizio 2.** UDP è un protocollo connection-less perché

- Ciascun datagramma UDP è trattato in modo indipendente dall'altro*
- UDP gestisce i dati come uno stream di datagrammi correlati*

**Esercizio 3.** I datagrammi UDP hanno un header di lunghezza fissa pari a:

- 60 byte*
- 8 byte*
- 20 byte*

**Esercizio 4.** Quando un'applicazione riceve un blocco di dati dal TCP l'applicazione sa che dati sono stati mandati come un singolo messaggio dalla sorgente

- Vero*
- Falso*

**Esercizio 5.** Nella fase di invio del messaggio, nell'header di quale livello viene aggiunta la porta di destinazione?

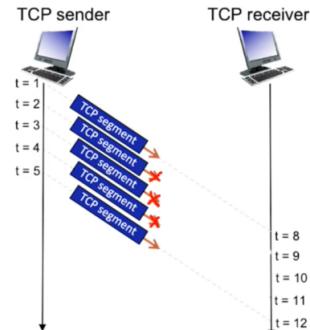
- Livello trasporto*
- Livello data link*
- Livello applicazione*
- Livello rete*

**Esercizio 6.** Oltre alla ritrasmissione dei pacchetti TCP implementa:

- Controllo di flusso*
- Controllo di congestione*
- Gestione della connessione*
- Tutti i servizi elencati*

**Esercizio 7.** Gli host A e B comunicano tramite una connessione TCP. A invia 5 segmenti. Si supponga che l'initial sequence number (ISN) con cui è stata aperta la connessione sia 500 e che i primi 5 segmenti contengono ciascuno 50 byte. Come mostrato in figura, 3 dei 5 segmenti non sono ricevuti da B.

- Indica il numero di sequenza dei 5 segmenti inviati da A e B
- Indica il numero di riscontro inviato da B in risposta ai segmenti ricevuti
  - a) 501, 551, 601, 651, 701    b) 551, 551
  - a) 500, 501, 502, 503, 504    b) 501, 505
  - a) 501, 502, 503, 504, 505    b) 502, 506
  - a) 500, 550, 600, 650, 750    b) 551, 551



**Esercizio 8.** In una connessione TCP A sta trasmettendo a B un segmento con numero di sequenza 68 e 20 byte di dati.

- In questo stesso segmento il numero di riscontro è necessariamente 88. Vero o falso?
- Ipotizzando che B riceve il segmento inviato da A e che B invii subito dopo un segmento ad A, questo conterrà un numero di riscontro pari in ogni caso a 88. Vero o falso?
  - a) Vero    b) Falso
  - a) Falso    b) Falso
  - a) Vero    b) Vero
  - a) Falso    b) Vero

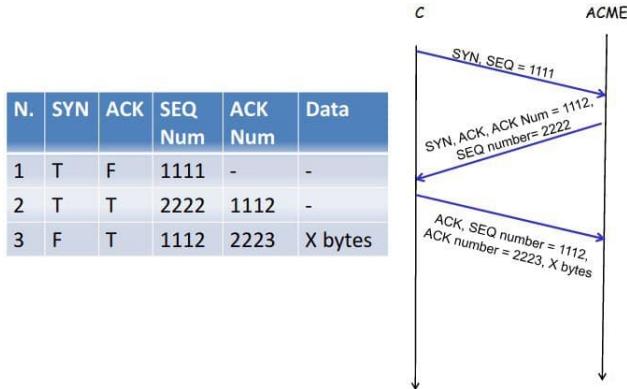
b) è falso perché non si hanno informazioni sullo scambio di segmenti precedenti, ad esempio il segmento precedente potrebbe non essere stato consegnato a B e quindi in quel caso B non può inviare un ack number 88

**Esercizio 9.** L'host A e B comunicano tramite una connessione TCP su cui sono stati già trasmessi correttamente alcuni segmenti. All'istante t A trasmette altri due segmenti a B. Il primo segmento ha numero di sequenza 2120 e il secondo 2150.

- Quanti dati sono trasportati nel primo segmento?
- Si ipotizza che il primo segmento vada perso e il secondo, contenente 500 bytes, arrivi a B. Qual è il numero di riscontro inviato da B ad A?

- Nessuna delle opzioni
- a) il primo segmento trasporta 30 byte   b) il numero di riscontro inviato è 2120
- a) il primo segmento trasporta 50 byte   b) il numero di riscontro inviato è 500
- a) il primo segmento trasporta 30 byte   b) il numero di riscontro inviato è 2620

**Esercizio 10.** Un client chiede la pagina web `www.acme.com/home/product.html` al server B di `www.acme.com` con una GET che è contenuta in un segmento TCP lungo X byte. Indicare - giustificando la risposta - i valori dei campi *sequence number*, *ack number*, e dei flags *ACK* ed *SYN* e lunghezze del campo *DATA*, in ciascuno dei segmenti C e B si scambiano per aprire la connessione nell'ipotesi che l'ack finale dell'apertura della connessione sia inviato in piggybacking assieme alla richiesta GET. Si supponga che in B ed in C rwnd (receive windows) sia molto grande, che non scada alcun timeout, che non ci siano errori di trasmissione, che nessun segmento vada perduto, e che il numero di sequenza iniziale di C sia 1111 e quello di B sia 2222

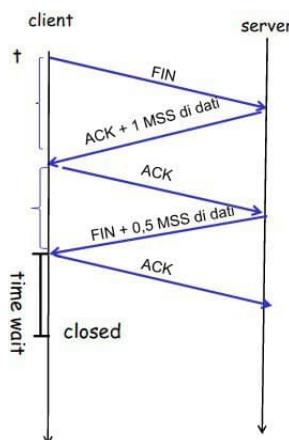


**Esercizio 11.** Un client C ha stabilito una connessione TCP con un server web S per scaricare una pagina web che consiste di tre oggetti. Al tempo  $t$ , subito dopo avere inviato la richiesta per il terzo oggetto, l'host di C invia a S un segmento con il flag FIN a true. Indicare - giustificando la risposta - il tempo minimo necessario al TCP di C per chiudere definitivamente la connessione supponendo che:

- la dimensione del terzo oggetto sia 1.5 MSS
- RTT sia costantemente 700 msec e il *maximum segment lifetime* sia 1100 msec
- tutti i segmenti vengano ricevuti corretti ed in ordine, e che il valore di cwnd del TCP sia 1MSS quando esso riceve il FIN inviato dall'host di C.

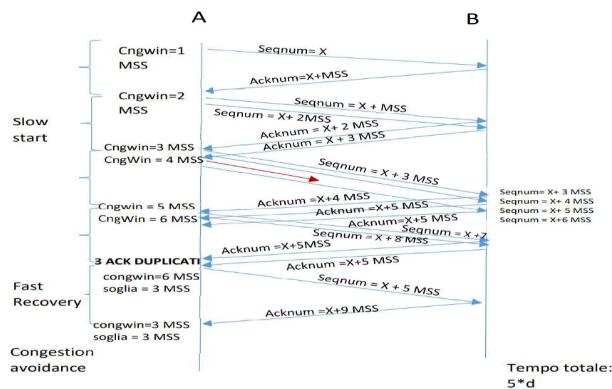
Trascurare i tempi di preparazione e di trasmissione dei segmenti e assumere che S abbia già a disposizione tutti i dati da inviare

*Il TCP di C riceve al tempo  $t + RTT$  il riscontro S1 del segmento FIN da lui inviato. Se S1 trasporta in piggybacking il primo MSS dei dati del terzo oggetto e il TCP di C invierà un riscontro C2 per tali dati, S potrà inviare l'ultima porzione di dati in un segmento con il flag FIN a true. Il TCP di C invierà un riscontro del FIN e considererà chiusa la connessione dopo aver atteso 2MSL, ovvero al tempo  $t + 2RTT + 2MSL = t + 3600msec$*



**Esercizio 12.** L'host A invia un file che consiste in 9 segmenti TCP full sized (i.e. ciascun segmento trasporta una quantità di dati pari a 1 MSS) a B su una connessione TCP. Il sesto segmento della trasmissione viene perso. Considerando TCP RENO, mostrare con un diagramma lo scambio di segmenti tra A e B indicando i valori di SEQ e ACK, e eventuali cambiamenti nel valore di congwin, soglia e stato del TCP Reno di A. Indicare infine il tempo totale per l'invio del file. Si assume che la connessione TCP sia stata instaurata, che non siano stati ancora inviati dati, che RTT valga  $d$  ms e, per semplicità, che il *retransmission timeout* sia  $> 2d$ . Si assuma inoltre che venga riscontrato ciasun segmento ricevuto (no ack posticipato). Indicare eventuali ipotesi semplificative fatte.

*Si ipotizza che la receive window sia sempre maggiore di congestion window e quindi trascurabile. Il trasferimento dati inizia nella fase di slow start con cong win = 1 MSS e soglia alta (es: 64 KB). Valore di sequenza per A all'inizio del trasferimento è X. B non invia dati utili ad A, quindi si può trascurare i sequence number per i segmenti da B ad A e riscontro da A. La congestion window aumenta esponenzialmente ad ogni RTT (1 MSS per ogni ACK) finché non viene ricevuto un ACK duplicato. Al terzo ACK duplicato il TCP va in FAST RECOVERY e reinvia il segmento mancante*



**Esercizio 13.** Dire in quali delle seguenti circostanze il TCP cambia la dimensione della finestra di congestione, e nel caso, come viene calcolata

Stato	Evento	Cambia la finestra?	Nuova dimensione della finestra
Slow Start	Ricezione di un ACK duplicato	No	-
Slow Start	Scatta un timeout	Si	1 MSS
Congestion Avoidance	Scatta un timeout	Si	1 MSS

**Esercizio 14.** Su un collegamento insistono una comunicazione TCP e una UDP. La banda del collegamento verrà condivisa equamente tra le due comunicazioni. Vero o Falso?

- Vero*  
 *Falso*

**Esercizio 15.** Un segmento TCP può avere una parte dati lunga zero byte?

- No*  
 *Sì*

**Esercizio 16.** In una connessione TCP quale segmento ha sia il flag SYN che il flag ACK settati? NB Si contano i segmenti in entrambe le direzioni

- terzo*  
 *quarto*  
 *secondo*  
 *primo*

**Esercizio 17.** Nel demultiplexing in un servizio ORIENTATO ALLA CONNESSIONE (TCP) il messaggio viene consegnato alla socket identificata da:

- solo indirizzo IP e porta destinazione
- indirizzo IP e porta destinazione e indirizzo IP e porta mittente
- solo indirizzo IP e porta mittente

**Esercizio 18.** Indicare quale delle seguenti affermazioni sulla checksum in UDP è VERA

- In UDP la checksum è calcolata considerando solo l'intestazione
- In UDP la checksum è opzionale e fornisce controllo degli errori end-to-end
- In UDP la checksum è obbligatoria e fornisce controllo degli errori ad ogni hop lungo il cammino da sorgente a destinazione

**Esercizio 19.** Si consideri TCP Reno. Il TCP di un host ha cwnd=10 MSS e soglia=20 MSS. Indicare cosa succede dopo i seguenti eventi: timeout e ricezione di tre riscontri non duplicati

- La finestra di congestione assume il valore di 4 MSS e la soglia 5 MSS
- La finestra di congestione assume il valore di 1 MSS e la soglia 4 MSS
- La finestra di congestione assume il valore di 8 MSS e la soglia 20 MSS
- La finestra di congestione assume il valore di 1 MSS e la soglia 20 MSS

# Chapter 4

## Livello di rete

### 4.1 Comunicazione e servizi

Il livello rete offre servizi allo strato di trasporto e utilizza i servizi dello strato di collegamento. È responsabile della consegna dei datagrammi tra gli host. Comunicazione a livello rete:

1. l'entità a livello rete riceve i segmenti dal livello di trasporto nell'host mittente, e incapsula i segmenti in datagrammi
2. i datagrammi sono inoltrati al prossimo nodo (host o router)
3. il router esamina i campi intestazione in tutti i datagrammi IP che lo attraversano e li inoltra da un collegamento in ingresso ad un collegamento in uscita
4. sul lato destinatario, consegna i segmenti al livello di trasporto (TCP o UDP)

A differenza del livello trasporto che è implementato negli host, il livello di rete è implementato nei nodi intermedi. Inoltre il livello di rete interconnette reti eterogenee e implementa poche funzioni. I servizi forniti a livello rete sono: **suddivisione in pacchetti; instradamento (routing)** processo decisionale di scelta del percorso verso una destinazione; **inoltro (forwarding)** trasferimento del pacchetto sull'appropriato collegamento di uscita; **controllo di errori; controllo di flusso; controllo di congestione; qualità del servizio; sicurezza** e anche a livello di rete si effettua multiplexing/demultiplexing

### 4.2 Internet Protocol

Il protocollo IP è un protocollo di tipo connection-less, nasce come un **servizio best effort** ma non garantisce che i pacchetti vengano ricevuti nell'ordine in cui sono stati inviati e non è garantita la consegna. Quindi non è affidabile e non prevede meccanismi di recupero di errore (*send and pray*). Inoltre non prevede garanzie sulla qualità del servizio, sul tempo di consegna dei datagrammi e sul controllo di flusso. Esistono però dei meccanismi che sono stati implementati per la qualità del servizio e il controllo della congestione (esempio *DiffServ*). Meccanismi IP:

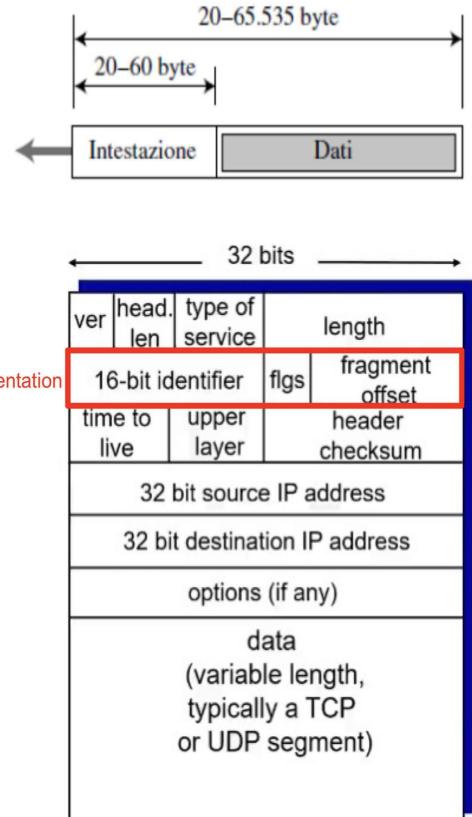
- **Indirizzamento**, strumento per identificare gli host nell'internet
- **Modello datagram**, per la consegna dei dati. Il datagramma permette a segmentare e riunificare dei pacchetti, controllo degli errori e verifica TTL

#### 4.2.1 Formato datagramma IP

I pacchetti usati dal protocollo IP vengono definiti **datagrammi IP**. Un datagramma è un pacchetto di lunghezza variabile composto da due parti: un'intestazione (header) e un campo dati (payload). I campi dell'intestazione sono:

- **Numero versione**, definisce la versione del protocollo IP (IPv4 o IPv6)
- **Lunghezza dell'intestazione**, il campo lunghezza intestazione (HLEN) formato da 4 bit definisce la lunghezza totale dell'intestazione del datagramma in parole da 4 byte

- **Tipo servizio**, sono 8 bit di cui 6 bit per i **servizi differenziati**, marcano il pacchetto in base alla classe di servizio, e altri 2 bit che implementano un meccanismo chiamato **explicit congestion notification** per il supporto a livello di rete e trasporto per la notifica di eventi di congestione
- **Lunghezza totale**, definisce la lunghezza totale (intestazione + dati) e può essere al massimo 16 bit
- **Identificatore, flag e offset**, sono campi per la frammentazione dei datagrammi IP
- **Time-to-live**, indica il massimo numero di salti (hop), cioè il numero di router visitati dal datagramma. Viene settato ad un certo valore (per esempio dal mittente) e ogni volta che il datagramma viene elaborato da un router, questo campo viene decrementato. Quando arriva a zero, il router scarta il datagramma. Valori tipici: 30, 64, 128, 255
- **Protocollo**, fornisce il multiplexing alla sorgente e il demultiplexing a destinazione
- **Checksum dell'intestazione**, viene calcolato il checksum solo dell'intestazione ad ogni router. Se si ottiene un errore si scarta il datagramma. IP lascia l'onere del controllo degli errori nei dati trasmessi al protocollo che è proprietario del payload, come ad esempio TCP o UDP
- **Indirizzi sorgente e destinazione**, sono campi lunghi 32 bit
- **Opzioni**, fino a 40 byte che possono essere usati per il test o il debug della rete
- **Payload**



#### 4.2.2 Frammentazione

Ogni protocollo di livello collegamento ha il proprio formato di frame. Una delle caratteristiche di ciascun formato è la dimensione massima del payload che può essere incapsulato nel frame: **Maximum Transfer Unit**. La MTU pone un limite alla lunghezza dei datagrammi IP e tecnologie/percorsi diversi possono porre limiti differenti.



Se un router riceve un datagramma la cui dimensione supera l'MTU della rete verso cui deve inoltrare quel datagramma, il router frammenta il datagramma IP in due o più datagrammi più piccoli, detti **frammenti**. Ciascun frammento è a sua volta un datagramma IP completo che viene trasmesso attraverso una serie di reti fisiche indipendentemente dagli altri. Il processo di frammentazione può essere ripetuto, ad esempio se un frammento deve essere inoltrato su un collegamento con MTU ancora più piccolo. Il riassemblaggio viene effettuato dall'entità rete a destinazione. Se i frammenti arrivano fuori ordine, il livello rete per ricomporre i frammenti nel pacchetto originale usa:

- **Identificatore** - 16 bit, identifica un datagramma che ha origine da un host sorgente. Quando un datagramma viene frammentato il valore nel campo identificazione viene copiato in tutti i frammenti ottenuti. Il numero di identificazione aiuta la destinazione a riassemblare il datagramma
- **Flag** - 3 bit, serve ad identificare l'ultimo frammento
  - il bit 0 è riservato
  - il bit 1 vale 0 se il pacchetto può essere frammentato, 1 altrimenti
  - il bit 2 vale 0 se il pacchetto è l'ultimo frammento, 1 altrimenti

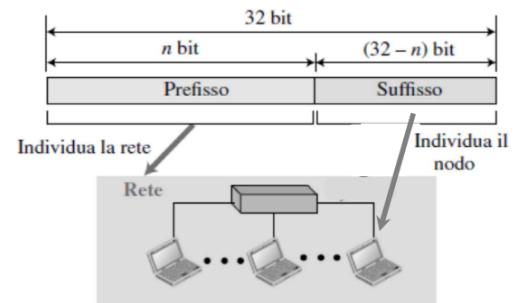
- **Offset** - 13 bit, mostra la posizione relativa di un frammento rispetto all'intero datagramma. I frammenti devono essere multipli di 8 byte e il primo frammento parte da offset uguale a 0

La ricostruzione del datagramma è un processo critico perché aumenta "la probabilità" di perdere un frammento e quindi la possibilità che un destinatario non possa ricostruire il datagramma. Una soluzione che è stata trovata per risolvere il problema è quello di non frammentare, infatti vi è il bit 1 "do not fragment" in IPv4, mentre IPv6 non supporta la frammentazione. Se qualche datagramma non arriva a destinazione, si butta via tutto il datagramma.

#### 4.2.3 Indirizzamento IP

Ogni host è connesso ad Internet attraverso un'interfaccia di rete, che è il confine fra l'host ed il collegamento su cui vengono inviati i datagrammi. Ad ogni interfaccia è assegnato un indirizzo IP. I router sono tipicamente connessi ad almeno due reti, quindi devono necessariamente essere connessi ad almeno due collegamenti. Gli indirizzi IP sono costituiti da 32 bit e possono essere rappresentati come sequenza di bit oppure in notazione decimale puntata. Ogni host ha un indirizzo univoco diviso in due parti:

- **Network ID (prefisso)** identifica una rete IP
- **Host ID (suffisso)** identifica l'host su quella rete IP.

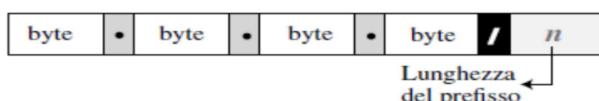


Il sistema di identificazione delle reti in IPv4 è stato inizialmente progettato come prefisso a lunghezza fissa. Questo schema, ormai obsoleto, viene indicato come **indirizzamento con classi**, dove l'intero spazio degli indirizzi era diviso in cinque classi:

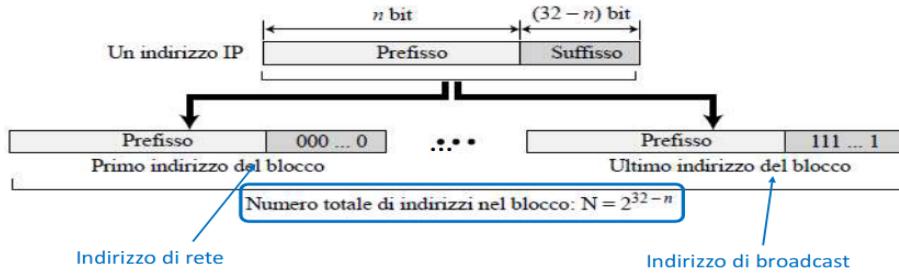
- **Classe A**, la lunghezza della parte di rete è di 8 bit, ma siccome il primo, che è 0, identifica il tipo di classe, si possono avere solo sette bit per l'identificazione delle reti. Ciò significa che ci sono solo  $2^7 = 128$  reti al mondo che possono avere un indirizzo di classe A
- **Classe B**, la lunghezza della parte di rete è di 16 bit, ma siccome i primi due bit, che sono  $(10)_2$  identificano la classe, possiamo avere solo 14 bit per identificare reti. Ciò significa che ci sono solo  $2^{14} = 16.384$  reti al mondo che possono avere un indirizzo di classe B
- **Classe C**, la lunghezza della parte di rete è di 24 bit, ma siccome tre bit definiscono la classe, si possono avere solo 21 bit per identificare le reti. Ciò significa che ci sono  $2^{21} = 2.097.142$  reti al mondo che possono avere un indirizzo di classe C. Raccoglie tutti gli indirizzi che iniziano con  $(110)_2$
- **Classe D**, non è divisa in prefisso e suffisso ed è usata per gli indirizzi di tipo **multicast**. Raccoglie tutti gli indirizzi che iniziano con  $(1110)_2$
- **Classe E**, non è divisa in prefisso e suffisso, e raccoglie tutti gli indirizzi che iniziano con  $(1111)_2$ . Riservata per uso futuro



La ragione per cui l'indirizzamento con classi è diventato obsoleto è l'esaurimento degli indirizzi. Il nuovo schema, chiamato **indirizzamento senza classi**, usa un prefisso di rete di lunghezza variabile. Nell'indirizzamento senza classi l'intero spazio degli indirizzi è diviso in blocchi di lunghezza variabile. Data l'assenza delle classi, la lunghezza del prefisso  $n$  viene aggiunta all'indirizzo separata da una barra slash. Tale notazione è definita **CIDR**. Esempio: 12.24.76.8/8 i primi 8 bit identificano l'indirizzo di rete, mentre, l'indirizzo dell'host in questa rete è 12.24.76.8



Quindi in un indirizzo IP,  $n$  bit sono per l'indirizzo di rete, i restanti  $32 - n$  bit identificano l'host in questa sottorete. A disposizione per gli host si hanno  $N = 2^{32-n}$  indirizzi. Non tutti sono assegnabili, in particolare il primo indirizzo del blocco con i bit del suffisso tutti settati a 0 (indirizzo di rete) e l'ultimo indirizzo del blocco con i bit del suffisso tutti settati a 1 (indirizzo di broadcast). Una rete con un prefisso di rete di  $n$  bit può avere al massimo  $2^{32-n} - 2$  host (si sottrae l'indirizzo di rete e di broadcast).



Un altro modo per distinguere quale parte di un indirizzo IP identifica la rete e quale l'host è la **subnet mask**, cioè un numero composto da 32 bit in cui i primi  $n$  bit a sinistra sono impostati a 1 e i rimanenti a 0

**Esempio 1.** L'indirizzo IP 150.217.8.42 ha netmask 255.255.255.0. Per trovare l'indirizzo di rete il calcolatore effettua una operazione di AND

Indirizzo IP	150.217.8.42	10010110	11011001	00001000	00101010
Subnet Mask	255.255.255.0	11111111	11111111	11111111	00000000
AND	150.217.8.0	<b>10010110</b>	<b>11011001</b>	<b>00001000</b>	<b>00000000</b>
La rete ha indirizzo <b>150.217.8.0/24</b>					

**Esempio 2.** Gli indirizzi 150.193.4.1/22, 150.193.8.18/22 e 150.193.11.3/22 appartengono alla stessa rete? Il modo più veloce è confrontare solo i 22 bit più a sinistra delle tre diverse sequenze

150.193.4.1/22	<b>10010110</b>	<b>11000001</b>	<b>00000100</b>	00000001
150.193.8.18/22	<b>10010110</b>	<b>11000001</b>	<b>00001000</b>	00010010
150.193.11.3/22	<b>10010110</b>	<b>11000001</b>	<b>00001011</b>	00000011

e verifica così che il primo indirizzo non è sulla stessa rete degli altri due

#### 4.2.4 Assegnamento dei blocchi di indirizzi

Il problema successivo nell'indirizzamento senza classi è rappresentato dall'assegnazione dei blocchi. Gli indirizzi IP sono gestiti da ICANN che non assegna indirizzi a singoli utenti Internet ma si occupa di assegnare grandi blocchi di indirizzi a ISP. Per il corretto funzionamento del CIDR devono essere applicate due restrizioni al blocco assegnato:

1. Il numero di indirizzi in ogni subnetwork deve essere una potenza di 2. La lunghezza del prefisso di ogni sottorete va calcolata con la formula  $n = 32 - \log_2 N$  dove  $N$  è il numero di indirizzi della sottorete
2. Si assegnano blocchi di indirizzo contigui. Se i blocchi sono di dimensioni diverse si parte dai blocchi più grandi

**Esempio 1.** Un ISP deve suddividere un blocco in 8 indirizzi (un blocco di 512 indirizzi per ciascuna organizzazione):

Blocco dell'ISP	<b>11001000</b>	<b>00010111</b>	<b>00010000</b>	00000000	200.23.16.0/20
-----------------	-----------------	-----------------	-----------------	----------	----------------

I primi 20 bit sono l'indirizzo di rete, quindi non li può cambiare. Tuttavia può organizzare lo spazio rimanente nei restanti 12 bit:  $2^{12}$ . Si calcola la lunghezza del prefisso:  $n = 32 - \log_2 512 = 23$

Organizzazione 0	<b>11001000</b>	<b>00010111</b>	<b>00010000</b>	00000000	200.23.16.0/23
Organizzazione 1	<b>11001000</b>	<b>00010111</b>	<b>00010010</b>	00000000	200.23.18.0/23
Organizzazione 2	<b>11001000</b>	<b>00010111</b>	<b>00010100</b>	00000000	200.23.20.0/23
...					
Organizzazione 7	<b>11001000</b>	<b>00010111</b>	<b>00011110</b>	00000000	200.23.30.0/23

**Esempio 2.** Un ISP richiede un blocco di 190 indirizzi e ottiene il blocco 14.24.74.0/24 (256 indirizzi). A questo punto l'ISP vuole ora partizionare il blocco ottenuto in tre sottoblocchi da 120, 60 e 10 indirizzi rispettivamente.

- Si parte dal sottoblocco più grande, la potenza di 2 maggiore di 120 è  $2^7 = 128$ . Quindi  $32-7 = 25$  bit per la rete. Il blocco è 14.24.74.0/25:

**00001110.00011000.01001010.00000000**

Primo indirizzo: 14.24.74.0

Ultimo indirizzo 14.24.74.127

- Secondo blocco, la potenza di 2 maggiore di 60 è  $2^6 = 64$ . Quindi  $32-6 = 26$  bit per la rete. Il blocco 14.24.74.128/26:

**00001110.00011000.01001010.10000000**

Primo indirizzo: 14.24.74.128

Ultimo indirizzo 14.24.74.191

- Terzo blocco, la potenza di 2 maggiore di 10 è  $2^4 = 16$ . Quindi  $32-4 = 28$  bit per la rete. Il blocco 14.24.74.192/28:

**00001110.00011000.01001010.11000000**

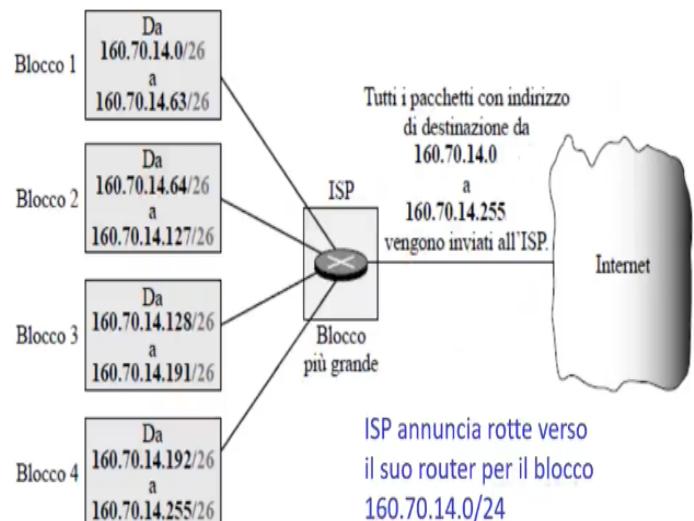
Primo indirizzo: 14.24.74.192

Ultimo indirizzo 14.24.74.207

#### 4.2.5 Aggregazione degli indirizzi

Con il CIDR si ha la possibilità di aggregare gli indirizzi per rendere più efficiente l'instradamento. La figura mostra come un ISP abbia assegnato a quattro organizzazioni dei piccoli blocchi di indirizzi. L'ISP combina questi quattro blocchi in un singolo blocco e annuncia al resto della rete solamente quest'ultimo blocco. L'annuncio indica che ogni pacchetto destinato a questo blocco più grande deve essere inviato a questo ISP. Sarà poi responsabilità dell'ISP inoltrare il pacchetto all'organizzazione corretta. Esistono 5 indirizzi speciali che vengono usati per scopi particolari:

- **This-host - 0.0.0.0**, usato quando un host ha necessità di inviare un datagramma ma non conosce il proprio indirizzo IP (indirizzo sorgente)
- **Limited-broadcast - 255.255.255.255**, usato quando un router o un host devono inviare un datagramma a tutti i dispositivi che si trovano all'interno della rete. I router bloccano la propagazione alla sola rete locale
- **Loopback - 127.0.0.1**, il datagramma con questo indirizzo di destinazione non lascia l'host local (localhost). Per test e debug
- **Indirizzi privati**, quattro blocchi riservati per indirizzi privati: 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16, 169.254.0.0/16
- **Indirizzi multicast**, blocco 224.0.0.0/4



#### 4.2.6 Dynamic Host Configuration Protocol

Un indirizzo IP può essere attribuito all'interfaccia di un host secondo due modalità: **configurazione manuale**, l'amministratore configura direttamente nell'host l'indirizzo IP ed inserisce ulteriori informazioni (indirizzo router e indirizzo IP di almeno un server DNS) o **DHCP**, un host quando si aggiunge ad una rete ottiene dinamicamente un indirizzo IP da un programma server in rete. DHCP è un protocollo client-server che utilizza UDP:

1. L'host invia in broadcast un messaggio **DHCP discover**. Questo messaggio viene encapsulato in datagramma con indirizzo sorgente impostato a 0.0.0.0 e indirizzo destinazione 255.255.255.255. Vi è anche un campo *transaction-ID* che mette in correlazione le richieste del client con le risposte del server
2. Il server DHCP risponde con un messaggio **DHCP offer** contenente l'indirizzo IP offerto per l'host e l'indirizzo IP del server. Il messaggio contiene anche un **lifetime** cioè per quanto tempo l'host potrà far uso di quell'indirizzo

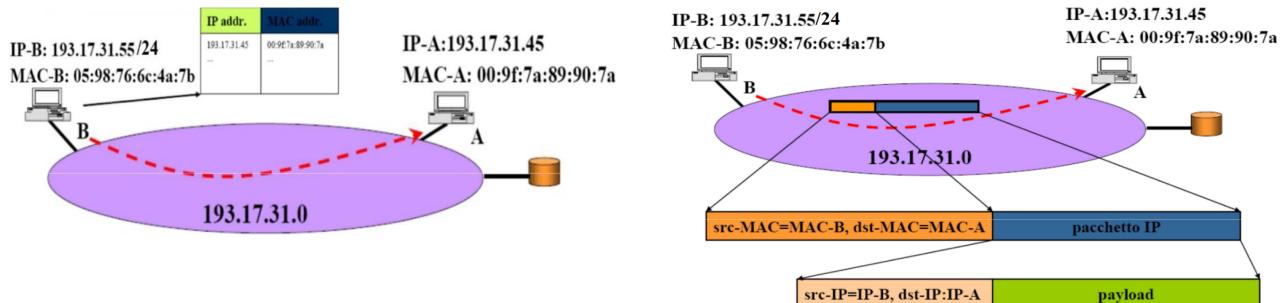
3. A questo punto l'host può effettivamente richiedere l'indirizzo IP al server con un messaggio di tipo **DHCP request**.
4. Infine il server DHCP risponde con un messaggio **DHCP ACK** se la richiesta va a buon fine. Se il server non è più in grado di mantenere la sua offerta, invia un messaggio di tipo **DHCP NACK** e il client deve ripetere il procedimento

Il passo 1-2 sono opzionali perché ad esempio se all'host già aveva ricevuto un indirizzo IP e deve richiedere che gli sia rinnovato, può fare direttamente una richiesta e aspettare la conferma del DHCP ACK.

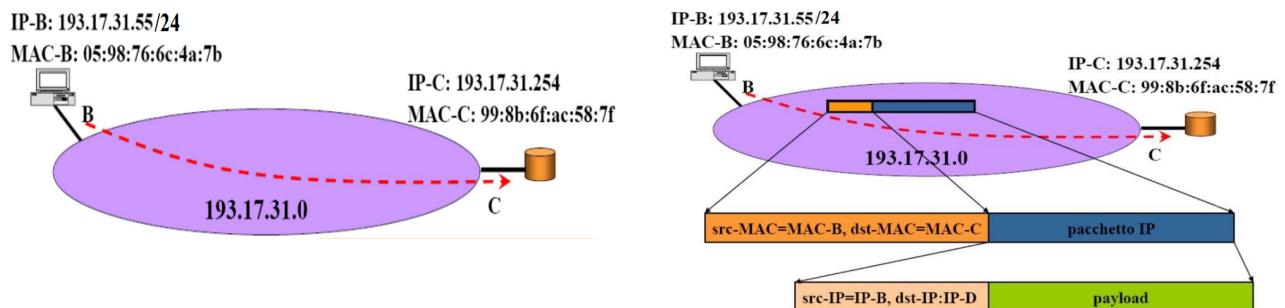
#### 4.2.7 Inoltro dei datagrammi IP

Ogni datagramma IP è soggetto a *forwarding* da parte dell'host di origine e del router che sta attraversando. Vi sono due tipi:

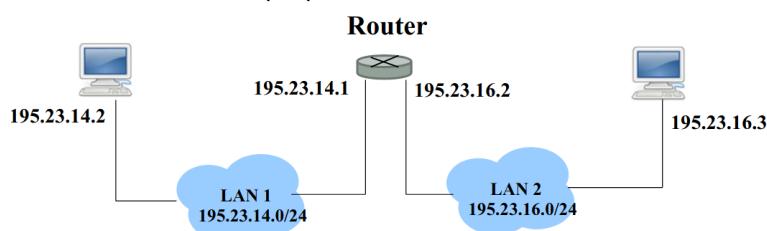
- **Inoltro diretto**, il pacchetto IP ha come destinazione un host nella propria rete IP. Il datagramma non esce dalla rete e l'invio è diretto sul destinatario, non viene interpellata nessun'altra entità. Esempio: l'entità IP di B deve spedire un pacchetto all'indirizzo IP-A. B conosce l'indirizzo IP-B della propria interfaccia e dal confronto con IP-A capisce che A si trova nella stessa rete. B consulta una tabella di corrispondenza tra indirizzi IP e indirizzi della rete (indirizzi MAC nel caso di rete locale) per reperire l'indirizzo MAC-A. L'entità IP di B passa il pacchetto al livello inferiore che crea un pacchetto con destinazione MAC-A



- **Inoltro indiretto**, il pacchetto IP ha come destinazione un host di un'altra rete IP. Viene delegato l'invio del datagramma ad un altro nodo, ovvero un router. Esempio: l'entità IP di B deve spedire un pacchetto all'indirizzo IP-D = 131.17.23.4. B conosce l'indirizzo IP-B della propria interfaccia e dal confronto con IP-D capisce che D non si trova nella stessa rete. B deve dunque inoltrare il pacchetto ad un router, di solito è configurato un solo default router. B recupera l'indirizzo MAC del router nella tabella di corrispondenza e passa il pacchetto al livello inferiore. Il pacchetto viene costruito e spedito sull'interfaccia



In entrambi i casi, condizioni necessarie perché tutto funzioni sono che: esista un cammino funzionante diretto, a livello collegamento, tra tutti gli host che appartengono ad una stessa sottorete e ogni host coinvolto abbia un indirizzo IP "giusto", cioè con un uguale *network ID*, e con *host ID* univoco nella sottorete. Ad ogni interfaccia verso la rete IP viene assegnato un indirizzo IP distinto. Il router è un apparato che svolge funzioni di *instradamento* a livello IP. Esso legge gli indirizzi IP, consulta la propria **tabella di inoltro** e decide dove mandare il pacchetto IP.



Una tabella di inoltro include: gli indirizzi delle reti raggiungibili (in notazione CIDR), una colonna per indicare se l'inoltro è diretto o indiretto (se non è indicato nulla si intende diretto), e infine l'interfaccia del router, ovvero l'interfaccia attraverso cui un datagramma deve essere inoltrato. Una volta arrivato il pacchetto, il router estrae l'indirizzo di destinazione, effettua la ricerca nella tabella per vedere se vi è una corrispondenza. In caso positivo inoltra il datagramma nell'interfaccia e con la modalità indicata dalla tabella. Normalmente l'ultima riga ha un **valore di default** nella prima colonna che rappresenta tutti gli indirizzi di destinazione che non combaciano con le righe precedenti.

**Esempio 1.** Un router R1 interconnette quattro sottoreti, e riceve un datagramma con IP destinazione 180.70.65.140:

10110100 01000110 01000001 10001100

1. La prima maschera (/26) è applicata all'indirizzo di destinazione

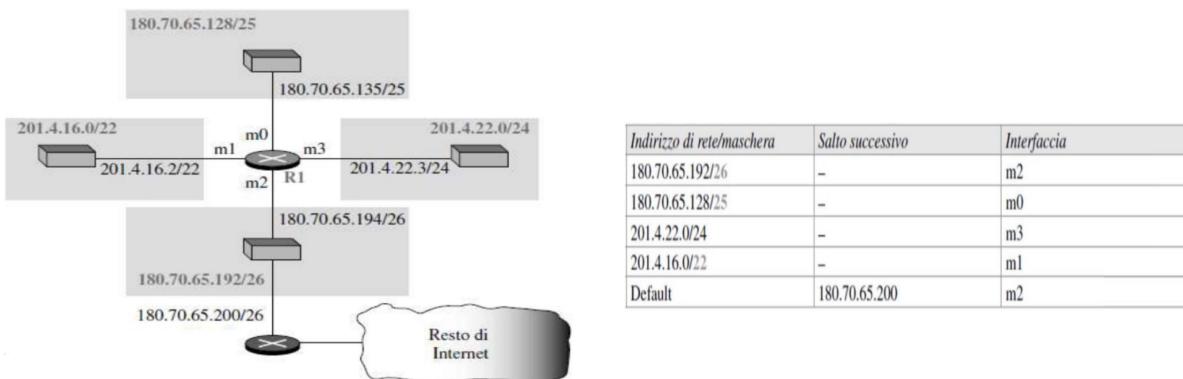
10110100	01000110	01000001	10001100
<b>11111111</b>	<b>11111111</b>	<b>11111111</b>	<b>11000000</b>
10110100	01000110	01000001	10000000
180	70	65	128

che non combacia con l'indirizzo di rete corrispondente

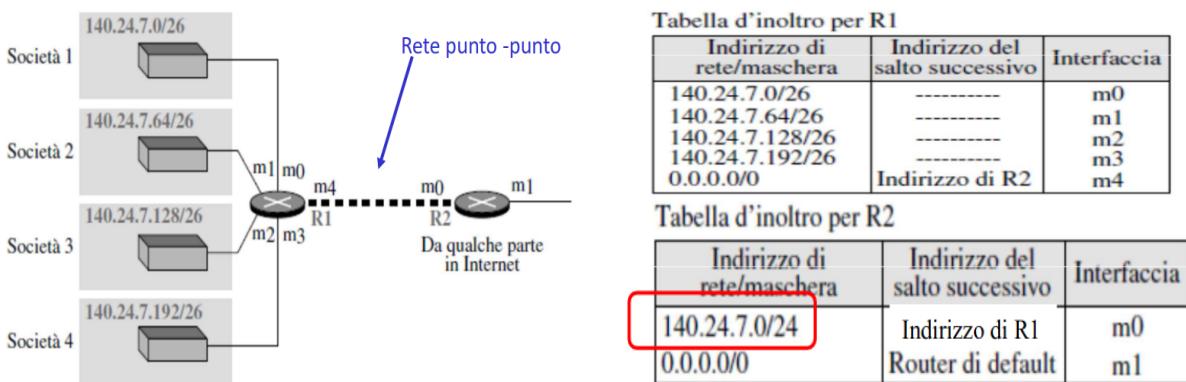
2. La seconda maschera (/25) si applica all'indirizzo di destinazione

10110100	01000110	01000001	10001100
<b>11111111</b>	<b>11111111</b>	<b>11111111</b>	<b>10000000</b>
10110100	01000110	01000001	10000000
180	70	65	128

che combacia con l'indirizzo di rete corrispondente, quindi si inoltra il datagramma in modalità diretta all'interfaccia m0



**Esempio 2.** Vi sono due router: R1 è connesso alle reti di quattro organizzazioni e R2 è da qualche parte in Internet lontano da R1. Per R2 qualunque datagramma con indirizzo di destinazione compresa da 140.24.7.0 a 140.24.7.255 è inviato attraverso l'interfaccia m0 (aggregazione degli indirizzi)



L'instradamento nell'indirizzamento senza classi sfrutta un altro principio, la **corrispondenza con la maschera più lunga**. Questo principio afferma che la tabella d'inoltro viene ordinata dalla maschera più lunga a quella più corta. In altre parole se ci sono tre maschere /27, /26 e /24, la maschera /27 deve essere nella prima riga e la /24 nell'ultima. Questo perché la maschera più lunga è più specifica

**Esempio 3.** Se un pacchetto arriva al router R2 con indirizzo di destinazione 140.24.7.200. La prima maschera nel router R2 è applicata e da l'indirizzo di rete 140.24.7.192. Il datagramma è quindi instradato correttamente all'interfaccia  $m_1$  e raggiunge la società 4. Se la tabella di inoltro non fosse memorizzata con il prefisso più lungo per primo, applicando la maschera /24 si avrebbe un instradamento sbagliato e l'invio del datagramma al router R1.

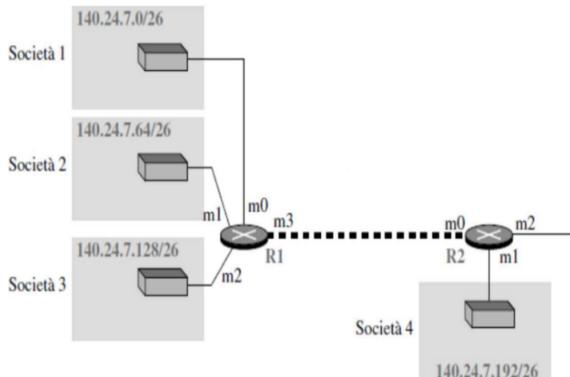


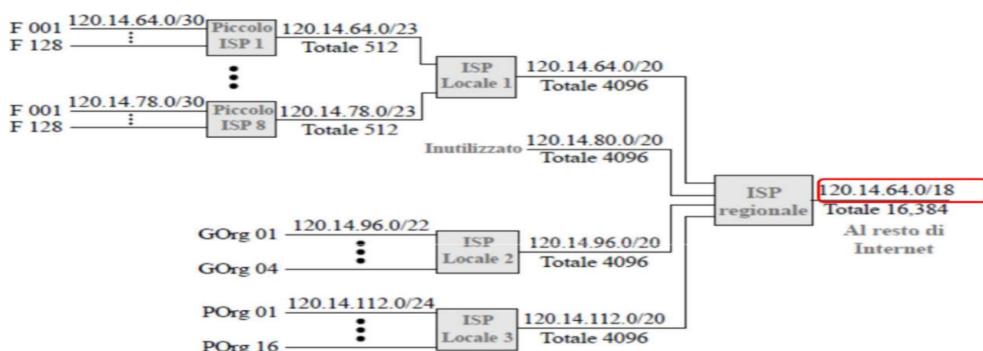
Tabella d'inoltro per R1

Indirizzo di rete/maschera	Indirizzo del salto successivo	Interfaccia
140.24.7.0/26	-----	$m_0$
140.24.7.64/26	-----	$m_1$
140.24.7.128/26	-----	$m_2$
0.0.0.0/0	Indirizzo di R2	$m_3$

Tabella d'inoltro per R2

Indirizzo di rete/maschera	Indirizzo del salto successivo	Interfaccia
140.24.7.192/26	-----	$m_1$
140.24.7.0/24	Indirizzo di R1	$m_0$
0.0.0.0/0	Router di default	$m_2$

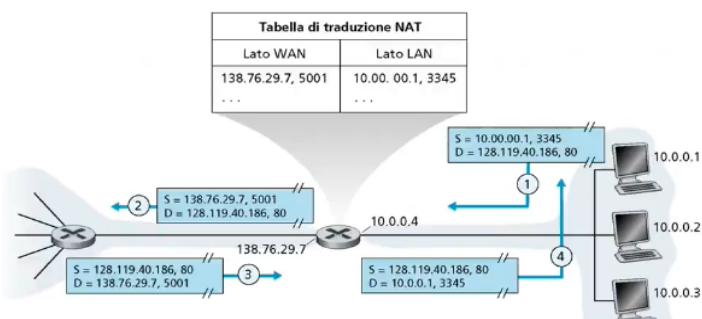
Per risolvere il problema delle tabelle di inoltro di dimensione eccessiva è possibile implementare una sorta di **router gerarchico** nelle tabelle di inoltro. Se la tabella d'inoltro ha qualche forma di gerarchia, analoga all'architettura di Internet, può diminuire di dimensione. Per esempio, ad un ISP regionale sono stati assegnati 16.384 indirizzi che partono da 120.14.64.0. ISP regionale ha deciso di dividere questo blocco in 4 sottoblocchi, ciascuno con 4096 indirizzi. Tre di questi sottoblocchi sono assegnati a tre ISP locali; il quarto invece è riservato per uso futuro. Gli ISP locali possono anche loro suddividere i blocchi in blocchi più piccoli, di diverse dimensioni, e assegnarli ad utenti individuali e organizzazioni di varie dimensioni. Il resto di Internet non è consapevole di tale divisione (e non gli interessa).



#### 4.2.8 Network Address Translation

Per abilitare il traffico verso/dalla rete Internet per una rete privata, si utilizza il **NAT**, che consente di usare una serie di indirizzi privati per la comunicazione interna e una serie di indirizzi Internet globali (almeno uno) per la comunicazione con il resto del mondo. Tutto il traffico *in uscita* dal router di accesso ha un indirizzo IP sorgente pubblico, quello del router. Tutto il traffico *in ingresso* alla sottorete ha come indirizzo IP di destinazione l'indirizzo pubblico del router di accesso. Tradurre l'indirizzo sorgente di un datagramma in uscita è banale, mentre non lo è per un datagramma in entrata. Il problema viene risolto dotando il router NAT di una **tabella di traduzione NAT**, le cui righe contengono le associazioni: (IP privato, porta) (IP pubblico e porta assegnata dal router).

- l'host 10.0.0.1 invia un datagramma a 128.119.40.186, 80
- il router NAT cambia l'indirizzo sorgente da 10.0.0.1, 3345 a 138.76.29.7.5001 e aggiorna la tabella
- arriva il messaggio di risposta con indirizzo di destinazione 138.76.29.7, 5001
- il router NAT controlla la tabella e modifica l'indirizzo di destinazione da 138.76.29.7, 5001 a 10.0.0.1, 3345



## 4.3 Internet Control Message Protocol

L'IPv4 non implementa alcun meccanismo per segnalare gli errori o correggerli e inoltre è sprovvisto di un meccanismo per effettuare richieste sullo stato di un sistema remoto. L'**ICMPv4** è stato creato per porre rimedio a queste carenze. Esso viene usato da host e router per scambiarsi messaggi di errore o altre situazioni che richiedono intervento. L'ICMPv4 è un protocollo di livello di rete ma i suoi messaggi non vengono passati direttamente al livello di collegamento ma vengono incapsulati all'interno di datagrammi IP prima di essere passati al livello inferiore. Gli host che implementano il protocollo IP devono supportare anche ICMP. I pacchetti ICMP vengono instradati dai router prima dei pacchetti IP ordinari, ed esistono delle regole:

- per pacchetti IP frammentati, i messaggi ICMP sono relativi al solo frammento con offset 0 (il primo frammento)
- i messaggi ICMP non sono mai inviati in risposta a pacchetti IP con indirizzo mittente che non rappresenta in modo univoco un host (es: 0.0.0.0, 127.0.0.1)
- i messaggi ICMP non sono mai inviati in risposta a pacchetti IP con destinazione IP broadcast o multicast

I messaggi ICMPv4 si distinguono in **messaggi di segnalazione errori** e **messaggi di richiesta/risposta**. Nell'header ICMP i campi *tipo* (8 bit) indica il tipo del messaggio e il campo *codice* (8 bit) indica invece il significato del messaggio. Una delle applicazioni che un host può utilizzare per verificare il funzionamento di un altro host è il programma **ping** che si basa sui messaggi di richiesta e risposta *echo* dell'ICMP. Un host invia una richiesta *echo* (tipo 8, codice 0) a un altro host che, se attivo, può rispondere con una risposta *echo* (tipo 0, codice 0). A questo punto l'host sorgente si può calcolare l'RTT. In maniera molto grossolana, il programma ping può anche misurare l'affidabilità e la congestione del router tra due host inviando una sequenza di messaggi richiesta/risposta.

ICMP Tipo	Codice	Descrizione
0	0	risposta al messaggio di echo (a ping) - <i>echo replay</i>
3	0	rete di destinazione irraggiungibile - <i>destination network unreachable</i>
3	1	host di destinazione irraggiungibile - <i>destination host unreachable</i>
3	2	protocollo di destinazione irraggiungibile - <i>destination protocol unreachable</i>
3	3	porta di destinazione irraggiungibile - <i>destination port unreachable</i>
3	6	rete di destinazione sconosciuta - <i>destination network unknown</i>
3	7	host di destinazione sconosciuto - <i>destination host unreachable</i>
4	0	strozzamento della sorgente (controllo della congestione) - <i>source quench</i>
8	0	richiesta di echo - <i>echo request</i>
9	0	annuncio dal router - <i>router advertisement</i>
10	0	scoperta del router - <i>router discovery</i>
11	0	TTL scaduto - <i>TTL expired</i>
12	0	cattiva intestazione IP - <i>IP header bad</i>

### 4.3.1 Tracerout

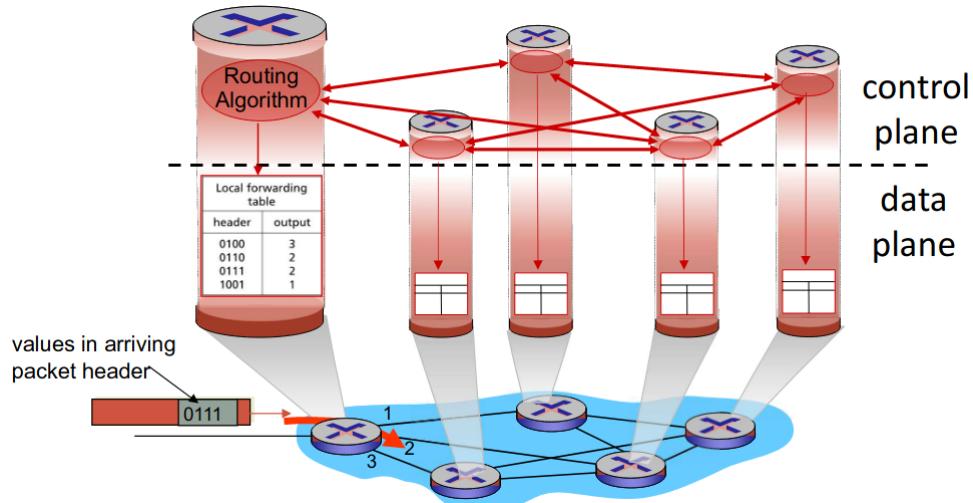
Il programma **traceroute** in UNIX può essere utilizzato per individuare il percorso di un datagramma dalla sorgente alla destinazione tramite l'identificazione dell'indirizzo IP di tutti i router che vengono visitati lungo il percorso. Solitamente il programma viene impostato per un massimo di 30 salti (router), che sono usualmente sufficienti per raggiungere la destinazione. Tracerout utilizza il protocollo UDP, quindi invia un datagramma ad una porta sui cui è improbabile che un processo sia in ascolto. I datagrammi sono configurati in modo da scatenare l'invio di messaggi di errore di due tipi: **tempo scaduto** e **porta destinazione non raggiungibile**. Se vi sono  $n$  router nel percorso, il programma traceroute invia  $n+1$  datagrammi con valori di TTL crescenti. I primi  $n$  messaggi vengono scartati dagli  $n$  router con messaggio di errore di *tempo scaduto*, uno per ogni router (il primo datagramma avrà TTL uguale a 1, il secondo uguale a 2, ...); l'ultimo messaggio è scartato dall'host di destinazione con messaggio di errore di *porta non raggiungibile*. Il programma traceroute impone inoltre un timer per trovare il tempo di round-trip di ciascun router e della destinazione. La maggior parte dei programmi traceroute invia tre messaggi a ogni dispositivo, con lo stesso valore di TTL, per poter effettuare una stima migliore del tempo di round-trip.

## 4.4 Data plane e control plane

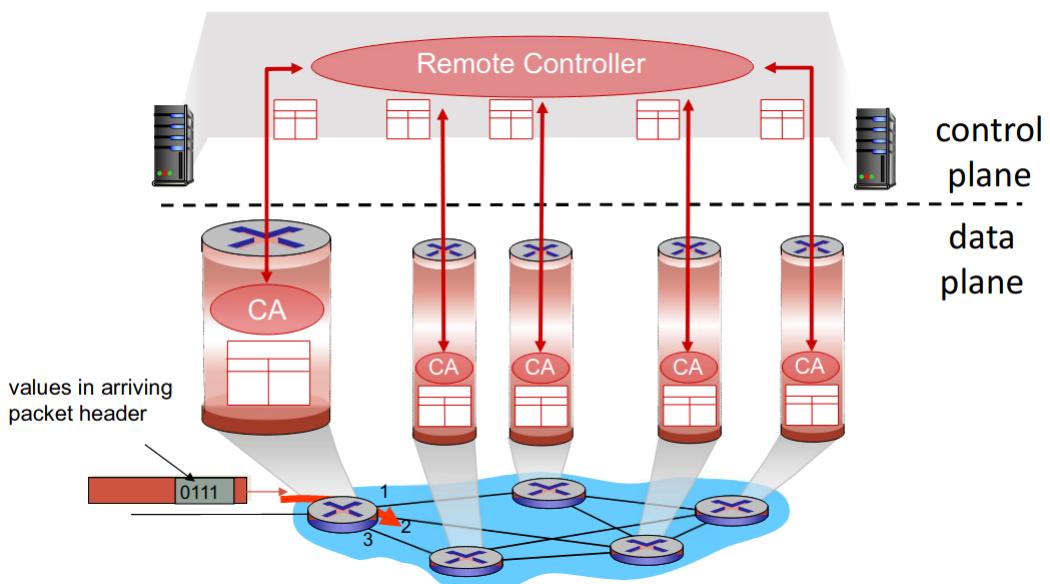
Quando si parla di rete si parla sempre di:

- **Forwarding**, trasferire i pacchetti sull'appropriato collegamento in uscita e si utilizza la tabella di inoltro. Questo viene effettuato a livello di **data plane**, ovvero la parte del router che si occupa del forwarding
- **Routing**, processo decisionale di scelta del percorso verso una destinazione. Determina i valori da inserire nella tabella di inoltro tramite **algoritmi di routing**. Questa parte decisionale viene eseguita a livello di **control plane**

I router devono acquisire delle informazioni dagli altri router per costruire un modello della rete, e su questo utilizzano l'algoritmo di routing. Quindi è un **approccio decentralizzato** dove non viene definito un unico punto di controllo ma ogni router ha sia la parte di *control plane* che di *data plane*, in modo tale che se un nodo diventa non più disponibile, gli altri nodi possono comunque continuare a lavorare



Da una decina di anni si sta affermando un approccio di **routing logicamente centralizzato** dove nei dispositivi di rete, la parte di control plane viene rimossa dal router e viene effettuata da un **remote controller** che può ricevere informazioni dai router per costruirsi la topologia della rete, costruirsi i percorsi e per ciascun router definire le tabelle di inoltro per poi inviarle ai router. Quindi c'è un punto decisionale che possiede tutta la parte di controllo e ai router resta come ruolo fondamentale quello del data plane. I router hanno una parte di controllo, chiamato **control agent** che comunica con il remote controller. In ogni caso questo approccio non sta sostituendo il routing decentralizzato, ma lo sta solo integrando.

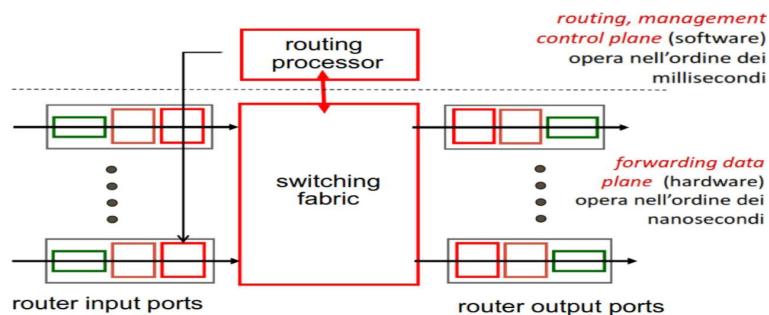


#### 4.4.1 Componenti router

All'interno di un router si possono identificare i seguenti componenti:

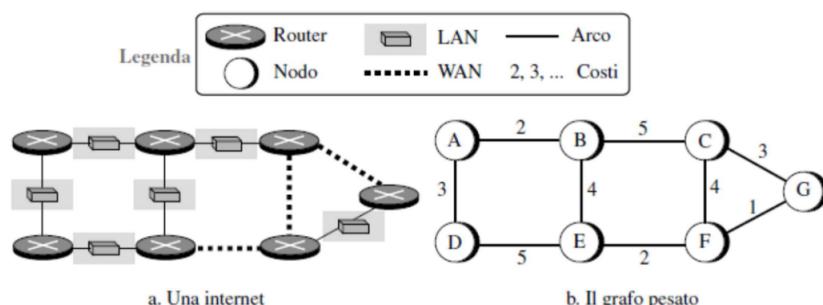
- **Porte di input**, dove vi sono tre parti fondamentali: *terminazione elettrica di linea*, ovvero la terminazione fisica del collegamento e dove si ricevono i bit; una parte per l'*elaborazione a livello di collegamento*, dove si ricevono i frame e infine vi è una parte dove si effettua la decisione di forwarding. La tabella di inoltro viene definita a livello di control plane e una copia viene mantenuta ad ogni porta in modo tale da decentralizzare e rendere più veloce la ricerca. Deciso qual è la porta di output dove deve essere trasferito il datagramma, si manda il datagramma nella struttura di commutazione (*switching fabric*), cioè la struttura che mette in comunicazione con la porta di output. Se la velocità con cui arrivano i datagrammi supera la velocità di inoltro nella struttura di commutazione, i pacchetti vengono messi in coda in dei buffer. Un parametro che contraddistingue la struttura di commutazione è la *velocità di commutazione* cioè la velocità con cui i pacchetti possono essere trasferiti dagli ingressi alle uscite. Esistono varie implementazioni di switching fabric come: *memory*, *bus* e *matrice di commutazione*
- **Porte di output**, è speculare alla porta di input e contiene un buffer in cui vengono inseriti i datagrammi in attesa di essere trasmessi come frame, quindi elaborati dal livello collegamento, e poi inviati come bit sul collegamento. Si effettua anche uno *scheduling* cioè l'ordinamento dei datagrammi in base a delle classi di priorità

I buffer vengono utilizzati per avere uno spazio per accodare i pacchetti che non si possono servire subito. Quando un pacchetto viene messo in coda, si ha un ritardo di accodamento. I buffer hanno dimensione finita e quando lo spazio disponibile finisce, i pacchetti vengono persi. Questo può avvenire sia nelle porte di input che di output



## 4.5 Routing

In una rete come Internet lo scopo del livello di rete è quello di consegnare un datagramma dalla sorgente alla destinazione o alle destinazioni. Se un datagramma è destinato ad una sola destinazione si parla di **routing unicast**. Se il datagramma è destinato a numerose destinazioni si parla di **routing multicast**. Infine, se il datagramma deve essere consegnato a tutti gli host della rete si parla di **routing broadcast**



Gli algoritmi di routing lavorano su delle rappresentazioni astratte, in particolare su dei grafi pesati, dove i nodi sono i router e gli archi rappresentano i collegamenti tra i router. I pesi degli archi rappresentano distanza tra i nodi, numero di salto, larghezza di banda disponibile, o altre metriche di latenza... Con la notazione  $c(x, x')$  si indica il costo dell'arco tra  $x$  e  $x'$ , ad esempio  $c(D, E) = 5$ . Per calcolare il costo tra  $x_s$  e  $x_d$ , se essi non sono adiacenti, si calcola come la somma dei costi dei nodi intermedi. Per esempio  $c(E, G) = 3$ . Non è detto che esiste un solo

percorso tra due nodi, quindi la domanda chiave dell'instradamento è *qual è il cammino a costo minimo tra due nodi?* Gli algoritmi di instradamento calcolano il cammino a costo minimo tra un nodo sorgente e un nodo destinazione. Il routing nel piano di controllo può essere:

- **Statico**, le entry della tabella vengono configurate manualmente dall'operatore. Tale metodo viene usato per reti di piccole dimensioni e la cui topologia non varia molto, dove è possibile prevedere tutti i possibili percorsi di un pacchetto IP nella rete
- **Dinamico**, esistono protocolli specifici che provvedono automaticamente ad inserire nella tabella del router le entry relative ai possibili percorsi. Viene usato nelle reti di medie-grandi dimensioni e con topologia variabile

Gli algoritmi di instradamento si possono classificare in:

- **Globali**, il router si costruisce la conoscenza della topologia di tutta la rete e a questo punto calcola i percorsi verso tutti i nodi della rete e i costi di tutti i collegamenti. Esempio di algoritmo globale:

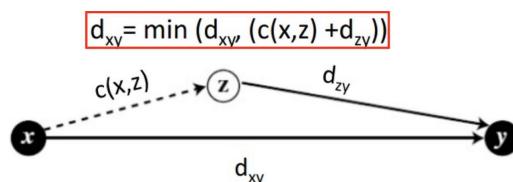
**Link State** ogni nodo si costruisce la vista della rete e quindi conosce lo stato di ciascun collegamento. Quindi tutti i nodi dispongono delle stesse informazioni e si calcolano con l'*algoritmo di Dijkstra* il cammino a costo minimo da un nodo origine a tutti gli altri nodi della rete. È un algoritmo iterativo, ovvero, dopo la  $k$ -esima iterazione i cammini a costo minimo verso  $k$  nodi di destinazione sono noti. I **link-state packet** sono dei pacchetti che notificano a tutti gli altri nodi lo stato dei collegamenti. Per esempio  $B$  può inviare un pacchetto in cui dice che il suo costo verso  $A$  è 2, e viceversa nella direzione opposta. Questa è una vista semplificata degli scambi dei messaggi, perché in realtà, un nodo può inviare ad un altro nodo anche tutto il **link-state database** che possiede, e quindi tutta

l'informazione sulla topologia della rete. Questo succede per esempio quando entra un nuovo nodo nella rete. L'algoritmo è composto dai seguenti passi:

1. si parte da un nodo  $u$  qualsiasi e lo si inserisci nell'insieme  $N' = \{u\}$ . Per tutti i nodi vicini  $v$ , se  $v$  è adiacente a  $u$  allora  $D(v)$  il costo del cammino da  $u$  a  $v$ , è dato da  $c(u, v)$  ovvero il costo del collegamento dal nodo  $u$  a  $v$ , altrimenti se non sono adiacenti  $D(v) = \infty$ .
2. a questo punto si entra in un loop in cui si cerca di determinare un nodo  $w$  che non è stato ancora inserito nell'insieme  $N'$  tale che  $D(w)$  sia minimo. Allora si aggiunge  $w$  a  $N'$  e si aggiornano le distanze verso tutti i nodi adiacenti a  $w$  e non ancora in  $N'$  applicando la formula  $D(v) = \min(D(v), D(w) + c(w, v))$  (*condizioni di Bellman*)
3. si ripete il passo 2 fino a che tutti i nodi sono in  $N'$

- **Decentralizzati**, quando nessun nodo conosce la topologia di tutta la rete, ma ha informazioni solo sui nodi e i collegamenti vicini. Ogni nodo elabora un vettore di stima dei costi verso tutti gli altri nodi della rete e il cammino a costo minimo viene calcolato in modo distribuito e iterativo scambiandosi informazioni con i nodi vicini. Esempio di algoritmo decentralizzato:

**Distance Vector** si basa sulla *equazione di Bellman-Ford* che trova il percorso a costo minimo tra un nodo sorgente  $x$  e un nodo destinazione  $y$ , tramite dei nodi intermedi vicini  $v$  supponendo che il costo  $c(x, v)$  tra nodo sorgente  $x$  e il vicino  $v$  sia noto e la distanza minima tra vicino  $v$  e destinazione  $y$  sia nota



$\min_v$ : minimo su tutti i vicini  $v$  di  $x$   
 $c(x, v)$ : costo da  $x$  al vicino  $v$   
 $d_{vy}$ : costo dal vicino  $v$  alla destinazione  $y$   
 $D_{xy}$ : stima del costo minimo da  $x$  a  $y$  per ogni  $y \in N$

Ciascun nodo  $x$  inizia con una stima delle distanze verso tutti i nodi in  $N$  e mantiene le seguenti informazioni:

- il costo verso ciascun vicino  $v$

- il suo vettore distanza  $D_x = [D_{xy} : \text{per ogni } y \in N]$
- i vettori distanza dei suoi vicini, per ogni vicino  $v$ ,  $x$  mantiene  $D_v = [D_{vy} : \text{per ogni } y \in N]$

L'idea di base dell'algoritmo è che ogni nodo invia una copia del proprio vettore distanza a ciascuno dei suoi vicini. Quando un nodo  $x$  riceve un nuovo vettore distanza  $DV$  da qualcuno dei suoi vicini  $v$ , lo salva e usa l'equazione di Bellman-Ford per aggiornare il proprio vettore distanza  $D_{xy} = \min\{c(x, v) + D_{vy}\}$  per ogni  $y \in N$ . Finché tutti i nodi continuano a cambiare i propri  $DV$  in maniera asincrona, ciascuna stima dei costi  $D_{xy}$  converge al costo minimo  $d_{xy}$ . Quindi è algoritmo **iterativo** e **asincrono** dove ogni iterazione locale è causata da un cambio del costo di uno dei collegamenti locali oppure da una ricezione da qualche vicino di un vettore distanza aggiornato. Inoltre è un algoritmo **distribuito** perché ogni nodo aggiorna i suoi vicini solo quando il proprio  $DV$  cambia e i vicini avvisano i rispettivi vicini solo se necessario. Un problema con il routing basato su distance vector è che i decrementi di costo si diffondono rapidamente, mentre gli aumenti di costo si propagano lentamente. Affinché un protocollo di routing lavori correttamente, se un collegamento è rotto (e quindi il costo del relativo arco diventa infinito), ogni altro router dovrebbe venirne a conoscenza immediatamente, ma nel routing basato su distance vector serve un po' di tempo. Questo problema è chiamato **conteggio all'infinito**. Una soluzione è il **split-horizon with poisoned reverse**, con questa strategia se un nodo  $x$  può raggiungere la destinazione  $z$  e lo fa attraverso un vicino  $v$  allora  $x$  inoltra a  $v$  il suo vettore distanza  $D_x[z] = \infty$ , dove l'infinito viene usato come avvertimento per indicare a  $v$  di non raggiungere  $z$  attraverso  $x$  perché quello che  $x$  conosce circa quel percorso viene da  $v$ . In altre parole, le rotte ricevute tramite un'interfaccia devono essere pubblicizzate indietro da quell'interfaccia con una metrica non raggiungibile.

Differenze tra i due algoritmi:

#### Complessità dei messaggi

LS con  $n$  nodi e  $E$  collegamenti, implica l'invio di  $O(nE)$  messaggi, la rete deve essere "inodata" dei messaggi sullo stato dei collegamenti e quindi può risultare costoso

DV richiede scambi tra nodi adiacenti e il tempo di convergenza può variare

#### Velocità di convergenza

L'algoritmo di Dijkstra sul LS ha complessità di  $O(n^2)$

La convergenza del DV può convergere lentamente, può presentare cicli d'instradamento e può presentare il problema del conteggio all'infinito

#### Robustezza: cosa avviene se un router funziona male?

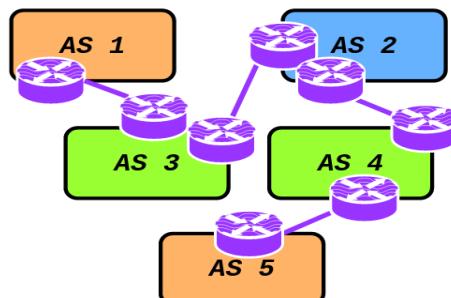
Nel LS un router può comunicare via broadcast un costo sbagliato per uno dei suoi collegamenti connessi (ma non per altri). I nodi si occupano di calcolare soltanto le proprie tabelle

Nel DV un nodo può comunicare cammini a costo minimo errati a tutte le destinazioni. La tabella di ciascun nodo può essere usata dagli altri e un calcolo errato si può diffondere per tutta la rete

### 4.5.1 Routing gerarchico

La visione di una rete costituita da un insieme di router omogenei interconnessi è semplicistica. Nella realtà i router sono organizzati in **sistemi autonomi (AS)**. Un AS è un gruppo di router sotto lo stesso controllo amministrativo e gli AS decidono autonomamente i protocolli e le politiche di routing che intendono adottare al loro interno. I protocolli di routing all'interno di un AS sono detti **Interior Gateway Protocol**, mentre i protocolli di routing fra AS sono detti **Exterior Gateway Protocol**. I sistemi autonomi si possono suddividere in almeno tre ruoli:

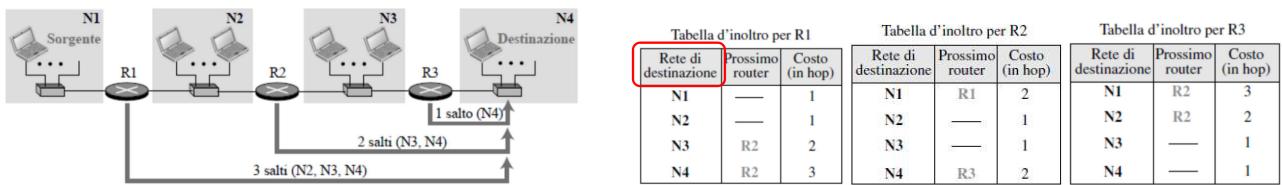
- **AS stub**, ha un solo collegamento verso un altro AS. Il traffico dati può essere generato da o destinato ad un AS stub ma non può accadere che i dati transitano attraverso l'AS. (AS 1, AS 5)
- **AS multihomed**, ha più di una connessione con altri AS, ma non consente al traffico dei dati di passare attraverso di esso. (AS 2)
- **AS di transito**, è collegato a vari AS e consente anche il transito del traffico dati. (AS 3, AS 4)



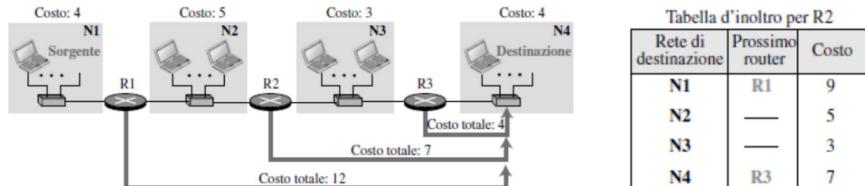
Gli *INTRA-AS routing protocol* determinano autonomamente rotte per le destinazioni *interne* ad AS:

- **Routing Information Protocol** è basato sull'algoritmo di tipo DV con scelta di implementazione di avere come costo massimo di un percorso 15 salti, inclusa la rete dove si trova la destinazione, il che significa che il valore 16 rappresenta l'infinito (nessuna connessione). RIP è implementato per mezzo di un processo demone (sempre attivo) che utilizza il protocollo UDP. I processi si scambiano dei distance vector ogni 30 secondi, oppure, se la tabella di inoltro cambia. Funzionamento:

- un nodo invia la sua tabella di inoltro ai propri vicini, ogni circa 30 secondi
- un nodo  $R$  riceve dal vicino  $V$  un avviso con  $D_V[y]$  e pone  $D_R[y] = 1 + D_V[y]$
- il nuovo percorso viene inserito in tabella:
  - \* se un percorso non è presente nella tabella, e quindi viene aggiunto
  - \* se  $D_V[y] + 1 < D_{R-old}[y]$  costo ricevuto inferiore a quello del vecchio percorso
  - \* se  $V$  è nextHop del vecchio e nuovo percorso, ma il costo è diminuito o incrementato



- **Open Shortest Path First**, basato sull'algoritmo di tipo LS. A differenza del RIP che la metrica era data dal numero dei salti, l'OSPF non ha nessun limite, ovvero, la metrica può deciderla l'amministratore. Rispetto al RIP, che normalmente viene usato in AS piuttosto piccoli, l'OSPF è stato progettato per poter gestire il routing in AS di tutte le dimensioni. Tuttavia vi è il problema che tutti i router inondino (*flooding*) l'intero AS con i loro LSP per creare l'LSDB globale. Per evitare ciò, l'AS può essere diviso in settori più piccoli chiamati *aree*, una delle quali fa da dorsale, cioè collega tra di loro le varie aree. Ogni area è un piccolo dominio indipendente per il flooding degli LSP. In pratica l'OSPF usa un altro livello di gerarchia nel routing: il primo livello è il sistema autonomo e il secondo è l'area al suo interno. L'OSPF è implementato come applicazione che sfrutta la comunicazione a livello di rete usando direttamente il protocollo IP.



Gli *INTRA-AS e INTER-AS routing protocol* determinano insieme rotte per le destinazioni *esterne* all'AS:

- **Border gateway protocol Versione 4**, si basa sull'algoritmo DV ed è l'unico protocollo di routing INTER-AS ora utilizzato in Internet. Ogni router all'interno degli AS sa come raggiungere tutte le reti che si trovano nel suo AS, ma non sa come raggiungere una rete che si trova in un altro AS. Per permettere ad ogni router di instradare correttamente i pacchetti, qualsiasi sia la destinazione, è necessario installare su tutti i router di confine dell'AS una variante del BGP chiamata **BGP esterno**. Tutti i router (non solamente quelli del confine) dovranno invece utilizzare la seconda variante chiamata **BGP interno**. La rotta è composta da un *prefisso* di rete raggiungibile e da una serie di *attributi* che caratterizzano il percorso. I due attributi più importanti sono:

- **AS\_PATH**, indica la sequenza degli AS che devono essere attraversati per arrivare a destinazione. È usato per scartare advertisement già ricevuti e scegliere tra più percorsi quello più breve
- **NEXT\_HOP**, indica l'indirizzo IP del primo router lungo un percorso annunciato a un dato prefisso di rete

Un router può ricevere più percorsi per una sottorete esterna, quindi per scegliere la rotta si seguono le seguenti regole:

1. si controlla l'attributo di "preferenza locale", **LOCAL\_PREF**, scelta da un amministratore o impostato dai router dell'AS. Vengono selezionati quelli coi valori più alti
2. a parità di LOCAL\_PREF si sceglie quello con AS\_PATH più breve
3. si sceglie quello con il NEXT\_HOP più vicino

## 4.6 IPv6

L'esaurimento degli indirizzi e alcune limitazioni di IPv4 hanno portato allo sviluppo di una nuova versione del protocollo. Questa nuova versione, chiamata **Internet protocol versione 6** porta i seguenti cambiamenti:

- **Spazio degli indirizzi di dimensione maggiore**, un indirizzo IPv6 è lungo 128 bit rispetto ai 32 bit dell'IPv4
- **Velocizzare elaborazione e forwarding pacchetti**, imposta una lunghezza fissa (40 byte) dell'header e eliminazione della checksum
- **Risparmiare ai router costo di frammentazione**, quindi i router dovranno fare una operazione in meno
- **Identificare datagramma**, in IPv6 è stato rimosso il campo tipo del servizio e al suo posto sono stati aggiunti due nuovi campi: *classe di traffico* ed *etichetta di flusso* per consentire alla sorgente di richiedere un trattamento speciale per il datagramma per migliorare la gestione del traffico

Anche se è pronta una nuova versione del protocollo IP, rimane da capire come può essere fatto il passaggio dalla versione corrente a quella nuova. L'IETF ha studiato tre strategie che possono essere implementate in questa fase:

- **Dual stack**, prevede che tutti gli host abbiano una doppia pila di protocolli per la comunicazione in rete per poter eseguire IPv4 e IPv6 simultaneamente fino a quando tutta la rete non sarà passata ad IPv6
- **Tunneling**, utilizzata quando due host che usano IPv6 vogliono comunicare tra loro ma i datagrammi devono attraversare una regione della rete che usa solo IPv4. La soluzione è quella di incapsulare ciascun datagramma IPv6 nel payload di un datagramma IPv4. Una volta superata la tratta IPv4, il datagramma IPv6 potrà essere nuovamente estratto
- **Traduzione dell'intestazione**, sarà necessaria quando la maggior parte di Internet sarà migrata ad IPv6, con alcuni sistemi residui ancora basati su IPv4. Se il mittente usa IPv6 ma il ricevente non è in grado di supportarlo, si effettua una traduzione completa dell'intestazione del datagramma, in modo da convertirla da IPv6 a IPv4, prima che il datagramma arrivi a destinazione

## 4.7 Esercizi

**Esercizio 1.** Qual è il campo nell'intestazione IPv4 che indica il numero massimo di hop nel percorso del datagramma?

- TTL
- flag
- offset

**Esercizio 2.** Host ha indirizzo IP 140.80.3.145 e maschera di sottorete 255.255.255.192, il suo indirizzo di rete è:

- 140.80.3.0
- 140.80.0.0
- nessuna delle altre opzioni
- 140.80.3.128

**Esercizio 3.** Un datagramma IP di dimensioni 1000 byte arriva a un router. Il router deve inoltrare questo pacchetto su un collegamento la cui MTU (unità di trasmissione massima) è di 100 byte. Supponiamo che la dimensione dell'intestazione IP sia di 20 byte. Il numero di frammenti in cui il datagramma IP sarà diviso per la trasmissione è:

- 12
- 10
- 13

*Il datagramma IP originale ha un payload di 980 byte (1000-20byte di header). Deve essere suddiviso in frammenti di lunghezza massima 100 byte (80 byte di payload + 20 byte di header). 80 byte è un multiplo di 8. Quindi il numero di frammenti è 980/80=12,25 quindi 13 frammenti.*

**Esercizio 4.** Quale dei seguenti campi dell'intestazione IP NON è tipicamente modificato da un router?

- indirizzo IP di destinazione*
- checksum*
- TTL*

**Esercizio 5.** Se il campo HLEN è 0101 la lunghezza dell'header di un datagramma IPv4 è

- nessuna delle opzioni*
- 60 byte*
- 5 byte*
- 20 byte*

**Esercizio 6.** Si considerino tre macchine, A, B e C con i seguenti indirizzi IP: A 100.10.5.2, B 100.10.5.5, e C 100.10.5.6. La subnet mask è 255.255.255.252 per tutte e tre le macchine. Quale delle seguenti affermazioni è vera?

- Solo A e B appartengono alla stessa sottorete*
- A,B e C appartengono alla stessa sottorete*
- Solo B e C appartengono alla stessa sottorete*
- A,B e C appartengono a tre sottoreti differenti*

*La subnet mask 255.255.255.252 indica che 30 bit sono usati per l'indirizzo di rete. L'indirizzo di rete di A è quindi 100.10.5.0, l'indirizzo di rete di B è 100.10.5.4, l'indirizzo di rete di C è 100.10.5.4. Quindi solo B e C appartengono alla stessa sottorete.*

**Esercizio 7.** Un ISP dispone del seguente blocco di indirizzi IP basati su CIDR: 245.248.128.0/20. L'ISP vuole dare metà di questo blocco di indirizzi all'organizzazione A e un quarto all'organizzazione B. Quale delle seguenti è una valida assegnazione di indirizzi ad A e B?

- 245.248.128.0/22 e 245.248.136.0/21*
- 245.248.136.0/24 e 245.248.132.0/21*
- 245.248.128.0/21 e 245.248.136.0/22*
- 245.248.132.0/22 e 245.248.132.0/21*

*Si inizia con il blocco di A (il più grande). Il blocco dell'organizzazione A deve avere 21 bit per l'indirizzo di rete, quindi l'indirizzo del blocco assegnato ad A sarà 245.248.128.0/21*

**11110101.11111000. 1000 0000. 0000 0000**

*(in grassetto i bit dedicati all'indirizzo di rete). Il blocco dell'organizzazione B deve avere 22 bit*

**11110101.11111000. 1000 1000. 0000 0000**

*quindi l'indirizzo del blocco assegnato a B sarà 245.248.136.0/22. PS nella risoluzione dell'esercizio non serve convertire in binario 245 e 248, li ho riportati per usare una notazione convenzionale.*

**Esercizio 8.** Quali informazioni fornisce un server DHCP ad un host?

- Indirizzo IP, netmask, indirizzo del router, indirizzo di livello collegamento*
- Indirizzo IP, netmask, indirizzo del router, indirizzo IP di almeno un server DNS*
- Indirizzo IP, netmask, indirizzo del router, indirizzo di localhost*

**Esercizio 9.** Dire quale delle seguenti affermazioni è vera:

- DHCP usa UDP e assegna un indirizzo IP ad un host per un intervallo di tempo definito*
- DHCP usa TCP e assegna un indirizzo IP ad un host per un intervallo di tempo definito*
- DHCP usa UDP e i messaggi DHCP non possono essere inviati ad un indirizzo IP broadcast*

**Esercizio 10.** Quale delle seguenti affermazioni è vera?

- Un messaggio di errore ICMP può essere generato solo per il primo frammento
- Un messaggio di errore ICMP può essere generato per un datagramma con indirizzo mittente 0.0.0.0
- Un messaggio di errore ICMP può essere generato a seguito dell'invio di un messaggio di errore ICMP.

**Esercizio 11.** Un router X genera un messaggio di errore ICMP relativo alla consegna di un datagramma IP. A chi è destinato il messaggio di errore ICMP?

- Mittente del datagramma
- Router precedente
- Destinatario del datagramma
- Router successivo

**Esercizio 12.** ICMP è un protocollo del livello

- Applicativo
- Collegamento
- Trasporto
- Rete

**Esercizio 13.** Indicare quali delle seguenti affermazioni NON è corretta

- il comando ping può essere usato per tracciare il percorso di un datagramma da sorgente a destinazione
- il comando traceroute invia una sequenza di datagrammi con TTL crescente (a partire da 1)
- il comando ping si basa sui messaggi di richiesta e risposta eco ICMP

**Esercizio 14.** Il compito di spostare il pacchetto dalla coda di input alla coda di output in un router viene svolto da:

- switching fabric
- porta di input e di output
- processore di routing

**Esercizio 15.** Indicare quale delle seguenti frasi NON è corretta

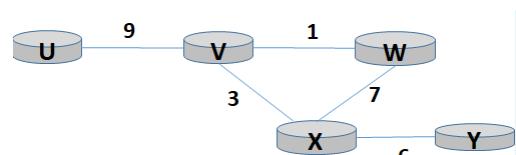
- le informazioni OSPF sono inviati in segmenti TCP
- le informazioni RIP sono inviate in datagrammi UDP
- RIP usa un algoritmo tipo Distance Vector
- OSPF usa un algoritmo tipo Link State

**Esercizio 16.** Il protocollo Open Shortest Path First (OSPF) può anche essere chiamato:

- Border Gateway Protocol
- Routing Information Protocol
- protocollo Link State

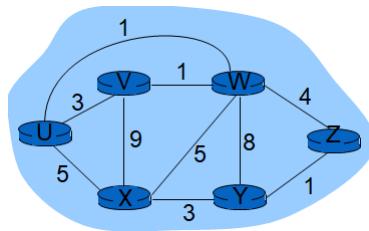
**Esercizio 17.** Considerando la topologia di rete in figura con 6 nodi, qual è il vettore distanza iniziale del nodo U?

- U 0; V 9; W inf; X inf; Y inf
- U 12; V 3; W 4; X 0; Y 6
- U inf; V 3; W 7; X 0; Y 6



**Esercizio 18.** Considerando la topologia di rete in figura e l'algoritmo link state, qual è il costo del cammino a costo minimo dal nodo U al nodo Z e quale nodo è il predecessore di Z?

- costo 9, predecessore Y
- costo 5, predecessore W
- costo 4, predecessore W



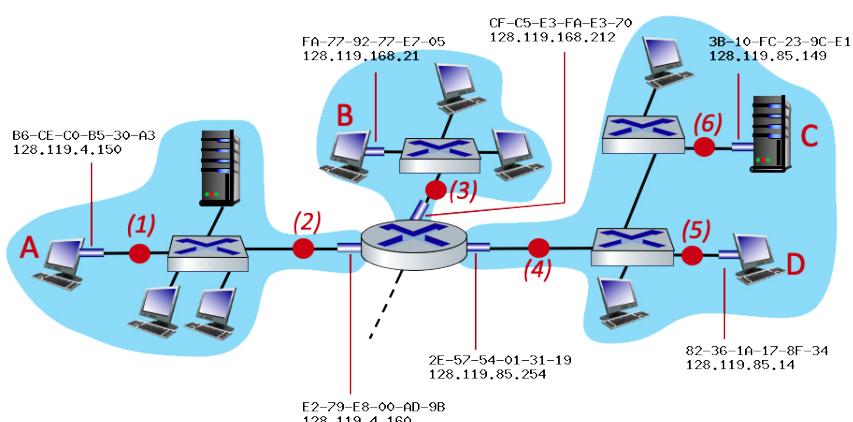
**Esercizio 19.** A fini dell'instradamento, la Internet pubblica è suddivisa in:

- Sistemi autonomi
- nessuna delle altre opzioni
- Aree

**Esercizio 20.** Quando un datagramma IP viene decapsulato (tolto l'header), l'unità dati che resta è un:

- frame
- segmento
- messaggio applicativo

**Esercizio 21.** Considera la figura qui sotto, in cui sono visualizzati gli indirizzi IP e MAC per i nodi A, B, C e D e per le interfacce del router. A invia un pacchetto a B. Qual è l'indirizzo MAC di destinazione del pacchetto al punto 1?



- B6-CE-C0-B5-30-A3
- FA-77-92-77-E7-05
- E2-79-E8-00-AD-9B

**Esercizio 22.** Indicare quale delle seguenti frasi su OSPF è corretta:

- OSPF usa l'algoritmo Distance Vector ed è un protocollo di routing INTRA-AS
- OSPF è un protocollo di routing INTER-AS
- OSPF usa l'algoritmo Link State ed è un protocollo di routing INTRA-AS

**Esercizio 23.** Un insieme di router sotto lo stesso controllo amministrativo è chiamato:

- Area
- Un sistema autonomo
- WAN

**Esercizio 24.** Il DHCP fornisce ad un client:

- URL*
- indirizzo MAC*
- indirizzo IP*

**Esercizio 25.** Qual è il campo nell'intestazione IPv4 utile al demultiplexing effettuato al livello di rete?

- campo Tipo di Servizio*
- campo Identificatore*
- campo Protocol*

**Esercizio 26.** Per una Intranet si ha a disposizione l'indirizzo di rete 150.210.60.0 e netmask: 255.255.252.0. Nella Intranet occorre configurare almeno 3 reti locali, ciascuna di 100 host. Indicare la maschera di rete per le sottoreti.

- 255.255.255.0
- 255.255.255.128
- 255.255.255.192

**Esercizio 27.** Cosa usano i router di una rete OSPF per conoscere la topologia di rete?

- messaggi DHCP*
- pacchetti Link State*
- messaggi RIP*

# Chapter 5

## Livello collegamento

### 5.1 Introduzione

Il livello collegamento muove i datagrammi da un nodo al nodo adiacente su un singolo link di comunicazione. I nodi all'interno delle reti sono fisicamente collegati da un mezzo trasmittivo, come un cavo. Compito del livello di collegamento è controllare come viene usato tale mezzo. Tipologie:

- **collegamento punto-punto**, dedicato a due soli dispositivi (mittente e ricevente)
- **collegamento broadcast**, condiviso tra più dispositivi. Quando un nodo trasmette un frame, il canale lo difonde e tutti gli altri nodi ricevono una copia

Il livello di collegamento viene implementato con una combinazione di hardware, software e firmware sulla scheda di rete (o adattatori), e può essere diviso in due sotto livelli: **data-link control** e **media access control**

### 5.2 Data-link control

Il sotto livello di data-link control si occupa di tutte le questioni comuni sia ai collegamenti punto-punto che a quelli broadcast. I servizi offerti sono:

- **Framing**, i protocolli encapsulano i datagrammi del livello di rete all'interno di un frame a livello di link. Un frame è composto da campo dati, intestazione ed eventuale trailer. Il framing separa i vari messaggi durante la trasmissione da una sorgente a una destinazione. Per identificare origine e destinatario si utilizzano indirizzi MAC
- **Consegna affidabile**, è considerata non necessaria nei collegamenti che presentano un basso numero di errori sui bit, mentre, è spesso utilizzata nei collegamenti soggetti a elevati tassi di errori
- **Controllo di flusso**, evita che il nodo trasmettente saturi quello ricevente
- **Controllo degli errori**, comprende sia il rilevamento che la correzione degli errori. Il nodo ricevente informa il mittente circa gli eventuali frame persi o danneggiati durante la trasmissione e coordina la ritrasmissione di tali frame da parte del mittente. È possibile grazie all'inserimento, da parte del nodo mittente, di bit di controllo di errori all'interno del frame

### 5.3 Protocollo ad accesso multiplo

Il problema del mezzo condiviso è che vi possono essere **collisioni** se un nodo riceve due o più segnali nello stesso istante. Per ovviare al problema sono stati ideati diversi protocolli che implementano vari algoritmi distribuiti che determinano come i nodi condividono il canale. Un protocollo di accesso multiplo *ideale*: canale broadcast con rate R bps con le seguenti caratteristiche:

1. quando un solo nodo trasmette, può inviare dati con un rate di R bps
2. quando M nodi vogliono trasmettere, ciascuno trasmette ad un rate medio R/M
3. decentralizzato, ovvero, non ci sono nodi speciali che coordinano la trasmissione
4. semplice

Vi sono tre classi principali di protocolli:

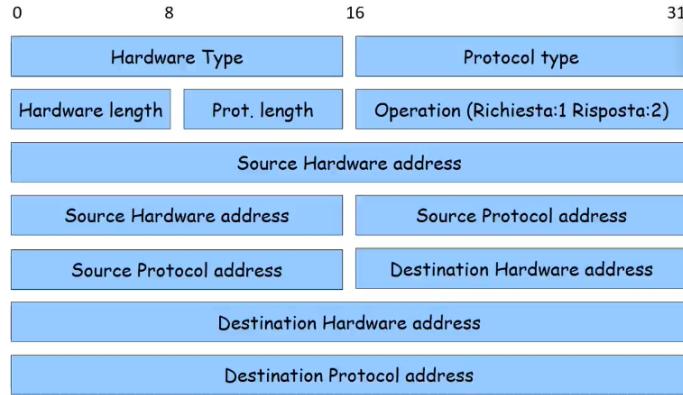
- **Suddivisione del canale**, si divide il canale in risorse più piccole. La risorsa può essere:
  - **Tempo, time division multiple access (TDMA)**. L'accesso al canale è misurato in intervalli di tempo dove ciascun intervallo è suddiviso in  $n$  slot. Ogni slot è dedicato ad un nodo e quando quest'ultimo trasmette lo può fare a R bps. Gli slot non utilizzati sono sprecati, quindi vi possono essere situazioni in cui non si ha un utilizzo ottimale del canale. Tuttavia, in questo contesto non vi sono le collisioni
  - **Frequenza, frequency division multiple access (FDMA)**. Lo spettro del canale viene diviso in bande di frequenza. Ad ogni nodo è assegnato una banda di frequenza fissa. Se ci sono nodi che non trasmettono, le rispettive bande di frequenza non sono utilizzate
- La risorsa è allocata al nodo in modo esclusivo.
- **Accesso casuale**, il canale è condiviso. Quando un nodo deve inviare dati trasmette al massimo rate del canale e non c'è coordinamento a priori tra i nodi quindi possono esserci collisioni. Un protocollo MAC ad accesso random specifica come rilevare le collisioni e come recuperare dopo le collisioni (per esempio trovare il momento giusto per ritrasmettere). Esempi di protocolli ad accesso casuale:
  - **Slotted ALOHA**, il tempo viene diviso in slot uguali e fissi. I nodi sono sincronizzati e possono trasmettere solo all'inizio dello slot, se si lasciano sfuggire tale momento, devono attendere fino all'inizio del ciclo successivo. Se due o più nodi trasmettono, ogni nodo rileva la collisione, e quindi deve ritrasmettere il frame nello slot successivo finché la trasmissione ha successo. I vantaggi sono che il singolo nodo attivo può trasmettere alla massima velocità consentita dal canale, il sistema è decentralizzato ed è semplice. Lo svantaggio è che se c'è un gran numero di nodi attivi, il canale è usato con successo solo il 37% del tempo
  - **ALOHA puro**, versione antecedente dello slotted ALOHA senza slot. L'idea è che ogni stazione può inviare un frame tutte le volte che ha qualcosa da inviare. Vi è una maggiore probabilità di collisione rispetto allo slotted ALOHA
  - **Carrier Sense Multiple Access**, richiede che ogni nodo ascolti il mezzo trasmisivo prima di trasmettere. Se il canale è occupato, ritarda la trasmissione. Il CSMA può ridurre la probabilità di collisioni, ma la possibilità esiste ancora a causa del ritardo di propagazione, ovvero, due nodi potrebbero non rilevare le comunicazioni reciproche. Le collisioni vengono gestite con il meccanismo di **collision detection**: un nodo incomincia a trasmettere però sempre ascoltando il canale; se un altro nodo sta trasmettendo, si interrompe la trasmissione.
- **Rotazione**, utilizzato con il meccanismo del **polling** dove il nodo *master* invita i nodi *slave* a trasmettere a rotazione. Un altro meccanismo è il **token passing** che viene effettuato con un token e il nodo che ha il token può trasmettere

## 5.4 Indirizzi a livello collegamento

Un indirizzo del livello collegamento è associato alla scheda di rete, non al nodo, ed è tipicamente permanente. Un indirizzo di livello collegamento è spesso chiamato **indirizzo fisico** o **indirizzo MAC**. Gli indirizzi di collegamento in Ethernet sono di 48 bit ed è espresso in notazione esadecimale. IEEE definisce ed assegna i primi 24 bit, mentre i rimanenti 24 bit vengono gestiti dalle aziende ed assegnati a livello locale. Quando un adattatore spedisce un frame, vi inserisce l'indirizzo MAC della scheda di destinazione. Nel caso di reti LAN Broadcast, tutti gli adattatori attestati sulla rete controllano l'indirizzo destinazione e passano i dati allo strato superiore solo se riconoscono il proprio indirizzo MAC nell'intestazione. Se un adattatore trasmittente vuole che tutte le schede di rete passino i dati agli strati superiori, immette nel campo indirizzo destinazione: FF-FF-FF-FF-FF-FF, ovvero l'indirizzo broadcast.

### 5.4.1 Address Resolution Protocol

Il protocollo **ARP** è definito a livello rete e associa gli indirizzi IP agli indirizzi di collegamento. ARP prende in input un indirizzo IP, individua l'indirizzo di collegamento corrispondente e lo passa al livello di collegamento. Ogni nodo IP nella LAN ha una **tavella ARP** che contiene la corrispondenza tra indirizzi IP e MAC, e un TTL che è un valore che indica quando bisognerà eliminare una data voce nella tabella (il tempo di vita tipico è di 20 minuti). La tabella non contiene necessariamente le corrispondenze per tutti i nodi della sottorete. La seguente figura mostra il formato di un pacchetto ARP:

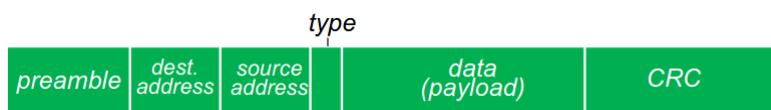


Il campo *hardware type* definisce il tipo del protocollo di livello collegamento che viene usato. Il campo *protocol type* definisce il protocollo di rete. I campi *source hardware address* e *source protocol address* definiscono, per il mittente, il suo indirizzo fisico e quello del protocollo di livello superiore (IP). Entrambi i campi sono di lunghezza variabile (*hardware lenght* e *protocol lenght*). I campi *destination hardware address* e *destination protocol address* definiscono l'indirizzo fisico e quello del protocollo di livello superiore, ma in questo caso per il ricevente. I pacchetti ARP vengono incapsulati direttamente all'interno dei frame di livello collegamento. Per questo motivo il frame dovrà indicare all'interno della sua intestazione che trasporta un pacchetto ARP invece di un datagramma del livello di rete. Il frame viene inviato con indirizzo MAC broadcast, quindi il pacchetto sarà ricevuto da tutte le schede di rete, e solo chi ha possiede l'indirizzo IP indicato risponderà. La risposta ARP è inviata in unicast.

## 5.5 Ethernet

Detiene una posizione dominante nel mercato delle LAN cablate. È stata la prima LAN ad alta velocità con vasta diffusione, essendo più semplice e meno costosa di token ring. Alla metà degli anni '90, la topologia più diffusa era quella a *bus*. Quasi tutte le odierne reti Ethernet sono progettate con topologia a *stella*, dove al centro della stella è allocato uno switch. Il frame Ethernet contiene sette campi:

- **Preambolo**, questo campo è composto di 8 byte, dove 7 byte hanno i bit 10101010 e l'ultimo è 10101011. Il suo compito è quello di allertare il sistema ricevente circa l'arrivo del frame e permettere di sincronizzare il suo orologio con quello del trasmittente
- **Indirizzo destinazione**, sono 6 byte e contiene l'indirizzo di collegamento della stazione o delle stazioni destinarie del frame
- **Indirizzo sorgente**, sono 6 byte contiene l'indirizzo di collegamento del mittente del frame
- **Tipo**, sono 2 byte e definisce il protocollo di livello superiore del pacchetto incapsulato all'interno del frame
- **Dati**, l'unità massima di trasferimento (MTU) varia da 46 byte ad un massimo di 1500 byte. Se il datagramma è più grande allora deve essere frammentato. Se il campo dati è più piccolo, il campo dati deve essere riempito (stuffed)
- **CRC**, consente all'adattatore ricevente di rilevare la presenza di un errore nei bit del pacchetto



Ethernet fornisce un servizio senza connessione, questo significa che ogni frame inviato è indipendente dal frame precedente e dal successivo. Non vi è handshake tra nodo mittente e destinatario, ovvero, il mittente si limita ad inviare un frame ogni volta che ne ha uno pronto, indipendentemente dal fatto che il destinatario sia pronto o meno a riceverlo. La trasmissione non è affidabile: nel caso in cui un frame viene scartato, per esempio per rilevamento di errore, il mittente non viene informato dell'accaduto. Dato che il protocollo IP, che usa i servizi offerti da Ethernet, è anch'esso un protocollo non affidabile e senza connessione, anche IP non verrà a conoscenza della perdita di dati. Se anche il protocollo di trasporto è non affidabile e senza connessione (ad esempio UDP), solamente il livello applicazione potrà accorgersi della perdita ed eventualmente porvi rimedio. Se, invece, a livello trasporto viene usato TCP, il mittente non riceverà alcun riscontro per il segmento contenuto nel frame perso e quindi lo invierà di nuovo. L'inaffidabilità di Ethernet è pari a quella di IP e UDP. È dovere dei protocolli di livello superiore riscontrare la perdita dei dati e porvi rimedio. Evoluzione di Ethernet:

- **Fast Ethernet**, velocità di trasmissione di 100 Mbps. È compatibile con la versione precedente, quindi il formato del frame e le sue dimensioni (massima e minima) rimangono invariati. Se la dimensione del frame è ancora 512 bit e la trasmissione è 10 volte più veloce, allora le collisioni devono essere rilevate 10 volte più velocemente. Tutto questo porta al fatto che la rete dovrebbe essere 10 volte più corta. Per questo motivo sono state previste due diverse soluzioni:

1. Abbandonare la topologia a bus, utilizzare un hub passivo con una topologia a stella e fissare la dimensione massima della rete a 250 metri
2. Utilizzare uno switch a livello collegamento dotato di buffer per memorizzare i frame e una connessione full-duplex per ciascun host. Questo permette di avere collegamenti di distanza maggiore e non avere collisioni

- **Gigabit Ethernet**, velocità di trasmissione di 1000 Mbps. Nato con lo scopo di aggiornare la velocità di trasferimento a 1 Gbps mantenendo invariata la lunghezza dell'indirizzo, il formato e la lunghezza massima e minima del frame. La topologia delle reti Gigabit Ethernet prevede la presenza di uno switch al centro della stella, e tutti i nodi collegati ai vari rami della stella

## 5.6 Dispositivi di interconnessione

È raro trovare degli host e delle reti che operano in maniera isolata; è molto più comune utilizzare dei dispositivi di interconnessione per collegare gli host e creare una rete, e per collegare le reti tra di loro creando una internet. Si studiano ora i tre tipi di dispositivi di interconnessione:

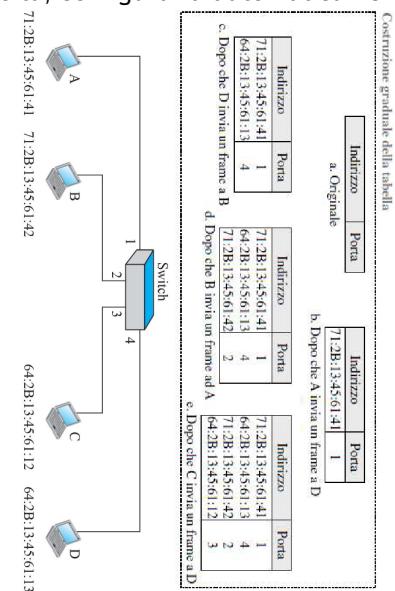
- **Repeater e hub**

Un **repeater** è un dispositivo che opera soltanto a livello fisico. Un repeater riceve un segnale, e prima che esso diventi troppo debole o danneggiato, *rigenera e risincronizza* la sequenza di bit originale. In passato quando le LAN Ethernet utilizzavano la topologia a bus, si usava un repeater per collegare tra loro due segmenti di una LAN. Oggi le LAN Ethernet usano la topologia a stella, e in questo caso il repeater è un dispositivo multiporta, chiamato **hub**, che può essere utilizzato per l'interconnessione delle varie stazioni collegate e allo stesso tempo fungere da ripetitore. L'hub inoltra il frame attraverso tutte le sue porte in uscita eccetto quella da cui ha ricevuto il segnale. Quindi tutte le stazioni LAN ricevono il frame, ma soltanto il destinatario del frame lo memorizza, tutte le altre lo scartano. Repeater e hub non hanno capacità di filtraggio

- **Switch di livello collegamento**

Uno **switch di livello collegamento** opera sia a livello fisico che a livello di collegamento. Come dispositivo di livello fisico rigenera il segnale che riceve. Come dispositivo di livello collegamento è in grado di verificare gli indirizzi MAC contenuti nel frame. Uno switch ha una tabella che viene utilizzata per il filtraggio. La tabella è dinamica ed associa automaticamente gli indirizzi MAC alle porte. Per creare una tabella dinamica, occorre che uno switch sia in grado di apprendere gradualmente dai frame che vengono inviati all'interno della rete. Per fare questo, lo switch analizza gli indirizzi sorgente e destinazione di ogni frame. L'indirizzo di destinazione viene utilizzato per decidere su quale interfaccia inoltrare il frame, quello sorgente invece serve per aggiungere nuove voci alla tabella e quindi, poco alla volta, configurarla automaticamente.

1. Quando la stazione *A* invia un frame alla stazione *D*, lo switch nella sua tabella non ha alcuna voce sia per *D* che per *A*. Il frame viene quindi inviato in broadcast su tutte le porte, ad esclusione di quella da cui è stato ricevuto. Analizzando l'indirizzo sorgente, lo switch apprende che la stazione *A* è collegata alla porta 1. Ciò indica che i frame destinati ad *A*, in futuro, dovranno essere inviati solo attraverso la porta 1. Lo switch aggiunge questa informazione alla sua tabella, che ora contiene la sua prima voce
2. Quando la stazione *D* invia un frame alla stazione *B*, lo switch non ha una voce relativa a *B*, così invia nuovamente in broadcast il frame. Tuttavia, questo frame permette di aggiungere un'ulteriore voce alla tabella: la porta a cui è collegata la stazione *D*
3. Il processo di apprendimento continua fino a quando la tabella non contiene informazioni su tutte le stazioni e le relative porte



Lo switch è *trasparente*, ovvero, gli host non sono a conoscenza della presenza degli switch, ed è *plug-and-play*, cioè gli switch non devono essere configurati. Tutte queste proprietà sono importanti perché permettono di dividere il dominio di collisione, in questo modo due coppie di nodi possono trasmettere simultaneamente senza collisioni.

- **Router**, è un dispositivo che opera su tre livelli: fisico, collegamento e rete. Come dispositivo a livello fisico rigenera il segnale che riceve. Come dispositivo a livello di collegamento, il router verifica gli indirizzi MAC contenuti nel frame. Come dispositivo a livello di rete, verifica gli indirizzi di questo livello. I router collegano reti diverse per ottenere una rete di reti. Ci sono tre grandi differenze tra un router, repeater e switch:

- un router ha un indirizzo fisico (indirizzo MAC) e logico (indirizzo IP) per ognuna delle sue interfacce
- un router opera soltanto su quei frame il cui indirizzo di destinazione (di livello collegamento) indicato nell'intestazione è uguale all'indirizzo di collegamento dell'interfaccia su cui arrivano
- i router cambiano l'indirizzo di collegamento del frame (sia la sorgente che la destinazione) quando li inoltrano

## 5.7 Esercizi

**Esercizio 1.** Il protocollo ARP (Address Resolution Protocol) viene utilizzato principalmente per la traduzione da:

- Nomi host a indirizzi IP*
- Indirizzi IP privati a indirizzi IP pubblici*
- Indirizzi MAC a indirizzi IP*
- Indirizzi IP a indirizzi MAC*

**Esercizio 2.** Quale dei seguenti è l'indirizzo di trasmissione ethernet utilizzato nelle richieste ARP?

- FF-FF-FF-FF-FF-FF*
- 08-00-20-11-AA-01*
- 255.255.255.255*
- 224.0.0.0*

**Esercizio 3.** Una richiesta ARP è incapsulata in:

- un frame*
- un segmento TCP*
- un datagramma IP*

**Esercizio 4.** Si considera uno switch con auto-apprendimento dotato di 5 porte a cui sono connessi 5 computer:

- il computer A con indirizzo MAC A alla porta 1
- il computer B con indirizzo MAC B alla porta 2
- il computer C con indirizzo MAC C alla porta 3
- il computer D con indirizzo MAC D alla porta 4
- il computer E con indirizzo MAC E alla porta 5

All'inizio la tabella di commutazione dello switch è vuota. Indicare il contenuto della tabella (nella forma porta: indirizzo MAC), dopo l'invio dei seguenti frame.

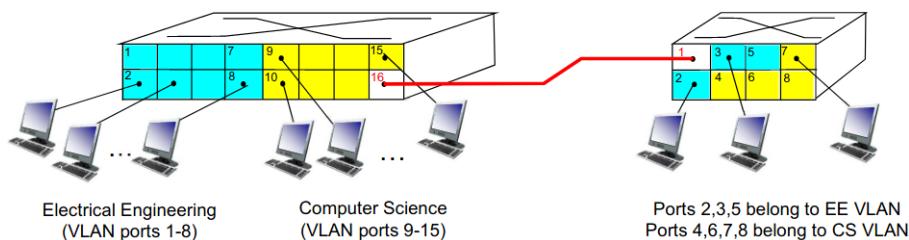
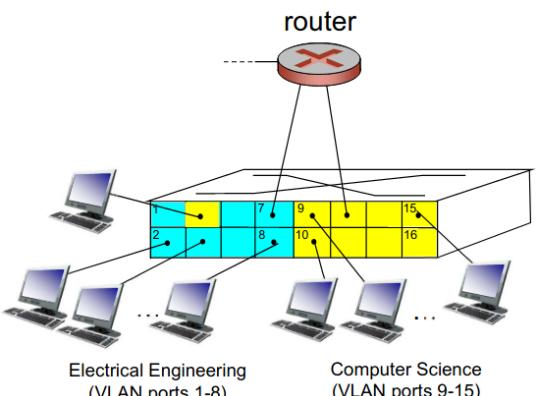
- da E a C
  - da C ad B
  - da A ad E
- 5:E, 3:C, 1:A*
  - 2:B, 4:D, 1:A*
  - 3:C, 5:E*
  - 5:E, 2:B, 4:D*

# Chapter 6

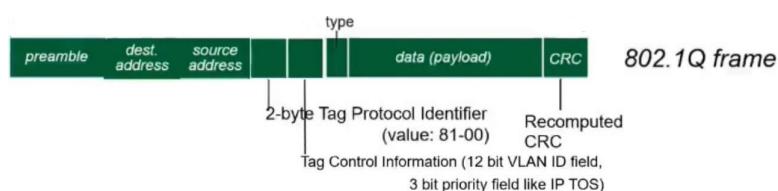
## VLAN, e cenni di applicazioni P2P e sicurezza

### 6.1 VLAN

Le LAN istituzionali moderne sono spesso configurate in modo gerarchico, così che ogni gruppo abbia la propria LAN commutata connessa alle LAN commutate degli altri gruppi attraverso una gerarchia di switch. Con tale configurazione possono nascere problemi di sicurezza e riservatezza. Per esempio, se un gruppo contiene il consiglio di amministrazione dell'azienda e un altro gruppo contiene impiegati scontenti che usano *Wireshark* per intercettare pacchetti, il gestore di rete potrebbe preferire che il traffico del consiglio non raggiunga mai gli host di tali impiegati. Un altro inconveniente è che se un dipendente si muovesse tra più gruppi, sarebbe necessario cambiare la posatura della rete per connetterlo a un altro switch. Questi problemi possono essere superati utilizzando uno switch che supporti una **VLAN**, tale switch permette di definire più reti locali virtuali su una singola infrastruttura fisica di rete locale. Gli host all'interno di una VLAN comunicano tra loro come se fossero tutti connessi allo switch. In una VLAN basata sulle porte, le interfacce dello switch vengono divise dal gestore di rete in gruppi, ognuno dei quali costituisce una VLAN le cui porte formano un **dominio broadcast**. Isolando completamente due VLAN, si è introdotto una nuova difficoltà: come trasmettere il traffico, per esempio, dal gruppo del dipartimento di Ingegneria elettronica al gruppo di Informatica? Un modo sarebbe quello di connettere una porta dello switch VLAN a un router esterno e configurarla in modo che appartenga a entrambe le VLAN. Per l'interconnessione tra switch VLAN è noto come **VLAN trunking**. In questo approccio una porta speciale per ogni switch viene configurata come porta di trunking per interconnettere i due switch VLAN.



Come fa uno switch a sapere che un frame che arriva a porta di trunking appartiene a una VLAN particolare? IEEE ha definito un formato esteso di frame Ethernet nello standard 802.1Q, per frame che attraversano un trunk VLAN. Il frame 802.1Q consiste nel frame standard Ethernet con aggiunta di un'etichetta VLAN di 4 byte nell'intestazione che trasporta l'identità della VLAN a cui il frame appartiene. L'etichetta VLAN consiste di un campo *tag protocol identifier* di 2 byte e un campo *tag control information* di 2 byte, contenente il campo di identificazione della VLAN a 12 bit e un campo di priorità a 3 bit.



## 6.2 Paradigma peer-to-peer

Il paradigma peer-to-peer indica un modello di architettura logica in cui i nodi non sono gerarchizzati unicamente sotto forma di client e server fissi, ma anche sotto forma di nodi paritari, potendo fungere al contempo da client e server verso gli altri nodi della rete. Il paradigma peer-to-peer è utilizzato in tante applicazioni per:

- distribuzione e memorizzazione di contenuti (file sharing, update di giochi...)
- condivisione di risorse di calcolo
- collaborazione e comunicazione
- bitcoin

Tutti i nodi hanno la stessa importanza, sono indipendenti gli uni dagli altri e sono localizzati ai bordi di Internet. Ogni peer ha funzioni di client e server ma in realtà possono essere presenti server centralizzati o nodi con funzionalità diverse rispetto agli altri nodi: **supernodi**. Il modello peer-to-peer ha la caratteristica principale che è un sistema altamente distribuito, dove il numero di nodi può essere dell'ordine delle centinaia di migliaia. I nodi sono altamente dinamici, ovvero, un nodo può entrare o uscire dalla rete P2P in ogni momento. **Problema**: coppie di peer comunicano direttamente tra loro, ogni peer però non è necessariamente sempre attivo e può cambiare indirizzo IP ogni volta che si connette. Quindi come trovare i peer? Come tenere traccia dei file disponibili nei vari peer?

### • Approccio 1: directory centralizzata

Una prima soluzione è quella di avere una directory centralizzata che contiene l'elenco dei peer e delle risorse condivise. In questo tipo di rete, un peer si registra per prima cosa presso un server centrale, al quale fornisce il proprio indirizzo IP e un elenco di file che vuole condividere. Un peer alla ricerca di un particolare file, invia una richiesta a un server centrale. Questo ricerca nella propria directory e risponde con l'indirizzo IP dei nodi che hanno una copia del file. Il peer contatta quindi uno (o più) dei nodi e preleva il file. La directory è costantemente aggiornata per tenere traccia dei nodi che entrano o escono dalla rete. Il vantaggio è che la gestione del sistema di directory è piuttosto semplice, ma presenza alcuni inconvenienti: l'accesso alla directory può generare un traffico enorme e rallentare tutto il sistema, i server centrali sono vulnerabili agli attacchi, e se dovesse subire danni, l'intero sistema collasserebbe

### • Approccio 2: reti decentralizzate

Una rete P2P decentralizzata non utilizza un sistema di directory centralizzato. Con questo modello i peer si organizzano formando una **overlay network**, ovvero una rete logica sovrastante alla rete fisica utilizzata per organizzare i peer. A seconda di come sono collegati i nodi della rete sovrastante, una rete P2P decentralizzata si classifica in:

- **Reti non strutturate**, i nodi sono collegati tra loro in modo casuale. L'organizzazione delle reti segue principi molto semplici e l'aggiunta o la rimozione di nodi è un'operazione semplice e poco costosa. Lo svantaggio è che effettuare una ricerca non è molto efficiente poiché una qualsiasi interrogazione deve essere inviata in **flooding**, ovvero, ogni nodo deve inviare la richiesta a tutti gli altri nodi con cui ha un collegamento diretto a tutta la rete. Questo produce un volume di traffico rilevante e, nonostante questo, la ricerca potrebbe avere esito negativo.
- **Reti strutturate**, i nodi vengono organizzati seguendo un preciso insieme di regole, in modo da migliorare l'efficienza di alcune operazioni come, ad esempio, la ricerca di un particolare contenuto. Ad ogni peer è assegnato un ID ed ogni peer conosce un certo numero di peer. Ad ogni risorsa condivisa viene assegnato un ID, basato su una funzione hash applicata al contenuto della risorsa ed al suo nome. Una risorsa viene memorizzata nel nodo che ha l'ID più vicino a quello della risorsa. Quindi l'inserimento di un contenuto e di un nodo diventano operazioni più costose, ma la ricerca di una risorsa specifica diventa molto efficiente

In tanti modelli sono state sperimentati approcci ibridi dove si cerca di combinare l'approccio centralizzato con quello decentralizzato. Per fare ciò si eleggono dei peer come dei **group leader** che hanno maggiori informazioni rispetto agli altri. Ogni group leader tiene traccia del contenuto dei suoi figli e scambia questi contenuti con gli altri group leader

### 6.2.1 BitTorrent

BitTorrent è un protocollo P2P molto diffuso per la distribuzione di file. *Idea*: dividere un file in pezzi (da 256 KB) e far ridistribuire ad ogni peer i dati ricevuti, fornendoli a nuovi destinatari, in questo modo: si riduce il carico di ogni sorgente; si riduce la dipendenza dal distributore originale e si fornisce ridondanza. La condivisione dei file avviene

tramite un processo collaborativo chiamato **torrent**. Ogni peer che partecipa a un torrent preleva parti del file, chiamate **chunk**, da un altro peer e contemporaneamente trasmette i propri chunk ad altri peer che non li possiedono ancora. L'insieme dei peer che fanno parte di un torrent viene chiamato **swarm**. BitTorrent nella versione originale prevede un ulteriore elemento detto **tracker** che tiene traccia delle operazioni dello swarm. Nella figura il file da condividere è suddiviso in cinque chunk e i peer 2 e 4 possiedono già il file completo, gli altri ne hanno solo alcune parti. Si supponga ora che un nuovo peer voglia scaricare quel file: contatta un motore di ricerca specializzato indicando il nome del file e riceve un metafile, chiamato **file torrent**, che contiene le informazioni relative alla suddivisione in parti del file e l'indirizzo del tracker che gestisce quel particolare torrent. Il nuovo peer può ora contattare il tracker per ricevere l'indirizzo di alcuni peer nel torrent, solitamente chiamati **neighbor**. Il nuovo peer è ora parte del torrent e può iniziare a scambiare chunk. Ottenuto il file completo di tutte le sue parti, può abbandonare il torrent oppure contribuire ad aiutare gli altri peer, inclusi quelli che si sono uniti al torrent dopo di lui, ad ottenere tutte le parti del file. Un peer può anche abbandonare il torrent prima di avere ottenuto tutte le parti del file, per riagganciarsi eventualmente in un momento successivo. Strategie principali:

- **tit for tat**, inviare dati ai peer che inviano dati, scegliendo quelli che stanno inviando dati a frequenza maggiore
- ciascun peer scambia dati con al più 4 vicini, priorità ai vicini che stanno inviando dati alla velocità più alta

Questi meccanismi però tendono ad escludere dei nodi, quindi ogni peer classifica i suoi vicini in **choked** e **interested** dove ogni 10 secondi ogni peer aggiorna la sua classifica e ogni 30 secondi sceglie un choked a caso e lo promuove. In questo modo i nuovi arrivati possono entrare. L'ultimo meccanismo è quello di scambiare per primi i chunk più rari

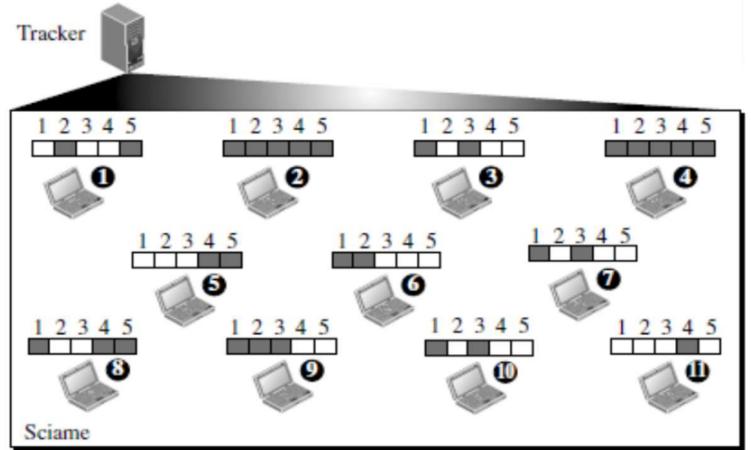
## 6.3 Sicurezza di rete

I tre obiettivi principali delle informazioni sono:

- **riservatezza**, solo mittente e destinatario devono essere in grado di comprendere il contenuto del messaggio trasmesso
- **integrità**, il contenuto delle comunicazioni non deve subire alterazioni non autorizzate durante la trasmissione
- **accessibilità**, l'informazione deve essere resa sempre disponibile alle entità autorizzate

Le minacce alla sicurezza delle reti sono molteplici, per esempio ci sono meccanismi per *intercettare* il traffico dati per spiare o rubarne il contenuto; attività di *spoofing* dove un attaccante finge di essere un utente diverso e invia una richiesta come se fosse quella dell'utente originario; *attacchi DoS e DDoS* dove si cerca di impedire agli utenti di accedere a certe risorse e altri tipi di attacchi... Tra gli elementi fondamentali di sicurezza vi è anche il **non ripudio**: proprietà che permette di assicurare che l'entità che ha generato i dati non possa in seguito negare di averlo fatto. Si studiano ora le contromisure adottabili per prevenire i problemi di sicurezza:

- **Crittografia**, si utilizza la **crittografia a chiave asimmetrica**, ovvero chiave (pubblica) per cifrare è diversa da quella per decifrare (privata) per scambiarsi la chiave di sessione. Per scambiarsi i dati si utilizza la **crittografia a chiave simmetrica** stessa chiave segreta usata per cifrare e decifrare
- **Funzioni hash**, risulta di fondamentale importanza per l'integrità di un messaggio. Per esempio un utente scrive un documento che non necessita di segretezza (dunque non deve essere cifrato), ma che deve rimanere integro, ovvero il suo contenuto non può essere modificato. Una funzione hash riceve un messaggio di lunghezza arbitraria e produce un **digest** che può essere utilizzata come verifica dell'integrità del messaggio
- **Message Authentication Code**, per garantire l'autenticazione del messaggio si crea un codice **MAC**. Mittente e destinatario condividono la chiave segreta K. Il mittente allega al messaggio il MAC concatenando la chiave segreta al messaggio: **h(KM)**. Il destinatario separa il messaggio dal MAC ed elabora un nuovo MAC dalla concatenazione della chiave segreta e del messaggio ricevuto. Confronta infine il MAC così ottenuto con il MAC ricevuto: se sono identici il messaggio è autentico e non è stato modificato



- **Firma digitale**, è un altro modo per garantire l'integrità e l'autenticazione di un messaggio. Funzionamento: il mittente firma il messaggio M cifrandolo con la sua chiave privata. La cifratura del messaggio M costituisce proprio la firma S del mittente. A questo punto il mittente invia sia il messaggio M sia la firma S al destinatario, il quale verifica l'autenticità del messaggio applicando la chiave pubblica del mittente. Se il messaggio che il destinatario ottiene è uguale a quello inviato dal mittente allora il messaggio è autentico.

### 6.3.1 Secure Sockets Layer

**SSL** è un protocollo di sicurezza ampiamente diffuso, ed è supportato da quasi tutti i browser. Fra i servizi forniti da SSL vi sono l'autenticazione di client e server, la riservatezza e l'integrità dei dati. Tipicamente SSL può ricevere dati da qualsiasi protocollo di livello applicazione, sebbene solitamente si utilizzi l'HTTP. I dati ricevuti dall'applicazione vengono firmati e crittografati, e poi passati a un protocollo affidabile di livello trasporto come il TCP. Le fasi principali dell'SSL sono:

- **handshake**, mittente e destinatario usano i loro certificati e chiavi private per autenticarsi a vicenda, e scambiarsi segreti condivisi. In questa parte si negozia la **cipher suite** per autenticare il client al server, e viceversa quando necessario, per scambiare informazioni utili alla definizione dei parametri crittografici segreti
- **derivazione delle chiavi**, mittente e destinatario usano il segreto condiviso per derivare il set di chiavi
- **trasferimento dati**, i dati da trasferire sono scomposti in serie di record

Se SSL viene utilizzato correttamente, tutto ciò che un utente malintenzionato sarà in grado di vedere è quale IP e porta sono collegati e approssimativamente quanti dati vengono inviati. Potrebbero essere in grado di terminare la connessione, ma sia il server che l'utente saranno in grado di dire che ciò è stato fatto da una terza parte. Tuttavia, non saranno in grado di intercettare alcuna informazione, il che lo rende essenzialmente un passaggio inefficace.

### 6.3.2 Sicurezza a livello rete: IPSec

IPSec è un insieme di protocolli sviluppato dall'IETF per fornire garanzie di sicurezza ai pacchetti di livello rete. IPSec serve ad autenticare e rendere confidenziali i pacchetto del protocollo IP. IPSec può operare in due modi:

- **modalità trasporto**, protegge ciò che viene fornito dal livello trasporto al livello rete. In altre parole, questa modalità protegge i dati encapsulati nei pacchetti di livello rete. La protezione è solo sui dati, l'intestazione del pacchetto IP non viene protetta. La modalità trasporto viene normalmente utilizzata quando occorre proteggere i dati scambiati tra mittente e destinatario
- **modalità tunnel**, permette di proteggere l'intero pacchetto IP. Con questa modalità l'intero pacchetto IP, inclusa l'intestazione, viene encapsulato in un pacchetto IPSec. Il tutto deve essere poi inserito in un pacchetto IP: verrà quindi aggiunta una nuova intestazione IP. La modalità tunnel viene solitamente utilizzata fra due router, fra un host e un router o fra un router e un host

IPSec definisce il **protocollo ESP** che fornisce autenticazione di sorgente, integrità e riservatezza. La formazione del pacchetto ESP:

1. il trailer ESP viene aggiunto al payload;
2. payload e trailer vengono crittografati;
3. viene aggiunta l'intestazione ESP;
4. intestazione ESP, payload ed ESP trailer vengono usati per generare i dati di autenticazione (che vengono aggiunti dopo trailer ESP);
5. viene aggiunta l'intestazione IP