

Reti di Calcolatori

Alessio Delgadillo

Anno accademico 2020-2021

Indice

1	Introduzione	4
1.1	Internet: una panoramica	4
1.1.1	Le reti	4
1.1.2	Commutazione (switching)	5
1.1.3	Internet	7
1.1.4	L'accesso a Internet	8
1.1.5	Metriche di riferimento	9
1.2	L'organizzazione dei protocolli in livelli	11
1.2.1	Stratificazione	11
1.2.2	OSI RM - <i>Open System Interconnection</i>	13
1.2.3	Lo stack protocollare TCP/IP	16
2	Livello applicazione	19
2.1	Introduzione	19
2.1.1	Protocollo dello strato di applicazione	19
2.1.2	Paradigmi del livello applicazione	20
2.2	Paradigma client/server	20
2.2.1	API: Application Programming Interface	21
2.2.2	Utilizzo dei servizi di livello trasporto	22
2.3	Applicazioni client/server standard	24
2.3.1	World Wide Web e HTTP	24
2.3.2	TELNET	31
2.3.3	Posta elettronica	33
2.3.4	Il DNS (Domain Name System)	43
2.3.5	FTP	50
2.3.6	Approfondimenti	56
2.4	Paradigma peer-to-peer	58
2.4.1	Reti P2P	58
2.4.2	BitTorrent: una rete P2P molto diffusa	62

3 Livello di trasporto	65
3.1 Introduzione	65
3.1.1 I servizi del livello trasporto	65
3.2 Il protocollo TCP (Transmission Control Protocol)	67
3.2.1 Servizi del protocollo TCP	67
3.2.2 Numeri di sequenza e di riscontro	68
3.2.3 Protocolli bidirezionali	69
3.2.4 Formato dei segmenti	69
3.2.5 La connessione TCP	71
3.2.6 Controllo degli errori	76
3.2.7 Finestra di trasmissione	82
3.2.8 Controllo di flusso	83
3.2.9 Controllo della congestione in TCP	84
3.3 Il protocollo UDP (User Datagram Protocol)	91
3.3.1 Struttura dei datagrammi utente	91
3.3.2 Servizi UDP	92
4 Livello di rete	95
4.1 Introduzione	96
4.1.1 Servizi a livello di rete	97
4.2 Protocolli di livello rete	99
4.2.1 Formato di datagrammi IPv4	100
4.2.2 Indirizzi IPv4	106
4.2.3 Inoltro dei datagrammi IP	119
4.2.4 ICMPv4	126
4.3 Architettura di un Router	130
4.4 Routing unicast	133
4.4.1 Concetti generali	133
4.4.2 Algoritmi di routing	135
4.4.3 Protocolli di routing unicast	145
4.5 IP versione 6	155
4.5.1 Formato dei datagrammi IPv6	156
4.5.2 Passaggio da IPv4 a IPv6	159
4.6 Approfondimenti	161
4.6.1 Checksum	161
4.6.2 NAT: problemi	161
5 Livello di collegamento	164
5.1 Introduzione	164
5.1.1 Nodi e collegamenti	164
5.1.2 Due tipi di collegamento	165

5.1.3	Due sotto-livelli	166
5.1.4	Servizi offerti	166
5.2	Indirizzamento a livello di collegamento	167
5.2.1	Address Resolution Protocol (ARP)	168
5.3	LAN cablate: protocollo Ethernet	172
5.3.1	Progetto IEEE 802	172
5.3.2	Ethernet Standard	172
5.3.3	Fast Ethernet	175
5.3.4	Gigabit Ethernet (802.3z)	176
5.4	Dispositivi di interconnessione	176
5.4.1	Repeater e hub	176
5.4.2	Switch di livello collegamento	177
5.4.3	Router	180
6	Sniffer & Portscanner	182
6.1	Introduzione	182
6.2	Sniffer	184
6.2.1	Cosa è uno sniffer?	184
6.2.2	Come lavora uno sniffer?	187
6.2.3	TCPDUMP	189
6.2.4	Wireshark	191
6.3	Port Scanner	192
6.3.1	NMAP	193
7	Cenni di Sicurezza di Rete	199
7.1	Introduzione	199
7.1.1	Obiettivi della sicurezza	199
7.2	Sicurezza della comunicazione	200
7.2.1	Attacchi	200
7.2.2	Cifratura a chiave simmetrica	202
7.2.3	Cifratura a chiave asimmetrica	204
7.2.4	Message digest	205
7.2.5	Message Authentication Code	206
7.2.6	Firma digitale	207
7.3	Comunicazione sicure nello stack protocollare	208
7.3.1	Sicurezza al livello rete: IPSec	208

Capitolo 1

Introduzione

1.1 Internet: una panoramica

1.1.1 Le reti

Una rete è un'interconnessione di dispositivi in grado di scambiarsi informazioni, quali *sistemi terminali* (*end system*), *router*, *switch* e *modem*. I sistemi terminali sono chiamati *host*. Un host può essere una macchina in genere di proprietà degli utenti e dedicata ad eseguire applicazioni, quale un computer desktop, un portatile, un cellulare o un tablet, oppure un server: tipicamente un computer con elevate prestazioni destinato a eseguire programmi che forniscono servizio a diverse applicazioni utente come, per esempio, la posta elettronica o il Web. I *router* sono dispositivi che collegano una rete ad altre reti, mentre gli *switch* (commutatori) collegano fra di loro più sistemi terminali a livello locale. È possibile che in una rete ci siano anche i modem, che trasformano la codifica dei dati. Tutti questi dispositivi vengono collegati utilizzando mezzi trasmissivi cablati o wireless, come cavi o onde radio, genericamente chiamati *link* (collegamenti).

LAN - *Local Area Network*

Una LAN (*Local Area Network*), o rete locale, è solitamente una rete privata che collega i computer in un singolo ufficio, un edificio o un complesso. Qualsiasi sistema terminale in una LAN deve avere un indirizzo che lo identifica univocamente nella rete. Un pacchetto inviato da un sistema terminale a 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, o mezzo *broadcast*, per cui il pacchetto inviato da un dispositivo veniva ricevuto da tutti gli altri. Il destinatario elaborava il pac-

chetto, 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 di inviare quindi il pacchetto al solo destinatario senza inviarlo agli altri dispositivi. Lo switch riduce il traffico nella LAN e consente a più coppie di dispositivi di comunicare contemporaneamente fra di loro, posto che non vi siano dispositivi sorgente o destinazione comune.

WAN - Wide Area Network

Anche una WAN (*Wide Area Network*), o rete geografica, è un'interconnessione di dispositivi in grado di comunicare, ma esistono notevoli differenze rispetto alle LAN. Quest'ultime sono solitamente di estensione limitata, mentre una WAN ha un'estensione decisamente superiore, potendo servire una città, una regione, una nazione o addirittura l'intero globo. Una LAN interconnette prevalentemente sistemi terminali, mentre una WAN interconnette anche switch, router o modem. Una LAN è solitamente di proprietà dell'organizzazione che la utilizza, mentre una WAN è generalmente creata e gestita da un operatore di telecomunicazioni, Internet Service Provider (ISP), che fornisce i suoi servizi alle organizzazioni che ne fanno uso. Oggi esistono due tipi di WAN: punto-punto e a commutazione (switched).

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

WAN a commutazione Una WAN *a commutazione (switched)* è una rete con più di due punti di terminazione. Essa viene utilizzata nelle dorsali dell'odierna rete globale.

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

1.1.2 Commutazione (switching)

Una internet (o internetwork) è data dall'interconnessione di reti, composte da link e dispositivi capaci di scambiarsi informazioni. In particolare i sistemi terminali appartenenti alla rete comunicano tra di loro per mezzo di dispositivi come switch e router che si trovano nel percorso (o rottta) tra il sistema sorgente e destinazione. In base al metodo adottato per determinare il percorso tra due sistemi terminali in comunicazione si possono distinguere due tipi di reti: a commutazione di circuito e a commutazione di pacchetto.

Reti a commutazione di circuito In una rete a commutazione di circuito (*circuit-switched network*) tra due dispositivi è sempre disponibile un collegamento dedicato, chiamato circuito; lo switch che collega i due dispositivi può solo attivarlo o disattivarlo. Sul percorso vengono dedicate risorse alla comunicazione e sono garantite per tutta la sua durata.

Svantaggi

- necessaria una fase di instaurazione (*setup*) della comunicazione;
- le risorse rimangono inattive se non utilizzate (non c'è condivisione).

Vantaggi

- performance (garantita)

Reti a commutazione di pacchetto In una rete di computer la comunicazione fra i due lati viene effettuata trasmettendo blocchi di dati chiamati *pacchetti*. Invece di avere una comunicazione continua, i due computer comunicano scambiandosi pacchetti di dati. Gli switch sono in grado di memorizzare i pacchetti, oltre che inoltrarli, essendo i pacchetti entità indipendenti che possono essere memorizzate e inviate successivamente. Mentre la commutazione di circuito prealloca l'utilizzo del collegamento trasmissivo con collegamenti garantiti, nella commutazione di pacchetto, pacchetto dopo pacchetto la capacità trasmissiva dei collegamenti sarà condivisa solo tra gli utenti che devono trasmettere sul collegamento. La sequenza dei pacchetti non segue uno schema prefissato, la condivisione di risorse avviene su richiesta (detta anche multiplexing statistico delle risorse).

Vantaggi

- le risorse vengono usate *a seconda delle necessità*

Svantaggi

- Contesa per le risorse
 - la richiesta di risorse può eccedere il quantitativo disponibile;
 - congestione: accodamento dei pacchetti, attesa per l'utilizzo del collegamento.
- Trasmissione *store-and-forward*

- il commutatore (es. router) deve ricevere l'intero pacchetto prima di poter cominciare a trasmettere sul collegamento in uscita: ritardo di store and forward
- attesa dei pacchetti in code di output (buffer): ritardo di coda
- i buffer hanno dimensione finita: perdita di pacchetti

1.1.3 Internet

Una internet (con *i minuscola*) è costituita da due o più reti interconnesse. L'internet più famosa è chiamata Internet (*I maiuscola*) 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. Nuove reti si aggiungono facilmente. Le reti degli host sono connesse a Internet attraverso una gerarchia di fornitori di servizi Internet (Internet Service Provider). Al livello più alto ci sono le dorsali, reti particolarmente estese di proprietà di qualche compagnia telefonica. Queste reti dorsali sono interconnesse tramite sistemi di comunicazione notevolmente complessi, chiamati *peering point*; si possono anche definire degli *Internet eXchange Point* (IXP): punto di incontro (può essere gestito da un'azienda terza) per il peering tra due o più ISP. Al secondo livello vi sono reti più piccole, chiamate reti dei provider, che utilizzano a pagamento i servizi delle dorsali. Le reti dei provider sono connesse alle dorsali e a volte ad altre reti di provider. Le reti private sono reti ai confini di Internet che utilizzano i servizi a pagamento forniti dalle reti dei provider. Le dorsali sono spesso chiamate ISP internazionali, le reti dei provider ISP nazionali o regionali.

Vista dei “servizi” Infrastruttura che fornisce servizi di comunicazione alle applicazioni:

- WWW, email, giochi, e-commerce, database, controllo remoto, voice over IP, ecc.
- applicazioni distribuite che coinvolgono più host

Si distinguono due tipi di servizi fondamentali

- senza connessione (*connectionless*): senza garanzia di consegna;
- orientati alla connessione (*connection-oriented*): garantiti in integrità, completezza e ordine.

Vista “delle entità software”

- **Applicazioni:** elaborano e si scambiano le informazioni
- **Protocolli:** regolamentano la trasmissione e la ricezione dei messaggi (es. TCP, IP, HTTP, FTP, PPP)
- **Interfacce:** definite in seguito, sono le “membrane” che separano gli strati della pila protocollare

Gli standard Internet e del Web

- **Internet Engineering Task Force (IETF):** l’organismo che studia e sviluppa i protocolli in uso su Internet. Si basa su gruppi di lavoro a cui chiunque può partecipare.
- **Internet Corporation for Assigned Names and Numbers (ICANN):** coordina il sistema dei nomi di dominio (DNS), assegna i gruppi di indirizzi 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 (es. HTML, XML, Semantic Web, ecc.)

1.1.4 L’accesso a Internet

Internet è una 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. Le reti di accesso sono di varia tipologia:

- accesso via rete telefonica
 - servizio dial-up
 - Digital Subscriber Line (DSL)
 - tecnologie per accesso in fibra ottica
- accesso tramite reti wireless
 - 3G, 4G, 5G
- collegamento diretto (es. collegamenti WAN dedicati ad alta velocità)

1.1.5 Metriche di riferimento

Un argomento di cui si parla spesso nell'ambito delle reti è la velocità della rete o di un collegamento, ovvero quanto velocemente si riesce a trasmettere e ricevere dati. In realtà, il concetto di velocità è molto ampio e coinvolge vari fattori che influiscono sulle prestazioni (*performance*) di una rete.

Aampiezza di banda e bit rate

Con il termine ampiezza di banda si indicano due concetti leggermente diversi ma strettamente legati. Al livello di caratterizzazione di un canale o di un sistema trasmittivo, l'ampiezza di banda (*bandwidth*), o banda, è una quantità che si misura in hertz e rappresenta la larghezza dell'intervallo di frequenze utilizzato dal sistema trasmittivo, ovvero l'intervallo di frequenze che un mezzo fisico consente di trasmettere senza danneggiare il segnale in maniera irrecuperabile. In generale, maggiore è l'ampiezza di banda, maggiore è la quantità di informazione che può essere veicolata attraverso il mezzo trasmittivo.

Quando viene espressa in bit al secondo (bps), la banda rappresenta invece il *bit* o *trasmision rate* (velocità di trasmissione), per brevità *rate*, ovvero la quantità di bit al secondo che un certo link garantisce di trasmettere. I due concetti sono legati ma non coincidenti, in quanto il bit rate di un link dipende sia dalla banda (in hertz) che dalla specifica tecnica di trasmissione, o formato di modulazione digitale utilizzato. In generale, il bit rate è proporzionale alla banda in hertz. Infine, per banda di un tipo di rete, si intende il bit rate garantito (nominalmente) dai suoi link.

Throughput

Il throughput indica quanto velocemente riusciamo effettivamente a inviare i dati tramite una rete. Sebbene a prima vista il rate e il throughput sembrino la stessa cosa, in realtà non lo sono. Un link può avere un rate di B bps, ma possiamo inviare solo T bps tramite quel link, con T sempre inferiore a B . In altre parole, il rate è una misura della potenziale velocità di un link; il throughput è una misura dell'effettiva velocità di un link (quanto velocemente riusciamo a inviare i dati in realtà). È importante notare che il throughput è definito come il numero dei bit che passano attraverso un punto della rete in un secondo. In un percorso da una sorgente a una destinazione un pacchetto può passare attraverso numerosi link, ognuno con un throughput diverso: il throughput medio è determinato dal collo di bottiglia. In generale in un percorso con n link in serie abbiamo

$$\text{Throughput} = \min\{T_1, T_2, \dots, T_n\}.$$

Si noti che il throughput dipende non solo dalla velocità di trasmissione del collegamento ma anche dalla quantità di dati (flussi di traffico aggiuntivi rispetto a quello di interesse), **effetti dei protocolli**, ecc.

Latenza (ritardo)

La latenza, o ritardo (delay), definisce quanto tempo serve affinché un intero messaggio arrivi completamente a destinazione dal momento in cui il primo bit parte dalla sorgente. I ritardi si possono dividere in quattro tipi: *ritardo di trasmissione*, *ritardo di propagazione*, *ritardo di elaborazione* e *ritardo di accodamento*. La latenza, o ritardo totale, deriva dalla seguente formula:

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

- **Ritardo di propagazione:** tempo che serve a un bit per viaggiare da un punto A a un punto B sul mezzo trasmittivo. Dipende dalla distanza (valori tipici da pochi microsecondi a centinaia di millesekondi)

$$Ritardo_{pr} = \text{distanza}/\text{velocità}_{\text{propagazione}}$$

- **Ritardo di trasmissione:** tempo necessario per immettere un pacchetto sulla linea

$$Ritardo_{tr} = \text{lunghezza del pacchetto}/\text{rate}_{\text{trasmissione}}$$

- **Ritardo di accodamento:** tempo in cui il pacchetto attende nella coda del router (dipende dalla congestione)
- **Ritardo di elaborazione:** tempo per l'elaborazione al nodo intermedio (in genere pochi microsecondi, o anche meno)

Perdita di pacchetti

Un'altra questione che influisce gravemente sulle prestazioni della comunicazione è il numero di pacchetti persi durante la trasmissione. Quando un router riceve un pacchetto mentre ne sta elaborando un altro, il pacchetto ricevuto deve essere memorizzato nel buffer di input in attesa del suo turno. Un router tuttavia ha un buffer di input di dimensioni limitate. Può arrivare il momento in cui il buffer è pieno e il pacchetto successivo deve essere saltato. L'effetto della perdita del pacchetto è che deve essere inviato nuovamente, il che a sua volta può creare un'eccedenza di dati e causare la perdita di altri pacchetti.

Ritardo end-to-end

Ipotizzando che ci siano $N-1$ router tra origine e destinazione e che siano trascurabili i ritardi di congestione, che il ritardo di elaborazione ai router e al nodo mittente sia $Ritardo_{el}$, il ritardo di trasmissione sia $Ritardo_{tr}$ e il ritardo di propagazione su ciascun collegamento valga $Ritardo_{pr}$

$$Ritardo_{nodo} = Ritardo_{el} + Ritardo_{tr} + Ritardo_{pr}$$
$$Ritardo_{end-to-end} = N \cdot Ritardo_{nodo}$$

Nel caso generale di ritardi eterogenei

$$Ritardo_{end-to-end} = Ritardo_{nodo1} + Ritardo_{nodo2} + \dots + Ritardo_{nodoN}$$

Prodotto rate-ritardo

Il rate e il ritardo sono due metriche delle prestazioni di un link. Tuttavia, ciò che è realmente importante nella comunicazione dei dati è il loro prodotto: numero massimo di bit che il link può contenere.

1.2 L'organizzazione dei protocolli in livelli

Un termine che si utilizza frequentemente quando si parla di Internet è protocollo. Un protocollo definisce le regole che il mittente e il destinatario, così come tutti i sistemi intermedi coinvolti, devono rispettare per essere in grado di comunicare. In situazioni particolarmente semplici potrebbe essere sufficiente un solo protocollo, in situazioni più complesse potrebbe essere opportuno suddividere i compiti fra più livelli (*layer*), nel qual caso è richiesto un protocollo per ciascun livello: si parla dunque di *layering di protocolli*, o di organizzazione (o strutturazione) dei protocolli in livelli.

1.2.1 Stratificazione

Scomposizione dei sistemi complessi:

- la struttura esplicita permette l'identificazione delle relazioni tra gli elementi di un sistema complesso
 - *modello di riferimento stratificato*
 - *suddivisione di funzioni e attori*
- la modularizzazione facilita la manutenzione e l'aggiornamento del sistema

- un modulo (o più precisamente livello/strato) svolge un insieme delimitato di compiti e nel sistema appare come una black box (input/output)
- ciascun livello offre servizi allo strato superiore implementando azioni all'interno del livello stesso e utilizzando i servizi del livello inferiore
- separazione tra servizi offerti (interfaccia) e implementazione: il cambiamento dell'implementazione di un servizio in un livello è sostanzialmente trasparente per il resto del sistema.

Principi base della stratificazione

- **Separation of Concern:** separazione degli interessi e delle responsabilità, fare ciò che compete, delegando ad altri tutto ciò che è delegabile
- **Information Hiding:** nascondere tutte le informazioni che non sono indispensabili affinché il committente possa comodamente definire l'operazione

In sintesi

- **Modello stratificato**
 - costituito da sistemi di consumatori/produttori
 - sistemi organizzati in strati funzionali (livelli)
 - ogni strato fornisce servizi allo strato superiore e usa i servizi di quello inferiore
 - ogni strato scambia informazioni direttamente solo con gli strati adiacenti
 - in ogni comunicazione i due strati omologhi svolgono funzioni reciproche
 - esistono sistemi (intermedi) che implementano solo alcune funzioni
- **Requisiti**
 - efficiente: minimizzare lo sforzo globale
 - efficace: massimizzare i risultati

- **Vantaggi**

- scomponere il problema in sottoproblemi più semplici da trattare: il singolo strato è più semplice del sistema nel suo complesso
 - * semplificazione della progettazione, implementazione e manutenzione del software
- rende i vari livelli indipendenti: posso modificare l'implementazione di uno strato senza dover cambiare gli altri strati (adiacenti e non), a patto che l'interfaccia non cambi
 - * i servizi forniti dagli strati inferiori possono essere usati da più entità negli strati adiacenti superiori
- definendo solamente servizi e interfacce, i livelli diversi possono essere sviluppati da soggetti diversi

Criteri di stratificazione

- Ogni livello logico di astrazione è realizzato in un apposito strato: un livello viene creato quando si rende necessario un diverso grado di astrazione.
- Ogni strato svolge una sola e ben definita funzione
- Il flusso dati attraverso le interfacce di ogni strato deve essere minimizzato
- Il numero degli strati deve essere minimizzato, compatibilmente con la loro complessità: numero 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

1.2.2 OSI RM - *Open System Interconnection*

Le prime reti di calcolatori nascono come *sistemi chiusi*: rete specializzata per specifici servizi e nella quale tutti i componenti sono dello stesso costruttore. Tuttavia, tali sistemi portano a problemi di interoperabilità poiché gli apparati non riescono ad interpretare i segnali degli altri in altre reti (parlano linguaggi diversi) e di conseguenza i programmi applicativi non riescono ad operare in ambiente distribuito. Da qui l'esigenza di realizzare dei *sistemi aperti*, ovvero dei sistemi in cui si stabiliscono delle regole comuni; vengono definiti degli standard con l'obiettivo di realizzare reti di calcolatori che potessero colloquiare con reti e dispositivi di altri fornitori.

Un sistema aperto (*open system*) è un sistema che implementa protocolli aperti:

- i dettagli dei protocolli sono disponibili pubblicamente;
- i cambiamenti sono gestiti da un'organizzazione la cui partecipazione è aperta al pubblico.

L'**International Organization for Standards (ISO)** ha specificato uno standard per l'interconnessione di sistemi aperti: modello di riferimento *Open System Interconnection* (OSI). OSI ha molto influenzato il modo di pensare ai protocolli stratificati.

Modello ISO/OSI

Il modello ISO/OSI prevede di dividere le funzionalità del protocollo di telecomunicazione in strati, o layers, ognuno dei quali svolge una parte piccola e indipendente dalle altre allo scopo di permettere una realizzazione o revisione delle singole funzionalità senza dover toccare le altre o anche di permettere una compatibilità a livelli diversi tra diverse implementazioni. La comunicazione tra i vari livelli è assicurata da chiamate standard; ogni livello è tenuto a rispondere in maniera corretta alle chiamate che gli competono e che verranno generate dai due livelli ad esso adiacenti (superiore e inferiore). La modalità con cui le funzioni competenti ad un livello vengono svolte non è visibile dall'esterno che ne è così svincolato.

Modello a strati

- Uno strato fornisce servizi allo strato superiore e riceve servizi dallo strato inferiore
- Lo strato *n-esimo* di una entità comunica con lo strato *n-esimo* di un'altra entità secondo un protocollo assegnato
- La comunicazione tra due strati avviene attraverso un'interfaccia

Definizioni

- **Strato:** modulo interamente definito attraverso i servizi, protocolli e le interfacce che lo caratterizzano (spesso indicato come livello)
- **Servizio:** insieme di primitive (operazioni) che uno strato fornisce ad uno strato soprastante

- **Interfaccia:** insieme di regole che governano il formato e il significato delle unità di dati (es. messaggi, segmenti, o pacchetti) che vengono scambiati tra due strati adiacenti della stessa entità
- **Protocollo:** insieme di regole che
 - permettono a due entità omologhe uno scambio efficace e efficiente delle informazioni
 - 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.

Pila di Protocolli - *Protocol Stack*

Strati di supporto alla rete e infrastruttura trasmissiva: trasmissione dei dati (realizzazione software e hardware)

- Protocollo di livello 1 **fisico**: trasmette un flusso di bit
- Protocollo di livello 2 **datalink**: consegna trame sul link
- Protocollo di livello 3 **rete**: instradamento del traffico
- Protocollo di livello 4 **trasporto**: trasferimento dati tra host

Strati di supporto all'elaborazione e interazione con utente (realizzazione software)

- Protocollo di livello 5 **sessione**: controllo del dialogo
- Protocollo di livello 6 **presentazione**: unificazione dati
- Protocollo di livello 7 **applicazione**: elaborazione dati

Inoltre, è possibile avere dei nodi intermedi: nello schema teorico di principio i primi 4 livelli superiori del modello ISO/OSI si riferiscono al dialogo *end-to-end*, cioè fra host terminali, mentre dal livello *rete* in giù si gestisce il trasferimento dati tra nodi intermedi.

Per quanto riguarda il livello *trasporto* abbiamo due modalità di servizio fondamentali:

- *connection-oriented*, nella quale c'è un'associazione logica tra due o più sistemi e, anche se siamo in una rete di commutazione di pacchetto, si riesce a costruire l'astrazione di connessione (instaurazione della connessione, trasferimento dei dati, chiusura della connessione);
- *connection-less*, dove i dati vengono trasferiti end-to-end ma senza stabilire una connessione.

Flusso dell'informazione

Per la rete, l'informazione ha origine al livello *applicazione* e discende i vari livelli fino alla trasmissione sul canale *fisico*. Ogni livello aggiunge all'informazione del livello superiore una propria sezione informativa, o più di una (*incapsulamento*): header che contiene informazioni riguardanti esclusivamente quel livello. Per i dati ricevuti si segue il cammino inverso: processo inverso. La definizione di *incapsulamento* è tale da garantire la possibilità di estrarre i dati precedentemente incapsulati.

Incapsulamento

- **Header:** qualificazione del pacchetto dati per questo livello;
- **Data:** payload proveniente dal livello superiore
- **Trailer:** generalmente usato in funzione di trattamento dell'errore (rivelazione, correzione)

1.2.3 Lo stack protocollare TCP/IP

TCP/IP è una famiglia di protocolli attualmente utilizzata in Internet. Si tratta di una gerarchia di protocolli, costituita da moduli interagenti, 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. Tale organizzazione dei protocolli viene chiamata *pila* o *stack protocollare TCP/IP*. Definita in origine in termini di quattro livelli software soprastanti a un livello hardware, la pila TCP/IP è oggi intesa come composta di 5 livelli:

1. *fisico*
2. *link*
3. *rete*
4. *trasporto*
5. *applicazione*

Il compito dei livelli *applicazione*, *trasporto* e *rete* è di tipo end-to-end, ovvero fra i due host terminali coinvolti nella comunicazione. Invece il compito dei livelli di *collegamento* e *fisico* è *hop-to-hop*, dove un *hop* è un host o un router. In altre parole il dominio dei compiti dei tre livelli superiori è la internet, mentre il dominio dei compiti dei due livelli inferiori è il link.

Livello applicazione

La comunicazione al livello applicazione avviene tra due *processi* (due programmi eseguiti a questo livello). Per comunicare, un processo invia una richiesta all'altro processo e ne riceve una risposta. Questo livello in Internet prevede numerosi protocolli predefiniti: HTTP, SMTP, FTP, ...

Livello di trasporto

Il livello di trasporto nell'host sorgente riceve il messaggio dal livello applicazione, lo incapsula in un pacchetto di livello di trasporto (chiamato *segmento*) e lo invia, tramite la connessione logica (virtuale), al livello di trasporto dell'host destinatario. Esistono più protocolli a livello di trasporto: TCP, UDP.

Livello di rete

Il livello di rete (network) è responsabile della comunicazione host-to-host e dell'instradamento e inoltro dei pacchetti attraverso i possibili percorsi. I protocolli di questo livello sono: IP, ICMP.

Livello di link

Il livello di collegamento (*data-link*) si occupa del trasferimento dati in frame attraverso il collegamento tra elementi di rete vicini.

Livello fisico

Il livello fisico si occupa di trasferire i singoli bit di un frame attraverso il link.

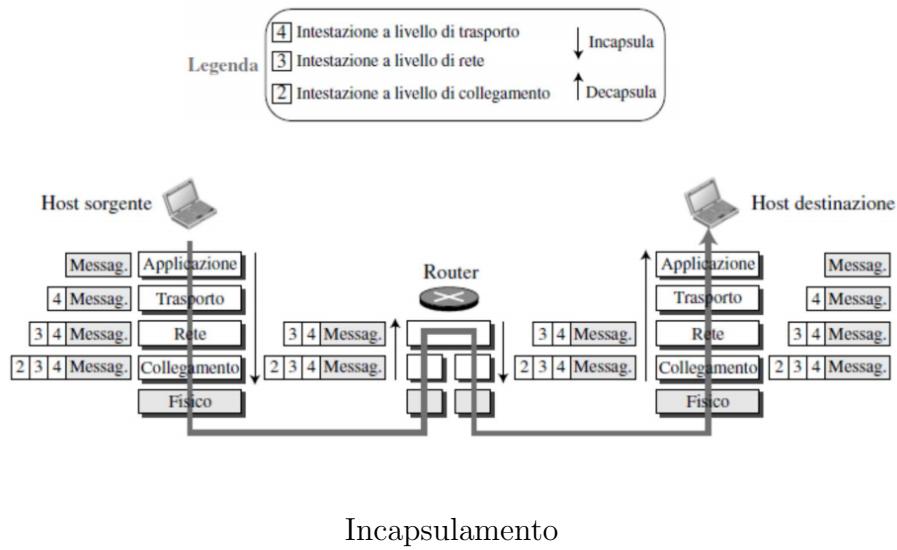
Incapsulamento

A livello applicazione i dati da scambiare vengono chiamati *messaggi*. Un messaggio solitamente non contiene alcuna *intestazione* (*header*) o *trailer*. Il messaggio viene passato al livello trasporto.

Il livello trasporto considera il messaggio ricevuto come *payload*, ovvero come carico dati, che deve trasportare e gli aggiunge il suo header; il risultato dell'incapsulamento è un *segmento*.

Il livello di rete considera il pacchetto di livello trasporto passatogli come proprio carico dati e gli aggiunge un'intestazione, il risultato è un pacchetto di livello di rete, chiamato *datagramma*.

Il livello di collegamento considera il pacchetto di livello di rete come payload e gli aggiunge la propria intestazione, il risultato è un pacchetto di livello di collegamento, chiamato *frame*.



ISO/OSI vs TCP/IP

- ISO/OSI
 - generale
 - definizione di servizio, interfaccia e protocollo
 - poco efficiente, alcuni livelli poco utili
 - standard difficili
 - telco-oriented
 - poca tempestività
- TCP/IP
 - standard de facto
 - implementation driven
 - specifiche non astratte e rigorose
 - modello non generale
 - oltre a TCP/IP, presenza di protocolli minori (per problemi ad-hoc), difficili da rimpiazzare

Capitolo 2

Livello applicazione

La rete Internet, nelle sue componenti sia hardware che software, è stata progettata e sviluppata per fornire servizi a livello applicazione. Questi servizi vengono forniti agli utenti della rete dal quinto livello della pila di protocolli TCP/IP. Lo scopo degli altri livelli è rendere possibile questi servizi.

2.1 Introduzione

Il livello applicazione fornisce servizi all'utente. La comunicazione è fornita per mezzo di una connessione logica: questo significa che i livelli applicazione nei due lati della comunicazione agiscono come se esistesse un collegamento diretto (immaginario) attraverso il quale poter inviare e ricevere messaggi.

2.1.1 Protocollo dello strato di applicazione

Definisce

- i **tipi di messaggi** scambiati a livello applicativo (es: di richiesta e di risposta)
- la **sintassi** dei vari tipi di messaggio (i campi del messaggio)
- la **semantica** dei campi (significato)
- le **regole** per determinare quando e come un processo invia messaggi o risponde ai messaggi

2.1.2 Paradigmi del livello applicazione

Dovrebbe ormai essere chiaro che per utilizzare Internet sono necessari due programmi che interagiscono l'uno con l'altro: uno eseguito su un certo computer in qualche parte del mondo, l'altro eseguito su un secondo computer in qualsiasi altro luogo. I due programmi hanno la necessità di scambiarsi i messaggi sfruttando l'infrastruttura di Internet. I due programmi applicativi devono essere entrambi in grado di richiedere e offrire servizi, oppure ciascuno deve occuparsi di uno dei due compiti? Con il passare del tempo, sono stati sviluppati due approcci (paradigmi) diversi: il paradigma client/server e quello peer-to-peer.

Client/server: il paradigma tradizionale

L'approccio tradizionale è chiamato *client/server*, ed è stato il paradigma più utilizzato fino a pochi anni fa. Con questo paradigma il fornitore di servizi è un programma applicativo chiamato processo *server*, sempre in esecuzione in attesa che un altro programma applicativo, chiamato processo *client*, apra una connessione via Internet e richieda un servizio.

Peer-to-peer: il nuovo paradigma

Un nuovo approccio, chiamato *paradigma peer-to-peer* (spesso abbreviato con *paradigma P2P*), è emerso come risposta alle necessità di alcune nuove applicazioni. In questo paradigma non è necessario che un processo server sia permanentemente in esecuzione in attesa che si connetta il processo client. Questa responsabilità viene condivisa tra i peer: un computer connesso a Internet può ricevere un servizio in un istante e offrire servizi in un altro momento. In alcuni casi, un computer può perfino fornire e ricevere servizi contemporaneamente.

Paradigma misto

Un'applicazione può sfruttare un mix dei due paradigmi combinando i vantaggi di ciascuno di essi.

2.2 Paradigma client/server

Nel paradigma client/server la comunicazione a livello applicazione avviene tra due programmi applicativi in esecuzione chiamati **processi**: un *client* e un *server*. Un client è un programma in esecuzione che inizia la comunicazione

invia una richiesta; un server è altro programma applicativo che attende le richieste dai client. Il server gestisce le richieste ricevute dai client, prepara i risultati e li rispedisce da un client. Questa definizione implica che il server sia in esecuzione quando arriva una richiesta da un client, mentre il client può essere in esecuzione solamente quando necessario.

2.2.1 API: Application Programming Interface

Un programma è solitamente scritto in un linguaggio di programmazione con un insieme predefinito di istruzioni che indicano al computer le operazioni da eseguire. Se si vuole sviluppare un programma capace di comunicare con un altro programma, è necessario un nuovo insieme di istruzioni per chiedere ai primi quattro livelli dello stack TCP/IP di aprire la connessione, inviare/ricevere dati e chiudere la connessione. Un insieme di istruzioni di questo tipo viene chiamato *API* (Application Programming Interface). Nella programmazione, un'interfaccia non è altro che un insieme di istruzioni usato per l'interazione tra due entità. In questo caso una delle entità è il processo a livello applicazione e l'altra è il sistema operativo che implementa i primi quattro livelli TCP/IP. Il produttore di un computer include i primi quattro livelli TCP/IP nel sistema operativo e fornisce le relative API. I processi eseguiti a livello applicazione sono così in grado di comunicare con il sistema operativo per inviare e ricevere messaggi via Internet. Sono state sviluppate diverse API di comunicazione, tra le quali l'*interfaccia socket* (*socket interface*).

Socket

Sebbene una socket possa apparire come un terminale o un file, non è un'entità fisica di questo tipo: è un'astrazione. Si tratta in realtà di una *struttura dati* creata e utilizzata dal programma applicativo. Si può dire che, per quanto riguarda il livello applicazione, comunicare fra un processo client e un processo server significa comunicare tra due socket create nei due lati della comunicazione.

Il client considera la socket come l'entità che riceve le richieste e fornisce le risposte; il server vede la socket come entità che gli sottopone le richieste e alla quale inviare le risposte.

Socket address

L'interazione fra client e server è una comunicazione bidirezionale. Nella comunicazione bidirezionale si rende necessaria un coppia di indirizzi: *locale*

(mittente) e *remoto* (destinatario). L’indirizzo locale in una direzione è l’indirizzo remoto nell’altra direzione e viceversa. Poiché la comunicazione nel paradigma client/server avviene fra due socket, è necessaria una coppia di socket address (indirizzo delle socket) per la comunicazione: un socket address locale ed uno remoto.

Un socket address deve innanzitutto identificare l’host sul quale il processo client o server è in esecuzione: un host in Internet viene definito univocamente dal proprio indirizzo IP, che solitamente è un intero a 32 bit. Ma sullo stesso computer potrebbero essere eseguiti contemporaneamente diversi processi client o server, di conseguenza è necessario utilizzare un altro identificatore per specificare il particolare client o server coinvolto nella comunicazione: un programma applicativo può essere identificato da un numero di porta, un intero a 16 bit. Un socket address sarà quindi composto dalla combinazione di un indirizzo IP e un numero di porta.

2.2.2 Utilizzo dei servizi di livello trasporto

Una coppia di processi fornisce servizi agli utenti di Internet, siano questi persone o applicazioni. La coppia dei processi, tuttavia, deve utilizzare i servizi offerti dal livello trasporto per la comunicazione, poiché non vi è una comunicazione fisica a livello applicazione. Nel livello trasporto dalla pila di protocolli TCP/IP sono previsti due protocolli principali: UDP e TCP.

Protocollo UDP

UDP fornisce un servizio di trasferimento di datagrammi utente (user datagram) inaffidabile, privo di connessione (connectionless). L’assenza di connessione significa che non viene stabilita un’associazione logica fra i due lati della comunicazione che si scambiano i messaggi. Ogni messaggio è un’entità indipendente incapsulata in un pacchetto chiamato datagramma utente. Per UDP non esiste alcuna relazione (connessione) fra datagrammi utente consecutivi provenienti dalla medesima sorgente e diretti allo stesso destinatario.

UDP non è un protocollo affidabile. Sebbene possa cercare di verificare che i dati non vengano corrotti durante la trasmissione, non chiede al mittente di rispedire gli eventuali datagrammi utente corrotti o persi. Nonostante le limitazioni, UDP presenta dei vantaggi per alcune applicazioni, tra cui il fatto che sia un protocollo “orientato al messaggio”, ovvero indichi in modo molto chiaro i confini dei messaggi trasmessi invece di creare un unico flusso di comunicazione.

Protocollo TCP

Il protocollo TCP fornisce un servizio basato su un flusso di byte (*byte-stream*) affidabile e *orientato alla connessione* (connection oriented). Prevede che le due entità in comunicazione stabiliscano per prima cosa una connessione logica scambiandosi alcuni pacchetti specifici per l'apertura della connessione. In questa fase, chiamata *handshaking* (stretta di mano), le due entità stabiliscono alcuni parametri fra cui la dimensione dei pacchetti dati che potranno essere scambiati, la dimensione dei buffer necessari per memorizzare temporaneamente i dati durante la comunicazione e via così. Completata la fase di handshaking, le due entità possono inviarsi dati suddivisi in cosiddetti *segmenti*, nelle due direzioni. Numerando i byte che vengono trasmessi, è possibile verificare la corretta sequenza dei byte ricevuti. Se per esempio alcuni byte risultassero persi o corrotti, il destinatario può richiederne la trasmissione, cosa che rende affidabile il protocollo TCP.

TCP è in grado di gestire anche il controllo del flusso e il controllo della congestione. Un inconveniente del protocollo TCP è il fatto di non essere orientato al messaggio, ovvero di non delimitare i messaggi scambiati bensì di fornire un flusso continuo di byte. È quindi un compito dell'applicazione quello di delimitare ed individuare i messaggi all'interno di questo flusso.

Che tipo di trasporto è richiesto da un'applicazione?

- **Throughput:** tasso al quale il processo mittente può inviare bit al processo ricevente.
 - alcune applicazioni (es. multimedia) richiedono un certo throughput minimo per essere efficaci
 - altre apps (*elastic apps*) possono far uso di poco o tanto throughput (a seconda di quanto ne trovano a disposizione)
- **Perdita dei dati**
 - alcune applicazioni (es. audio) possono tollerare alcune perdite
 - altre applicazioni richiedono un trasferimento dati affidabile al 100%
- **Sensibilità ai ritardi:** alcune applicazioni (es. teleconferenza, giochi interattivi) richiedono un basso ritardo per essere efficaci

2.3 Applicazioni client/server standard

2.3.1 World Wide Web e HTTP

World Wide Web

L'idea alla base del Web venne proposta per la prima volta da Tim Berners-Lee nel 1989 al CERN (Conseil Européen pour la Recherche Nucléaire) per consentire ai vari ricercatori sparsi nel mondo di accedere in modo facile alle pubblicazioni scientifiche. Il Web venne aperto al mondo commerciale nei primi anni '90.

Oggi 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. L'enorme diffusione e la crescita esplosiva del Web sono in stretta relazione con due termini nella frase precedente: *distribuiti* e *collegati*. La distribuzione favorisce la crescita del Web: qualsiasi server Web nel mondo può aggiungere un nuova pagina alla collezione e renderla disponibile a tutti gli utenti di Internet senza dover sovraccaricare un ipotetico server centralizzato.

Il collegamento delle pagine web è ottenuto tramite una tecnica chiamata *hypertext* (ipertesto), che venne introdotta molti anni prima dell'avvento di Internet. Il concetto alla base dell'ipertesto è il collegamento (detto link). L'idea 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 proprio 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 e video.

Uniform Resource Identifier

Una URI è una forma generale (definita dagli standard IETF) per identificare una risorsa presente sulla rete.

- **Uniform:** uniformità della sintassi dell'identificatore anche se i meccanismi per accedere alle risorse possono variare
- **Resource:** qualsiasi cosa abbia un'identità (documento, servizio, immagine, collezione di altre risorse)
- **Identifier:** informazioni che permettono di distinguere un oggetto da altri oggetti (entro un ambito definito di identificazione)

Ci sono due tipi di URI:

- **Uniform Resource Locator (URL)**: sottoinsieme di URI che identifica le risorse attraverso il loro meccanismo di accesso (es. protocollo HTTP)
- **Uniform Resource Name (URN)**: sottoinsieme di URI che devono rimanere globalmente unici e persistente anche quando la risorsa cessa di esistere e diventa non disponibile

URL (Uniform Resource Locator)

Una pagina Web, essendo un file, deve avere un identificatore univoco per distinguerla dalle altre pagine. Per identificare una pagina Web sono necessari tre identificatori: host, porta e percorso. Tuttavia, prima di definire la pagina Web è necessario indicare al browser quale applicazione client/server si desidera utilizzare, specificando il cosiddetto protocollo. Per identificare una pagina Web sono quindi necessari quattro parametri: il primo indica lo strumento che deve essere utilizzato per recuperare la pagina Web, la combinazione degli altri tre identifica l'oggetto ricercato (pagina Web).

- **Protocollo.** Il primo identificatore è un'abbreviazione che indica il protocollo client/server necessario per il trasferimento.
- **Host.** L'identificatore dell'host può essere un indirizzo IP (in notazione decimale puntata) o il nome univoco assegnato al dominio di un host.
- **Porta.** La porta, un intero di 16 bit, è predefinito per molte applicazioni client/server standard.
- **Percorso.** Il percorso identifica la posizione e il nome del file nel filesystem. Il formato di questo identificatore dipende normalmente dal sistema operativo utilizzato.

Questi quattro elementi sono raggruppati assieme sotto la forma di *URL*, come ad esempio:

- **protocol://host/path** utilizzato nella maggior parte dei casi
- **protocol://host:porta/path** utilizzato quando è necessario specificare il numero di porta

URI assolute e relative

Le URI possono essere assolute o relative

- **URI assoluta:** identifica una risorsa indipendentemente dal contesto in cui è usata;
- **URI relativa:** informazioni per identificare una risorsa in relazione ad un'altra URL (è priva dello schema e della authority).

Le URI relative non viaggiano sulla rete, ma sono interpretate dal browser in relazione al documento di partenza.

HTTP (HyperText Transfer Protocol)

HTTP (HyperText Transfer Protocol) è un protocollo che definisce come devono essere scritti i programmi client/server per accedere alle pagine Web. Un client HTTP invia una richiesta, un server HTTP restituisce una risposta. Il server utilizza la porta 80, il client utilizza un numero di porta effimero. HTTP sfrutta i servizi del TCP il quale, come visto in precedenza, è un protocollo affidabile e orientato alla connessione. Questo significa che, prima che possa avvenire una qualsiasi transazione, il client e il server devono stabilire una connessione. Al termine della transazione la connessione dovrebbe essere chiusa. Il client e il server non devono preoccuparsi degli eventuali errori nei messaggi scambiati o della perdita di dati, perché è il protocollo TCP che essendo affidabile si occupa di tutti questi aspetti.

HTTP è nato con un scopo generico, non solo quello di gestire l'ipertesto, ma volendo anche gestire sistemi basati sul concetto di oggetti o risorse distribuite. Ha un insieme di metodi di richiesta pensati espressamente per la condivisione e la collaborazione sugli ipertesti e tali metodi sono estendibili. È un protocollo *stateless* (senza memoria di stato): le coppie richiesta/risposte sono indipendenti, cioè ogni richiesta viene eseguita indipendentemente da quelle che l'hanno preceduta. Inoltre, permette la rappresentazione e la negoziazione dei formati dei dati.

Connessioni persistenti e non persistenti

Connessione: un circuito logico di livello trasporto stabilito tra due programmi applicativi per comunicare tra loro.

La natura di ipertesto delle pagine Web può richiedere una serie di richieste e risposte. Se le pagine Web, gli oggetti ai quali si intende accedere, sono localizzate su server differenti, non vi è altra soluzione che creare una

connessione TCP per ogni server e ottenere ciascun oggetto richiesto. Tuttavia, se più oggetti sono localizzati sullo stesso server, esistono due possibilità: recuperare ciascun oggetto usando una nuova connessione TCP o recuperare tutti gli oggetti con un'unica connessione. Il primo metodo viene indicato con l'espressione *connessioni non persistenti*, il secondo metodo con *connessioni persistenti*. HTTP nelle versioni precedenti alla 1.1 utilizzava connessioni non persistenti. Questo è cambiato in HTTP 1.1 dove le connessioni sono solitamente persistenti, ma l'utente può decidere diversamente.

- **Connessione non persistente:** viene stabilita una nuova connessione TCP per ciascuna coppia richiesta/risposta.
- **Connessione persistente:** in questo caso il server, dopo aver mandato una prima risposta, lascia la connessione aperta in attesa di ulteriori richieste. Il server può chiudere la connessione su richiesta del client o allo scadere di un timeout.

L'impiego di connessioni persistenti consente di risparmiare tempo e risorse. Infatti il tempo necessario per l'apertura e la chiusura di un gran numero di connessioni viene eliminato. Inoltre si ha una gestione più efficace del controllo di congestione: più tempo per regolare la frequenza di invio del traffico in base alla congestione di rete percepita.

Pipelining

Serve per migliorare ulteriormente le prestazioni. Consiste nell'invio da parte del 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 risposte. Se una richiesta richiede tempo per essere processata, le risposte alle richieste successive sono bloccate (*Head of line blocking*)
- Il client non può inviare in pipeline richieste che usano metodi HTTP non *idem-potenti*.

Server Web che rispettano HTTP 1.1 devono supportare il pipelining, tuttavia è implementato in pochi browser.

Formato dei messaggi

Il protocollo HTTP definisce il formato dei messaggi di richiesta e di risposta. Ogni messaggio è formato da quattro parti. La prima parte nel messaggio

di richiesta viene chiamata *riga di richiesta*, mentre nel messaggio di risposta viene chiamata *riga di stato*. Le altre tre parti hanno lo stesso nome in entrambi i tipi di messaggio, ma le similarità si limitano al nome, poiché possono avere contenuti diversi.

Messaggio di richiesta

Nella riga di richiesta vi sono tre campi separati da uno spazio e terminati da due caratteri (ritorno carrello e fine linea). I campi sono

- **metodo**: definisce i tipi di richiesta.
- **URL**: definisce l'indirizzo e il nome della pagina Web corrispondente.
- **versione**: indica la versione del protocollo

Dopodiché posso avere più *linee di intestazione* nelle quali ho il nome del campo e un valore, separati da uno spazio, seguiti da ritorno carrello e fine linea. Infine, preceduto da una linea vuota, vi è il *corpo dell'entità*

Header

Gli *header* sono insiemi di coppie (nome:valore) che specificano alcuni parametri del messaggio trasmesso o ricevuto:

- **General Header**: relativi alla trasmissione; per esempio data, codifica, connessione, ...
- **Entity Header**: relativi all'entità trasmessa; per esempio content-type, content-length, data di scadenza, ...
- **Request Header**: relativi alla richiesta; per esempio chi fa la richiesta, a chi viene fatta, che tipo di caratteristiche il client è in grado di accettare, autorizzazione, ...
- **Response Header**: nel messaggio di risposta; per esempio server, autorizzazione richiesta, ...

Messaggio di risposta

La prima riga di un messaggio di risposta è chiamata riga di stato. In questa riga vi sono tre campi separati da spazi e terminati da ritorno carello e fine linea. Il primo campo indica la versione del protocollo HTTP. Il campo di stato (*status code*) definisce lo stato della richiesta, ed è costituito da tre cifre. La frase di stato esplicita in forma testuale il significato del codice.

Status code

1xx: Informational	Request received, continuing process
2xx: Success	The action was successfully received, understood, and accepted
3xx: Redirection	Further action must be taken in order to complete the request
4xx: Client Error	The request contains bad syntax or cannot be fulfilled
5xx: Server Error	The server failed to fulfill an apparently valid request

Content negotiation

Le risorse possono essere disponibili in più rappresentazioni (lingua, formato di dati, dimensione, etc.).

Content negotiation: meccanismo per selezionare la rappresentazione appropriata quando viene servita una richiesta (uso di Request e Entity headers).

Metodi di HTTP

Metodo	Azione
GET	Richiede un documento al server
HEAD	Richiede le informazioni relative a un documento
PUT	Invia un documento dal client al server
POST	Invia informazioni dal client al server
TRACE	Restituisce una copia della richiesta inviata
DELETE	Elimina la pagina Web
OPTIONS	Richiede informazioni relative alle opzioni disponibili

Metodi sicuri e idem-potenti

- **Safe methods:** metodi che non hanno effetti “collaterali”, es. non modificano la risorsa
 - GET, HEAD, OPTIONS, TRACE
- **Idem-potent methods:** i metodi possono avere la proprietà di *idempotenza* negli effetti collaterali se $N > 0$ richieste identiche hanno lo stesso effetto di una richiesta singola
 - GET, HEAD, PUT, DELETE, OPTIONS, TRACE

Web caching: il server proxy

HTTP supporta i *server proxy*, ovvero dei computer che conservano una copia delle risposte alle richieste recenti. Il client HTTP invia una richiesta al server 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 server proxy riduce il carico sul server originale, spesso riduce il traffico in rete e la latenza. Naturalmente il client, per poter utilizzare il server proxy, deve essere configurato in modo da accedere al proxy anziché direttamente al server Web.

Si noti che il server proxy agisce sia da server sia da client. Quando riceve una richiesta da un client per il quale ha una risposta, agisce come server inviando la risposta al client. Quando riceve una richiesta da un client per il quale non ha risposta, agisce prima come client inviando la richiesta al server Web, successivamente, quando riceve la risposta, agisce come server e invia la risposta al client.

Cookie

Il Web è stato inizialmente progettato come ambiente “senza stato” (stateless): il server risponde alle richieste inviate dal client senza mantenere alcuna traccia del contesto, senza memorizzare alcuna informazione relativa alla sequenza di richieste inviate dal client, che sono quindi totalmente isolate. Questa modalità è perfettamente in linea con gli obiettivi iniziali del Web: poter ottenere facilmente documenti pubblicamente disponibili. Questo approccio oggi è decisamente inadeguato, visto che il Web ha anche altre funzioni:

- alcuni siti Web sono usati per il commercio elettronico in negozi virtuali;
- alcuni siti Web devono limitare l'accesso ai soli utenti registrati;
- altri siti Web sono utilizzati come portali;
- altri siti ancora sono utilizzati a scopo pubblicitario.

Creazione e memorizzazione dei cookie

I dettagli relativi alla creazione e alla memorizzazione dei cookie variano molto a seconda dell'implementazione, ma il meccanismo di massima è sempre lo stesso:

- il server quando riceve una richiesta memorizza le informazioni (relative al client) in una stringa o in un file. Tali informazioni possono comprendere il nome di dominio del client, i contenuti di un cookie precedente inviato dal client insieme alla richiesta (informazioni precedentemente raccolte dal server sul client quali nome, numero di registrazione, e così via), l'orario e altro;
- il server include il cookie nella risposta che invia al client;
- il browser che riceve la risposta memorizza il cookie in una directory adibita a questo scopo, a sua volta suddivisa per nome di dominio del server.

Utilizzo dei cookie

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. Si noti che i contenuti del cookie non sono mai mal interpretati dal browser o resi disponibili all’utente: il cookie è “preparato” e “consumato” dal server.

2.3.2 TELNET

Un programma server può fornire un servizio specifico al corrispondente programma client. È tuttavia improponibile avere una coppia client/server per ogni tipo di servizio, il numero di server crescerebbe a dismisura. In altre parole si tratta di un approccio non scalabile. È decisamente più ragionevole creare una coppia di applicativi client/server che permetta a un utente di accedere su un computer remoto e quindi di utilizzare tutti gli applicativi già presenti sul server.

TELNET, abbreviazione di TErminal NETwork, è uno dei primi protocolli di login remoto. 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 di accesso sono trasmesse in chiaro (senza alcuna forma di crittografia). Questo significa che un attaccante potrebbe semplicemente intercettare le credenziali ed ottenere l’accesso al sistema remoto. Per questo motivo, attualmente si usa prevalentemente un altro protocollo chiamato Secure Shell (SSH).

Login locale e remoto

L’azione con la quale un utente si collega a un computer locale è detta *login locale*. Se l’utente preme un tasto della tastiera connessa a un terminale del sistema, il carattere viene inviato al driver del terminale che a sua volta lo trasmette al sistema operativo del calcolatore; il sistema operativo, infine, interpreta la sequenza di caratteri ricevuti e invoca il programma applicativo richiesto all’utente.

Tuttavia, gli utenti che vogliono utilizzare programmi applicativi su dispositivi remoti devono effettuare un *login remoto*, per esempio per mezzo della coppia client/server TELNET. I caratteri introdotti dall’utente vengono accettati dal driver del terminale e inviati al client TELNET che trasforma la sequenza in una successione di caratteri con una codifica standard, che viene poi trasmessa allo stack TCP/IP locale.

I comandi o i messaggi di testo che viaggiano attraverso Internet e giungono allo stack TCP/IP del dispositivo remoto, dove vengono trasmessi al sistema operativo e da questo al server TELNET. Quest’ultimo, infine, li trasforma in una sequenza di caratteri utilizzabile dalla macchina remota. A questo punto sorge una piccola complicazione data dal fatto che nel server non è presente un driver di terminale. I caratteri provenienti dal client non possono essere recapitati direttamente al sistema operativo remoto perché questo è in grado di accettare soltanto caratteri provenienti da un driver di un terminale, che è assente (nel sistema remoto non è presente alcun terminale). Per questa ragione viene aggiunto un ulteriore software, detto driver dello pseudoterminale, con il compito di accettare i caratteri dal server e di trasmetterli al sistema operativo. Dopo questo passaggio intermedio il sistema operativo consegna infine i caratteri all’applicazione opportuna.

TELNET: NVT

Problema: TELNET deve poter operare con il numero massimo di sistemi e quindi gestire dettagli di sistemi operativi eterogenei. I terminali possono differire gli uni dagli altri per:

- il set e codifica di caratteri;
- la larghezza della linea e lunghezza della pagina;
- i tasti funzione individuati da diverse sequenze di caratteri (escape sequence), per esempio una diversa combinazione di tasti per interrompere un processo, caratteri ASCII diversi per terminare le righe di testo.

Soluzione: definizione di un terminale virtuale. Sulla rete si considera un unico terminale standard e in corrispondenza di ogni host, si effettuano la conversione da terminale locale a terminale virtuale e viceversa.

TELNET assume che sui due host sia in esecuzione un ***Network Virtual Terminal*** (NVT). La connessione TCP è instaurata tra questi due terminali NVT.

L’NVT è un dispositivo “immaginario” che fornisce una rappresentazione astratta di un terminale canonico. Gli host, sia client che server, traducono le loro caratteristiche locali così da apparire esternamente come un NVT e assumono che l’host remoto sia un NVT.

Il *Network Virtual Terminal* definisce un set di caratteri e di comandi universale che permette di trasformare il set di caratteri in uso localmente in un set di caratteri universali. Lo standard prevede che i terminali NVT si scambino i dati in formato ASCII a 7-bit, però la comunicazione è a 8-bit: in questo modo si differenziano i dati dalle sequenze di comandi veri e propri. Inoltre, i comandi sono anticipati da un ottetto speciale con tutti i bit settati a 1 chiamato *Interpret as Command* (IAC). Tale meccanismo si chiama *inband signalling*: i comandi e i dati sono mandati sulla stessa connessione.

2.3.3 Posta elettronica

La posta elettronica (o e-mail) consente agli utenti di scambiarsi messaggi. La natura di questa applicazione è tuttavia differente da altre applicazioni tratte fino a questo momento. In un’applicazione come HTTP, il programma server è sempre in esecuzione, in attesa di una richiesta dal client. All’arrivo della richiesta, il server provvede a fornire il servizio richiesto. Vi sono una richiesta e un’risposta. Nel caso della posta elettronica la situazione è differente: il destinatario di un messaggio potrebbe non essere disponibile ad accettare messaggi di posta in quel momento (utente impegnato o computer spento); quindi lo scambio del messaggio deve prevedere questo disaccoppiamento tra mittente e destinatario, un po’ come succede nella posta tradizionale.

Architettura

Per spiegare l’architettura della posta elettronica si prende come riferimento lo scenario illustrato nella Figura 2.1.

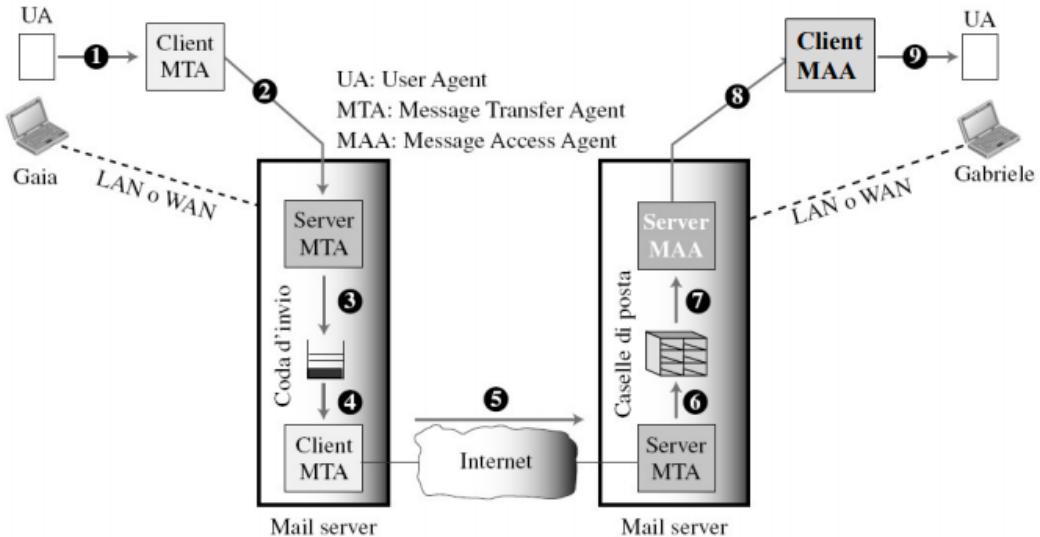


Figura 2.1: Scenario classico

Abitualmente il mittente e il destinatario della posta elettronica, Gaia e Gabriele, sono collegati via LAN o WAN a due server di posta. L'amministratore ha creato una *mailbox* (casella di posta) per ogni utente dove memorizzare i messaggi ricevuti. Una mailbox non è altro che una porzione di spazio sulla memoria di massa del server, sottoforma di un file (o di una directory) con opportune restrizioni per far sì che solamente il proprietario possa accedere. Inoltre, l'amministratore ha creato anche una coda (spesso chiamata spool) dove memorizzare i messaggi in attesa di essere inviati. Gaia e Gabriele utilizzano tre *agenti* differenti: un User Agent (UA), un Mail Transfer Agent (MTA) e un Message Access Agent (MAA). Gaia, per inviare un messaggio a Gabriele, utilizza un programma UA per preparare il messaggio e inviarlo al proprio server di posta. Quest'ultimo memorizza il messaggio in attesa di essere inviato all'interno di un'apposita coda. Il messaggio, tuttavia, deve essere inviato tramite Internet dal mail server di Gaia al mail server di Gabriele usando un MTA. Sono quindi necessari due MTA in ciascun mail server: uno client e uno server. Analogamente alla maggior parte dei programmi client/server presenti in rete, il server deve essere in esecuzione permanente perché non può sapere quando il client richiederà una connessione. Il client, invece, può essere eseguito dal sistema quando in coda c'è un messaggio in attesa di essere spedito. L'user agent presso il sito di Gabriele gli consente di ricevere e leggere messaggi. Infine Gabriele utilizza un client MAA per ottenere i messaggi da un server MAA in esecuzione sul secondo server.

Ci sono due aspetti importanti da considerare. Primo Gabriele è obbligato a passare per il server di posta e non può utilizzare il server MTA direttamente. Per farlo, dovrebbe tenere il proprio computer costantemente acceso; nel caso in cui vi fosse connesso tramite WAN, dovrebbe mantenere la connessione attiva in permanenza: entrambe le situazioni sono improponibili.

Secondo, si noti che Gabriele richiede un’ulteriore coppia di programmi client-server: i programmi per accedere ai messaggi di posta. Il programma client-server MTA, infatti, è un’applicazione di tipo *push*: il cliente “spinge” il messaggio al server. Gabriele invece richiede un programma di tipo *pull*: il client deve “tirare” il messaggio dal server.

Server postale (schema di principio)

I mail server adottano una tecnica denominata *spooling*:

- L’utente invia un messaggio, il sistema ne pone una copia in memoria insieme all’id mittente, l’id destinatario, l’id della macchina di destinazione e il tempo di deposito.
- Il sistema avvia il trasferimento alla macchina remota. Il sistema (client) stabilisce una connessione TCP con la macchina 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.

Indirizzi

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 in due parti: una parte locale e un nome di dominio (domain name), separati dal simbolo @.

La parte locale dell’indirizzo indica il nome di una mailbox in cui vengono memorizzati tutti i messaggi ricevuti dall’utente, in attesa che vengano prelevati dal client MAA. La seconda parte contiene il nome del dominio.

Un'organizzazione sceglie uno o più host (a volte chiamati server di posta o mail exchanger) per la ricezione e l'invio della posta elettronica.

User Agent

Il suo scopo è quello di semplificare all'utente il processo di invio e ricezione dei messaggi. Un user agent è un programma che consente di scrivere, leggere, rispondere e inoltrare i messaggi, ed è in grado inoltre di gestire le mailbox nei computer degli utenti.

Message Transfer Agent: SMTP

Si può affermare che la posta elettronica è una di quelle applicazioni che richiedono tre impieghi del paradigma client/server: la prima e la seconda sono MTA (Message Transfer Agent), la terza è un MAA (Message Access Agent).

Il protocollo che definisce in modo formale l'interazione tra il client e il server MTA è chiamato SMTP (*Simple Mail Transfer Protocol*). SMTP è utilizzato in due diverse occasioni: fra mittente ed il suo server di posta e tra i due server di posta. L'obiettivo di SMTP è il **trasferimento affidabile e efficiente** di mail. È indipendente dal sistema di trasmissione usato e richiede solo il trasferimento di stream di byte ordinato e affidabile (l'RFC discute SMTP su TCP).

Una caratteristica di SMTP è la capacità di trasportare mail attraverso più reti. Un messaggio di mail può passare attraverso server intermedi nel percorso da mittente a destinatario finale.

Quando un client SMTP vuole trasferire un messaggio, stabilisce un **cana-
le 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**). Un client SMTP determina l'indirizzo di un host appropriato che ospita un server SMTP risolvendo il nome della destinazione in un mail server destinazione (risoluzione di un nome in indirizzo IP attraverso il DNS).

Possibili problemi che possono emergere nel trasferimento mail da un client a un server tramite SMTP:

- connessione con mailserver del mittente (server inesistente o irraggiungibile)
- connessione con mailserver destinatario (server inesistente o irraggiungibile)
- inserimento in maibox destinatario (user unknown, mailbox full)

ma 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, filtro antispam) rimuove il messaggio.

Quello che fa SMTP è semplicemente definire come deve avvenire l'interazione per mezzo di comandi e risposte: i comandi sono inviati da un client MTA a un server MTA, viceversa le risposte. I comandi terminano tutti con la medesima coppia di caratteri (ritorno carrello e fine linea).

Comandi Il loro formato è composto da una keyword (parola chiave) e uno o più argomenti:

Keyword: argomento (o argomenti)

<i>Keyword</i>	<i>Argomenti</i>	<i>Descrizione</i>
HELO	Nome dell'host mittente	L'host mittente si identifica
MAIL FROM	Mittente del messaggio	Identifica il mittente del messaggio
RCPT TO	Destinatario	Identifica il destinatario del messaggio
DATA	Corpo del messaggio	Invia il contenuto del messaggio
QUIT		Termina la sessione SMTP
RSET		Interrompe la transazione in atto
VRFY	Nome del destinatario	Verifica la validità dell'indirizzo del destinatario

Risposte Costituite da un codice a tre cifre seguito eventualmente da un testo, vengono inviate dal server al client.

<i>Codice</i>	<i>Descrizione</i>
Risposte con esito positivo	
220	Servizio pronto
221	Il servizio è in procinto di chiudere il canale di trasmissione
250	Comando richiesto completato
251	Utente non locale al mail server, il messaggio verrà inoltrato
Risposte intermedie positive	
354	Puoi cominciare l'invio del messaggio
Risposte transitorie con esito negativo	
421	Servizio non disponibile
450	Mailbox non disponibile
451	Comando interrotto; errore locale
452	Comando interrotto; memoria di massa insufficiente
Risposte con esito negativo permanente	
500	Errore di sintassi; comando non riconosciuto
501	Errore di sintassi nei parametri o argomenti
502	Comando non disponibile
503	Sequenza di comandi errata
550	Comando non eseguito; mailbox non disponibile
551	Utente non locale
552	Azione richiesta interrotta; spazio insufficiente
554	Transazione fallita

Fasi della consegna La consegna di un messaggio avviene in tre fasi distinte: apertura della connessione, trasferimento del messaggio di posta e chiusura della connessione. L'uso del termine “connessione” per il protocollo SMTP è parzialmente ambiguo visto che si tratta dello stesso termine utilizzato per TCP (quindi a livello trasporto). È bene notare come la connessione SMTP e quella TCP, entrambe necessarie, avvengano a livelli diversi della pila di protocolli.

Apertura della connessione. Il server SMTP dà il via alla fase di apertura della connessione non appena il client effettua la connessione TCP con la porta nota 25. Questa fase può essere suddivisa in tre passi successivi.

1. Il server invia il codice 220 (servizio pronto) per indicare al client che è pronto alla ricezione di messaggi, oppure il codice 421 (servizio non disponibile) in caso contrario.
2. Il client si identifica per mezzo del comando HELO seguito dal suo nome di dominio; questo passo è necessario per informare il server del nome di dominio del client.

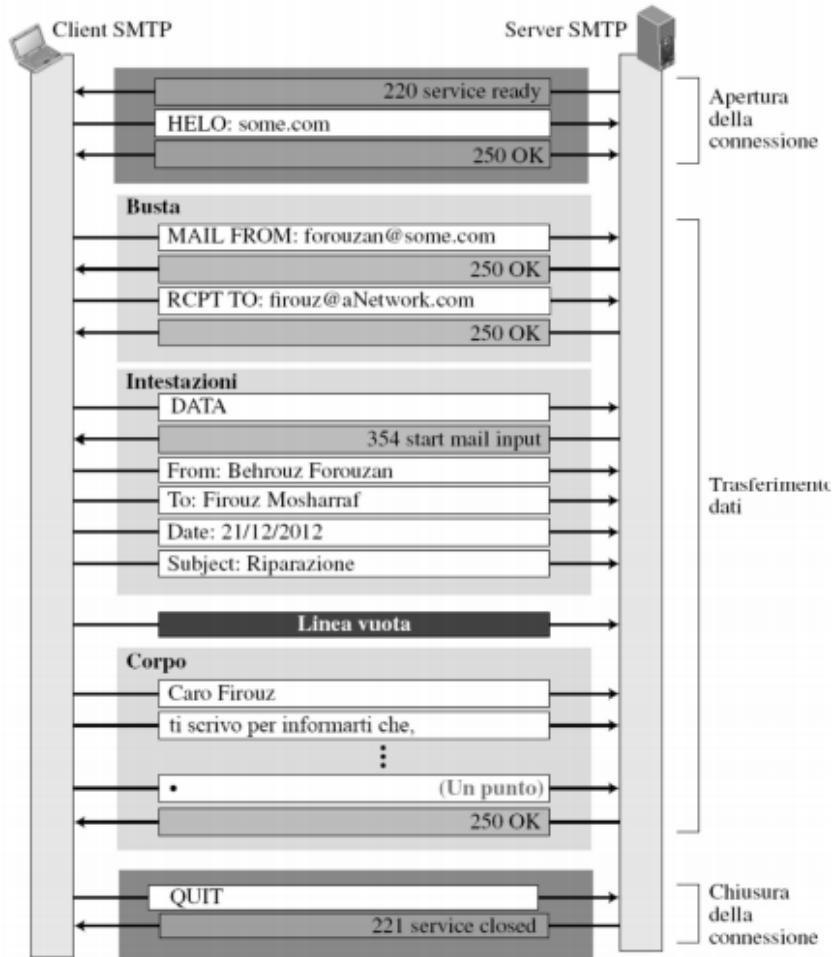
3. Il server invia il codice 250 (comando richiesto completato) o altri codici a seconda della situazione particolare.

Trasferimento del messaggio. Dopo l'apertura della connessione tra il client e il server SMTP, un singolo messaggio dal mittente a uno o più destinatari può essere inviato. Le operazioni svolte in questa fase possono essere raggruppate in otto passi successivi; se vi sono più destinatari i passi tre e quattro vengono ripetuti.

1. Il client invia il comando MAIL FROM per indicare il mittente del messaggio; come argomento viene usato l'indirizzo e-mail del mittente (mailbox e nome del dominio). Questo passo è necessario per fornire al server l'indirizzo del mittente cui inviare eventuali messaggi d'errore.
2. Il server risponde con il codice 250 o con un altro codice appropriato alla situazione particolare.
3. Il client, per mezzo del comando RCPT TO (abbreviazione di recipient), invia l'indirizzo e-mail destinatario.
4. Il server risponde con il codice 250 o con un altro codice appropriato alla situazione particolare.
5. Il client invia il comando DATA per iniziare il trasferimento del messaggio.
6. Il server risponde con il codice 354 (inizia l'invio del messaggio) o con un codice appropriato alla situazione.
7. Il client invia il contenuto del messaggio come sequenza di righe, ciascuna delle quali termina con la coppia ritorno carrello e fine linea. Il messaggio termina con una riga contenente solo un punto.
8. Il server risponde con il codice 250 (OK) o con un altro codice appropriato alla situazione particolare.

Chiusura della connessione. Il client, una volta trasferito il messaggio, chiude la connessione. Questa fase comporta due passaggi successivi:

1. Il client invia il comando QUIT.
2. Il server risponde con il codice 221 o con un altro codice appropriato alla situazione.



Fasi della consegna

Formato dei messaggi mail Il messaggio è suddiviso in intestazioni (header) e corpo (body): le intestazioni contengono l'indirizzo del mittente, quello del destinatario, l'argomento del messaggio e alcune altre informazioni, mentre il corpo contiene il messaggio vero e proprio.

Multipurpose Internet Mail Extensions (MIME)

La posta elettronica ha una struttura molto elementare, ma la sua semplicità comporta che il protocollo possa gestire soltanto messaggi nel formato NVT ASCII a 7 bit. Vi sono quindi delle limitazioni; per esempio non possono essere inviati messaggi in lingue che non siano supportate dai caratteri ASCII a 7 bit come per esempio il francese, il tedesco, l'ebraico, il russo, il cinese o il giapponese e non possono essere inviati file binari contenenti audio o video.

Il protocollo MIME (Multipurpose Internet Mail Extensions) è un protocollo supplementare che permette l'invio dei messaggi in formati diversi dall'ASCII. Esso agisce lato mittente convertendo tutti i dati in formato non ASCII al formato NVT ASCII e consegnando il risultato al client MTA per l'invio. Il messaggio viene ritrasformato al formato originale presso il destinatario.

In conclusione si può pensare al protocollo MIME come a un insieme di funzioni che si occupano di tradurre dati non ASCII in formato NVT ASCII e viceversa.

Intestazione MIME Il protocollo MIME definisce cinque tipi di intestazioni specifiche che si aggiungono a quelle originali previste dal protocollo di posta elettronica.

Intestazioni della e-mail
MIME-Version: 1.1
Content-Type: tipo/sottotipo
Content-ID: identificatore del messaggio
Content-Description: spiegazione testuale del contenuto non testuale
Corpo della e-mail

Intestazioni MIME

MIME-version. In questa intestazione viene dichiarata la versione MIME usata. La versione attuale è la 1.1.

Content-Type. Intestazione che definisce il tipo di dati nel corpo del messaggio; il tipo del contenuto e il sottotipo sono separati da una barra. A seconda del sottotipo l'intestazione può contenere altri parametri.

Content-Transfer-Encoding. Definisce il metodo utilizzato per la codifica del messaggio in binario per il trasporto.

Content-ID. Intestazione che individua univocamente una parte del messaggio in messaggi composti da multiple parti.

Content-Description. Intestazione che indica se il corpo del messaggio contiene un'immagine, un file audio o un video.

Message Access Agent: i 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 essendo un protocollo di tipo push: esso “spinge” il messaggio dal client al server. In altri termini, la direzione della maggior parte dei dati (messaggi) è 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. Il terzo stadio utilizza dunque un *Message Access Agent*.

Attualmente si utilizzano due protocolli per l’accesso ai messaggi: POP3 (Post Office Protocol, versione 3) e IMAP4 (Internet Mail Access Protocol, versione 4).

POP3 Il protocollo Post Office Protocol versione 3 è molto semplice ma ha funzionalità limitate. Il software client POP3 è installato sul computer del destinatario, mentre il software POP3 server viene installato sul server di posta. L’accesso alla porta avviene quando il client scarica i messaggi (le e-mail) dalla propria casella di posta che si trova sul server. Il client apre una connessione TCP verso il server sulla porta nota 110, quindi invia il proprio nome utente e relativa password per accedere alla casella di posta. A questo punto l’utente può richiedere la lista dei messaggi presenti e prelevarli, uno alla volta.

Il protocollo POP3 ha due modalità: delete (elimina) e keep (conserva). Nella prima i messaggi sono automaticamente eliminati dalla mailbox dopo il prelievo; nella seconda, invece, vengono conservati.

IMAP4 Un altro protocollo di accesso alla posta è l’Internet Mail Access Protocol versione 4 (IMAP4). Questo protocollo è simile al POP3 ma ha più funzionalità: è più potente e complesso.

POP3 è limitato sotto vari aspetti. Non consente all’utente di organizzare la propria posta e di gestire più cartelle sul server. POP3, inoltre, non consente all’utente di controllare parte del contenuto del messaggio prima del suo prelievo completo. IMAP4, invece, 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. Una funzionalità particolarmente utile quando vi sono limitazioni di banda e i messaggi comprendono contenuti multimediali particolarmente voluminosi;
- 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.

2.3.4 Il DNS (Domain Name System)

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. Si tratta di una situazione analoga alla rete telefonica, che è progettata per utilizzare dei numeri di telefono, non dei nomi.

Vista la dimensione di Internet, un sistema centralizzato non potrebbe gestire tutte le associazioni necessarie. Inoltre se tale sistema centralizzato si dovesse guastare, colllasserebbe l'intera rete. La soluzione attualmente adottata consiste nel suddividere questa enorme massa di informazioni e distribuire le varie parti ottenute su calcolatori sparsi nel mondo. L'host che ha bisogno di associare un indirizzo a un nome, o viceversa, contatta il calcolatore più vicino e gli invia una richiesta opportuna. Questo metodo viene usato dal Domain Name System (DNS).

Un utente vuole utilizzare un client per il trasferimento di file per accedere al file server corrispondente, in esecuzione su un host remoto. L'utente conosce solo il nome simbolico del file server, per esempio *afilesource.com*. Lo stack TCP/IP ha bisogno invece dell'indirizzo IP del file server per stabilire una connessione.

I sei passi seguenti consentono di associare l'indirizzo IP al nome simbolico dell'host:

1. l'utente comunica il nome dell'host al client di trasferimento file;
2. il client di trasferimento file trasmette il nome dell'host al client DNS;
3. qualsiasi computer, una volta avviato, conosce l'indirizzo IP di un server DNS. Il client DNS invia dunque un messaggio al server DNS contenente la richiesta di traduzione del nome simbolico del file server.
4. il server DNS risponde con l'indirizzo IP del file server desiderato;
5. il client DNS comunica l'indirizzo IP al client di trasferimento file;
6. il client di trasferimento file utilizza ora l'indirizzo IP ricevuto per accedere al file server.

Si noti che anche se lo scopo è quello di scambiare un file, prima che questo sia possibile è necessario sfruttare un altro servizio presente su Internet, attraverso l'interazione tra un client e un server DNS.

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 associare in modo univoco un nome a un indirizzo, in modo tale che anche il nome individui senza ambiguità un calcolatore. Vi sono due modi possibili per organizzare tale spazio: spazio con struttura piatta e spazio con struttura gerarchica. In uno *spazio dei nomi con struttura piatta* viene associato un nome differente a ciascun indirizzo; ogni nome è quindi una sequenza di caratteri senza alcuna struttura interna. Due nomi possono avere una sottosequenza di caratteri in comune, ma il fatto che l'abbiano o meno non ha alcuna ulteriore implicazione o significato. Il problema fondamentale di questo tipo di spazio, in una rete grande come Internet, è la necessità che esso sia controllato a livello centrale per evitare ambiguità e duplicazioni. In uno *spazio dei nomi gerarchico* ogni nome è composto da diverse parti: la prima può definire la natura dell'organizzazione, la seconda il suo nome, la terza i vari dipartimenti all'interno dell'organizzazione e così via. Con uno spazio di questo tipo gli enti o società che si occupano di assegnare e controllare i nomi possono avere una struttura decentralizzata: l'ente centrale si occuperà della parte di nome che corrisponde alla natura dell'organizzazione e alla sua denominazione, per la parte restante del nome la responsabilità di gestione può essere demandata all'organizzazione stessa. Questa può aggiungere prefissi o suffissi al nome per individuare i suoi host, o più in generale le sue risorse, senza preoccuparsi di eventuali coincidenze con i prefissi scelti da altre organizzazioni. Anche se due diverse 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.

Spazio dei nomi di dominio Lo spazio dei nomi di dominio (domain name space) è alla base della costruzione dello spazio dei nomi con struttura gerarchica. In questo schema i nomi hanno una struttura ad albero con la radice in cima e un numero di livelli variabile tra 0 (la radice) e 127.

Etichette Ogni nodo è individuato da un'etichetta (label) costituita al più da 63 caratteri. Alla radice (root) è associata un'etichetta vuota, a tutti i nodi collegati a un medesimo nodo da rami diversi sono associate etichette diverse, garantendo così l'univocità dei nomi di dominio.

Nomi di dominio Ogni nodo dell'albero ha un nome di dominio (domain name), ovvero una sequenza di etichette separate da punti (.), che lette da

sinistra a destra forniscono tutte le etichette associate ai vari nodi a partire da quello in questione fino alla radice. L'ultima etichetta è quella della radice, la stringa nulla, quindi tutti i nomi di dominio terminano con un punto. Tutte le etichette che si trovano immediatamente sotto la radice vengono definite *top-level* (di livello più alto) e quindi si parlerà di domini top-level. Una sequenza di etichette che termina con una stringa nulla è detta *nome di dominio pienamente qualificato* o *FQDN* (Fully Qualified Domain Name). Si osservi che il nome termina con una stringa nulla e quindi l'ultimo carattere risulta essere un punto (.). Una sequenza di etichette che non termina con una stringa nulla è detta *nome di dominio parzialmente qualificato* o *PQDN* (Partially Qualified Domain Name). Un PQDN, quindi, inizia da un nodo ma non raggiunge la radice; esso viene usato quando il nome da risolvere e il client appartengono allo stesso dominio di livello superiore. In questo caso, infatti, il protocollo atto alla risoluzione del nome può aggiungere automaticamente la parte mancante, detta suffisso, al fine di ottenere un FDQN.

Domini Un dominio (domain) è un sottoalbero dello spazio dei nomi di dominio che viene identificato dal nome di dominio del nodo in cima al sottoalbero; si osservi che un dominio può essere suddiviso in ulteriori domini, detti talvolta sottodomini.

Informazioni degli spazi di dominio Le informazioni contenute nello spazio dei nomi di dominio devono essere memorizzate in un dispositivo fisico. Salvare questa enorme mole di dati su un solo computer sarebbe allo stesso tempo inaffidabile, in quanto il computer potrebbe guastarsi, e inefficiente, poiché per ottenere informazioni relative ai nomi di dominio tutti i calcolatori della rete mondiale dovrebbero rivolgersi a una stessa macchina. È quindi necessario trovare una soluzione alternativa.

Gerarchie dei Name Server

Il problema appena posto è stato risolto ripartendo le informazioni relative ai nomi di dominio tra diversi calcolatori detti *server DNS* o anche *name server*. L'intero spazio è stato suddiviso in diversi domini differenziati al primo livello dell'albero; in altri termini la radice resta comune e a ogni nodo di primo livello è stato associato un dominio. Poiché i domini così ottenuti continuano a essere molto vasti, essi sono stati suddivisi in domini più piccoli, detti *sottodomini*. Ogni server è responsabile di un dominio o di un sottodominio: vi è quindi una gerarchia di server che implementa la gerarchia di nomi.

Zone Poiché non è possibile memorizzare la gerarchia completa dei nomi di dominio su un singolo server, questa viene suddivisa fra numerosi server. Una *zona* è tutto ciò di cui è responsabile un server. Si può definire una zona come una parte contigua dell'intero albero. Se un server accetta tutte le responsabilità per un dominio e non effettua suddivisioni in sottodomini, allora la zona e il dominio coincidono. Ogni server ha un database, detto *file di zona*, in cui vengono elencate tutte le informazioni relative ai nodi presenti nella sua zona di competenza. Se, invece, il server suddivide il proprio dominio in sottodomini, allora la zona e il dominio si differenziano. In questo caso il server ha delegato una parte della sua autorità ad uno o più altri server.

Le informazioni relative ai nodi nei sottodomini sono immagazzinate nei server di livello più basso e il server originario si limita a gestire i riferimenti a questi server di livello inferiore. Ovviamente il server originario non risulta totalmente privo di responsabilità: a esso è ancora associata una zona, ma le informazioni dettagliate sui nodi di tale zona sono immagazzinate nei server di livello più basso.

Server radice Un server radice (*root server*) è un server che ha per zona l'intero albero. Solitamente i root server delegano la loro responsabilità ad altri server e si limitano a immagazzinare riferimenti relativi a questi server. Attualmente vi sono vari server radice distribuiti sulla rete Internet, ciascuno dei quali gestisce l'intero spazio dei nomi di dominio. I root server sono fisicamente distribuiti in tutto il mondo.

Server primari e secondari

I server DNS possono essere primari o secondari. Un *server primario* possiede un file relativo alla zona sotto la sua responsabilità (e autorità): la creazione, la gestione e l'aggiornamento del file di zona sono di sua competenza. Il file di zona è memorizzato nel suo disco locale.

Un *server secondario* riceve tutte le informazioni relative a una certa zona da un altro server, primario o secondario, e le memorizza in un file nel suo disco. I server secondari non creano né aggiornano i file di zona; sono i server primari che effettuano queste operazioni e poi trasmettono il nuovo file ai server secondari.

I server primari e secondari hanno entrambi la medesima autorità sulla zona di loro competenza (spesso si dice che sono “autorevoli” o “autoritativi”), in altri termini sono allo stesso livello di autorità; ma l'introduzione di un server secondario porta a una duplicazione, che può risultare utile in caso di guasti al server primario. Si osservi anche che un certo server può fungere

da primario per una certa zona e da secondario per un'altra; gli aggettivi primario e secondario, quindi, vanno riferiti alla zona a cui ci si riferisce.

Server DNS nella rete Internet

Il protocollo DNS può essere utilizzato su diverse piattaforme; nel caso di Internet lo spazio dei nomi di dominio (l'albero) è stato originariamente diviso in tre parti: *domini generici*, *nazionali* e *inversi*. Tuttavia, a causa della rapidissima espansione di Internet, è divenuto troppo complesso tenere traccia dei domini inversi, che potevano essere utilizzati per trovare il nome dell'host corrispondente partendo da un indirizzo IP. I domini inversi sono oggi obsoleti.

Domini generici I **domini generici** suddividono gli host in base al loro scopo generale; ogni nodo dell'albero definisce un dominio che non è altro che un indice nel database dello spazio dei nomi di dominio.

Domini nazionali Il formato dei **domini nazionali** prevede un'etichetta di primo livello costituita da abbreviazioni formate da due caratteri e che individuano una nazione. Il significato della seconda etichetta varia da nazione a nazione: negli Stati Uniti serve a individuare lo Stato dell'Unione.

Risoluzione

Il processo che permette di associare un indirizzo IP a un nome è detto **risoluzione** di un nome (address resolution). Il DNS è progettato come applicazione client/server. Un host che voglia associare un nome a un indirizzo IP (o viceversa, quando è possibile) si rivolge a un programma client DNS che è detto **resolver** (risolutore). Il resolver invia un'opportuna richiesta al server DNS più vicino; questi, se ha l'informazione, comunica l'indirizzo o il nome cercato al resolver, altrimenti si rivolge a un altro server o comunica al resolver l'indirizzo di un altro server a cui fare riferimento. Ricevuta la risposta, il resolver l'analizza per accertarsi che non contenga errori e trasmette il risultato al processo che aveva richiesto la risoluzione. La modalità di svolgimento della **risoluzione** può essere **ricorsiva** o **iterativa**.

Risoluzione ricorsiva Supponiamo che un programma applicativo eseguito su un host di nome *pads.cs.unibo.it* voglia trovare l'indirizzo IP di un altro host chiamato *engineering.mcgraw-hill.com* a cui inviare un messaggio. L'host sorgente è connesso alla rete dell'Università di Bologna, mentre l'host destinatario è connesso alla rete McGraw-Hill.

Il programma applicativo sull'host sorgente chiede al resolver (il client) DNS l'indirizzo IP dell'host destinatario. Il resolver, che non conosce tale indirizzo, invia una query al server DNS locale fornito dall'università. Ipotizziamo che anche questo server non conosca l'indirizzo IP dell'host destinatario. Invia quindi la query a un server DNS radice, il cui indirizzo IP si suppone noto a questo server DNS locale. I server radice solitamente non mantengono le associazioni fra nomi e indirizzi IP, ma devono conoscere almeno un server per ogni dominio top-level. La query viene quindi inviata a questo server di dominio top-level. Ipotizziamo che questo server non conosca l'associazione nome/indirizzo IP di questo particolare destinatario, ma conosca l'indirizzo IP del server DNS locale dell'azienda McGraw-Hill. La query viene così inviata a questo nuovo server che conosce l'indirizzo IP dell'host destinatario. L'indirizzo IP viene ora inviato al server DNS top-level, quindi al server radice, poi al server DNS dell'università che potrebbe memorizzarlo nella propria cache per eventuali richieste future, ed infine viene restituito all'host sorgente.

Risoluzione iterativa Nel caso della risoluzione 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. Solitamente la risoluzione iterativa ha luogo fra due server locali, il resolver originale riceve la risposta finale dal server locale.

Caching Un server, ogni volta che riceve una richiesta di risoluzione per un nome che non si trova all'interno del suo dominio, deve cercare nel proprio database l'indirizzo di un altro server al quale eventualmente inoltrare la richiesta. Ridurre questo tempo di ricerca significa aumentare l'efficienza. Questo avviene per mezzo di un meccanismo detto *caching*. Quando un server si rivolge a un secondo server per la risoluzione di un indirizzo archivia la sua risposta in una memoria temporanea, o memoria cache, al fine di poter usare questa informazione per successive richieste di risoluzione. Quando un server riceve una richiesta e la risposta è già presente in memoria cache, questa viene inviata al client segnalando però che si tratta di una risposta non autorevole (*unauthoritative*).

La tecnica del caching permette di velocizzare le procedure di risoluzione, ma può anche generare problemi; infatti se un server conservasse troppo a lungo un'informazione nella sua memoria cache, correrebbe il rischio di inviare ai client informazioni obsolete. Per evitare questi problemi, i server autorevoli aggiungono alle informazioni da loro trasmesse un campo detto *tempo residuo* (time to live, TTL), che stabilisce per quanti secondi il server

ricevente può conservare l'informazione nella sua memoria cache. Trascorso questo tempo la corrispondenza non è più valida, le informazioni vengono eliminate dalla memoria cache ed eventuali nuove interrogazioni verranno trasmesse al server autorevole. Il DNS richiede che ogni server mantenga un contatore TTL per ciascuna associazione memorizzata. La memoria cache deve essere poi controllata periodicamente per eliminare le associazioni con TTL scaduto.

Record di risorsa

A ogni nome di dominio (un nodo dell'albero dello spazio dei nomi) è associato un record di risorsa. Il database del server dei nomi non è altro che una collezione di record risorsa che vengono inviati al client che ne fa richiesta. Un record risorsa è formato da cinque campi:

(Nome di dominio, Tipo, Classe, TTL, Valore)

Il campo *nome di dominio* identifica il record risorsa. Il campo *valore* contiene l'informazione memorizzata relativa al nome di dominio. Il campo *TTL* indica il numero di secondi per cui l'informazione deve essere ritenuta valida. Il campo *classe* definisce il tipo di rete: in questo contesto si è esclusivamente interessati alla classe IN (Internet). Il *tipo* definisce infine come interpretare il campo valore.

<i>Tipo</i>	<i>Interpretazione del campo valore</i>
A	Indirizzo IPv4 a 32 bit
NS	Identifica i server autoritativi in una zona
CNAME	Indica che un nome di dominio è un alias per il nome di dominio ufficiale
SOA	Specifica una serie di informazioni autoritative riguardo una zona
MX	Indica il server di posta del dominio
AAAA	Indirizzo IPv6

Tipi di record

Messaggi DNS

I messaggi DNS sono di due tipi, *interrogazione (query)* e *risposta (response)*, e hanno lo stesso formato.

Il campo identificatore è utilizzato dal client per associare la risposta all'interrogazione. Il campo flag 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. La sezione richiesta, che è inclusa nell'interrogazione ed è ripetuta nel messaggio di risposta, consiste di uno o più record di richiesta. La sezione risposta, presente esclusivamente nei messaggi di risposta, consiste di uno o più record di risorsa. La sezione autorevole fornisce informazioni (nome di dominio) di uno o più server autorevoli per l'interrogazione. La sezione supplementare fornisce informazioni addizionali che potrebbero essere utili al risolutore.

Identificazione	Flag
Numero dei record di richiesta	Numero dei record di risposta (tutti 0 nei messaggi di interrogazione)
Numero dei record di autoritativi (tutti 0 nei messaggi di interrogazione)	Numero dei record supplementari (tutti 0 nei messaggi di interrogazione)
Sezione richiesta	
Sezione risposta (record risorsa)	
Sezione autoritativa	
Sezione supplementare	

Struttura dei messaggi DNS

Protocollo di livello trasporto

Il sistema DNS può usare sia il protocollo UDP che quello TCP; in entrambi i casi il server usa il numero di porta noto 53. Il protocollo UDP viene usato quando la dimensione del messaggio di risposta è inferiore a 512 byte, perché molto spesso i datagrammi utente UDP non possono superare i 512 byte di dimensione massima. In caso contrario viene usato il protocollo TCP.

2.3.5 FTP

FTP (File Transfer Protocol) è il protocollo standard offerto da TCP/IP per la copia di file da un host a un altro.

- Servizio diverso dall'accesso condiviso on-line (accesso simultaneo da parte di più programmi ad un singolo file)
- Trasferimento file: si ottiene una copia locale (si effettuano modifiche in locale) ed eventuale trasferimento del file modificato in remoto.

Sebbene il trasferimento di un file possa sembrare un'operazione semplice, vi sono, in realtà, numerosi problemi ai quali bisogna prestare attenzione: i due sistemi coinvolti, per esempio, possono adottare convenzioni differenti per la denominazione dei file, oppure gestire in modo differente la struttura delle directory. Problemi di questo tipo vengono risolti in modo semplice ed elegante dal protocollo FTP.

FTP fornisce funzionalità aggiuntive rispetto al semplice trasferimento file:

- accesso interattivo: l'utente può navigare e cambiare/modificare l'albero di directory nel file system remoto;
- specifica del formato dei dati da trasferire (es. file di testo o file binari);
- autenticazione: il client può specificare login e password.

Sebbene sia possibile trasferire file con HTTP, FTP rimane la scelta migliore per trasferire file voluminosi.

Il client ha tre componenti: interfaccia utente, processo di controllo e processo di trasferimento dati. Il server ha, invece, due sole componenti: processo di controllo e processo di trasferimento dati. La connessione di controllo viene realizzata tra i due processi di controllo e quella per lo scambio dei dati tra i due processi relativi.

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 separazione del trasferimento dei dati e dei comandi rende il controllo FTP piuttosto efficiente: la connessione che si occupa delle informazioni di controllo può usare regole molto semplici, così che lo scambio di informazioni si riduce allo scambio di una riga di comando (o di risposta) per ogni interazione. La connessione che si occupa dello scambio di dati, invece, deve fare uso di regole più complesse a causa della varietà delle informazioni scambiate.

Inoltre, FTP è un protocollo *stateful*: il server deve tenere traccia dello stato dell'utente (connessione di controllo associata ad un account, directory attuale, ecc.).

Durata delle due connessioni

Le due connessioni in FTP hanno durate differenti. La connessione di controllo rimane aperta durante l'intera sessione FTP interattiva. La connessione dati viene aperta e chiusa per ciascun trasferimento file: viene aperta a ogni comando che comporta il trasferimento di uno o più file, per essere chiusa al

termine del trasferimento. Quando un utente inizia una sessione FTP viene aperta la sessione di controllo; mentre questa è aperta, la connessione dati può essere aperta e chiusa più volte se vengono trasferiti più file. Il protocollo FTP usa due porte TCP note: la porta 21 è usata per la connessione di controllo e la 20 per lo scambio dei dati.

Connessione di controllo

1. Il client FTP contatta il server FTP alla porta 21
2. Il client ottiene l'autorizzazione sulla connessione di controllo
3. Il client invia i comandi sulla connessione di controllo (es. cambio di directory, invio file, ecc.)

La connessione di controllo è *persistente*

Il protocollo FTP usa lo stesso approccio usato da TELNET: la comunicazione attraverso la connessione di controllo avviene per mezzo di caratteri con una codifica standard chiamata NVT ASCII, sia per i comandi sia per le risposte. Pur nella sua semplicità questo metodo è perfettamente adeguato perché l'interazione avviene mediante l'invio di un comando o di una risposta alla volta. Ogni riga termina con la coppia di caratteri ritorno carrello e fine linea.

Durante questa connessione di controllo i comandi vengono inviati dal client al server e le risposte vengono inviate dal server al client. I comandi sono inviati al processo di controllo del client FTP sotto forma di caratteri ASCII maiuscoli e possono opzionalmente essere seguiti da un argomento.

Ogni comando FTP genera almeno una risposta; le risposte sono composte da due parti: un numero di tre cifre e un testo. La parte numerica costituisce il codice della risposta, quella di testo contiene i parametri necessari o informazioni supplementari. La prima cifra fornisce lo stato del comando. La seconda cifra definisce l'area alla quale si applica lo stato. La terza cifra fornisce informazioni supplementari.

Connessione per lo scambio dei dati

Per creare la connessione TCP per il trasferimento dati sono possibili due modalità:

- **Active mode:**

1. Il client, non il server, effettua un'apertura passiva usando una porta effimera (rimarrà in attesa di connessione); ciò viene fatto dal client perché è tale processo che invierà i comandi per il trasferimento dei file.
2. Il client invia questo numero di porta al server per mezzo del comando PORT.
3. Il server, ricevuto il numero di porta, effettua un'apertura attiva (aprirà la connessione) usando la porta nota 20 e quella effimera segnalata dal client.

- **Passive mode:** il client chiede al server di mettersi in ascolto su una porta per una connessione dati tramite il comando PASV, ottiene questo numero di porta dal server e lo usa per aprire la connessione con il server (porta 20 ma non necessariamente)

- Il server fa un'apertura passiva (si mette in ascolto di richieste di connessione su una porta), il client fa un'apertura attiva.

La connessione dati è *non persistente*, una per ciascun trasferimento.

Comunicazione lungo la connessione per lo scambio dei dati La connessione per lo scambio dei dati ha caratteristiche e scopi diversi da quella di controllo. Lo scopo ultimo è il trasferimento di file: il client deve definire il tipo del file, la struttura dei dati e la modalità di trasmissione al fine di risolvere i problemi di eterogeneità tra il client e il server. Il trasferimento del file viene preparato mediante uno scambio di informazioni lungo la connessione di controllo.

Trasferimento di file Il trasferimento dei file avviene lungo la connessione per lo scambio dei dati sotto il controllo dei comandi trasmessi lungo la connessione di controllo. Bisogna ricordare, comunque, che trasferire un file con il protocollo FTP vuol dire effettuare una delle operazioni seguenti: copiare un file dal server al client (retrieving di un file), copiare un file dal client al server (storage di un file) e ottenere una lista dei file presenti sul server (listing).

<i>Comando</i>	<i>Argomenti</i>	<i>Descrizione</i>
ABOR		Interruzione del comando precedente
CDUP		Sale di un livello nell'albero delle directory
CWD	Nome della directory	Cambia la directory corrente
DELE	Nome del file	Cancella il file
LIST	Nome della directory	Elenca il contenuto della directory
MKD	Nome della directory	Crea una nuova directory
PASS	Password utente	Password
PASV		Il server sceglie la porta
PORT	Numero di porta	Il client sceglie la porta
PWD		Mostra il nome della directory corrente
QUIT		Uscita dal sistema
RETR	Nome di uno o più file	Trasferisce uno o più file dal server al client
RMD	Nome della directory	Cancella la directory
RNTO	Nome (del nuovo) file	Cambia il nome del file
STOR	Nome di uno o più file	Trasferisce uno o più file dal client al server
USER	Identificativo	Identificazione dell'utente

Principali comandi FTP

<i>Codice</i>	<i>Descrizione</i>	<i>Codice</i>	<i>Descrizione</i>
125	Connessione dati aperta	250	Azione sul file OK
150	Stato del file OK	331	Nome dell'utente OK, in attesa della password
200	Comando OK	425	Non è possibile aprire la connessione dati
220	Servizio pronto	450	Azione sul file non eseguita; file non disponibile
221	Servizio in chiusura	452	Azione interrotta; spazio insufficiente
225	Connessione dati aperta	500	Errore di sintassi; comando non riconosciuto
226	Connessione dati in chiusura	501	Errore di sintassi nel parametro o negli argomenti
230	Login dell'utente OK	530	Login dell'utente fallito

Risposte FTP

Modalità di trasmissione

- **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.

Conclusione

FTP può essere usato per amministrare i siti web: se è necessario spostare file su server remoti, sui server di un host provider, è possibile usare un client FTP per trasferire i file dalla postazione locale alla postazione in remoto; ovviamente è necessaria l'autenticazione. Tuttavia ci sono alcuni server che supportano l'*anonymous FTP*, ovvero una connessione FTP senza autenticazione:

- login anonymous;
- password guest (o email).

Tipicamente consentono di accedere solo ad una parte del file system e permettono solo un subset di operazioni (es. no PUT).

Ci sono anche altri protocolli per il trasferimento di file, per esempio basati su SSH. Rimanendo su FTP la specifica più sicura e più vicina è l'*FTPS* (securing FTP with TLS): meccanismo che può essere usato da client e server FTP per implementare sicurezza e autenticazione usando il protocollo TLS (RFC 2246).

2.3.6 Approfondimenti

Come migliorare il tempo di caricamento di una pagina agendo a livello di protocollo?

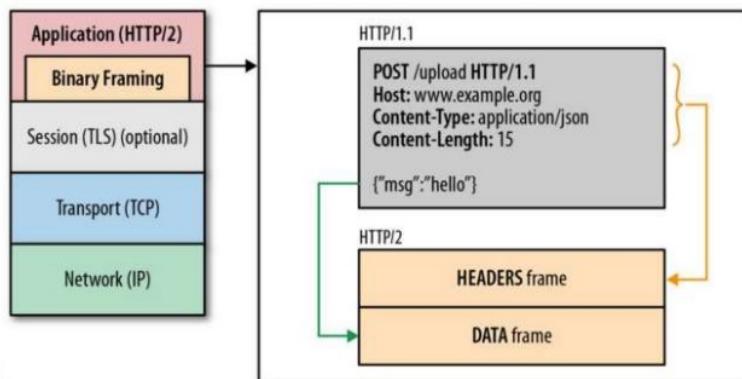
Con HTTP/1.1: i browser instaurano più connessioni TCP in parallelo (tipicamente con un valore massimo di 6 connessioni contemporanee)

- Comportamento non equo
- Il controllo di congestione è poco efficace

HTTP/2

1. Multiplexing delle richieste su un'unica connessione TCP

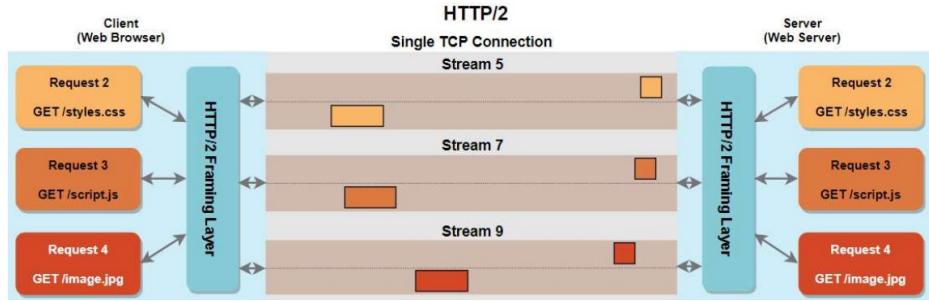
- **Frame**: l'unità di comunicazione in HTTP/2. Una sequenza di frame mappa un messaggio HTTP (es. un frame può essere un HEADER frame, DATA frame, ...)



Frame HTTP/2

- **Stream**: un flusso bidirezionale di frame di un'unica connessione TCP, rappresenta una comunicazione richiesta-risposta

- Mediante l'astrazione degli stream è possibile effettuare il **multiplexing delle richieste** (più stream sulla stessa connessione TCP)



Multiplexing delle richieste

2. Definizione delle priorità:

è possibile associare ad ogni stream

- un **peso** che ne indica la priorità
- una **dipendenza** verso altri stream

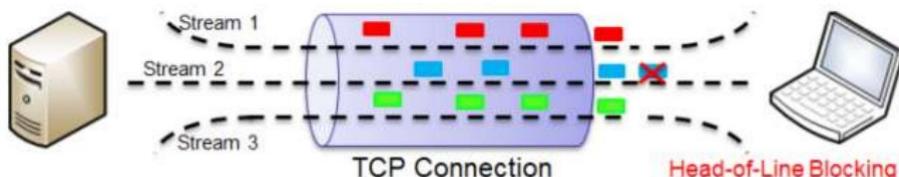
3. Compressione delle intestazioni

4. *Server Push*:

permette al server di inviare risorse aggiuntive per una singola richiesta da parte del client

HTTP/2 risolve il problema di uso inefficiente di una connessione TCP, in particolare per pagine con tante risorse.

Limite: *Head Of Line Blocking*, una perdita (3 ACK/timeout) provoca lo stallo della connessione (e quindi di tutti gli stream).



QUIC Protocollo di trasporto che sfrutta UDP. Aggiunge:

- controllo del flusso e della congestione;
- rilevazione delle perdite e ritrasmissione.

Astrazione dello stream gestito a livello di trasporto. Gli stream sono indipendenti tra loro, la perdita o rallentamento di uno stream non influisce sul progredire degli altri streams.

HTTP/3

- Evoluzione di HTTP/2 che usa i servizi di QUIC
- Mantiene la semantica di HTTP/2
- Gli stream non sono trasparenti al livello di trasporto

2.4 Paradigma peer-to-peer

Il paradigma client/server è stato analizzato nella prima parte di questo capitolo; nei paragrafi precedenti si sono descritte alcune applicazioni client/server standard. In questo paragrafo si analizza il paradigma peer-to-peer (P2P). Il primo esempio di condivisione di file P2P risale al 1987 quando Wayne Bell creò WWIVnet, la componente di rete del software WWIV (World War Four). Nel luglio del 1999, Ian Clarke progettò Freenet, una struttura decentralizzata e resistente alla censura per l'immagazzinamento e la diffusione di informazioni. Freenet ha l'obiettivo, per mezzo di un'architettura P2P, di fornire libertà di parola con forte protezione dell'anonimato.

Il paradigma P2P divenne molto popolare con Napster (1999-2001), un servizio di condivisione di file musicali creato da Shawn Fanning. Sebbene la copia e la diffusione non autorizzata di file musicali da parte degli utenti abbia portato a una condanna per violazione dei diritti di copyright costringendo Napster alla chiusura, esso preparò il terreno per i meccanismi di condivisione dei file P2P che apparvero successivamente. Il rilascio della prima versione di Gnutella avvenne nel marzo dell'anno 2000. Venne seguita da FastTrack (usato da Kazaa), BitTorrent, WinMX e GNUnet rispettivamente in marzo, aprile, maggio e novembre del 2001.

2.4.1 Reti P2P

Gli utenti Internet che intendono condividere le proprie risorse divengono “peer” (pari) e formano una rete.

- Tutti i nodi (peer) hanno la stessa importanza (in linea di principio): nodi indipendenti (autonomi) e localizzati ai bordi (edge) di Internet.
- Sistemi altamente distribuiti: il numero di nodi può essere dell'ordine delle centinaia di migliaia.

Quando uno dei peer nella rete ha un file (per esempio audio o video) da condividere, lo rende disponibile agli altri. Chi è interessato può connettersi

al computer dove il file è memorizzato e prelevarlo; una volta prelevato, lo può a sua volta rendere disponibile ad altri peer. Via via che altri peer entrano a far parte della rete e prelevano il file, il gruppo ha a disposizione sempre più copie. Poiché il gruppo di peer può crescere e ridursi dinamicamente, il problema è come poter traccia della disponibilità dei vari file tra i peer. Per risolvere questo problema è necessario prima suddividere le reti P2P in due categorie: centralizzate e decentralizzate.

Reti centralizzate

In una rete P2P *centralizzata*, il sistema di directory - ovvero l'elenco dei peer e delle risorse condivise - utilizza il paradigma client/server, ma la memorizzazione e lo scaricamento dei file avvengono usando il paradigma P2P. Per questa ragione le reti centralizzate P2P sono a volte chiamate *reti P2P ibride*. Napster è stato un esempio di rete P2P centralizzata. 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. Per evitare il collasso del sistema, Napster utilizzava numerosi server per questo scopo.

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.

Nelle reti centralizzate la gestione del sistema di directory è piuttosto semplice ma presenta vari inconvenienti. L'accesso alla directory può generare un traffico enorme e rallentare tutto il sistema. I server centrali sono vulnerabili agli attacchi e, se dovessero subire danni, l'intero sistema crollerebbe. Napster perse la causa relativa alla violazione del copyright proprio a causa della componente centrale del sistema, che fu ritenuta direttamente responsabile delle attività illegali condotte dagli utenti, e fu costretto a chiudere nel luglio del 2001.

Reti decentralizzate

Una rete P2P *decentralizzata* non utilizza un sistema directory centralizzato. Con questo modello i peer si organizzano formando un *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 (la modalità di organizzazione), una rete P2P decentralizzata si classifica in strutturata o non strutturata.

Reti non strutturate I nodi in una rete *non strutturata* sono collegati tra loro in modo casuale. Una ricerca in una rete P2P non strutturata non è molto efficiente poiché una qualsiasi interrogazione (ad esempio proprio per trovare un file) deve essere inviata in *flooding* (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. Tuttavia, l'aggiunta o la rimozione dei nodi è un'operazione semplice e poco costosa; tali reti sono infatti adatte per gestire nodi con un comportamento fortemente transiente (tassi di join/leave elevati). Esempi di questo tipo di rete sono Gnutella e Freenet.

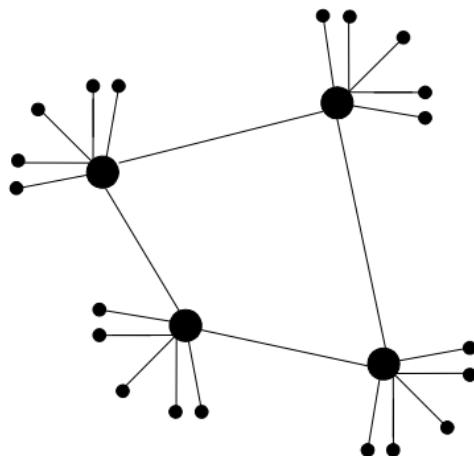
La rete Gnutella è un esempio di rete P2P decentralizzata e non strutturata. Non è strutturata nel senso che il meccanismo di directory è distribuito tra nodi. Quando il nodo A desidera accedere a un oggetto (per esempio un file) contatta uno dei suoi vicini, uno qualsiasi dei nodi con cui è collegato nella rete di overlay. Il nodo A invia quindi un messaggio di richiesta (query) al suo vicino W. La richiesta contiene le informazioni necessarie per identificare l'oggetto richiesto (per esempio il nome del file). Se il nodo W conosce l'indirizzo del nodo X che possiede l'oggetto, trasmette un messaggio di risposta (response) che contiene l'indirizzo del nodo X. Il nodo A può ora utilizzare un protocollo di trasferimento come HTTP per ottenere una copia dell'oggetto dal nodo X. Se il nodo W non conosce l'indirizzo del nodo X, invia in flooding la richiesta di A a tutti i suoi nodi vicini. Alla fine uno dei nodi della rete risponderà al messaggio di interrogazione, e il nodo A potrà accedere al nodo X. Sebbene in Gnutella il meccanismo di flooding sia implementato in modo da evitare un traffico eccessivo, una delle ragioni per cui Gnutella è difficilmente scalabile è proprio per l'impiego di questa tecnica.

Uno dei problemi non ancora affrontati è come possa il nodo A conoscere l'indirizzo di almeno un vicino. Questo viene effettuato nella cosiddetta fase di bootstrap (accesso iniziale del client alla rete P2P) attraverso varie strategie. Nella più semplice il software include un elenco di nodi (peer) che il nodo A può memorizzare immediatamente come vicini.

Copertura gerarchica

- Cerca di combinare il “meglio” delle reti centralizzate e del query flooding:
 - no server con tutti i contenuti (solo “bootstrap servers”);
 - i peer non sono tutti uguali (esistono i group leader).
- Ogni peer

- è group leader (se “potente” in banda o risorse)
- oppure viene assegnato a un group leader
- Connessioni TCP
 - tra peer e il suo group leader
 - tra (alcune) coppie di group leader
- Group leader tiene traccia del contenuto dei suoi “figli”
 - *Group leader* \sim *Napster-like mini-server*
 - *Leader-to-leader connections* \sim *Gnutella-like overlay network*



Copertura gerarchica

- Ogni file associato con un suo hash e un suo descrittore
- Client invia una query di keyword a suo group leader
- Group leader risponde con dei “match” del tipo

$$<\text{hash del file}, \text{indirizzo IP}>$$
- Se leader inoltra la query ad altri leader, questi rispondono con altri match
- Il cliente sceglie quindi i file da scaricare

Reti strutturate In una rete strutturata i nodi vengono organizzati seguendo un ben preciso insieme di regole, in modo da migliorare l'efficienza di alcune operazioni come, ad esempio, la ricerca di un particolare contenuto. L'aggiunta o la rimozione di nodi è un'operazione costosa. Un famoso protocollo che utilizza tecniche di questo tipo è BitTorrent.

Reti strutturate - DHT

- Sistemi con Distributed Hash Table (DHT)
- Ad ogni peer è assegnato un ID ed ogni peer conosce un certo numero di peer
- Ad ogni risorsa condivisa (pubblicata) viene assegnato un ID, basato su una funzione hash applicata al contenuto della risorsa ed al suo nome
- Routing della risorsa pubblicata verso il peer che ha l'ID più simile a quello della risorsa
- La richiesta per la risorsa specifica sarà instradata verso il peer che ha l'ID più simile a quello della risorsa

2.4.2 BitTorrent: una rete P2P molto diffusa

BitTorrent è un protocollo P2P progettato da Bram Cohen per la condivisione di file particolarmente voluminosi. Il termine *condivisione* (sharing) in questo contesto è tuttavia usato in modo differente dai classici protocolli di condivisione di file. Non si ha semplicemente un peer che consente a un altro peer di prelevare un intero file: in questo caso è un gruppo di peer che collabora per fornire ad altri peer del gruppo una copia del file. La condivisione dei file avviene tramite un processo collaborativo chiamato *torrent*. Ogni peer che partecipa a un torrent preleva parti del file (da 256 KB), chiamate chunk, da un altro peer e contemporaneamente trasmette i propri chunk ad altri peer che non li possiedono ancora. Questa divisione dei file permette di

- ridurre il carico di ogni sorgente;
- ridurre la dipendenza dal distributore originale;
- fornire ridondanza.

Viene implementata una strategia chiamata *tit-for-tat* (occhio per occhio) che assomiglia molto a quella di un gioco per bambini. L'insieme dei peer

che fanno parte di un torrent viene chiamato *swarm* (sciame). In uno sciame, un peer che possiede una copia dell'intero file viene chiamato *seed* (seme); un nodo che ha solo una parte del file e desidera ottenere il file completo si chiama invece *leech* (sanguisuga). Uno sciame è quindi composto da vari seed e leech.

BitTorrent con tracker

BitTorrent nella versione originale prevede un ulteriore elemento detto *tracker*. Questo, come indicato dal nome, tiene traccia delle operazioni dello sciame, come si vedrà in seguito. La Figura 2.2 illustra un esempio di torrent con seed, leech e tracker.

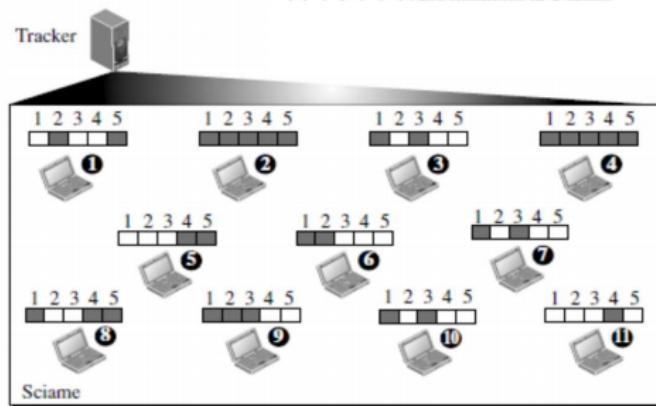


Figura 2.2: Esempio di torrent

Nella figura il file da condividere è suddiviso in cinque chunk. I peer 2 e 4 possiedono già il file completo, gli altri ne hanno solo alcune parti: le parti possedute da un nodo nella figura sono ombreggiate. L'uploading e il downloading delle parti continua, mentre alcuni peer possono abbandonare il torrent e altri peer possono aggiungersi.

Si supponga ora che un nuovo peer voglia scaricare proprio 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 *vicini* (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.

Sebbene i meccanismi di accesso, condivisione e abbandono di un torrent possano sembrare molto semplici, il protocollo BitTorrent utilizza un insieme di politiche per garantire equità, per incoraggiare i peer a partecipare attivamente allo scambio, per evitare di sovraccaricare un peer con troppo richieste e per consentire ai peer di effettuare scambi alla velocità più ampia possibile.

Per evitare di sovraccaricare un peer e garantire un certo livello di equità, ciascun peer generalmente non può avere più di quattro vicini con cui effettuare scambi. I peer inoltre classificano i vicini come *choked* (soffocati) oppure no, e come *interested* (interessati allo scambio) oppure no.

Ogni peer divide i propri vicini nei quattro gruppi derivanti dalle combinazioni di: choked oppure no, interested oppure no. Il gruppo non choked è quello al quale il peer attualmente è connesso e con il quale scambia chunk. Il gruppo chocked invece è l'insieme dei vicini ai quali non è attualmente connesso ma potrebbe connettersi in futuro.

Ogni 10 secondi ogni peer controlla uno dei peer nel gruppo degli interessati ma soffocati per verificare se può ottenere una migliore velocità di trasferimento. Se questo peer ha una velocità superiore a uno qualsiasi dei peer non soffocati entra a far parte di quest'ultimo gruppo, mentre il peer non choked con la velocità più bassa viene spostato nel gruppo dei soffocati. Così facendo i nodi nel gruppo non soffocato sono sempre quelli con la migliore velocità tra quelli controllati. Questa strategia divide i vicini in sottogruppi, nei quali i nodi vicini con velocità di trasferimento dati compatibile comunicano l'uno con l'altro. In questo meccanismo si può vedere l'implementazione della strategia di scambio tit-for-tat menzionata in precedenza.

Per consentire a un peer, che è appena entrato in uno sciame ma che non ha ancora nulla da condividere, di ricevere chunk dagli altri partecipanti, viene implementata una politica di *unchoking ottimistico*. Significa che tutti i peer promuovono casualmente, ogni 30 secondi, un peer dal gruppo soffocato, indipendentemente dalla sua velocità di uploading, inserendolo nel gruppo dei non soffocati.

Inoltre, il protocollo di BitTorrent cerca di bilanciare il numero di chunk che ogni peer possiede in ciascun istante utilizzando una strategia chiamata *rarest-first* (prima il più raro). Questa strategia prevede che ogni peer tenti per prima cosa di prelevare le parti con il minor numero di copie esistenti fra i vicini. Così facendo, i chunk rari vengono scambiati molto rapidamente e si giunge in fretta a una distribuzione piuttosto omogenea.

Capitolo 3

Livello di trasporto

Il livello di trasporto nella pila TCP/IP è posizionato fra il livello applicazione e il livello rete: fornisce quindi servizi al livello applicazione e riceve servizi dal livello rete. Il livello trasporto agisce da intermediario fra un programma client e il rispettivo programma server, creando un collegamento tra processi. Il livello trasporto è il cuore della pila TCP/IP: è il meccanismo end-to-end che consente di trasferire dati da un punto all'altro in Internet.

3.1 Introduzione

Il livello trasporto si colloca fra il livello rete e il livello applicazione, e fornisce un servizio di comunicazione tra processi fra i due livelli applicazione, uno sull'host locale e l'altro sull'host remoto. Il servizio di comunicazione viene fornito tramite una connessione logica, che consente ai due livelli applicazione (che possono essere situati in qualsiasi parte del mondo) di lavorare come se ci fosse una connessione diretta attraverso la quale inviare e ricevere i messaggi.

3.1.1 I servizi del livello trasporto

Comunicazione tra processi

Il primo compito di un protocollo di livello trasporto è supportare la *comunicazione tra processi*. Un processo è un'entità (programma in esecuzione) di livello applicazione che usa i servizi del livello trasporto.

Il livello rete si occupa della comunicazione tra dispositivi, ovvero della comunicazione tra host. Un protocollo di rete consegna i messaggi al computer destinatario, ma questo tipo di consegna è incompleta, perché i messaggi devono giungere ai processi cui erano indirizzati. È a questo punto che

intervengono i protocolli del livello trasporto, che completano la consegna passando i messaggi ai processi applicativi destinatari.

Indirizzamento: i numeri di porta

Sebbene la comunicazione tra processi possa essere realizzata in diversi modi, la più diffusa è sicuramente quella basata sul paradigma client/server.

Va osservato, però, che la maggior parte dei sistemi operativi attuali è multiutente e multiprocesso, quindi l'host locale ha spesso diversi processi client attivi, così come l'host remoto può avere diversi processi server attivi. Affinché si possa stabilire una comunicazione tra i due dispositivi, è necessario un metodo per individuare: l'host locale, l'host remoto, il processo locale e il processo remoto. Gli host vengono individuati per messo del loro indirizzo IP. I processi necessitano di ulteriori identificatori, detti *numero di porta*. I protocolli TCP/IP usano come numeri di porta dei numeri interi compresi tra 0 e 65535 (codificabili in 16 bit).

Al client viene assegnato un numero di porta che viene detto *effimero*, ovvero “di breve durata”, dato il breve tempo di vita di un processo client. I numeri di porta effimeri sono superiori a 1023.

Anche al server deve essere associato un numero di porta, che, però, non può essere scelto a caso, perché tale numero deve essere in qualche modo noto al client. Una soluzione potrebbe essere l'invio di un pacchetto al fine di scoprire questo numero di porta, ma ciò creerebbe un carico inutile sulla rete. Nei protocolli TCP/IP si usano numeri universali per i server, che sono detti *numeri di porta noti (well-known)*. Ogni client conosce il numero di porta noto del server corrispondente.

Multiplexing e demultiplexing

Il termine *multiplexing* (molti a uno) fa riferimento al caso in cui un'entità riceve informazioni da più di una sorgente; il termine *demultiplexing* fa riferimento al caso in cui un'entità trasmette informazioni a più di un destinatario. Il livello trasporto effettua multiplexing nel sito mittente e demultiplexing nel sito destinatario. Le operazioni di multiplexing e demultiplexing si basano sui socket address dei processi e pertanto dipendono dal numero di porta su cui i processi sono attivi.

Demultiplexing senza connessione Lo strato di trasporto dell'host ricevente consegna il datagramma UDP alla socket identificata da IP e porta destinazione. I datagrammi con IP e/o porta mittente differenti ma stessi IP e porta destinatari vengono consegnati alla stessa socket.

Demultiplexing orientato alla connessione L'host ricevente usa i quattro parametri per inviare il segmento alla socket appropriata. Un host server può supportare più socket contemporanee (socket differenti per ogni connessione).

3.2 Il protocollo TCP (Transmission Control Protocol)

Il protocollo TCP (Transmission Control Protocol) è un protocollo orientato alla connessione e affidabile. Per fornire un servizio orientato alla connessione TCP prevede esplicitamente i meccanismi di apertura della connessione, di trasferimento dei dati e di chiusura della connessione.

3.2.1 Servizi del protocollo TCP

Il protocollo TCP fornisce un servizio di trasporto *affidabile* tra processi basato sui numeri di porta. TCP effettua il multiplexing in trasmissione e il demultiplexing in ricezione. È un protocollo *orientato alla connessione*, pertanto è necessario stabilire una connessione per ogni coppia di processi prima di avviare la comunicazione tra processi.

Quando un processo sull'host A desidera inviare e ricevere dati da un altro processo sull'host B, avvengono le seguenti tre fasi:

1. I due processi TCP stabiliscono una connessione logica tra di essi.
2. Vengono scambiati dei dati in entrambe le direzioni.
3. La connessione viene terminata.

Si noti che si tratta di una connessione logica, non fisica. Il segmento TCP viene incapsulato in un datagramma IP e può essere ricevuto fuori sequenza, corrotto o addirittura smarrito e quindi ritrasmesso. Ciascun datagramma IP può essere instradato verso il destinatario su percorsi differenti - non esiste un collegamento fisico diretto. TCP crea un ambiente orientato al flusso e si occupa di consegnare al ricevente i byte nella sequenza corretta.

Oltre a essere orientato alla connessione, il protocollo TCP è anche *orientato al flusso di dati (stream-oriented)*, ovvero permette al livello applicazione di trasmettere un flusso di dati. Un processo, quando utilizza UDP, richiede l'invio di una serie di messaggi indipendenti l'uno dall'altro. UDP aggiunge la propria intestazione a ciascuno di questi messaggi e li consegna al livello rete

(protocollo IP) per la trasmissione. Ciascun messaggio del livello applicazione (prodotto da un processo in esecuzione) viene incapsulato in un pacchetto UDP che a sua volta verrà incapsulato in un datagramma IP. UDP e IP non riconoscono alcuna relazione fra i vari datagrammi.

TCP, al contrario, consente al processo in trasmissione di fornire i dati in un flusso continuo di byte, che viene ricevuto come tale dal processo in ricezione. TCP crea un ambiente nel quale i due processi sembrano essere collegati tramite un “tubo” che trasporta la sequenza di byte attraverso Internet. Il processo in trasmissione produce (immette) il flusso, mentre il processo in ricezione lo consuma (legge).

Poiché i processi in trasmissione e in ricezione non scrivono o leggono i dati necessariamente alla stessa velocità, occorrono dei buffer in cui si possono memorizzare i segmenti inviati e ricevuti. Vi sono due buffer, uno di trasmissione e uno di ricezione, per ciascuna direzione. Si vedrà nel seguito che questi buffer sono necessari anche ai meccanismi di controllo di flusso e di gestione degli errori previsti da TCP.

Sebbene la bufferizzazione possa gestire la differenza di velocità fra i processi produttore e consumatore, è necessario un passo ulteriore per poter trasmettere i dati. Il livello rete, quale fornitore di servizi al protocollo di livello trasporto, deve inviare i dati in pacchetti, e non come flusso di byte. A livello trasporto, TCP raggruppa quindi un certo numero di byte in unità chiamate *segmenti*. TCP aggiunge un'intestazione formata da varie informazioni di controllo a ciascun segmento, che consegna al livello rete per la trasmissione. I segmenti vengono incapsulati in datagrammi IP e trasmessi. Tutte queste operazioni risultano essere perfettamente trasparenti al processo ricevente. Si noti che i segmenti non sono necessariamente tutti della stessa dimensione.

Un altro aspetto da considerare è che il protocollo TCP offre un servizio *full-duplex*, nel quale i dati possono fluire contemporaneamente in entrambe le direzioni. Ciascuna entità TCP ha quindi i propri buffer di trasmissione e ricezione e i segmenti possono circolare in entrambe le direzioni.

3.2.2 Numeri di sequenza e di riscontro

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. TCP numera ciascun byte di dati che viene trasmesso in una connessione. Tale numerazione è indipendente in ciascuna direzione. Quando TCP riceve i byte di dati da un processo, li memorizza nel buffer di trasmis-

sione e li numera. La numerazione non deve necessariamente iniziare da 0 bensì da un numero arbitrario fra 0 e $2^{32} - 1$ come indirizzo del primo byte.

TCP, dopo aver numerato i byte, assegna un numero di sequenza a ogni segmento che viene inviato. Il numero di sequenza, per ciascuna direzione, viene definito come segue:

1. Il numero di sequenza del primo segmento, detto ISN (*Initial Sequence Number* - Numero di Sequenza Iniziale), è un numero casuale.
2. Il numero di sequenza di qualsiasi altro segmento è il numero di sequenza del segmento precedente sommato al numero di byte (reali o fintizi) ivi contenuti.

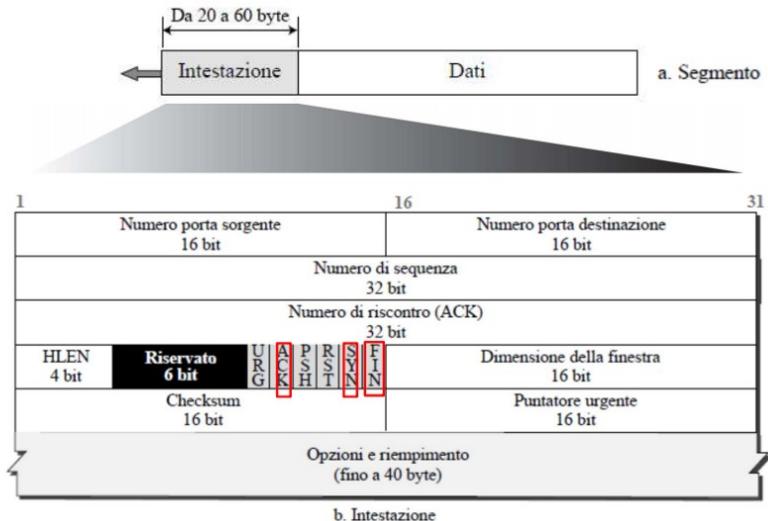
L'intestazione dei segmenti, oltre al numero di sequenza contiene anche un numero di riscontro (acknowledgment, per brevità ack). Poiché la comunicazione in TCP è full-duplex, a connessione stabilita entrambe le entità possono inviare e ricevere contemporaneamente dati. Entrambe numerano i byte dati, in genere con un numero di partenza differente. Per ciascuna direzione, il numero di sequenza indica il numero del primo byte contenuto nel segmento. Entrambe le entità utilizzano un numero di riscontro per confermare i byte ricevuti. Tuttavia il numero di riscontro indica il numero del prossimo byte che l'entità si aspetta di ricevere (e non il numero dell'ultimo byte ricevuto).

3.2.3 Protocolli bidirezionali

Poiché i pacchetti dati vengono trasmessi in entrambe le direzioni, anche i riscontri devono viaggiare in entrambe le direzioni. Per migliorare l'efficienza dei protocolli bidirezionali viene utilizzata una tecnica chiamata *piggybacking* (“viaggiare in spalla a qualcuno”). Quando un pacchetto trasporta dei dati da A a B, può trasportare anche i riscontri relativi ai pacchetti ricevuti da B; viceversa, quando un pacchetto trasporta dei dati da B ad A, può trasportare anche i riscontri relativi ai pacchetti ricevuti da A.

3.2.4 Formato dei segmenti

Il protocollo TCP riceve i dati in forma di messaggi dal livello applicazione e li incapsula in *segmenti*. Il segmento consiste di un'intestazione di dimensione compresa tra i 20 e i 60 byte, seguito dai dati provenienti dal programma applicativo. La dimensione dell'intestazione è di 20 byte in assenza di opzioni, fino a 60 byte altrimenti.



Formato dei segmenti TCP

- *Numero di porta sorgente.* Campo a sedici bit contenente il numero di porta del processo mittente sull'host che invia il segmento.
- *Numero di porta destinazione.* Campo a sedici bit contenente il numero di porta del processo destinatario sull'host che riceve il segmento.
- *Numero di sequenza.* Campo a 32 bit contenente il numero attribuito al primo byte di dati contenuto nel segmento.
- *Numero di riscontro (acknowledgement number).* Campo a 32 bit che contiene il numero di sequenza del byte che il destinatario si aspetta di ricevere.
- *Lunghezza dell'intestazione (HLEN).* Campo a quattro bit che indica il numero di parole di quattro byte presenti nell'intestazione TCP.
- *Flag di controllo.* Campo contenente 6 bit di controllo (flag); è possibile che più di un bit sia attivo simultaneamente.
 - URG: il campo *puntatore urgente* contiene dati significativi da trasferire in via prioritaria.
 - ACK: il campo *numero di riscontro* contiene dati significativi
 - PSH: funzione Push (trasferimento immediato dei dati in un segmento dal traporto al livello applicativo).

- RST: reset della connessione.
 - SYN: sincronizza il numero di sequenza.
 - FIN: non ci sono altri dati dal mittente - chiusura della connessione
- *Dimensione della finestra.* Campo a 16 bit che indica la dimensione della finestra di cui l'altro host coinvolto nella trasmissione deve disporre.
 - *Checksum.* Campo a 16 bit contenente il checksum; viene usato per il controllo degli errori.
 - *Opzioni.* Vi possono essere da 0 a 40 byte di informazioni opzionali nell'intestazione del pacchetto TCP.

3.2.5 La connessione TCP

Il protocollo TCP è orientato alla connessione, ovvero stabilisce un percorso virtuale fra il mittente e il destinatario. Tutti i segmenti appartenenti a un messaggio vengono spediti lungo questo percorso logico. Utilizzare un percorso virtuale per l'intero messaggio semplifica il processo di conferma e di ritrasmissione dei segmenti smarriti o danneggiati. Ci si potrebbe chiedere come TCP possa essere orientato alla connessione dato che utilizza i servizi di IP, un protocollo privo di connessioni. In realtà la connessione TCP è virtuale, non fisica. TCP opera a un livello più alto: utilizza i servizi di IP per consegnare i singoli segmenti al destinatario, ma è il TCP stesso che controlla la connessione. Un segmento smarrito o danneggiato viene ritrasmesso da TCP, ma IP è inconsapevole di tale ritrasmissione. Un segmento ricevuto fuori sequenza viene bufferizzato da TCP fino a quando riceve il segmento mancante: IP è ignaro di questo riordinamento.

La trasmissione orientata alla connessione nel protocollo TCP richiede tre fasi: apertura della connessione, trasmissione dei dati e chiusura della connessione.

Apertura della connessione

TCP trasmette i dati in modalità full-duplex: due entità TCP connesse in due macchine differenti sono in grado di trasmettere dati l'una all'altra simultaneamente. Questo comporta che ciascuna entità deve inizializzare la comunicazione e ottenere l'approvazione dell'entità corrispondente prima di poter trasferire qualsiasi dato.

La procedura di apertura della connessione in TCP è detta **three way handshake** (stretta di mano a tre vie). Consideriamo un esempio in cui

un programma applicativo, chiamato client, vuole collegarsi a un altro programma applicativo, chiamato server, usando il protocollo di livello trasporto TCP.

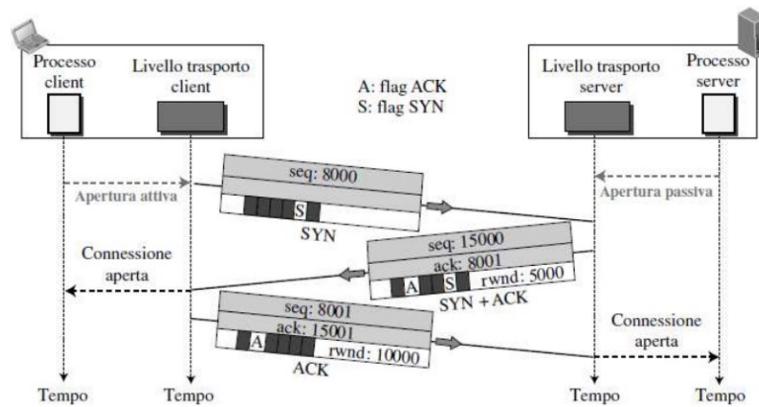
Il processo server deve essere attivo, e pronto ad accettare connessioni mediante il livello TCP, per cui deve richiedere al TCP una *apertura passiva*. Nonostante il server TCP sia in grado di aprire connessioni con un qualsiasi dispositivo connesso alla rete mondiale, non può aprire la connessione da solo, ma dietro la richiesta di un processo applicativo.

La richiesta fatta dal processo client, invece, è detta richiesta di *apertura attiva*: un processo client che voglia aprire una connessione con un processo server su un particolare dispositivo ne informa il suo TCP, il quale dà inizio alla procedura three way handshake.

La procedura comporta i tre passi seguenti:

1. Il client invia il primo segmento, un segmento SYN nel quale viene impostato il solo flag SYN, che serve per la sincronizzazione dei numeri di sequenza. Il client sceglie un numero random come primo numero di sequenza, chiamato numero di sequenza iniziale, o ISN, e lo invia al server. Si noti che questo segmento non contiene alcun numero di riscontro e non definisce la dimensione della finestra: quest'ultima è significativa solo nel caso in cui il segmento include un riscontro. Si noti che il segmento SYN è un segmento di controllo e non contiene dati utente, tuttavia usa un numero di sequenza: quando inizierà il trasferimento dei dati, il numero di sequenza verrà incrementato di un'unità. Si potrebbe dire che il segmento SYN non trasporta dati reali, ma solo un byte immaginario.
2. Il server invia il secondo segmento, un segmento con i due flag SYN e ACK attivati. Il suo scopo è duplice: in primo luogo è un segmento SYN per la comunicazione nell'altra direzione. Il server utilizza questo segmento per inizializzare un numero di sequenza necessario per numerare i byte inviati dal server al client. In un secondo luogo il server usa il segmento 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. Dato che contiene un riscontro, deve anche definire la dimensione della finestra di ricezione (*rwnd*) che il client dovrà utilizzare. Essendo un segmento SYN esso deve essere confermato, quindi usa un numero di sequenza.
3. Il client invia il terzo segmento, un segmento ACK, che conferma l'avvenuta ricezione del secondo segmento mediante il flag ACK. Si noti che il segmento ACK non usa alcun numero di sequenza se non contiene

dati utenti, ma alcune implementazioni consentono a questo terzo segmento, nella fase di apertura della connessione, 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.



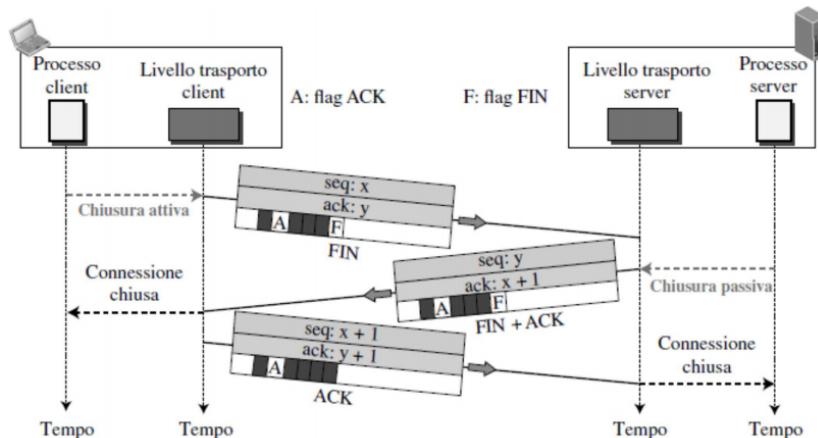
Apertura della connessione mediante il three-way handshake

Chiusura della connessione

Ciascuna delle due parti coinvolte nello scambio dei dati (client o server) può chiudere la connessione, sebbene la chiusura sia solitamente iniziata dal client. La maggior parte delle implementazioni attuali fornisce due opzioni per la chiusura della connessione: handshake a tre vie e handshake a quattro vie con l'opzione di half-close.

1. Il client TCP, dopo aver ricevuto un comando di chiusura del processo applicativo client, invia un primo segmento FIN nel quale viene impostato il flag FIN. Si noti che il segmento FIN può contenere l'ultima parte dei dati inviati dal client o può essere un semplice segmento di controllo. Se si tratta di un segmento di controllo, usa un solo numero di sequenza perché richiede un riscontro.
2. Il server TCP, dopo aver ricevuto il segmento FIN, notifica la situazione al suo processo applicativo e invia il segmento FIN + ACK, per riscontrare le ricezione del segmento FIN al client e per annunciare la chiusura della connessione nella direzione opposta. Questo segmento può anche contenere gli ultimi dati da parte del server; se non trasporta dati, consuma solo un numero di sequenza perché è necessario riscontrare la ricezione.

- Il client TCP invia l'ultimo segmento dell'handshake, un segmento ACK, per riscontrare la ricezione del segmento del segmento FIN dal server TCP. Questo segmento contiene il numero di riscontro, che vale uno più il numero di sequenza ricevuto nel segmento FIN dal server; non può trasportare dati e non consuma numeri di sequenza.



Chiusura della connessione mediante three-way handshake.

Half-close

In TCP, uno dei due processi può smettere di inviare dati mentre ne sta ancora ricevendo: si tratta della cosiddetta *half-close* (chiusura a metà). Sebbene entrambe le entità possano eseguire la half-close, essa è solitamente iniziata dal client. Questo può avvenire quando il server ha bisogno di tutti i dati prima di poter procedere alla loro elaborazione. Un esempio classico è l'ordinamento: quando il client invia i dati al server affinché siano ordinati, il server deve ricevere tutti i dati prima di poter iniziare l'ordinamento. Il client può dunque chiudere la connessione nella direzione di uscita una volta trasmessi tutti i dati, mentre la connessione nella direzione di ingresso deve rimanere aperta per poter ricevere i risultati. Il server, dopo aver ricevuto i dati, ha bisogno di più tempo per poterli ordinare, la sua connessione in uscita deve quindi rimanere aperta.

Il client chiede l'half-close della connessione inviando un segmento FIN. Il server accetta la richiesta di chiusura inviando un segmento ACK. Il trasferimento dei dati dal client al server termina, ma il server può continuare a inviare dati. Quando il server ha terminato l'invio dei dati elaborati, invia un segmento FIN, che viene riscontrato da un segmento ACK dal client.

Dopo il completamento della half-close è ancora possibile trasmettere i dati dal server al client e i riscontri dal client al server, ma il client non può inviare altri dati.

Stato TIME-WAIT TIME-WAIT 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. Il tempo di permanenza in questo stato è di 2MSL.

La MSL è la *stima del massimo periodo di tempo che un pacchetto IP può vivere sulla rete*; questo tempo è limitato perché ogni pacchetto IP può essere ritrasmesso dai router un numero massimo di volte (detto hop limit).

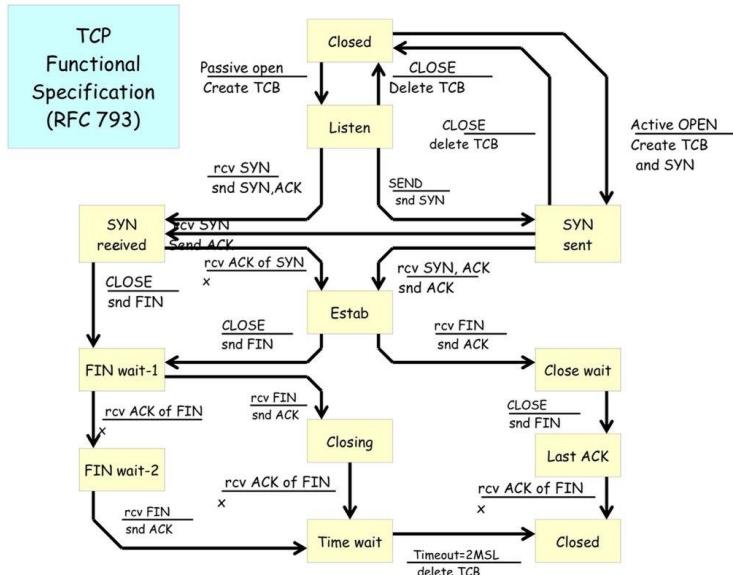


Diagramma delle transizioni di stato

Lo stato TIME-WAIT viene utilizzato dal protocollo per due motivi principali:

- **implementare in maniera affidabile la terminazione della connessione** in entrambe le direzioni
 - Se l'ultimo ACK della sequenza viene perso, chi esegue la chiusura passiva manderà un ulteriore FIN, chi esegue la chiusura attiva deve mantenere lo stato della connessione per poter reinviare l'ACK.

- consentire l'eliminazione dei segmenti duplicati in rete

<i>Stato</i>	<i>Descrizione</i>
CLOSED	Assenza di connessione.
LISTEN	Ricevuta richiesta di apertura passiva; server in attesa di un messaggio SYN da parte del client.
SYN-SENT	Richiesta di connessione SYN inviata, in attesa di ACK.
SYN-RCVD	SYN + ACK inviati; in attesa di ACK.
ESTABLISHED	Connessione aperta, trasferimento dati in atto.
FIN-WAIT-1	Primo FIN inviato; in attesa di ACK.
FIN-WAIT-2	Ricevuto ACK al primo FIN; in attesa di secondo FIN.
CLOSE-WAIT	Ricevuto primo FIN, ACK inviato; in attesa che il processo richieda la chiusura.
TIME-WAIT	Ricevuto secondo FIN, ACK inviato, in attesa del timeout 2MSL.
LAST-ACK	Secondo FIN inviato; server in attesa di ACK.
CLOSING	Le due parti hanno deciso simultaneamente di chiudere la connessione

Stati del protocollo TCP

3.2.6 Controllo degli errori

Il protocollo TCP è un protocollo di trasporto affidabile; ovvero garantisce al livello applicazione che i dati inviati verranno consegnati nel giusto ordine, senza errori, senza smarrimenti o duplicazioni.

Il controllo degli errori garantisce affidabilità e prevede meccanismi per l'individuazione e la rispedizione dei segmenti corrotti, per la rispedizione dei segmenti smarriti, per l'identificazione e lo scarto dei segmenti duplicati e per la memorizzazione dei segmenti ricevuti fuori sequenza fino a quando non arrivano i segmenti mancanti. I tre semplici strumenti usati dal protocollo TCP per individuare gli errori di trasmissione sono il checksum, i messaggi di riscontro e il timeout.

Checksum

Il protocollo TCP prevede che ciascun segmento contenga un campo checksum di 16 bit, utilizzato per identificare i segmenti corrotti. Se un segmento è

corrotto, questo viene ignorato dal destinatario e considerato smarrito. TCP utilizza un checksum di 16 bit, obbligatorio in ogni segmento.

Riscontri

TCP utilizza i riscontri per confermare la ricezione dei segmenti dati e dei segmenti di controllo che non contengono dati ma che usano un numero di sequenza. I segmenti di riscontro (ACK) non sono mai riscontrati.

Il protocollo TCP è stato originariamente progettato per riscontrare la ricezione dei segmenti in modo cumulativo: il destinatario notifica il numero di byte che si attende, ignorando tutti gli eventuali segmenti giunti fuori sequenza e memorizzati. Questa tecnica viene a volte indicata come *riscontro cumulativo positivo* (ACK). Il termine positivo si riferisce al fatto che i segmenti scartati, smarriti o duplicati non vengono notificati. Il campo ACK a 32 bit nell'intestazione TCP è utilizzato per i riscontri cumulativi e il suo valore è da considerarsi valido solo quando è impostato il flag corrispondente ACK.

Un numero crescente di implementazioni stanno aggiungendo un nuovo tipo di riscontro chiamato *selettivo* (SACK). Il SACK non sostituisce l'ACK, ma notifica ulteriori informazioni al mittente: segmenti che non sono in sequenza e segmenti duplicati. Tuttavia, dato che non è possibile aggiungere questo tipo di informazioni nell'intestazione TCP, SACK viene implementato come opzione al temine dell'intestazione.

Quando, il destinatario, genera i riscontri? Nell'evoluzione di TCP sono state definite varie regole, utilizzate poi da varie implementazioni. Di seguito si riportano le regole più comuni; l'ordine non riflette necessariamente l'importanza.

1. Quando un'entità invia un segmento dati all'entità corrispondente, deve includere (piggybacking) un riscontro che fornisca il prossimo numero di byte che prevede di ricevere. Questa regola diminuisce il numero di segmenti necessari e riduce quindi il traffico.
2. Quando il destinatario non ha dati da inviare e riceve un segmento nell'ordine corretto (ovvero con il numero di sequenza atteso) e il segmento precedente è già stato riscontrato, ritarda l'invio del segmento ACK fino a quando non arriva un altro segmento o fino a quando non scade un tempo massimo predefinito (normalmente di 500 ms). In altre parole il destinatario ritarda l'invio di un segmento ACK se ha un unico segmento nella sequenza corretta da riscontrare. Anche questa regola (detta *ACK posticipato* o *delayed*) evita che i segmenti ACK creino un traffico eccessivo, e fa uso di un timer di ACK-posticipato.

3. Quando arriva un segmento con numero di sequenza atteso e il segmento precedente non è stato riscontrato, il destinatario invia immediatamente un segmento ACK. In altre parole, non ci devono mai essere più di due segmenti nell'ordine corretto non riscontrati. Questo serve a evitare l'inutile ritrasmissione dei segmenti, che potrebbe produrre congestione nella rete.
4. Quando arriva un segmento fuori sequenza, il destinatario invia immediatamente un segmento ACK notificando il numero di sequenza atteso nel prossimo segmento. Questo consente la *ritrasmissione rapida* (fast retransmission) di eventuali segmenti smarriti.
5. Quando arriva un segmento mancante, il destinatario invia un segmento ACK per notificare il prossimo numero di sequenza atteso. Questo serve a notificare al mittente l'avvenuta ricezione di un segmento precedentemente annunciato come smarrito.
6. Quando arriva un segmento duplicato, il destinatario lo scarta e invia immediatamente un riscontro con il numero di sequenza del prossimo segmento atteso. Questo risolve alcuni problemi legati alla perdita dei segmenti di conferma.

Ritrasmissione dei segmenti

Il cuore del meccanismo di controllo degli errori è la ritrasmissione dei segmenti: quando un segmento viene inviato, viene memorizzato in una coda in attesa di essere riscontrato. Alla scadenza del timer di ritrasmissione o quando il mittente riceve tre ACK duplicati per il precedente segmento nella coda, quel segmento viene trasmesso. Analizziamo ora i due casi nel dettaglio.

Il TCP mittente inizializza un *timer di ritrasmissione* o RTO (Retransmission Time-Out), per ogni segmento inviato. Allo scadere del tempo limite senza averne ricevuto un riscontro, il segmento all'inizio della coda (il segmento con il numero di sequenza inferiore) viene ritrasmesso e si fa ripartire il timer. Si noti che si ipotizza sempre che $S_f < S_n$, dove S_f (*SendFirst*) è il numero di sequenza del primo byte in attesa di essere riscontrato e S_n (*Send next*) è il prossimo byte da inviare.

La regola relativa alla trasmissione dei segmenti è sufficiente se il valore RTO non è eccessivo. A volte, tuttavia, viene smarrito un segmento e il destinatario riceve un numero tanto elevato di segmenti fuori sequenza da non poterli memorizzare (per la dimensione limitata del buffer). Per risolvere questo problema, la maggior parte delle implementazioni attuali osservano la regola dei *tre ACK duplicati* e ritrasmettono immediatamente

il segmento mancante. Questa funzione viene chiamata *ritrasmissione veloce*. Con questa versione quando vengono ricevuti tre ACK duplicati (ovvero l'ACK originale più tre copie identiche) di un segmento, quello successivo viene immediatamente ritrasmesso senza attendere il timeout.

Quando un segmento viene ritardato, smarrito o scartato, i segmenti che lo seguono giungono fuori sequenza (non nell'ordine atteso). La maggior parte delle implementazioni attuali del TCP non scartano i segmenti fuori sequenza, ma li memorizzano temporaneamente in attesa di ricevere il segmento mancante. Si noti tuttavia che i segmenti fuori sequenza non possono essere trasmessi al processo, dato che TCP garantisce la consegna dei dati nell'ordine corretto.

Calcolo del timeout

Il tempo di timeout (RTO) è fondamentale per il funzionamento di TCP. Deve essere maggiore di RTT (Round Trip Time): tempo trascorso da quando si invia un segmento a quando se ne riceve il riscontro. Viene calcolato analizzando gli RTT dei segmenti non ritrasmessi (Sample RTT, stimato per un segmento trasmesso - non per ogni invio).

$$RTT_{ESTIMATED} = (1 - \alpha) \cdot RTT_{ESTIMATED} + \alpha \cdot RTT_{SAMPLE}$$

RTT_{SAMPLE} può fluttuare. Si considera $RTT_{ESTIMATED}$ come la combinazione dei precedenti valori di $RTT_{ESTIMATED}$ con il nuovo valore RTT_{SAMPLE} . Il valore di α viene posto a 1/8 in modo da rendere via via meno importanti gli RTT dei pacchetti più vecchi (RFC 2988).

$$RTT_{ESTIMATED} = 0,875 \cdot RTT_{ESTIMATED} + 0,125 \cdot RTT_{SAMPLE}$$

Oltre al valore RTT stimato è necessario anche una stima della variabilità di RTT data dalla seguente formula

$$RTT_{DEV} = (1 - \beta)RTT_{DEV} + \beta|RTT_{SAMPLE} - RTT_{ESTIMATED}|$$

Stima di quanto RTT_{SAMPLE} si discosta da $RTT_{ESTIMATED}$. Il valore β viene posto a 1/4 (RFC 2988).

Una volta ottenuti questi valori, il timeout viene normalmente calcolato come

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

In molte implementazioni, dopo un errore (es. ACK non ricevuto) si raddoppia il timeout: si tratta di un primo meccanismo di controllo della congestione.

Alcuni scenari

Operatività normale Il primo scenario illustra il trasferimento di dati bidirezionali fra due sistemi. Il client TCP invia un segmento, il server TCP ne invia tre. Per il primo segmento del client e tutti e tre i segmenti del server si applica la prima regola: vi sono dati da inviare, quindi il segmento indica il prossimo byte atteso. Quando il client riceve il primo segmento dal server, non ha altri dati da inviare: invia dunque un segmento ACK. Tuttavia la seconda regola stabilisce che il riscontro sia ritardato (posticipato) di 500 ms in attesa di eventuali altri segmenti. Alla scadenza del timer viene inviata la conferma, dato che il client non può sapere se stanno arrivando altri segmenti e non può ritardare indefinitamente il riscontro. All'arrivo del segmento successivo viene inizializzato un altro timer di ACK-posticipato. Questa volta, prima che scada il tempo impostato, arriva il terzo segmento che comporta l'invio del riscontro in conformità alla terza regola. Il timer RTO non è stato considerato, poiché nessun segmento è stato smarrito o ritardato. Si assume semplicemente che il timer RTO sia gestito correttamente.

Segmento smarrito In questo scenario si illustra il caso in cui un segmento viene smarrito o corrotto. I casi di segmento smarrito o corretto vengono trattati nello stesso modo dal destinatario: il segmento smarrito viene perso da qualche parte nella rete, il segmento corrotto viene eliminato dal destinatario stesso. Entrambi vengono considerati persi. Si suppone che il trasferimento di dati sia unidirezionale: una postazione trasmette, l'altra riceve. In questo scenario il mittente invia i segmenti 1 e 2 che sono immediatamente riscontrati con un ACK (regola 3). Il terzo segmento viene smarrito. Il destinatario riceve il segmento 4 che risulta fuori sequenza: ne memorizza i dati nel suo buffer ma lascia uno spazio per indicare che non vi è continuità nei dati. Invia immediatamente un riscontro al mittente indicando il prossimo byte atteso (regola 4). Si noti che l'entità TCP destinataria consegna al processo applicativo solo dati correttamente ordinati.

Il TCP mittente mantiene un timer RTO per tutto il periodo di connessione. Alla scadenza del timer il TCP mittente ritrasmette il segmento 3, che questa volta arriva correttamente e viene riscontrato (regola 5).

Ritrasmissione veloce In questo scenario si vuole esemplificare il concetto di *ritrasmissione veloce*. Questo scenario è simile al precedente ad eccezione del valore RTO che in questo caso è superiore.

Quando il destinatario riceve un qualsiasi segmento successivo a quello atteso (in questo caso il terzo segmento) invia il riscontro indicante il segmento atteso (regola 4). Il mittente riceve quattro riscontri con il medesimo

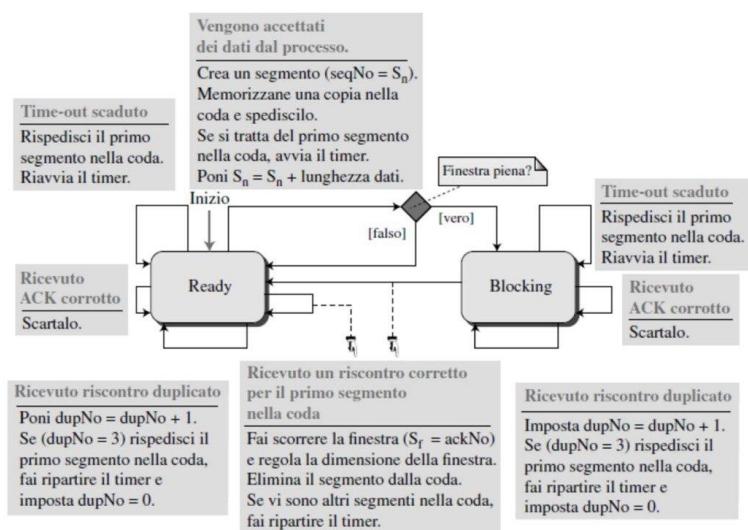
valore (tre duplicati). Sebbene il timer non sia ancora scaduto, la regola di ritrasmissione veloce richiede che il segmento 3, il segmento indicato come atteso da tutti e quattro i riscontri, sia immediatamente ritrasmesso. Una volta ritrasmesso il segmento, il timer viene fatto ripartire.

Correzione automatica di un ACK smarrito Questo scenario illustra il caso in cui un riscontro smarrito viene automaticamente rimpiazzato dal successivo. Con la tecnica di riscontro cumulativo utilizzata dal TCP, la perdita di una conferma può passare inosservata al TCP mittente: il riscontro successivo corregge automaticamente lo smarrimento del riscontro.

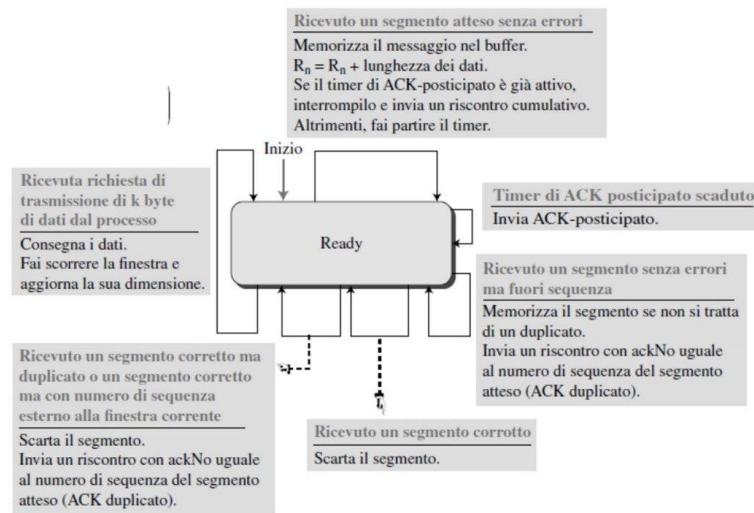
Riscontro smarrito corretto con la rispedizione del segmento Se il riscontro successivo subisce un lungo ritardo o non vi è riscontro successivo (il riscontro smarrito è l'ultimo), il timer RTO scade senza che si sia ricevuto alcun riscontro. Il mittente ritrasmette quindi il segmento, che risulta duplicato. Il destinatario ignora il segmento che riceve duplicato ma ritrasmette immediatamente l'ultimo ACK per informare il mittente che i segmenti sono arrivati a buon fine.

Si noti che viene ritrasmesso solo un segmento sebbene due segmenti non siano stati riscontrati. Il mittente, alla ricezione dell'ACK ritrasmesso, capisce che entrambi i segmenti sono stati ricevuti correttamente dato che il riscontro è cumulativo.

FSM semplificata per il mittente TCP



FSM semplificata per il lato destinatario TCP

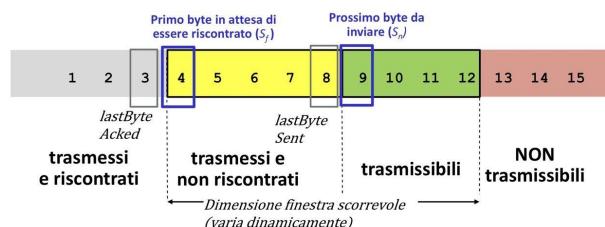


3.2.7 Finestra di trasmissione

I dati inviati dal processo a livello applicativo sono mantenuti nel buffer di invio.

La trasmissione dei dati si basa sulla finestra di trasmissione (*sliding window*).

- finestra sovrapposta sulla sequenze da trasmettere
- negoziata dinamicamente
- viene fatta avanzare alla ricezione di un ACK.



Finestra di trasmissione

3.2.8 Controllo di flusso

Il *controllo di flusso* consente di bilanciare la velocità con cui un produttore genera i dati con la velocità con la quale un consumatore li utilizza. Il protocollo TCP separa il controllo di flusso dal controllo degli errori; in questo paragrafo si descrive il controllo del flusso astraendo dal controllo degli errori: si assume dunque che il canale logico fra il mittente e il destinatario TCP sia di errori.

I dati generati dal processo applicativo mittente vengono prima passati (pushed) al TCP mittente, da questo trasmessi al TCP destinatario e infine da quest'ultimo al processo applicativo ricevente. I feedback del controllo di flusso, invece, vengono trasmessi dal TCP ricevente al TCP mittente e da questo al processo applicativo mittente. La maggior parte delle implementazioni TCP non fornisce feedback del flusso di controllo dal processo destinatario al TCP destinatario, ma consente al processo destinatario di richiedere a suo piacimento i dati dal TCP destinatario. In altri termini il TCP destinatario controlla il TCP mittente, il quale a sua volta controlla il processo mittente.

Il feedback del controllo di flusso dal TCP mittente al processo applicativo mittente è ottenuto tramite il semplice rifiuto dei dati da parte del TCP mittente quando la sua finestra è completa. La descrizione del controllo di flusso si concentra quindi sul feedback inviato dal TCP destinatario al TCP mittente.

Ogni host imposta buffer di invio e di ricezione. Il processo applicativo destinatario legge i dati dal buffer di ricezione (non necessariamente nell'istante in cui arrivano). Si intende con controllo di flusso la capacità del mittente di evitare la possibilità di saturare il buffer del ricevitore: mette in relazione la frequenza di invio del mittente con la frequenza di lettura dell'applicazione ricevente.

TCP implementa questa caratteristica tramite una variabile detta **finestra di ricezione (*rwnd*)** mantenuta nel mittente: questa variabile fornisce un'idea di quanto spazio è ancora a disposizione nel buffer del ricevitore. Tale valore è comunicato nel campo window dell'header TCP.

Spazio disponibile nel buffer del destinatario

$$rwnd = RcvBuffer - (LastByteReceived - LastByteRead)$$

rwnd è dinamica. L'host destinatario comunica la dimensione di *rwnd* al mittente.

Il mittente si assicura che

$$LastByteSent - LastByteAcked < rwnd$$

quantità di dati trasmessi e non ancora riscontrati.

N.B. se $rwnd = 0$, il mittente manda segmenti “sonda” di 1 byte per ricevere l’aggiornamento sulla dimensione di $rwnd$.

3.2.9 Controllo della congestione in TCP

Finestra di congestione

Quando si è descritto il controllo di flusso nel protocollo TCP, si è detto che la dimensione della finestra di invio è controllata dal destinatario tramite il valore $rwnd$, che viene indicato in ogni segmento trasmesso nella direzione opposta. L’impiego di questa strategia garantisce che la finestra del ricevente non venga mai sovraccaricata con i dati ricevuti. Questo tuttavia non garantisce che i buffer intermedi, i buffer nei router, non si congestionino. Un router potrebbe ricevere dati da più di un mittente. Indipendentemente dalla capacità del buffer di un router, è sempre possibile che questo venga saturato e che si trovi costretto a scartare i segmenti di qualche specifico mittente TCP. In altre parole non vi è congestione agli estremi, ma vi può essere congestione nei nodi intermedi. TCP deve preoccuparsi della possibile congestione della rete poiché la perdita di un gran numero di segmenti può influire significativamente sul meccanismo di controllo degli errori. La perdita di numerosi segmenti comporta la loro rispedizione, peggiorando così la congestione della rete e portando alla fine al collasso della comunicazione.

TCP è un protocollo end-to-end che sfrutta i servizi di IP. La congestione nei router è un problema che riguarda IP e che dovrebbe essere gestito a livello rete. Tuttavia IP è un semplice protocollo senza controllo della congestione. TCP è dunque necessariamente costretto a occuparsi del problema.

TCP non può ignorare la congestione nella rete: non può inviare un numero di segmenti eccessivo alla rete, altrimenti ne verrebbe penalizzato. Ma TCP non può nemmeno essere troppo conservativo, nel senso di inviare un numero troppo limitato di segmenti nell’unità di tempo, poiché non sfruttarebbe al meglio la capacità della rete. TCP deve quindi prevedere delle strategie per accelerare la trasmissione dei dati quando non vi è congestione e ridurre la trasmissione quando rileva la congestione.

Per controllare il numero di segmenti da trasmettere, TCP utilizza una seconda variabile chiamata $cwnd$ (*congestion window - finestra di congestione*), il cui valore dipende dal livello di congestione nella rete. Le due variabili $cwnd$ e $rwnd$ congiuntamente definiscono la dimensione della finestra di invio di TCP. La prima variabile è legata alla congestione della rete, la seconda

alla congestione delle postazioni terminali. La dimensione finale della finestra corrisponde al minimo dei due valori.

$$\text{Dimensione della finestra} = \min(rwnd, cwnd)$$

Rilevare la congestione

Prima di analizzare come il valore di *cwnd* debba essere impostato e aggiornato, è necessario descrivere come un mittente TCP possa identificare il possibile verificarsi della congestione nella rete. Il mittente interpreta come sintomi di congestione due eventi: il timeout e la ricezione di tre riscontri duplicati.

Il primo evento è dunque il *timeout*. Se un mittente TCP non riceve un riscontro per un segmento o un gruppo di segmenti prima dello scadere del timeout, suppone che tale segmento o i segmenti siano stati smarriti a causa della congestione.

Il secondo evento è la ricezione di *tre riscontri duplicati* (quattro ACK con il medesimo numero di ritorno). Si ricordi che un destinatario TCP invia un riscontro duplicato per segnalare un ritardo nella ricezione di un segmento, ma tre riscontri duplicati sono un'indicazione di segmento smarrito, che può essere dovuto alla congestione nella rete. La congestione nel caso dei tre riscontri duplicati, significa che un segmento è stato smarrito, ma anche che altri tre segmenti sono stati ricevuti: la rete è al limite della congestione o è appena rientrata da una congestione.

È interessante notare che il mittente TCP usa solo un tipo di feedback dal destinatario per rilevare la congestione: i riscontri. L'assenza della ricezione rapida e regolare dei riscontri, evidenziata dallo scadere del timeout, è il sintomo di una congestione severa; la ricezione di tre riscontri duplicati è il sintomo di una congestione debole.

Le strategie di gestione della congestione

La strategia generale del protocollo TCP per gestire la congestione si basa su tre fasi: *slow start* (partenza lenta), *congestion avoidance* (prevenzione della congestione) e *fast recovery* (recupero veloce).

Slow start: incremento esponenziale L'algoritmo *slow start* (partenza lenta) si basa sull'idea che la dimensione della finestra di congestione (*cwnd*) viene inizializzata con la dimensione del segmento massimale (MSS, Maximum Segment Size), ma viene aumentata di un valore corrispondente a MSS a ogni segmento riscontrato.

Il nome dell'algoritmo è decisamente fuorviante: l'algoritmo parte lentamente ma procede con velocità esponenzialmente crescente. Si supponga che $rwnd$ sia molto più elevato di $cwnd$, in modo tale che la finestra del mittente sia sempre uguale a $cwnd$. Si ipotizza anche che tutti i segmenti siano della stessa dimensione di MSS byte. Per semplicità si ignora anche la strategia dei riscontri in ritardo e si ipotizza che ogni segmento sia riscontrato individualmente.

Il mittente inizia con $cwnd = 1$, quindi può inviare un solo segmento. Ricevuto il primo riscontro, il segmento riscontrato viene eliminato dalla finestra di invio: nella finestra si è quindi liberato spazio per un segmento. Anche la dimensione della finestra di congestione è incrementata di 1 perché la ricezione del riscontro è un buon sintomo di assenza di congestione nella rete. La dimensione della finestra vale ora 2 ed è possibile inviare due segmenti. Quando questi vengono confermati, la dimensione della finestra viene aumentata di 1 per ciascun riscontro ricevuto, quindi $cwnd$ vale ora 4 e così via. In altri termini, la dimensione della finestra di congestione con questo algoritmo è una funzione del numero di riscontri ricevuti e può essere determinata come segue:

Se arriva un riscontro, $cwnd = cwnd + 1$

Valutando la dimensione di $cwnd$ in funzione del tempo di andata e ritorno (RTT), si trova che la velocità di crescita è esponenziale, dunque molto aggressiva, in termini di ciascun RTT:

$$\begin{array}{ll} \text{Inizio} & \rightarrow cwnd = 1 \rightarrow 2^0 \\ \text{Dopo 1 RTT} & \rightarrow cwnd = cwnd + 1 = 1 + 1 = 2 \rightarrow 2^1 \\ \text{Dopo 2 RTT} & \rightarrow cwnd = cwnd + 2 = 2 + 2 = 4 \rightarrow 2^2 \\ \text{Dopo 3 RTT} & \rightarrow cwnd = cwnd + 4 = 4 + 4 = 8 \rightarrow 2^3 \end{array}$$

La progressione dell'algoritmo non può continuare indefinitamente: è necessaria una soglia per arrestare questa fase. Il mittente tiene traccia di una variabile, chiamata *ssthresh* (*slow start threshold* o soglia di slow start): quando la dimensione della finestra espressa in byte raggiunge questa soglia, slow start termina e inizia la fase successiva.

È necessario menzionare il fatto che la progressione dell'algoritmo slow start è più lenta nel caso di riscontri posticipati. Si ricordi che $cwnd$ è incrementato di 1 per ciascun riscontro. Di conseguenza, se due segmenti vengono riscontrati cumulativamente, il valore di $cwnd$ si incrementa solo di 1, non di 2. La crescita è ancora esponenziale, ma non è una potenza di 2. Con un riscontro ogni due segmenti, diviene una potenza di 1,5.

Congestion avoidance: additive increase La dimensione della finestra di congestione aumenta esponenzialmente nella fase di slow start, ma per evitare la congestione il ritmo di crescita deve essere rallentato.

TCP prevede un altro algoritmo, chiamato *congestion avoidance*, che incrementa in modo lineare anziché esponenziale il valore di *cwnd*. Quando la dimensione della finestra di congestione raggiunge la soglia di slow start (*ssthresh*) dove *cwnd* = i , la fase di slow start si arresta e inizia la fase di incremento lineare (additive increase). Con questo algoritmo, ogni volta che viene riscontrata l'intera finestra di segmenti la dimensione della finestra di congestione viene incrementata di 1.

Il mittente inizia con *cwnd* = 4. Questo significa che il mittente può inviare solo quattro segmenti, prima di ricevere un riscontro. In questo caso, dopo che il mittente ha ricevuto i riscontri per tutti i segmenti della finestra, la dimensione della finestra viene aumentata di 1. La dimensione della finestra è quindi 5. Dopo aver inviato 5 segmenti e aver ricevuti i 5 riscontri corrispondenti, la dimensione della finestra di congestione diviene 6 e così via. Anche con questo algoritmo la dimensione della finestra di congestione è una funzione del numero di riscontri e viene determinata come segue:

$$\text{Se arrivo un riscontro, } cwnd = cwnd + (1/cwnd)$$

In altri termini, la dimensione della finestra cresce solamente di una porzione $1/cwnd$ di un MSS (in byte). Ovvero, tutti i segmenti nella finestra precedente devono essere riscontrati per aumentare la finestra di 1 MSS.

Valutando la dimensione *cwnd* in termini del tempo di andata e ritorno (RTT), si trova che la velocità aumenta in modo lineare, ovvero molto più lentamente che nello slow start.

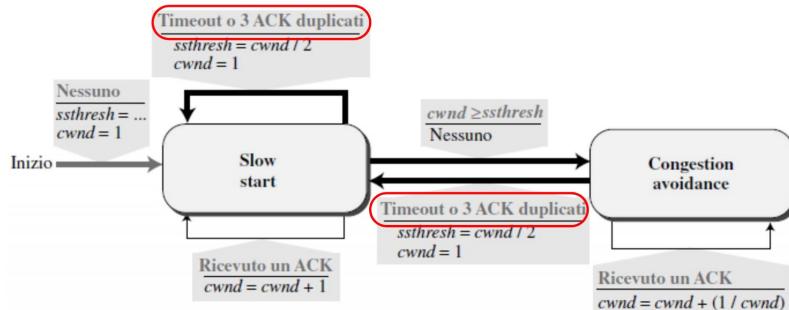
Inizio	→	$cwnd = i$
Dopo 1 RTT	→	$cwnd = i + 1$
Dopo 2 RTT	→	$cwnd = i + 2$
Dopo 3 RTT	→	$cwnd = i + 3$

Fast recovery L'algoritmo *fast recovery* (recupero veloce) è opzionale in TCP. La vecchia versione di TCP non lo utilizzava, al contrario delle nuove versioni. Esso inizia quando arrivano tre riscontri duplicati che vengono interpretati come indizio di leggera congestione nella rete. Come il congestion avoidance, anche questo algoritmo incrementa linearmente, ma incrementa la dimensione della finestra di congestione quando riceve un riscontro duplicato (dopo i tre riscontri duplicati che fanno partire questo algoritmo). Si può dire:

$$\text{Se arriva un riscontro duplicato, } cwnd = cwnd + (1/cwnd)$$

TCP Taho

La prima versione di TCP, chiamata TCP Taho, usava solo due algoritmi per la gestione della congestione: slow start e congestion avoidance. La figura illustra la FSM relativa a questa versione di TCP; è necessario premettere che si sono eliminate alcune azioni poco significative, per esempio l'incremento e l'azzeramento del numero di riscontri duplicati, per rendere la FSM più facilmente comprensibile.



TCP Taho

TCP Taho reagisce ai due sintomi di congestione, il timeout e i tre riscontri duplicati, nello stesso modo. In questa versione, una volta aperta la connessione, TCP avvia l'algoritmo *slow start* e imposta la variabile *ssthresh* a un valore pre-concordato (solitamente multiplo di MSS) e *cwnd* a 1 MSS. In questo stato, come già detto, ogni volta che arriva un riscontro la dimensione della finestra di congestione viene aumentata di 1. È evidente che si tratta di una strategia molto aggressiva che incrementa esponenzialmente la dimensione della finestra, cosa che può comportare una congestione.

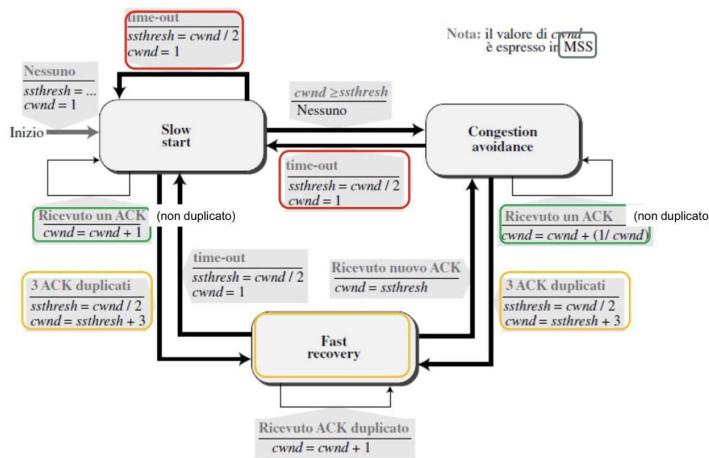
Se viene rilevata la congestione (scadenza del timeout o ricezione di tre riscontri duplicati), TCP interrompe immediatamente la crescita esponenziale, fa ripartire l'algoritmo *slow start* con un nuovo threshold uguale alla metà del valore corrente di *cwnd* e reimposta la dimensione della finestra a 1. In altri termini, non solo TCP riparte dall'inizio ma aggiorna anche il threshold. Se non viene rilevata congestione fino al raggiungimento del threshold, TCP considera di aver raggiunto il massimo delle possibili ambizioni e di non poter continuare a ingrandire la finestra a tale velocità. Si sposta quindi nello stato di *congestion avoidance*, dove la dimensione della finestra viene incrementata di 1 ogni volta che viene che riceve un numero di riscontri pari alla dimensione corrente della finestra. Si noti che non vi è un limite alla dimensione della finestra che può essere raggiunta in questo stato: la

crescita lineare continua fino alla chiusura della connessione o fino a quando viene rilevata la congestione. Se la congestione viene rilevata in questo stato, TCP reimposta il valore di $ssthresh$ alla metà del valore corrente di $cwnd$ e si sposta nuovamente nello stato *slow start*.

Sebbene in questa versione di TCP la dimensione di $ssthresh$ venga continuamente aggiornata a ogni rilevazione di congestione, essa non diviene necessariamente minore del valore precedente.

TCP Reno

Questa versione di TCP tratta i due sintomi della congestione, il timeout e la ricezione dei tre riscontri duplicati, in modo differente. Quando avviene un timeout, TCP passa allo stato *slow start* (o riparte in questo stato se già vi si trova). Quando riceve tre riscontri duplicati, TPC passa allo stato *fast recovery* e vi rimane fintantoché arrivano altri riscontri duplicati. Lo stato *fast recovery* è in un certo senso uno stato intermedio fra gli stati *slow start* e *congestion avoidance*. Si comporta come lo *slow start*, nel senso che $cwnd$ cresce esponenzialmente, ma il valore iniziale di $cwnd$ è uguale a $ssthresh$ più 3 MSS (anziché 1). Quando TCP passa nello stato *fast recovery*, possono avvenire tre eventi principali. Se continuano ad arrivare riscontri duplicati, TPC rimane nello stato ma $cwnd$ cresce esponenzialmente. Se avviene un timeout, TCP assume che vi sia una reale congestione nella rete e passa allo stato *slow start*. Se arriva un nuovo riscontro (non duplicato), TCP passa allo stato *congestion avoidance*, ma riduce il valore di $cwnd$ al valore $ssthresh$ come se i tre riscontri duplicati non fossero stati ricevuti e la transizione fosse dallo stato *slow start* allo stato *congestion avoidance*.



TCP Reno

Additive increase, multiplicative decrease La versione più diffusa di TCP è attualmente la Reno. In questa versione si è osservato che nella maggior parte dei casi la congestione viene rilevata e gestita tramite la rilevazione dei tre riscontri duplicati. Anche se vi sono alcuni eventi di timeout, TCP recupera rapidamente la situazione grazie alla crescita esponenziale. In altri termini, in una connessione TCP di lunga durata, se si ignorano gli stati *slow start* e la breve crescita esponenziale durante il *fast recovery*, la finestra di congestione TCP viene calcolata come $cwnd = cwnd + (1/cwnd)$ quando arriva un riscontro (congestion avoidance) e $cwnd = cwnd/2$ quando viene rilevata la congestione, come se lo *slow start* non esistesse e il *fast recovery* fosse ridotto a zero. La prima espressione viene chiamata *incremento additivo* (additive increase), la seconda *decremento moltiplicativo* (multiplicative decrease). La dimensione della finestra di congestione, una volta terminata la fase di slow start iniziale, segue un profilo a dente di sega chiamato *AIMD* (Additive Increase, Multiplicative Decrease).

Throughput di TCP

Il *Throughput* del protocollo TCP dipende dal comportamento della finestra di congestione. Esso è facile da calcolare se si assume che *cwnd* sia una funzione costante di RTT: con questa ipotesi poco realistica si ha $throughput = cwnd/RTT$, poiché TCP invia *cwnd* byte di dati e ne riceve i riscontri corrispondenti in un tempo RTT. Ma il comportamento di TCP non è costante: è piuttosto un profilo a denti di sega, con numerosi minimi e massimi relativi. Se i denti fossero tutti uguali, potremmo asserire che

$$throughput = [(massimo + minimo)/2]/RTT$$

Sapendo che il valore del massimo è il doppio del valore del minimo poiché a ogni rilevazione di congestione il valore *cwnd* è impostato alla metà del suo valore precedente, il throughput può essere calcolato come

$$throughput = 0,75 \cdot W_{max}/RTT$$

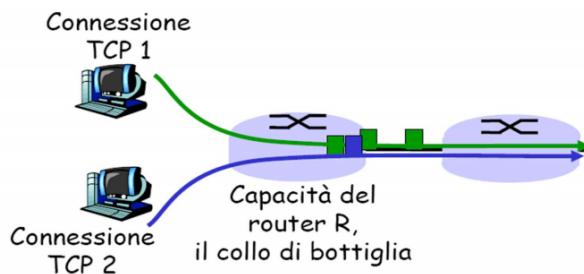
dove W_{max} è la dimensione media della finestra quando avviene la congestione.

Equità (fairness)

Ipotesi:

- K connessioni TCP insistono su un unico link di capacità R bit/s
- Le connessioni hanno gli stessi valori di MSS e RTT
- Non ci sono altri protocolli che insistono sullo stesso link

Risultato: Ognuna delle connessioni TCP tende a trasmettere R/K bit/s



3.3 Il protocollo UDP (User Datagram Protocol)

UDP (User Datagram Protocol) è un protocollo del livello trasporto inaffidabile e privo di connessione. Non aggiunge nulla ai servizi di IP eccetto la comunicazione tra processi. Per quale motivo un processo dovrebbe utilizzare UDP, se offre così pochi servizi? Gli svantaggi hanno come contropartita alcuni vantaggi. UDP è protocollo molto semplice con un overhead (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 interazioni fra il mittente e il destinatario rispetto all'impiego del TCP. Essendo meno complesso e offrendo meno servizi, è più indicato in contesti dove occorra un completo controllo della temporizzazione (applicazioni time-sensitive come la trasmissione di dati multimediali).

3.3.1 Struttura dei datagrammi utente

I pacchetti UDP sono chiamati *datagrammi utente*; la loro intestazione è costituita da 4 campi di 2 byte (16 bit) per un totale di 8 byte fissi. I primi due campi definiscono i numeri di porta di mittente e destinatario. Il terzo

campo definisce la lunghezza totale, intestazione più dati, del datagramma utente. I 16 bit possono definire una lunghezza totale da 0 a 65535 byte, ma in realtà la dimensione è inferiore perché il datagramma utente UDP viene inserito in un datagramma IP con lunghezza totale di 65535 byte. L'ultimo campo può contenere il checksum opzionale.

3.3.2 Servizi UDP

UDP fornisce il servizio di *comunicazione tra processi* utilizzando i socket address (indirizzo IP + numero di porta), ed eseguendo il *multiplexing e demultiplexing* dei pacchetti. Il concetto di numero di porta è già stato introdotto e utilizzato, ma non si è detto nulla a proposito come le porte vengano realizzate. L'idea di base, usata nel protocollo UDP, è quella di coda. Quando un processo client viene avviato richiede al sistema operativo la disponibilità di una porta. Il sistema operativo crea due code, una d'ingresso e una d'uscita, associate al processo. In alcuni sistemi operativi viene creata la sola coda d'ingresso.

Il protocollo UDP, per inviare un messaggio da un processo a un altro, *incapsula* e *decapsula* il messaggio. Tuttavia la comunicazione tra processi realizzata da UDP è *priva di connessione*, il che vuol dire che ogni datagramma viene trasmesso in modo indipendente dagli altri. Non esiste alcuna relazione tra i datagrammi anche nel caso in cui essi provengano dallo stesso processo mittente e siano indirizzati allo stesso processo destinatario. Il fatto che i datagrammi utente non vengano numerati e che la trasmissione avvenga senza apertura e chiusura di una connessione, risulta in una totale indipendenza dei percorsi seguiti dai vari datagrammi.

Una delle conseguenze della trasmissione senza connessioni è che il processo mittente non può inviare un flusso di dati al protocollo UDP e aspettarsi che questi lo suddivida in datagrammi correlati. I processi devono inviare all'UDP richieste di dimensioni sufficientemente piccole da poter essere inserite ciascuna in un singolo datagramma utente; solo i processi che usano messaggi di piccole dimensioni, inferiori a 65507 byte (65535 meno gli 8 byte per l'intestazione UDP e 20 byte per l'intestazione IP) possono utilizzare il protocollo UDP.

Il protocollo UDP è un protocollo di trasporto estremamente semplice. Non vi è alcun controllo del flusso e quindi nessun meccanismo di finestre scorrevoli: il destinatario può quindi trovarsi in situazione di congestione. L'assenza di controllo di flusso implica che siano i processi che usano UDP, se necessario, a dover fornire questo servizio.

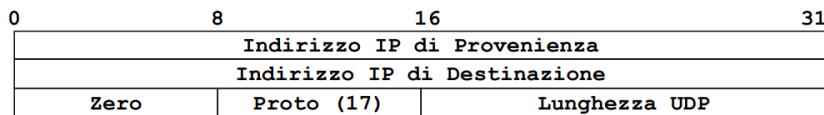
Inoltre, essendo UDP un protocollo privo di connessione, non fornisce alcun controllo della congestione. UDP è nato per applicazioni che inviano pac-

chetti in maniera sporadica e di dimensione limitate: non potrebbero quindi creare congestione nella rete. Questa ipotesi può non essere sempre verificata nelle applicazioni moderne, dove UDP viene utilizzato per il trasferimento di audio e video in tempo reale.

Riguardo al controllo degli errori, l'unico meccanismo in UDP è il checksum; al mittente, quindi, non viene notificata la perdita o l'eventuale duplicazione di un messaggio. In caso di errore evidenziato dal checksum il destinatario elimina il pacchetto ma non invia alcuna notifica. A questa carenza del protocollo UDP devono inevitabilmente sopperire i processi che lo impiegano.

Checksum

Il checksum in UDP riguarda tre parti: pseudoheader, intestazione e dati provenienti dal livello applicazione. Lo *pseudoheader* è una parte dell'intestazione del pacchetto IP in cui viene incapsulato il datagramma utente, con alcuni dei campi impostati a zero.



Pseudoheader – usato per calcolo checksum

Se il checksum non includesse lo pseudoheader, 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. Il campo protocollo permette di distinguere tra i pacchetti UDP e quelli TCP. Se i processi possono usare entrambi questi protocolli, i numeri di porta possono essere gli stessi. Il valore del campo protocollo corrispondente all'UDP è 17; se tale valore viene modificato durante la trasmissione, il calcolo del checksum evidenzia questa anomalia e UDP scarta il pacchetto. I pacchetti quindi non rischiano di essere consegnati al protocollo sbagliato.

Calcolo del checksum

- Mittente
 - Campo checksum a 0
 - Tratta il contenuto del datagramma UDP come una sequenza di parole da 16 bit

- Checksum: somma le parole di 16 bit in complemento a uno
 - Complemento a uno del risultato della somma
 - Il mittente pone il valore della checksum nel campo del checksum del datagramma UDP
- Ricevente
- Calcola la checksum del segmento ricevuto
 - Se il valore del checksum è diverso da 0 allora è stato rilevato un errore e il messaggio viene scartato.

	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
<hr/>																
a capo	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1
somma	1	0	1	1	1	0	1	1	1	0	1	1	1	0	0	0
checksum	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

Calcolo del checksum

Capitolo 4

Livello di rete

Il livello rete nello stack TCP/IP è responsabile della consegna dei datagrammi tra gli host. Questo procedimento comprende numerosi aspetti, come ad esempio la progettazione degli indirizzi logici necessari per identificare in maniera univoca gli host connessi ad Internet. Inoltre, richiede anche degli opportuni protocolli di instradamento che permettano ai pacchetti del livello di rete di giungere dalla sorgente fino alla destinazione.

4.1 Introduzione

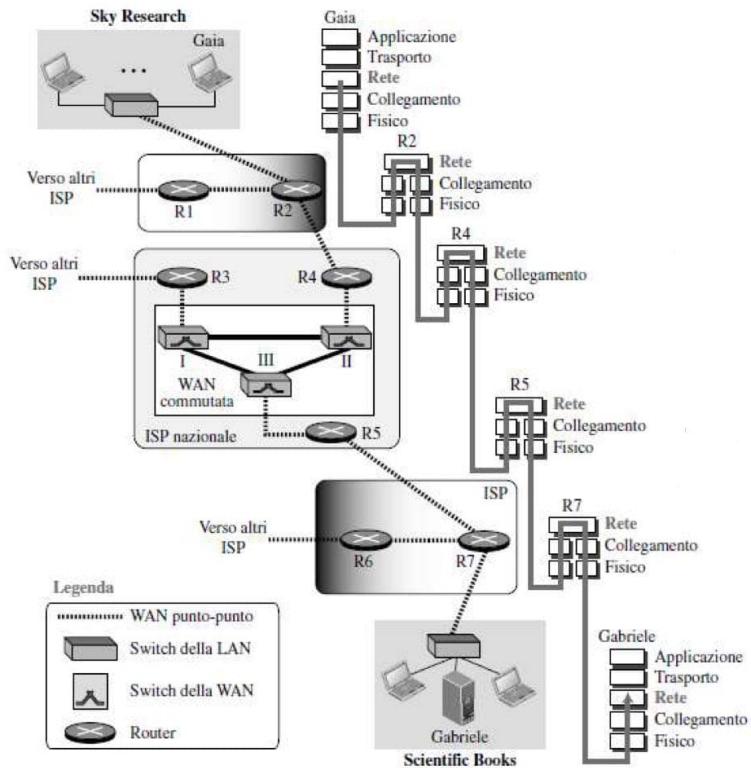


Figura 4.1: Comunicazione a livello rete

La Figura 4.1 mostra la comunicazione tra Gaia e Gabriele a livello di rete. La figura mostra come Internet sia formata da molte reti (o collegamenti) connessi tramite opportuni dispositivi. In altre parole Internet è un'internetwork (una rete di reti), cioè una combinazione di LAN e WAN. Per comprendere meglio il ruolo del livello di rete (o del livello di internetwork), dobbiamo considerare i dispositivi di interconnessione (router o switch) che connettono tra loro LAN e WAN.

Come mostra la figura, il livello di rete è presente nell'host sorgente, nell'host destinatario e in tutti i router lungo il percorso (R2, R4, R5 e R7). Nell'host sorgente (Gaia), il livello di rete accetta un pacchetto (più precisamente un segmento o un datagramma utente) dal livello trasporto, incapsula il pacchetto in un datagramma del livello rete e lo consegna al livello di collegamento. Nell'host di destinazione (Gabriele), il datagramma viene decapsulato, il pacchetto viene estratto e consegnato al livello trasporto.

Sebbene gli host sorgente e destinazione siano coinvolti in tutti e cinque i livelli della pila di protocolli TCP/IP, i router utilizzano solamente tre livelli durante l'attività vera e propria di instradamento dei pacchetti. Tuttavia i router possono aver bisogno dei livelli di trasporto ed applicazione per alcuni scopi particolari (ad esempio per coordinarsi con altri router). Un router presente lungo il percorso tra sorgente e destinazione normalmente viene rappresentato con almeno due livelli di collegamento e due livelli fisici, in quanto riceve un pacchetto da una rete e lo invia su un'altra rete.

4.1.1 Servizi a livello di rete

Suddivisione in pacchetti

Il primo compito del livello di rete è sicuramente la suddivisione dei dati in pacchetti (packetizing): alla sorgente incapsulare i dati ricevuti dal livello superiore (payload) in un pacchetto (detto anche datagramma) del livello di rete e, viceversa, alla destinazione decapsulare il payload del pacchetto. In altre parole, uno dei compiti del livello di rete è quello di portare un payload dalla sorgente alla destinazione senza modificarlo o utilizzarlo. Il livello di rete funge quindi solamente da vettore, proprio come un ufficio postale, che è responsabile della consegna dei pacchi da un mittente a un ricevente senza modificare o utilizzare il loro contenuto.

L'host sorgente riceve il payload da un protocollo di livello superiore, aggiunge un'intestazione (header) che contiene gli indirizzi sorgente e destinazione oltre ad alcune altre informazioni richieste dal protocollo del livello di rete e consegna il pacchetto al livello di collegamento. La sorgente non è autorizzata a modificare il payload a meno che esso non sia troppo grande per il trasferimento e quindi necessiti di essere suddiviso in frammenti più piccoli (frammentazione).

L'host di destinazione riceve il pacchetto di livello rete dal suo livello di collegamento, lo apre e consegna il payload al protocollo di livello superiore corrispondente. Se il pacchetto è stato frammentato alla sorgente o nei router lungo il percorso, il livello di rete deve attendere l'arrivo di tutti i frammenti di quel pacchetto, riassembrarli e solo a questo punto consegnare il payload al protocollo di livello superiore.

I router lungo il percorso non sono autorizzati ad aprire i pacchetti che hanno ricevuto a meno che tali pacchetti non debbano essere frammentati. Inoltre, i router non possono nemmeno modificare gli indirizzi sorgente e destinazione. Essi verificano semplicemente gli indirizzi allo scopo di inoltrare il pacchetto alla rete successiva lungo il percorso. Tuttavia, se un pacchetto

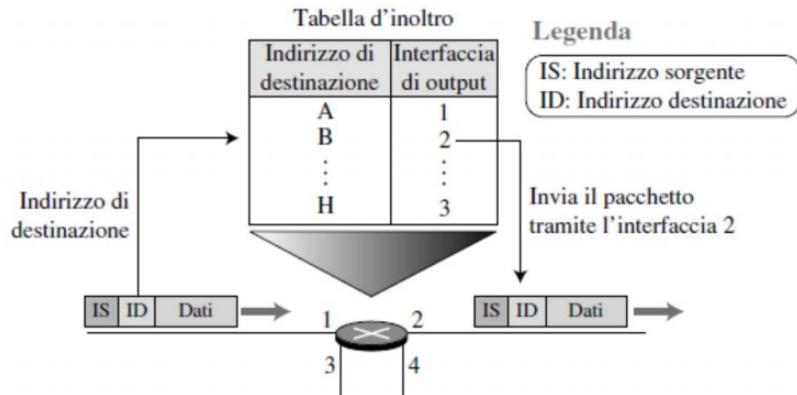
deve essere frammentato, l'intestazione sarà copiata in tutti i frammenti ma saranno anche necessari alcuni cambiamenti.

Instradamento (routing)

Un altro compito del livello di rete, tanto importante quanto il primo, è l'instradamento (*routing*). Il compito del livello di rete è quello di instradare il pacchetto dalla sua sorgente alla destinazione. Una rete fisica non è altro che la combinazione di varie reti (LAN e WAN) e dei router che le collegano. Ciò significa che, normalmente, c'è più di un percorso che va dalla sorgente alla destinazione. Il livello di rete deve trovare il *migliore* tra tali possibili percorsi. La scelta del percorso migliore deve avvenire per mezzo di opportune strategie. Nella Internet di oggi questo avviene tramite l'esecuzione di alcuni *protocolli di routing* (*protocolli di instradamento*), che hanno lo scopo di aiutare i router a condividere e coordinare la loro conoscenza sulla rete. Il risultato è la creazione di tabelle di instradamento che verranno utilizzate per decidere come instradare i pacchetti al momento del loro arrivo nei router. I protocolli d'instradamento, di cui parleremo successivamente in questo capitolo, devono essere eseguiti prima che qualsiasi altra comunicazione possa avvenire.

Inoltro (forwarding)

Se routing significa applicare strategie ed eseguire protocolli per creare tabelle di instradamento in ogni router, l'inoltro (*forwarding*) si può definire come l'azione eseguita dai router quando un pacchetto arriva ad una delle sue interfacce. La tabella d'inoltro (*forwarding table*) che viene normalmente utilizzata da un router per completare tale azione viene talvolta definita, in modo piuttosto ambiguo, *routing table*. Quando un router riceve un pacchetto da una delle reti a cui è collegato direttamente, deve inoltrare il pacchetto ad un'altra delle reti a cui è collegato (nell'instradamento unicast) o a più reti a cui è collegato (nel caso di instradamento multicast). Per prendere questa decisione il router utilizza alcune informazioni che si trovano nell'intestazione del pacchetto, ovvero l'indirizzo di destinazione o un'etichetta, per trovare la giusta interfaccia di output all'interno della tabella d'inoltro.



Controllo degli errori

Sebbene il controllo degli errori possa essere realizzato anche a livello di rete, i progettisti di tale livello in Internet hanno deciso di ignorare questo aspetto. Una delle ragioni di tale decisione è il fatto che i pacchetti a livello di rete possono essere frammentati in ciascun router, un fatto che renderebbe il controllo degli errori piuttosto inefficiente visto che dovrebbe lavorare su dati parziali.

I progettisti del livello rete, tuttavia, hanno aggiunto un campo denominato checksum nel pacchetto, al fine di controllare eventuale errori presenti sull'intestazione, ma non c'è alcuna verifica per quanto riguarda il payload. Questa checksum può rilevare cambiamenti o errori nell'intestazione del pacchetto che avvengono durante il trasferimento tra coppie di dispositivi o tra la sorgente e la destinazione.

Dobbiamo aggiungere che, nonostante il livello di rete in Internet non fornisca direttamente il controllo degli errori, Internet utilizza un protocollo ausiliario (chiamato ICMP) che fornisce una sorta di controllo degli errori nel caso un pacchetto venga scartato o contenga informazioni sconosciute nell'intestazione.

4.2 Protocolli di livello rete

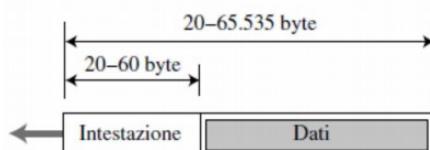
In questa sezione si vedrà come il livello di rete è implementato nella pila di protocolli TCP/IP. Nel corso del tempo i protocolli di livello rete hanno avuto molte versioni differenti. In questa sezione si vedrà la versione attuale (4), mentre nell'ultima parte del capitolo sarà brevemente trattata la nuova versione (6), che è all'orizzonte, ma ancora scarsamente utilizzata.

Nella versione 4 il livello di rete può essere visto come formato da un protocollo principale e da tre protocolli ausiliari. Il protocollo principale, l'Internet Protocol versione 4 (IPv4) è responsabile della suddivisione in pacchetti, dell'inoltro (forwarding) e della consegna (delivery) dei datagrammi a livello di rete. L'Internet Control Message Protocol versione 4 (ICMPv4) aiuta l'IPv4 a gestire alcuni errori che possono avvenire nella consegna a livello di rete. L'Internet Group Management Protocol (IGMP) viene utilizzato per supportare l'IPv4 nella gestione del multicasting. Infine, l'Address Resolution Protocol (ARP) viene usato per far interagire il livello di rete e quello di collegamento. Più in dettaglio, ARP ha il compito specifico di permettere l'associazione tra indirizzi di livello rete e indirizzi di livello di collegamento.

L'IPv4 è un protocollo inaffidabile e senza connessione, basato su datagrammi. IPv4 offre un servizio di consegna di tipo best-effort, ovvero il protocollo fa del suo meglio per consegnare i dati che sono stati spediti ma non offre alcuna garanzia. Con il termine *best-effort* si intende che i datagrammi IPv4 possono essere danneggiati, persi, arrivare fuori ordine o in ritardo e possono anche generare congestione nella rete. Se è importante l'affidabilità allora è necessario associare ad IPv4 un protocollo di livello trasporto che sia in grado di garantirla aggiungendo alcune funzionalità, come ad esempio il TCP.

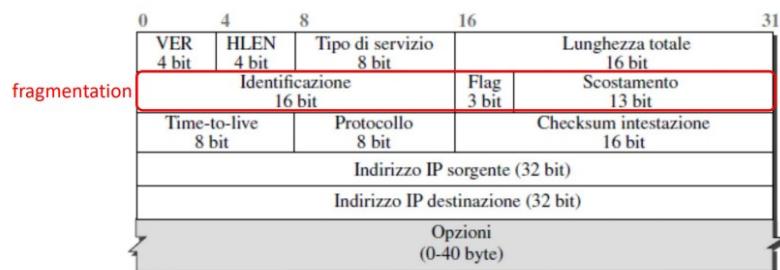
Come detto in precedenza, l'IPv4 è un protocollo senza connessione per reti a commutazione di pacchetto che utilizza un approccio basato su datagrammi. Questo significa che ogni datagramma viene gestito in modo del tutto indipendente e quindi può seguire un percorso diverso tra l'origine e la destinazione. Ciò implica che i datagrammi inviati dalla stessa sorgente alla stessa destinazione potrebbero arrivare fuori ordine. Inoltre alcuni potrebbero andare persi o essere danneggiati durante la trasmissione. Anche in questo caso IPv4 deve far affidamento su un protocollo di livello superiore (per esempio, il livello trasporto) per risolvere questi problemi.

4.2.1 Formato di datagrammi IPv4



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). L'intestazione è lunga da 20 a 60 byte e contiene le informazioni essenziali per il routing e la consegna dei datagrammi. Per comodità di lettura normalmente si rappresenta l'intestazione TCP/IP sotto forma di righe di 4 byte (32 bit) ciascuna.

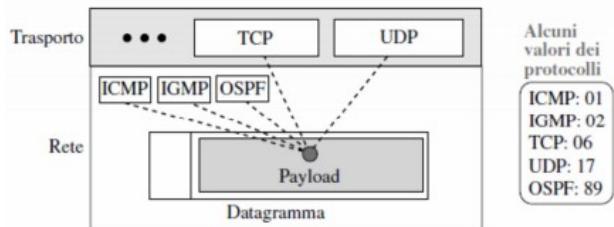


Formato dell'intestazione

- **Numero versione.** Questo campo (VER), formato da 4 bit, definisce la versione del protocollo IP. Visto che si sta trattando IPv4 ovviamente il valore sarà 4.
- **Lunghezza dell'intestazione.** Il campo lunghezza intestazione (HLEN), formato da 4 bit, definisce la lunghezza totale dell'intestazione del datagramma in parole formato da 4 byte. Il datagramma IPv4 ha un'intestazione di lunghezza variabile. Quando un dispositivo riceve un datagramma deve sapere dove finisce l'intestazione e dove iniziano i dati incapsulati nel datagramma. La lunghezza dell'intestazione, come numero di byte, deve essere rappresentata facendo uso dei soli 4 bit che formano questo campo. Per questo motivo, si ricorre ad una rappresentazione sotto forma di parole da 4 byte. Questo significa che la lunghezza totale dell'intestazione viene divisa per 4 e il valore inserito nel campo. Il ricevente ovviamente deve moltiplicare il valore di questo campo per 4 per trovare l'effettiva lunghezza totale.
- **Tipo servizio.** Secondo la progettazione originaria dell'intestazione IP, questo campo doveva specificare il tipo di servizio, ovvero il modo in cui il datagramma doveva essere gestito dai router lungo il percorso. Alla fine degli anni '90, l'IETF ha ridefinito il campo per fornire i cosiddetti *servizi differenziati*.

- **Time-to-live (TTL, tempo di vita).** A causa di alcuni malfunzionamenti dei protocolli di instradamento (che si vedranno in seguito), può accadere che un datagramma circoli su Internet, visitando alcune reti più di una volta, ma senza raggiungere la destinazione finale. Ovviamente questo porta alla creazione di traffico del tutto inutile. Il campo TTL viene usato per controllare il numero massimo di salti (hop), cioè il numero di router visitati dal datagramma. Quando un host sorgente invia il datagramma, esso memorizza un valore iniziale in questo campo. Questo valore è approssimativamente pari a due volte il numero massimo di salti tra due host qualsiasi presenti su Internet. Ogni router che elabora il datagramma decrementa di un'unità questo numero. Se il TTL, dopo essere stato diminuito, giunge a zero allora il router scarta il datagramma.
- **Lunghezza totale.** Questo campo, da 16 bit, definisce la lunghezza totale (intestazione più dati), espressa in byte, del datagramma IP. Un campo di 16 bit può rappresentare una lunghezza massima di 65.535 byte (nel caso in cui tutti i bit siano impostati a 1). Tuttavia la dimensione dei datagrammi normalmente è molto inferiore a questo massimo. Questo campo è utile al dispositivo ricevente per determinare se il pacchetto in ricezione è arrivato completamente.
- **Identificazione, flag e scostamento di frammentazione (offset).** Questi tre campi riguardano la frammentazione dei datagrammi IP che avviene quando la loro dimensione è maggiore rispetto a quella che la tecnologia di livello collegamento sottostante è in grado di trasportare.
- **Protocollo.** In TCP/IP, la sezione relativa ai dati di un pacchetto, chiamata *payload*, incapsula l'intero pacchetto di un altro protocollo (solitamente di livello superiore). Un datagramma, per esempio, può portare un pacchetto che appartiene a un protocollo di livello trasporto come UDP (quindi un datagramma utente) o TCP (quindi un segmento). Un datagramma può anche portare un pacchetto di un altro protocollo che usa direttamente il servizio offerto da IP, come ad esempio alcuni protocolli di instradamento o alcuni protocolli ausiliari. Gli enti di gestione Internet hanno assegnato ad ogni protocollo che fa uso del servizio IP un numero univoco composto da 8 bit, numero che viene di volta in volta inserito nel campo protocollo all'interno dell'intestazione. Quando il payload viene incapsulato in un datagramma nella sorgente, il numero di protocollo corrispondente viene inserito in questo campo; quando il datagramma arriva a destinazione, il valore di

questo campo permette di definire a quale protocollo va consegnato il payload estratto dal datagramma. In altre parole, questo campo fornisce il multiplexing alla sorgente e il demultiplexing a destinazione. È bene notare che il campo protocollo a livello di rete ha un ruolo simile ai numeri di porta a livello trasporto.



Multiplexing e demultiplexing utilizzando il valore del campo protocollo

- **Checksum dell'intestazione.** IP è un protocollo non affidabile, ad esempio non controlla se i dati incapsulati in un datagramma vengono danneggiati durante la trasmissione. IP lascia l'onere del controllo degli errori nei dati trasmessi al protocollo che è proprietario del payload, come ad esempio UDP o TCP. L'intestazione del datagramma, tuttavia, viene aggiunta dall'IP e il controllo degli errori è quindi una sua responsabilità. Gli errori nell'intestazione IP possono essere un grosso problema. Se l'indirizzo IP di destinazione è danneggiato, il pacchetto sarà consegnato all'host sbagliato. Se il campo protocollo è danneggiato, il payload sarà consegnato al protocollo errato. Se i campi relativi alla frammentazione sono danneggiati, il datagramma non sarà riassemblato correttamente a destinazione, e così via. Per tali ragioni IP aggiunge un campo checksum che riguarda la sola intestazione ma non il payload. È bene ricordare che, siccome il valore di alcuni campi, come TTL, la frammentazione e le opzioni, possono cambiare da router a router, il checksum deve essere ricalcolato in ogni router. Il checksum, che viene inserito in un campo da 16 bit, è ottenuto come il complemento della somma degli altri campi dell'intestazione, il tutto utilizzando l'aritmetica complemento a uno.
- **Indirizzi sorgente e destinazione.** Questi campi, lunghi 32 bit, definiscono rispettivamente l'indirizzo IP della sorgente e quello della destinazione. L'host sorgente dovrebbe conoscere a priori il suo indirizzo IP. L'indirizzo IP di destinazione invece è noto al protocollo che utilizza il servizio offerto da IP oppure viene ottenuto per mezzo del

DNS. Da notare che il valore di questi campi deve rimanere immutato durante tutto il tragitto del datagramma IP dell'host sorgente a quello di destinazione.

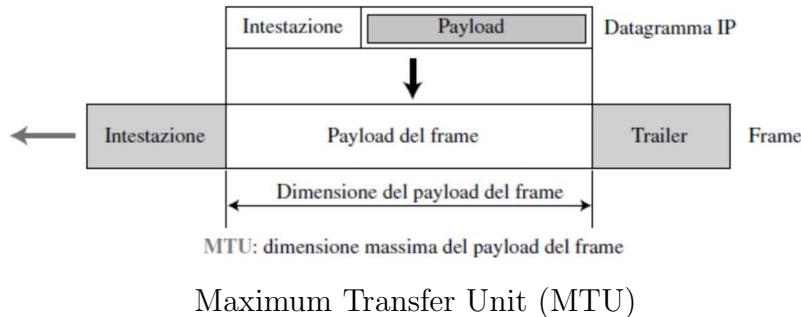
- **Opzioni.** L'intestazione di un datagramma può avere fino a 40 byte di opzioni. Le opzioni possono essere usate per il test o il debug della rete.
- **Payload (dati).** I dati trasportati ovviamente sono la ragione principale per la creazione del datagramma. Il payload è il pacchetto che deriva dagli altri protocolli che usano il servizio dell'IP, siano questi di livello superiore (per esempio, TCP o UDP) o di pari livello (per esempio, ICMP) nello stack TCP/IP.

Frammentazione

Per giungere alla destinazione, un datagramma IP può dover viaggiare attraverso varie reti, ognuna con caratteristiche differenti. Ogni router toglie il datagramma dal frame di livello collegamento che ha ricevuto, lo elabora e poi lo incapsula in un nuovo frame. Il formato e la dimensione del frame ricevuto dipendono dalla tecnologia fisica e da quella di livello di collegamento utilizzati per trasportare il frame. Viceversa, il formato e la dimensione del frame inviato dipendono dalla tecnologia fisica e da quella di livello di collegamento della rete su cui il frame verrà inviato. Per esempio se un router collega una LAN a una WAN, riceverà un frame in formato LAN e invierà un nuovo frame in formato WAN.

Maximum Transfer Unit (MTU) 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. In altre parole, quando un datagramma è racchiuso in un frame, la dimensione totale del datagramma deve essere inferiore rispetto a questa misura massima, che è definita dalle restrizioni imposte dall'hardware e dal software usati nella rete.

Per rendere il protocollo IP indipendente dai livelli sottostanti, i progettisti hanno deciso di fissare la lunghezza massima del datagramma IP a 65.535 byte. Questo rende la trasmissione efficiente nel caso si usi un protocollo di livello collegamento con una MTU di tali dimensioni. Tuttavia, altre tecnologie di rete richiedono che il datagramma sia suddiviso in parti più piccole per poter essere trasportato. Questo procedimento viene chiamato *frammentazione*.



Quando un datagramma IP viene frammentato ogni frammento ha la propria intestazione, nella maggior parte dei casi i campi dell'intestazione sono uguali rispetto al datagramma originario ma alcuni devono essere cambiati. Un datagramma frammentato può essere a sua volta frammentato se incontra una rete con un MTU ancora più piccola. In altre parole un datagramma può essere frammentato più volte prima di raggiungere la sua destinazione finale.

Un datagramma può essere frammentato dall'host sorgente o da qualsiasi altro router lungo il percorso verso la destinazione. Il *riasssemblaggio* del datagramma, tuttavia, viene fatto solo dall'host di destinazione, in quanto ogni frammento diventa un datagramma indipendente.

I diversi frammenti del datagramma originario possono viaggiare lungo percorsi differenti; questi percorsi non sono prevedibili e controllabili a priori ma, alla fine, tutti i frammenti che appartengono allo stesso datagramma dovrebbero arrivare all'host di destinazione finale. Oltretutto, il riassemblaggio dei pacchetti durante il percorso provocherebbe una significativa perdita di efficienza.

Quando parliamo di frammentazione non intendiamo solamente che il payload del datagramma IP viene frammentato. Infatti, la maggior parte dell'intestazione, ad eccezione di alcune opzioni, deve essere copiata nell'intestazione di tutti i frammenti. L'host o il router che frammenta un datagramma IP deve cambiare i valori di tre campi dell'intestazione: i flag, lo scostamento di frammentazione e la lunghezza totale. Il resto dei campi dell'intestazione deve invece essere solamente copiato. Naturalmente il valore del checksum va ricalcolato ad ogni hop indipendentemente dalla frammentazione.

Campi relativi alla frammentazione Abbiamo detto in precedenza che tre campi nell'intestazione del datagramma IP si riferiscono alla frammentazione: *identificazione*, *flag* e *scostamento di frammentazione (offset)*.

Il campo *identificazione*, da 16 bit, identifica un datagramma che ha origine da un host sorgente. La combinazione tra indirizzo IP sorgente e campo

identificazione deve definire in modo unico un datagramma quando lascia l'host sorgente. Per garantire l'unicità, il protocollo IP utilizza un contatore per etichettare i datagrammi. Il contatore viene inizializzato a un numero positivo. Quando il protocollo IP invia un datagramma, copia il valore corrente del contatore nel campo identificazione ed incrementa il contatore di uno. Quando un datagramma viene frammentato il valore del campo identificazione viene copiato in tutti i frammenti ottenuti. In altre parole tutti i frammenti hanno lo stesso numero di identificazione, che è anche lo stesso del datagramma originale. Il numero d'identificazione aiuta la destinazione a riassemblare il datagramma. Esso sa che tutti i frammenti con lo stesso valore d'identificazione vanno assemblati in un unico datagramma IP.

Il *campo flag*, da 3 bit, in realtà definisce tre flag distinti, ognuno composto da un singolo bit. Il bit all'estrema sinistra è riservato (non viene attualmente usato). Il secondo bit è definito *do not fragment* (non frammentare). Se il suo valore è 1 allora il dispositivo non deve frammentare il datagramma. Se non è in grado di trasmettere il datagramma attraverso nessuna delle reti fisiche allora lo scarta ed invia un messaggio d'errore ICMP all'host sorgente. Se il suo valore è 0, se necessario, il datagramma può essere frammentato. Il terzo bit detto *more fragments*, nel caso sia impostato a 1 indica che questo datagramma non è l'ultimo frammento della serie e che ci sono altri frammenti dopo di lui. In altre parole non si tratta dell'ultimo frammento di un datagramma originario. Se il suo valore è 0 significa che è l'ultimo (o l'unico) frammento.

Il *campo scostamento di frammentazione (offset)*, da 13 bit, mostra la posizione relativa di questo frammento rispetto all'intero datagramma. È l'offset dei dati nel datagramma originario misurato in unità di 8 byte. Questo avviene perché la lunghezza del campo offset è di soli 13 bit e non può rappresentare una sequenza di byte maggiore di 8191. Ciò costringe gli host o i router che frammentano i datagrammi a scegliere la dimensione dei frammenti in modo che sia divisibile per 8, ad esclusione ovviamente dell'ultimo frammento originato dalla suddivisione di un datagramma.

4.2.2 Indirizzi IPv4

L'identificatore usato dal protocollo IP nella pila TCP/IP per individuare il collegamento di ciascun dispositivo ad Internet è chiamato indirizzo Internet (Internet Address) o indirizzo IP. Un indirizzo IPv4 è un indirizzo formato da 32 bit che definisce in modo unico e universale il collegamento di un host o un router ad Internet. L'indirizzo IP è l'indirizzo del collegamento, non dell'host o del router, in quanto se il dispositivo viene spostato in un'altra rete, molto probabilmente, l'indirizzo IP verrà cambiato.

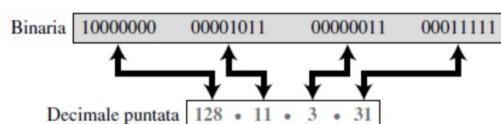
Gli indirizzi IP sono unici nel senso che ogni indirizzo definisce uno e un solo collegamento ad Internet. Se un dispositivo ha due collegamenti ad Internet, ad esempio tramite due reti diverse, allora avrà due indirizzi IPv4 distinti. Gli indirizzi IPv4 sono universali nel senso che ciascun host che vuole collegarsi a Internet deve necessariamente farne uso.

Spazio degli indirizzi (address space)

Un protocollo come l'IPv4 che utilizza degli indirizzi ha bisogno di definire uno spazio di indirizzi, cioè il numero totale di indirizzi utilizzati dal protocollo. Se un protocollo usa b bit per rappresentare un indirizzo, lo spazio degli indirizzi è pari a 2^b perché ogni bit può assumere solo due valori distinti (0 o 1). IPv4 usa indirizzi a 32 bit, il che significa che lo spazio degli indirizzi è 2^{32} o 4.294.967.296 (più di quattro miliardi). Se non ci fosse alcuna restrizione, oltre 4 miliardi di dispositivi potrebbero essere collegati a Internet.

Notazione

Esistono tre notazioni comunemente usate per rappresentare gli indirizzi IPv4: la notazione binaria (base 2), la notazione decimale puntata (base 256) e quella esadecimale (base 16). Nella *notazione binaria* un indirizzo IPv4 viene rappresentato con 32 bit. Per facilitare la lettura dell'indirizzo normalmente vengono inseriti uno o più spazi tra ogni ottetto (8 bit). Spesso ci si riferisce a ciasun ottetto come ad un byte. Per rendere l'indirizzo IPv4 più compatto e facile da leggere normalmente viene scritto in forma decimale, con un punto che separa i byte. Questo formato viene definito *notazione decimale puntata (dotted-decimal notation)*. Da notare che siccome ogni byte (ottetto) è formato da soli 8 bit, ogni numero della notazione decimale puntata è compreso tra 0 e 255. A volte è possibile trovare un indirizzo IPv4 espresso in notazione esadecimale. In questo caso ogni cifra esadecimale equivale a quattro bit. Ciò significa che un indirizzo a 32 bit ha 8 cifre esadecimale. Tale notazione viene spesso utilizzata nella programmazione di rete.

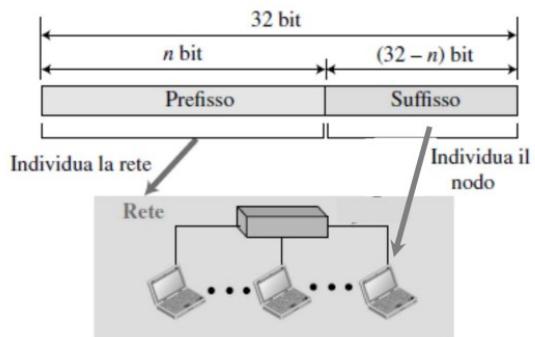


Due diverse notazione per l'indirizzamento IPv4

Gerarchia nell'indirizzamento

In ogni rete di comunicazione in cui è prevista la consegna di un messaggio da un mittente ad un destinatario, come una rete telefonica o una rete postale, il meccanismo di indirizzamento è gerarchico. In una rete postale, l'indirizzo di spedizione comprende la nazione, la città, la via, il numero civico e il nome del destinatario. allo stesso modo un numero telefonico è diviso in codice nazione, prefisso e numero identificativo.

Anche gli indirizzi IPv4 sono gerarchici, ma divisi in sole due parti. La prima parte dell'indirizzo, chiamata *prefisso* (*prefix*), identifica la rete, mentre la seconda parte dell'indirizzo, chiamato *suffisso* (*suffix*), identifica il nodo nella rete (o meglio il collegamento di un dispositivo a Internet).



Gerarchia nell'indirizzamento

Un prefisso può avere una lunghezza fissa o variabile. Il sistema di identificazione delle reti in IPv4 è stato inizialmente progettato come prefisso a lunghezza fissa. Questo schema, che ormai è obsoleto, viene indicato come indirizzamento con classi (*classful addressing*). Il nuovo schema, chiamato indirizzamento senza classi (*classless addressing*), usa un prefisso di rete di lunghezza variabile.

Indirizzamento con classi

Come detto in precedenza, inizialmente l'indirizzamento IPv4 era stato progettato con un prefisso di lunghezza fissa, ma vista la necessità di supportare sia reti piccole che grandi, al posto di una sola lunghezza di prefisso ne erano state previste 3 ($n = 8$, $n = 16$ e $n = 24$). L'intero spazio degli indirizzi era stato diviso in cinque classi (classe A, B, C, D ed E). Questo schema è ora definito come *indirizzamento con classi* (*classful addressing*).



Suddivisione dello spazio degli indirizzi nell'indirizzamento con classi

Nella classe A la lunghezza della parte di rete è di 8 bit, ma siccome il primo bit, che è 0, identifica il tipo di classe, possiamo avere solo sette bit per l'identificazione delle reti. Ciò significa che ci sono $2^7 = 128$ reti al mondo che possono avere un indirizzo di classe A.

Nella 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.

Tutti gli indirizzi che iniziano con $(110)_2$ appartengono alla classe C, nella quale la lunghezza della parte di rete è di 24 bit, ma siccome tre bit definiscono la classe, possiamo avere solo 21 bit per identificare le reti. Ciò significa che ci sono $2^{21} = 2.097.152$ reti al mondo che possono avere un indirizzo di classe C.

Gli indirizzi che iniziano con $(1110)_2$ appartengono alla classe D, una classe che non è divisa in prefisso e suffisso ed è usata per gli indirizzi di tipo multicast. Infine, la classe E raccoglie tutti gli indirizzi che iniziano con $(1111)_2$. Come per la classe D, la classe E non è divisa in prefisso e suffisso ed è riservata per uso futuro.

Esaurimento degli indirizzi La ragione per cui l'indirizzamento con classi è diventato obsoleto è l'esaurimento degli indirizzi. Siccome gli indirizzi non sono stati assegnati in modo appropriato, Internet ha dovuto affrontare il problema dell'esaurimento degli indirizzi disponibili per società ed individui che volevano collegarsi in rete. Per comprendere meglio il problema pensiamo alla classe A. Tale classe può essere assegnata solo a 128 organizzazioni al mondo, ma a causa della struttura della classe A ognuna deve avere un singola rete con 16.777.216 nodi (computer collocati in questa unica rete). Siccome ci sono ben poche organizzazioni così grandi, la maggior parte degli indirizzi

in questa classe è andata sprecata (gran parte degli indirizzi erano assegnati ma rimanevano inutilizzati). Gli indirizzi di classe B sono stati progettati per le organizzazioni di media dimensione, ma anche in questo caso molti degli indirizzi sono rimasti inutilizzati. Gli indirizzi di classe C hanno un difetto di progettazione completamente diverso. Il numero di indirizzi disponibili in ogni rete (256) era talmente piccolo che la maggiore parte delle organizzazioni non lo trovavano adeguato per le proprie esigenze. Infine, gli indirizzi di classe E erano riservati per uso futuro, sprecando l'intera classe.

Subnetting e supernetting Per mitigare il problema dell'esaurimento degli indirizzi sono state proposte due strategie che, in certa misura, sono state anche implementate: il subnetting e il supernetting. Nel subnetting un blocco di classe A o B viene diviso in varie sottoreti (subnet). Ogni subnet ha un prefisso di lunghezza maggiore rispetto alla rete originale. Per esempio se una rete in classe A viene suddivisa in quattro subnet, ogni subnet ha un prefisso di $n_{\text{sub}} = 10$. Il vantaggio è che se in una rete non vengono utilizzati tutti gli indirizzi, il subnetting consente di dividere gli indirizzi disponibili tra varie organizzazioni. Questa idea non ha funzionato in quanto la maggior parte delle grandi organizzazioni non era disponibile a dividere il proprio blocco di indirizzi per darne alcuni ad organizzazioni più piccole.

Mentre il subnetting è stato ideato per dividere un grande blocco di indirizzi in blocchi più piccoli, il supernetting è stato ideato per combinare numerosi blocchi di classe C in un blocco più grande, che potesse soddisfare organizzazioni per le quali un blocco di classe C (256) era troppo piccolo. Questa idea nella pratica non ha funzionato perché complicava il routing dei pacchetti.

Vantaggi dell'indirizzamento con classi Anche se l'indirizzamento con classi presentava numerosi problemi ed è diventato obsoleto, aveva un vantaggio: una volta individuato un indirizzo IP, si poteva facilmente risalire alla classe dell'indirizzo e, siccome la lunghezza del prefisso di ogni classe è fissa, scoprire immediatamente la lunghezza del prefisso. In altre parole, la lunghezza del prefisso nell'indirizzamento con classi fa parte dell'indirizzo stesso, non c'è quindi la necessità di aggiungere altre informazioni per determinare la parte di prefisso e quella di suffisso.

Indirizzamento senza classi

Il subnetting e il supernetting nell'indirizzamento con classi non erano realmente in grado di risolvere il problema dell'esaurimento degli indirizzi. Con

la crescita di Internet era chiaro che, come soluzione a lungo termine, sarebbe servito uno spazio degli indirizzi più grande. Tuttavia, questa soluzione richiede necessariamente che la lunghezza degli indirizzi IP venga aumentata, il che significa anche che il formato dei datagrammi IP deve essere modificato. Anche se la soluzione a lungo termine è già stata ideata ed implementata (IPv6), è stata studiata anche una soluzione a breve termine. In questa soluzione si continua a usare lo stesso spazio degli indirizzi ma si cambia la loro distribuzione, attraverso un approccio più equo. La soluzione a breve termine è ancora basata sugli indirizzi IPv4 ma non fa più uso della divisione in classi. In altre parole la suddivisione in classi è stata rimossa per compensare la scarsità di indirizzi disponibili.

C'è stata un'altra motivazione che ha favorito l'indirizzamento senza classi. Negli anni '90 sono emersi gli Internet Service Provider (ISP). Un ISP è un'organizzazione che fornisce accesso Internet a individui, piccole imprese e aziende di medie dimensioni che non vogliono entrare direttamente a far parte di Internet e fornire a loro volta servizi Internet. Sono aziende che per semplicità vogliono solamente utilizzare i servizi messi a disposizione da un ISP. Un ISP ottiene quindi un insieme piuttosto ampio di indirizzi e lo suddivide in gruppi di indirizzi (con dimensione 1, 2, 4, 8, 16, etc.), fornendo questi gruppi di indirizzi ai suoi clienti a seconda delle loro tipologie ed esigenze.

Nel 1996 gli enti di gestione di Internet hanno annunciato una nuova struttura chiamata *indirizzamento senza classi*. In questa forma di indirizzamento vengono utilizzati dei blocchi di lunghezza variabile che non appartengono a nessuna classe. Possiamo avere un blocco da 1, 2, 4, 128 indirizzi e così via.

Nell'indirizzamento senza classi l'intero spazio degli indirizzi è diviso in blocchi di lunghezza variabile. Il prefisso in un indirizzo definisce il blocco (individua la rete): il suffisso definisce il nodo (individua il dispositivo). In teoria possiamo avere un blocco di 2^0 , 2^1 , 2^2 , ..., 2^{32} indirizzi. Una delle restrizioni, di cui parleremo più avanti, è che il numero degli indirizzi in un blocco deve essere una potenza di 2. Ad un'organizzazione si può quindi assegnare un blocco di indirizzi di dimensione adeguata alle sue esigenze.

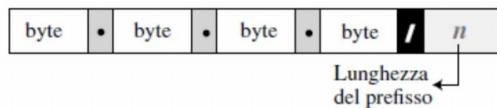
A differenza dell'indirizzamento con classi, la lunghezza del prefisso nell'indirizzamento senza classi è variabile. Possiamo avere una lunghezza del prefisso che va da 0 a 32 bit. La dimensione della parte di rete è inversamente proporzionale alla lunghezza del prefisso. Un prefisso piccolo implica una rete più grande (con molti nodi al suo interno); un prefisso grande implica una rete più piccola (con pochi nodi).

Bisogna sottolineare che l'idea dell'indirizzamento senza classi si può applicare facilmente all'indirizzamento con classi. Un indirizzo di classe A si può pensare come un indirizzo senza classi nel quale la lunghezza del prefisso è di 8 bit. Un indirizzo di classe B è un indirizzo senza classi con prefisso di

16 bit e così via. In altre parole l'indirizzamento con classi è un caso speciale dell'indirizzamento senza classi.

Lunghezza del prefisso: notazione slash (barra) La prima domanda cui dobbiamo rispondere nell'indirizzamento senza classi è come trovare la lunghezza del prefisso di un determinato indirizzo. Siccome la lunghezza del prefisso non è più parte integrante dell'indirizzo è necessario aggiungere questa informazione in modo separato. In questo caso la lunghezza del prefisso, n , viene aggiunta all'indirizzo separata da una barra (slash). La numerazione viene informalmente definita notazione slash e formalmente *classless interdomain routing (CIDR)*.

In altre parole, un indirizzo nell'indirizzamento senza classi non è in grado di definire da solo il blocco (o la rete) a cui appartiene, è necessario indicare esplicitamente anche la lunghezza del prefisso.

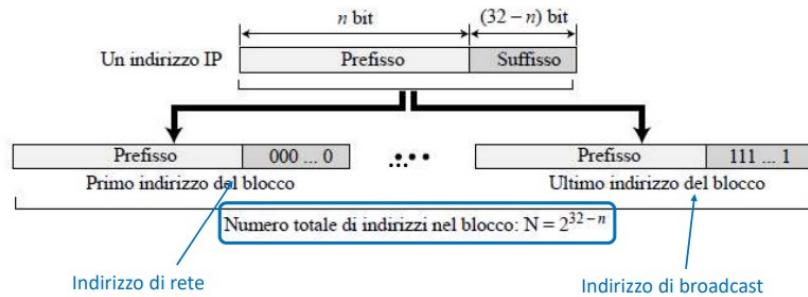


Esempi:
12.24.76.8/8
23.14.67.92/12
220.8.24.255/25

Numerazione slash (CIDR).

Estrazione delle informazioni da un indirizzo Dato un indirizzo IP appartenente ad un blocco di indirizzi, normalmente vogliamo ricavare tre informazioni riguardo il suo blocco di appartenenza: il numero di indirizzi che contiene, il primo indirizzo del blocco e l'ultimo. Siccome la lunghezza del prefisso, n , è nota, possiamo facilmente ottenere queste tre informazioni.

1. Il numero di indirizzi nel blocco è dato da $N = 2^{32-n}$.
2. Per trovare il primo indirizzo, teniamo invariati i primi n bit partendo da sinistra e impostiamo a 0 tutti i bit restanti a destra (sono $32 - n$).
3. Per trovare l'ultimo indirizzo, teniamo invariati i primi n bit partendo da sinistra e impostiamo a 1 tutti i bit restanti a destra (anche in questo caso sono $32 - n$).



Estrazione delle informazioni nell’indirizzamento senza classi.

Maschera dell’indirizzo (address mask) Un altro modo per trovare il primo e l’ultimo indirizzo del blocco è usare la maschera dell’indirizzo (address mask), cioè un numero composto da 32 bit in cui i primi n bit a sinistra sono impostati a 1 e il resto dei bit $(32 - n)$ sono impostati a 0. Il motivo principale per utilizzare una notazione di questo tipo è che può essere usata da un programma per calcolare in modo efficiente le informazioni di un blocco, usando solamente tre semplici operatori sui bit: NOT, AND e OR.

1. Il numero degli indirizzi nel blocco è $N = \text{NOT}(\text{maschera}) + 1$;
2. Il primo indirizzo nel blocco = (qualsiasi indirizzo del blocco) **AND** (maschera)
3. L’ultimo indirizzo nel blocco = (qualsiasi indirizzo del blocco) **OR** (**NOT**(maschera)).

Indirizzo di rete (network address)

Il primo indirizzo, detto *indirizzo di rete (network address)*, è particolarmente importante in quanto è usato nell’instradamento dei datagrammi verso la rete di destinazione. Per il momento supponiamo che Internet sia composta da m reti e da un router con m interfacce. Quando un datagramma giunge al router da un qualsiasi host sorgente, il router deve sapere a quale rete va inviato il datagramma e quindi da quale interfaccia va inviato. Quando il datagramma arriva alla rete di destinazione, raggiungerà l’host di destinazione utilizzando un’altra strategia che vedremo solo in seguito. Dopo che il network address è stato trovato, il router consulta la sua tabella d’inoltro per trovare l’interfaccia corrispondente dalla quale inviare il datagramma. L’indirizzo di rete in realtà è lo strumento per l’identificazione delle reti: ogni rete è identificata per mezzo del suo network address.

Assegnazione dei blocchi di indirizzi

Il problema successivo nell’indirizzamento senza classi è rappresentato dall’assegnazione dei blocchi. La responsabilità dell’assegnazione ricade su un ente globale chiamato Internet Corporation of Assigned Names and Numbers (ICANN). L’ICANN non assegna indirizzi a singoli utenti Internet, si occupa invece di assegnare grandi blocchi di indirizzi a ISP (o organizzazioni comparabili per dimensione ad un ISP). Per il corretto funzionamento del CIDR devono essere applicate due restrizioni al blocco assegnato.

1. Il numero degli indirizzi assegnati, N , deve essere una potenza di 2. La ragione è che $N = 2^{32-n}$ o $n = 32 - \log_2 N$. Se N non è una potenza di 2, il valore di n non sarà intero.
2. Quando c’è una richiesta per un blocco di indirizzi di una certa dimensione, il blocco assegnato deve essere uno spazio libero contiguo nello spazio degli indirizzi. Tuttavia c’è una restrizione nella scelta del primo indirizzo nel blocco. Il primo indirizzo deve essere divisibile per il numero degli indirizzi nel blocco. La ragione è che il primo indirizzo deve essere composto dal prefisso seguito da $(32 - n)$ zeri. Il valore decimale del primo indirizzo quindi è

$$\text{Primo indirizzo} = (\text{prefisso in decimale}) \cdot 2^{32-n} = (\text{prefisso in decimale}) \cdot N$$

Subnetting Più livelli di gerarchia possono essere creati utilizzando il subnetting. Un’organizzazione (o un ISP) a cui viene assegnato un blocco di indirizzi può dividerlo in vari sottoblocchi ed assegnare ognuno di questi ad una sottorete (subnet). Da notare che nulla vieta all’organizzazione di creare più livelli. Una sottorete può essere divisa a sua volta in varie sotto-reti. E così via anche per le sottoreti.

All’interno di una rete, le sottoreti vanno progettate con attenzione, in modo da permettere l’instradamento dei datagrammi. Supponiamo che il numero totale degli indirizzi assegnato ad un’organizzazione sia N , la lunghezza del prefisso sia n , il numero di indirizzi assegnato ad ogni sotto rete sia N_{sub} e la lunghezza del prefisso di ogni sottorete sia n_{sub} . Per un funzionamento corretto delle sottoreti è necessario seguire questi passi:

1. il numero degli indirizzi in ogni subnetwork deve essere una potenza di 2;
2. la lunghezza del prefisso di ogni sottorete va calcolata secondo la formula:

$$n_{sub} = 32 - \log_2 N_{sub}$$

3. l'indirizzo di partenza di ogni sottorete dovrebbe essere divisibile per il numero di indirizzi presenti in quella sottorete. Ciò si può ottenere assegnando come prima cosa gli indirizzi alle sottoreti più grandi.

Dopo aver progettato le sottoreti, le informazioni relative ad ogni sottorete, come il primo e l'ultimo indirizzo, si possono trovare usando il procedimento descritto in precedenza.

Aggregazione degli indirizzi Uno dei vantaggi della strategia CIDR è la possibilità di *aggregare gli indirizzi* (address aggregation, address summarization, route summarization). Quando alcuni blocchi d'indirizzi vengono combinati per creare un blocco più grande, l'instradamento può essere effettuato sulla base del prefisso del blocco combinato. L'ICANN assegna un grande blocco di indirizzi a un ISP, l'ISP suddivide il blocco in sottoblocchi più piccoli e li assegna ai propri clienti.

Indirizzi speciali Prima di terminare l'argomento degli indirizzi IPv4 dobbiamo parlare di cinque indirizzi speciali che vengono usati per scopi particolari: l'indirizzo *this-host*, l'indirizzo *limited-broadcast*, l'indirizzo *loopback*, gli indirizzi *privati* e quelli *multicast*.

- **Indirizzo *this-host*.** L'unico indirizzo del blocco **0.0.0.0/32** viene chiamato indirizzo *this-host* (*questo host*). Viene usato ogni volta che un host ha la necessità di inviare un datagramma IP ma non conosce il proprio indirizzo da usare come indirizzo sorgente.
- **Indirizzo *limited-broadcast*.** L'unico indirizzo nel blocco **255.255.255.255/32** viene chiamato indirizzo *limited-broadcast*. Viene usato ogni volta che un router o un host devono inviare un datagramma a tutti i dispositivi che si trovano all'interno della rete. I router della rete, bloccano i datagrammi che hanno questo indirizzo come destinazione, in questo modo la sua diffusione sarà limitata alla rete locale.
- **Indirizzo di *loopback*.** Il blocco **127.0.0.0/8** viene chiamato indirizzo di *loopback*. Un datagramma con uno degli indirizzi in questo blocco come indirizzo di destinazione non lascia mai l'host di spedizione, rimane al suo interno. Questi indirizzi vengono usati per il test e debug del software in esecuzione sull'host locale. Ad esempio è possibile scrivere un programma client e uno server in cui uno degli indirizzi nel

blocco sia usato come indirizzo server. Questo permetterà di testare i programmi client e server usando lo stesso host, prima di installarli su computer diversi.

- **Indirizzi privati.** Quattro blocchi sono stati riservati come indirizzi privati: **10.0.0.0/8**, **172.16.0.0/12**, **192.168.0.0/16**.
- **Indirizzi multicast.** Il blocco **224.0.0.0/4** è riservato agli indirizzi multicast.

Dynamic Host Configuration Protocol (DHCP)

Abbiamo visto che una grande organizzazione o un ISP possono ricevere un blocco di indirizzi direttamente dall'ICANN e che una piccola organizzazione può riceverlo direttamente da un ISP. Dopo che un blocco di indirizzi è stato assegnato ad un'organizzazione, l'amministratore di rete può assegnare manualmente gli indirizzi ai singoli host o router. Tuttavia l'assegnazione degli indirizzi può anche essere automatizzata usando il Dynamic Host Configuration Protocol (DHCP). Il DHCP è un programma di livello di applicazione, basato sul paradigma client/server, che in pratica aiuta il TCP/IP a livello di rete.

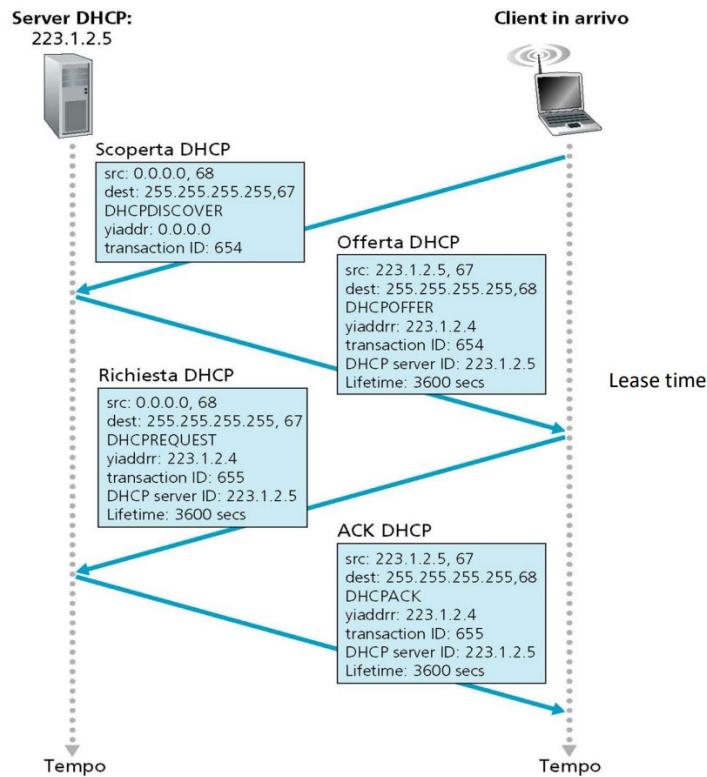
Il DHCP ha una diffusione amplissima in rete e spesso viene definito *protocollo plug-and-play*. Può essere usato in molte situazioni. Ad esempio, un amministratore di rete può configurare il DHCP per assegnare indirizzi IP permanenti agli host e ai router. Il DHCP può anche essere configurato per fornire indirizzi IP temporanei (a richiesta) agli host. Quest'ultima modalità è particolarmente utile, ad esempio, per fornire un indirizzo IP temporaneo all'ospite di un hotel che vuole collegare ad Internet il suo laptop o smartphone. Inoltre, consente anche ad un ISP cui sono stati assegnati 1000 indirizzi di fornire servizi a 4000 famiglie, supponendo che non più di un quarto dei suoi clienti usi Internet contemporaneamente.

In aggiunta al suo indirizzo IP, un computer necessita anche di conoscere il proprio prefisso di rete (o la maschera di rete dell'indirizzo). La maggior parte dei computer ha bisogno anche di altre due informazioni: l'indirizzo di un router di default che deve essere usato per comunicare con le altre reti (oltre a quella locale) e l'indirizzo di almeno un server per la risoluzione dei nomi (DNS). Riassumendo, normalmente servono quattro informazioni fondamentali: l'indirizzo del computer, il prefisso, l'indirizzo del default router e l'indirizzo IP del server dei nomi di dominio. DHCP può essere usato per fornire tutte queste informazioni all'host.

Funzionamento del DHCP

1. L'host che vuole entrare in rete crea un messaggio di tipo **DHCP-DISCOVER** nel quale solo il campo transaction-ID viene impostato usando un numero generato casualmente. L'host non può impostare alcun altro campo in quanto non ha le informazioni necessarie per farlo. Questo messaggio viene incapsulato in un datagramma utente UDP con porta sorgente impostata a 68 e la porta destinazione impostata a 67, entrambe sono porte note. Il datagramma utente viene quindi incapsulato in un datagramma IP con indirizzo sorgente 0.0.0.0 (“this-host”) e l’indirizzo di destinazione impostato a 255.255.255.255 (indirizzo broadcast). Il motivo è semplice, l'host al momento non conosce né il proprio indirizzo né quello del server.
2. Il server o i server DHCP (ce ne possono essere più di uno) rispondono con un messaggio di tipo **DHCPOFFER** nel quale il campo your-IP-address contiene l’indirizzo IP offerto per l’host che ne ha fatto richiesta e il server-IP-address comprende l’indirizzo IP del server. Il messaggio contiene anche un lease time (tempo di concessione), cioè per quanto tempo l’host potrà far uso di quell’indirizzo IP. Questo messaggio di risposta viene incapsulato in un datagramma utente con gli stessi numeri di porta della richiesta, ma ovviamente in ordine inverso. Il datagramma utente a sua volta viene incapsulato in un datagramma IP con l’indirizzo del server come indirizzo IP sorgente, ma l’indirizzo destinazione è ancora un indirizzo broadcast. Questo è utile anche per consentire agli altri server DHCP di ricevere l’offerta e fornire un’offerta migliore, qualora possano.
3. L’host richiedente riceve una o più offerte e seleziona la migliore. A questo punto l’host deve inviare un messaggio di tipo **DHCPREQUEST** al server che ha inviato l’offerta migliore. Tutti i campi con valore noto vengono impostati. Il messaggio viene incapsulato in un datagramma utente con numeri di porta uguali al primo messaggio di richiesta. Il datagramma utente viene incapsulato in un datagramma IP con indirizzi sorgente e destinazione uguali al primo messaggio di richiesta. Questo è necessario per annunciare a tutti gli altri server la cui offerta non è stata accettata.
4. Infine, il server selezionato risponde con un messaggio di tipo **DHC-PACK** al client se l’indirizzo IP offerto è valido. Se il server non è più in grado di mantenere la sua offerta (per esempio se nel frattempo l’indirizzo è stato offerto ad un altro host), il server invia un messaggio

di tipo **DHCPOFFER** e il client deve ripetere il procedimento. Anche questo messaggio è broadcast per far sapere agli altri server se la richiesta è stata accettata o rifiutata.



NAT

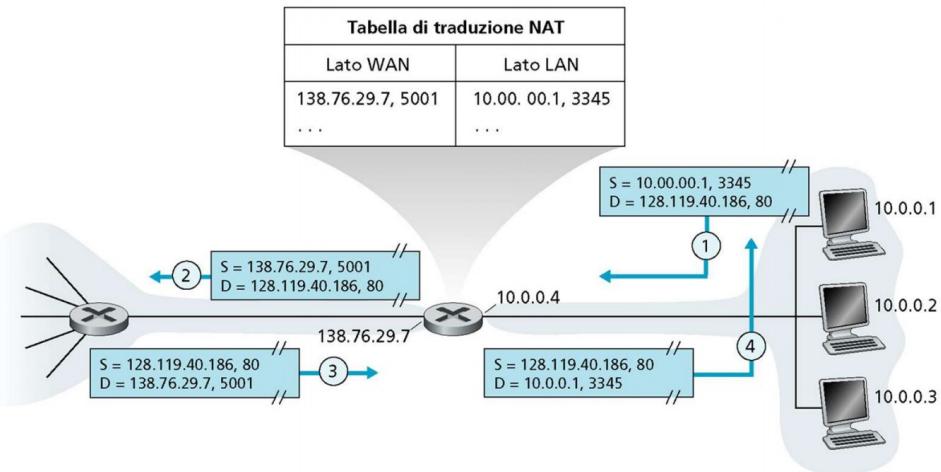
NAT ovvero Network Address Translation (RFC 2663, 3022), permette di trasmettere su Internet il traffico proveniente da sistemi attestati su sottoreti private, in cui sono assegnati indirizzi IP privati.

L'accesso di una rete privata su Internet è realizzato attraverso un router abilitato alla NAT, questo router **ha almeno un** indirizzo IP pubblico

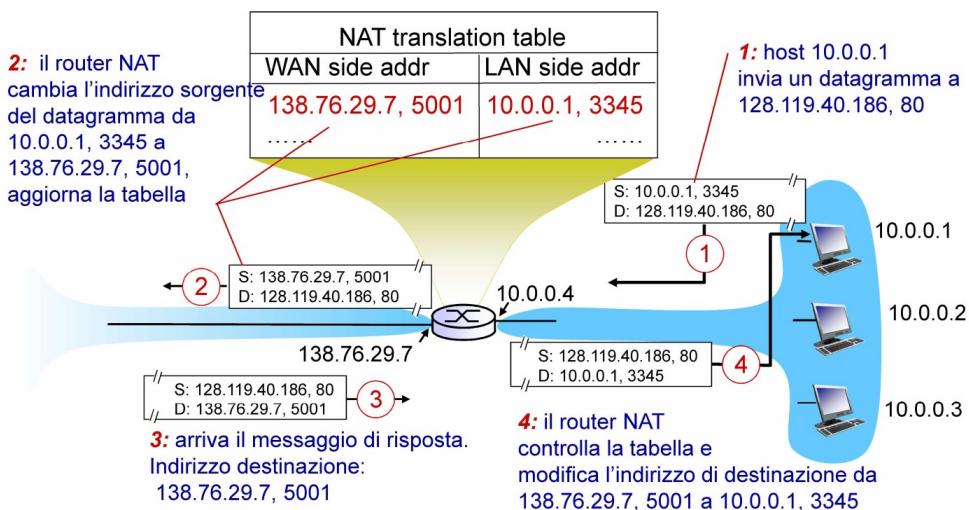
- Tutto il traffico (datagram) in uscita dal router di accesso ha un indirizzo IP sorgente pubblico, quello del router.
- Tutto il traffico in ingresso alla subnet ha come indirizzo IP di destinazione l'indirizzo pubblico del router di accesso.

Il router di accesso ha in memoria una tabella di traduzione NAT, le cui righe contengono le associazioni

(IP privato, porta) (IP pubblico e porta assegnata dal router)



Network Address Translation (NAT)



Traduzione dell'indirizzo

4.2.3 Inoltro dei datagrammi IP

All'inizio del capitolo è stato discusso il tema dell'inoltro (forwarding) a livello di rete. In questa sezione, il concetto sarà esteso per includere anche il ruolo

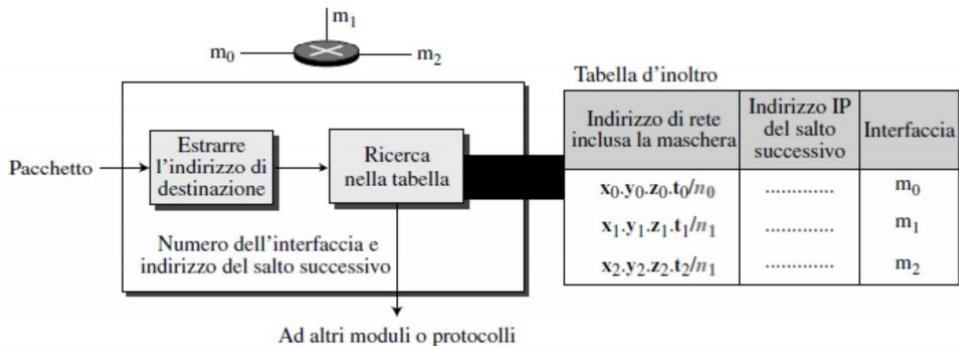
degli indirizzi IP nel forwarding. Come detto in precedenza, inoltrare significa collocare il pacchetto nel giusto percorso che lo porterà a destinazione. Dato che oggi Internet è costituita da una combinazione di collegamenti (reti), inoltrare significa inviare il pacchetto al salto (hop) successivo (che può essere la destinazione finale o un dispositivo d'interconnessione intermedio).

Inoltro basato sull'indirizzo di destinazione

Questo approccio richiede che l'host o il router che deve effettuare il forwarding abbia una tabella d'inoltro. Quando un host ha un datagramma da inviare o quando un router ha ricevuto un datagramma da inoltrare, esso fa riferimento a questa tabella per trovare il salto successivo in cui inviare il datagramma.

Nell'indirizzamento senza classi, l'intero spazio degli indirizzi è un'unica entità; come dice il nome non ci sono classi. Nella pratica ciò significa che l'inoltro richiede una riga di informazioni per ciascun blocco coinvolto. Le informazioni nella tabella devono essere cercate in base all'indirizzo di rete (il primo indirizzo nel blocco). Sfortunatamente, l'indirizzo di destinazione presente nel datagramma non fornisce alcun indizio in merito all'indirizzo di rete. Per risolvere il problema, occorre includere la maschera (ln) nella tabella. In altre parole, una tabella d'inoltro per l'indirizzamento senza classi deve includere quattro informazioni: la maschera, l'indirizzo di rete, il numero dell'interfaccia e l'indirizzo IP del router successivo (quest'ultimo è necessario per trovare l'indirizzo a livello di collegamento del salto successivo). Tuttavia, abbiamo visto che spesso le prime due informazioni sono combinate.

Il compito del modulo di forwarding è quello di effettuare le ricerche nella tabella. In ciascun riga, gli n bit a sinistra dell'indirizzo di destinazione (prefisso) sono lasciati invariati e il resto dei bit (suffisso) sono impostati a 0. Se l'indirizzo risultante (chiamato *indirizzo di rete*) combacia con l'indirizzo nella prima colonna, allora l'informazione nelle successive due colonne viene estratta. In caso contrario la ricerca continua. Normalmente l'ultima riga ha un valore di default nella prima colonna (non mostrato nella figura) che rappresenta tutti gli indirizzi di destinazione che non combaciano le righe precedenti (in generale si dice quindi che è una "riga di default").



Modulo d'inoltro semplificato per l'indirizzamento senza classi

Forwarding diretto/indiretto L'inoltro può essere di due tipi: diretto e indiretto.

Si parla di inoltro diretto quando il pacchetto IP ha come destinazione un host nella propria rete (o subnet) IP.

- l'invio è diretto sul destinatario;
- l'indirizzo di destinazione a livello link è quello del destinatario (MAC address);
- non viene interpellata nessun'altra entità.

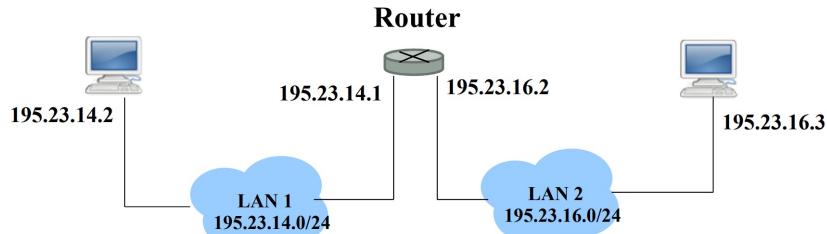
Nell'inoltro indiretto il pacchetto IP ha come destinazione un host di un'altra rete (o subnet) IP.

- Viene delegato l'invio ad "un altro";
- "l'altro" si chiama router;
- l'indirizzo di destinazione a livello link è quello del router.

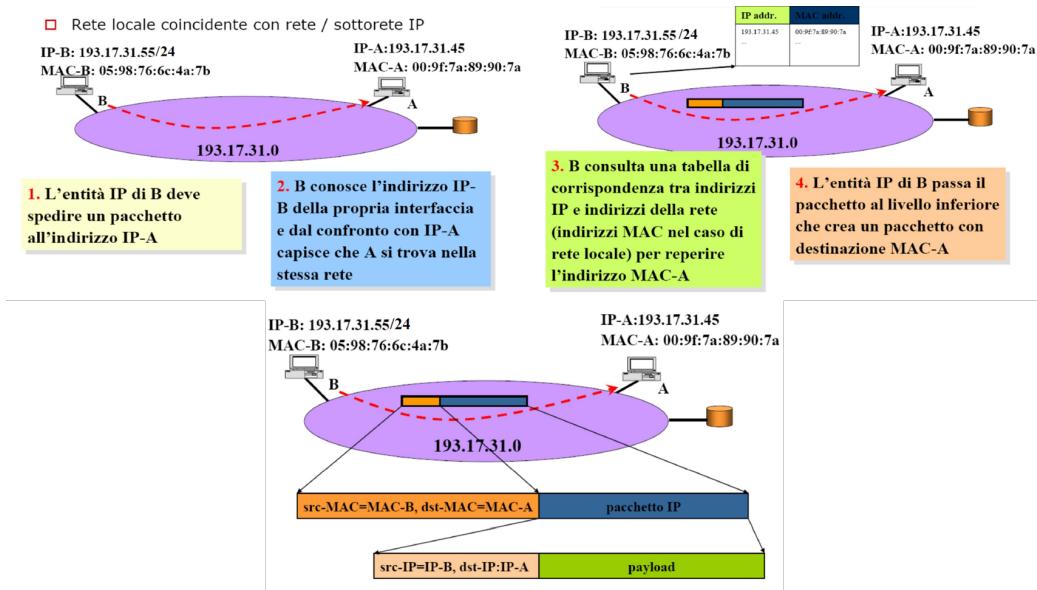
Si osserva come, in entrambi i casi, **condizioni necessarie** perché tutto funzioni sono che:

- esista un cammino (funzionante e) diretto, a livello data-link, tra tutti gli host che appartengono ad una stessa sottorete;
- ogni host coinvolto abbia un indirizzo IP "giusto", cioè con uguale net ID (cioè appartenga alla stessa sottorete) e con host ID univoco nella sottorete.

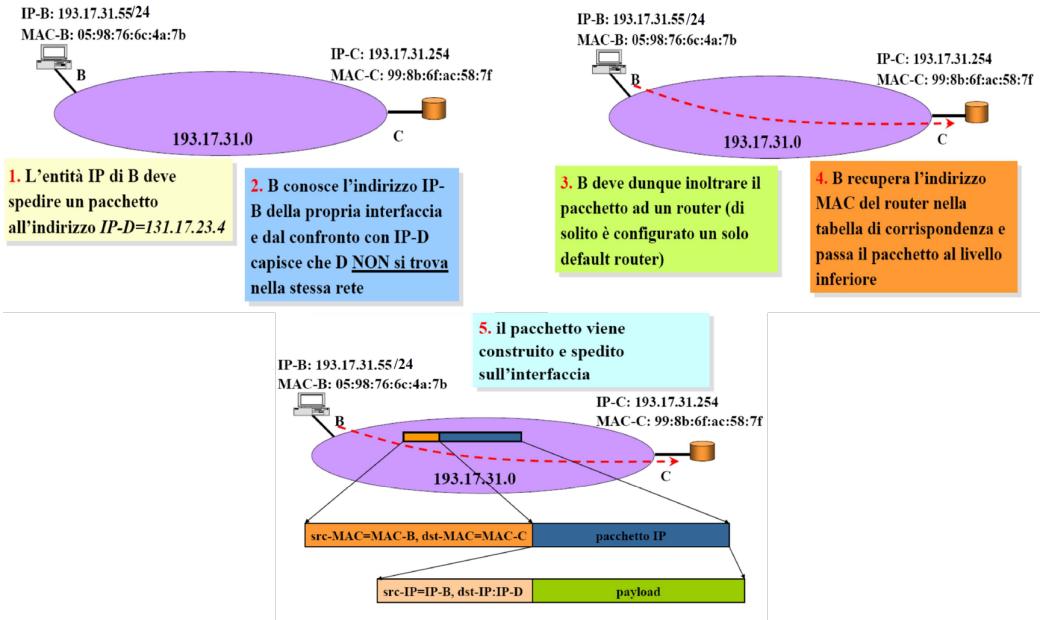
Le due condizioni insieme diventano condizione necessaria e sufficiente perché la comunicazione “funzioni”.



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 forwarding e decide dove mandare il pacchetto IP.



Inoltro diretto



Inoltro indiretto

Aggregazione degli indirizzi Quando viene utilizzato l'indirizzamento con classi, nella tabella d'inoltro è presente solamente una riga per ciascuna rete esterna all'organizzazione a cui appartiene il router. La riga è in grado di rappresentare pienamente la rete esterna anche se questa fa uso di subnetting. Quando un datagramma arriva al router, questo verifica la riga corrispondente ed effettua l'inoltro in base alle informazioni ricavate. Quando invece si utilizza l'indirizzamento senza classi, si assiste ad un aumento del numero delle righe nella tabella d'inoltro. Questo perché lo scopo dell'indirizzamento senza classi è proprio quello di dividere lo spazio degli indirizzi in blocchi più piccoli e gestibili. L'aumento nel numero delle righe da controllare porta ad un maggior tempo necessario per le ricerche nella tabella d'inoltro. Per attenuare questo problema è stato ideato il meccanismo di aggregazione degli indirizzi.

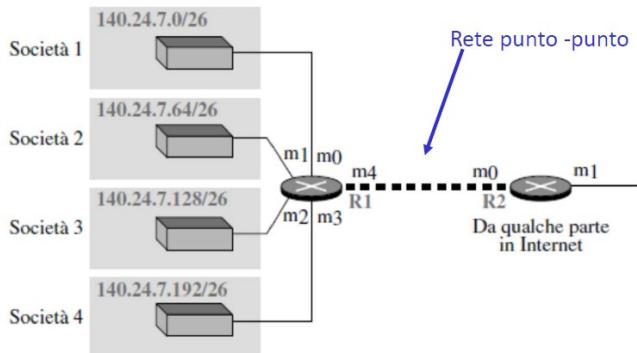


Tabella d'inoltro per R1

Indirizzo di rete/maschera	Indirizzo del salto successivo	Interfaccia
140.24.7.0/26	-----	m0
140.24.7.64/26	-----	m1
140.24.7.128/26	-----	m2
140.24.7.192/26	-----	m3
0.0.0.0/0	Indirizzo di R2	m4

Tabella d'inoltro per R2

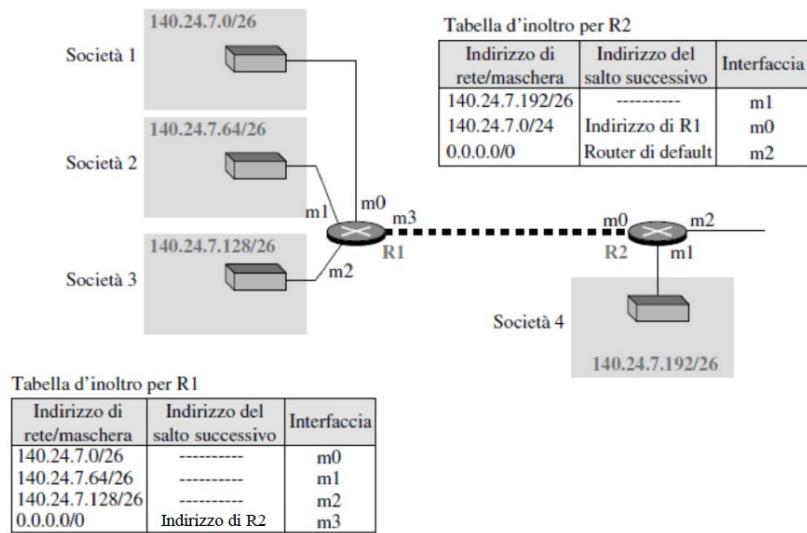
Indirizzo di rete/maschera	Indirizzo del salto successivo	Interfaccia
140.24.7.0/24	Indirizzo di R1 Router di default	m0
0.0.0.0/0	-----	m1

Aggregazione degli indirizzi

R1 è connesso alle reti di quattro organizzazioni, ciascuna con un blocco di 64 indirizzi. R2 è da qualche parte in Internet, lontano da R1. È evidente che R1 ha una tabella d'inoltro più lunga di R2 poiché ciascun datagramma che gli arriva deve essere indirizzato correttamente all'organizzazione appropriata. R2, invece, può avere una tabella d'inoltro molto piccola. Per R2, qualunque datagramma con indirizzo di destinazione compreso tra 140.24.7.0 e 140.24.7.255 è inviato attraverso l'interfaccia m_0 , senza considerare in alcun modo il destinatario specifico (a quale organizzazione dovrà alla fine essere consegnato il datagramma). Questo meccanismo è chiamato aggregazione degli indirizzi, poiché i blocchi degli indirizzi di quattro società sono stati aggregati in un solo blocco più grande. Nel caso ciascuna organizzazione avesse blocchi di indirizzi che non possono essere aggregati in un blocco più grande (ad esempio blocchi non contigui), R2 sarebbe costretto ad avere una tabella d'inoltro molto più lunga.

Corrispondenza con la maschera più lunga Che cosa accade se una delle organizzazioni nella figura precedente non è geograficamente vicina alle altre tre? Ad esempio, se l'organizzazione 4 non può essere connessa al router R1 per qualche ragione, possiamo ancora utilizzare l'idea di aggregazione di indirizzo e assegnare il blocco 140.24.7.192/26 alla società 4? La risposta è sì poiché l'instradamento nell'indirizzamento senza classi sfrutta un altro principio, la *corrispondenza con la maschera più lunga* (*longest mask matching*).

Questo principio afferma che la tabella d'inoltro viene ordinata dalla maschera più lunga alla maschera più corta. In altre parole, se ci sono 3 maschere, /27, /26 e /24, la maschera /27 deve essere nella prima riga e la /24 deve essere nell'ultima (senza considerare la regola di default). Vediamo se questo principio permette di risolvere la situazione nella quale l'organizzazione 4 è separata dalle altre tre.



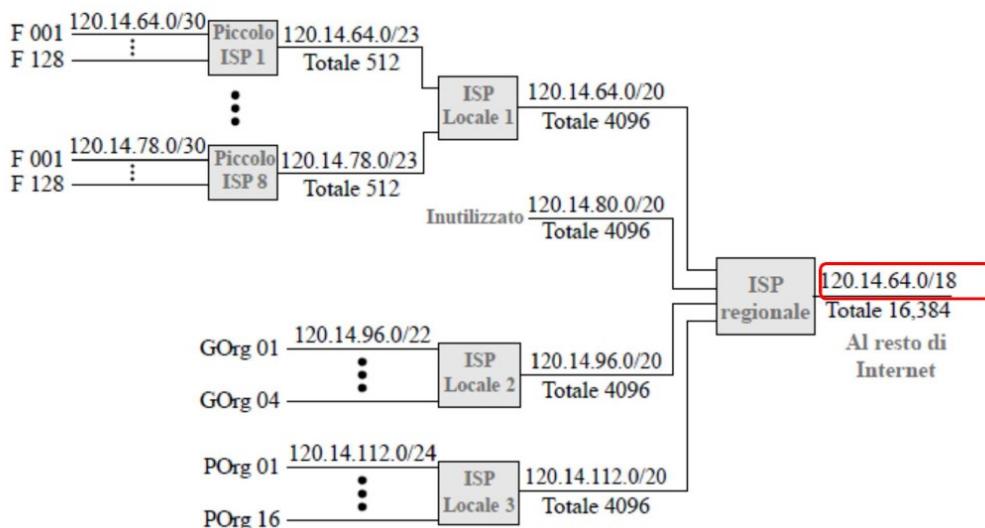
Corrispondenza con la maschera più lunga

Supponiamo che un pacchetto arrivi al router R2 con indirizzo di destinazione 140.24.7.200. La prima maschera nel router R2 è applicata e dà l'indirizzo di rete 140.24.7.192. Il datagramma è quindi instradato correttamente all'interfaccia m_1 e raggiunge l'organizzazione 4. Tuttavia, se la tabella d'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.

Routing gerarchico Per risolvere il problema delle tabelle d'inoltro di dimensione eccessiva è possibile implementare una sorta di gerarchia nelle tabelle d'inoltro. Oggi Internet ha una struttura in qualche modo gerarchica, essendo divisa in dorsali e ISP nazionali. Gli ISP nazionali sono suddivisi in ISP regionali, che a loro volta sono partizionati in ISP locali. Se la tabella d'inoltro ha una qualche forma di gerarchia, analoga all'architettura di Internet, può diminuire di dimensione.

Prendiamo il caso di un ISP locale. A un ISP locale può essere assegnato un singolo, ma ampio, blocco di indirizzi con un certa lunghezza di prefisso. L'ISP locale può suddividere questo blocco in blocchi più piccoli, di diverse dimensioni, e assegnarli ad utenti individuali e organizzazioni di varia dimensione. Se il blocco assegnato all'ISP locale inizia con a.b.c.d/n, l'ISP può creare blocchi che iniziano con e.f.g.h/m dove m può variare per ciascun cliente ed è maggiore di n.

Come può questo ridurre la dimensione della tabella d'inoltro? Molto semplicemente, il resto di Internet non è consapevole di questa divisione (e non c'è bisogno che lo sia). Tutti i clienti dell'ISP locale sono individuati come a.b.c.d/n per il resto di Internet. Ogni datagramma destinato a uno degli indirizzi in questo blocco piuttosto ampio è instradato all'ISP locale. In ogni router del mondo vi è una sola riga per tutti questi clienti; tutti loro appartengono allo stesso gruppo. Ovviamente, all'interno dell'ISP locale, il router deve riconoscere i sottoblocchi ed instradare i datagrammi al destinatario giusto. Se uno dei clienti non è una grande organizzazione, al suo interno può creare un ulteriore livello gerarchico usando il subnetting e dividendo il suo sottoblocco in blocchi ancora più piccoli.



Routing gerarchico ed ISP

4.2.4 ICMPv4

L'IPv4 non implementa alcun meccanismo per segnalare gli errori o correggerli. Cosa accade se qualcosa va storto? Che cosa accade se un router deve scartare un datagramma perché non riesce a trovare un percorso per la desti-

nazione finale o perché il campo time-to-live ha raggiunto il valore zero? Che cosa accade se un host di destinazione non ha ricevuto tutti i frammenti di un datagramma entro un determinato limite di tempo prestabilito? Queste sono tutte situazioni nelle quali si è verificato un errore, ma il protocollo IP non ha meccanismi integrati per renderlo noto all'host mittente.

Inoltre, il protocollo IP è sprovvisto di un meccanismo per effettuare richieste sullo stato di un sistema remoto. Ad esempio, un host, può a volte aver bisogno di determinare se un router o un altro host è attivo. Altre volte un amministratore di rete può aver bisogno di informazioni relative ad un host o un router.

L'*Internet Control Message Protocol* versione 4 (ICMPv4) è stato creato per porre rimedio a queste carenze. ICMP deve essere pensato come la controparte di IP ed è esso stesso un protocollo del livello di rete. Tuttavia, i suoi messaggi non vengono passati direttamente al livello di collegamento, come ci si potrebbe aspettare. I messaggi ICMP vengono incapsulati all'interno di datagrammi IP prima di essere passati al livello inferiore. Quando un datagramma incapsula un messaggio ICMP, il valore del campo protocollo nel datagramma IP è impostato a 1, per indicare che nel payload del datagramma è presente un messaggio ICMP.

Messaggi

I messaggi ICMPv4 sono suddivisi in due ampie categorie: *messaggi di segnalazione errori* e *messaggi di richiesta (query messages)*. I messaggi di segnalazione riportano i problemi che un router o un host (la destinazione) possono incontrare quando elaborano un datagramma IP. I messaggi di richiesta, invece, permettono ad un host o ad un amministratore di rete di chiedere informazioni ad un router o ad un host. Ad esempio, un host può individuare gli altri host nella rete locale e un router può indicare ad un nodo come reindirizzare i propri datagrammi.

Formato dei messaggi Un messaggio ICMPv4 ha un'intestazione di 8 byte e una sezione dati di lunghezza variabile. Anche se il formato dell'intestazione è diverso per ogni tipo di messaggio, i primi 4 byte sono comuni a tutti. Il primo campo, il tipo ICMP, definisce il tipo di messaggio. Il campo codice specifica la ragione del particolare tipo di messaggio. L'ultimo campo comune è quello del checksum. Il resto dell'intestazione è specifico per ciascun tipo di messaggio. La sezione dati nei messaggi di errore trasporta informazioni utili per individuare il datagramma originale che ha provocato l'errore. Nei messaggi di richiesta invece la sezione dati trasporta informazioni aggiuntive che dipendono dal tipo di interrogazione.

ICMP Tipo	Codice	Descrizione
0	0	risposta al messaggio di eco (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 eco - <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>

Tipi di messaggio ICMP

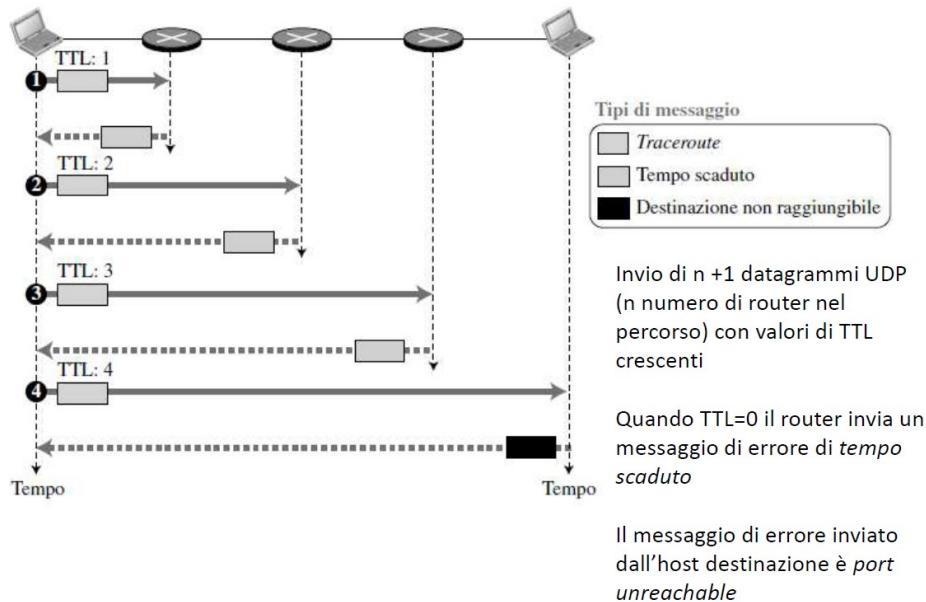
Ping Una delle applicazioni che un host può utilizzare per verificare il funzionamento di un altro host è il programma *ping*. Il programma *ping* si basa sui messaggi di richiesta e risposta eco dell'ICMP. Un host invia una richiesta eco (tipo 8, codice 0) a un altro host che, se attivo, può rispondere con una risposta eco (tipo 0, codice 0). 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.

```
$ ping pads.cs.unibo.it
PING chernobog.pads.cs.unibo.it (130.136.132.11) 56(84) bytes of data.
64 bytes from chernobog.pads.cs.unibo.it (130.136.132.11): icmp_req=1 ttl=52 time=34.2 ms
64 bytes from chernobog.pads.cs.unibo.it (130.136.132.11): icmp_req=2 ttl=52 time=33.1 ms
64 bytes from chernobog.pads.cs.unibo.it (130.136.132.11): icmp_req=3 ttl=52 time=34.0 ms
64 bytes from chernobog.pads.cs.unibo.it (130.136.132.11): icmp_req=4 ttl=52 time=33.9 ms
64 bytes from chernobog.pads.cs.unibo.it (130.136.132.11): icmp_req=5 ttl=52 time=33.3 ms
--- chernobog.pads.cs.unibo.it ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 33.177/33.758/34.220/0.417 ms
```

Invio di un messaggio ping al sito pads.cs.unibo.it

Traceroute Il programma *traceroute* in UNIX o *tracert* in Windows 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.

Il programma *traceroute* ha un funzionamento molto diverso da *ping*. *Ping* è basato su due messaggi query; *traceroute* è invece implementato per mezzo di due messaggi di segnalazione degli errori: tempo scaduto e porta destinazione non raggiungibile.



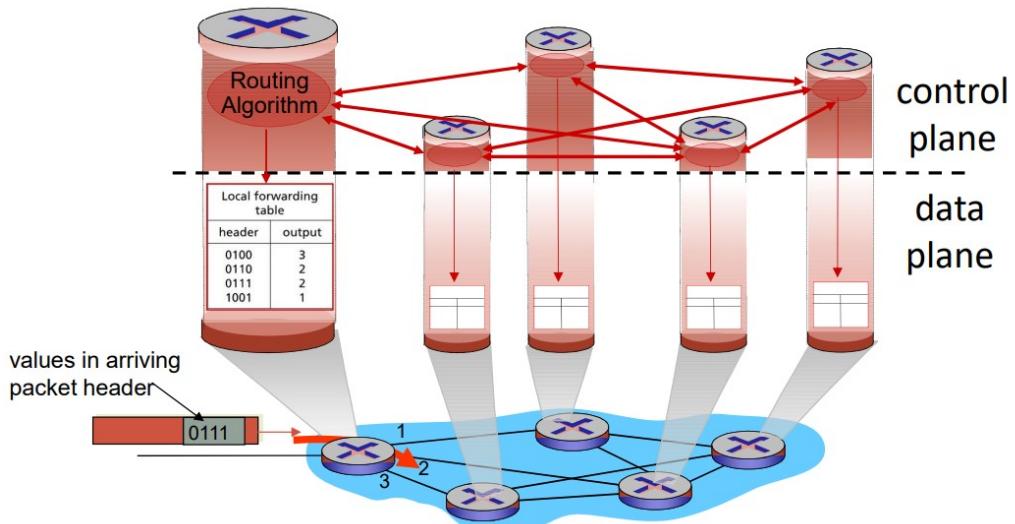
Esempio di funzionamento del programma traceroute

Il programma *traceroute* imposta 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. Quanto segue mostra un esempio di funzionamento del programma *traceroute* che utilizza tre messaggi per ogni dispositivo e ottiene quindi tre RTT.

\$ traceroute printers.com				
traceroute to printers.com (13.1.69.93), 30 hops max, 38 byte packets				
1 route.front.edu	(153.18.31.254)	0.622 ms	0.891 ms	0.875 ms
2 ceneric.net	(137.164.32.140)	3.069 ms	2.875 ms	2.930 ms
3 satire.net	(132.16.132.20)	3.071 ms	2.876 ms	2.929 ms
4 alpha.printers.com	(13.1.69.93)	5.922 ms	5.048 ms	4.922 ms

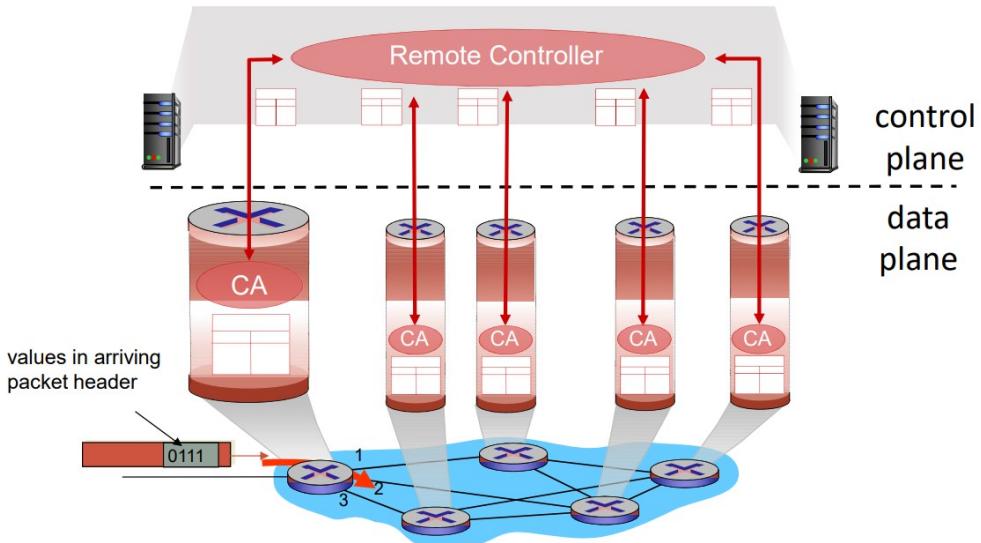
4.3 Architettura di un Router

<i>Forwarding (inoltro)</i>	<i>Routing (instradamento)</i>
Trasferire i pacchetti sull'appropriato collegamento in uscita	Processo decisionale di scelta del percorso verso una destinazione
Usa la tabella d'inoltro	Determina i valori da inserire nella tabella d'inoltro tramite gli algoritmi di <i>routing</i>
Avviene al livello del piano di dati	Avviene al livello del piano di controllo



Routing decentralizzato:

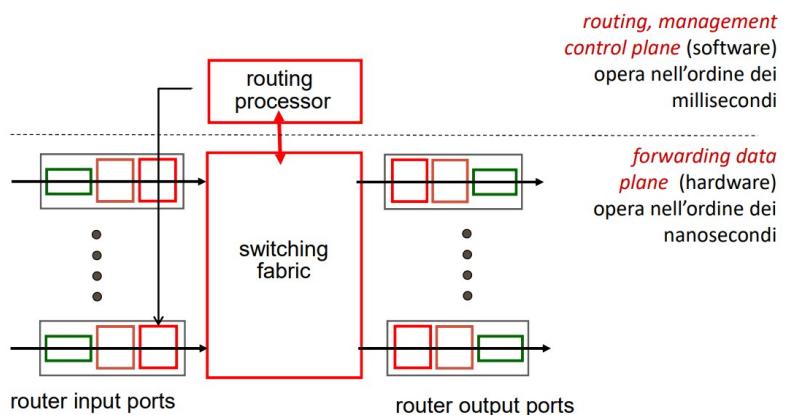
- Algoritmo di routing in esecuzione su ciascun router
- I router si scambiano messaggi (protocolli di routing)



Routing logicamente centralizzato (*Software-defined Networking (SDN)*)
Un controller remoto interagisce con Control Agents locali:

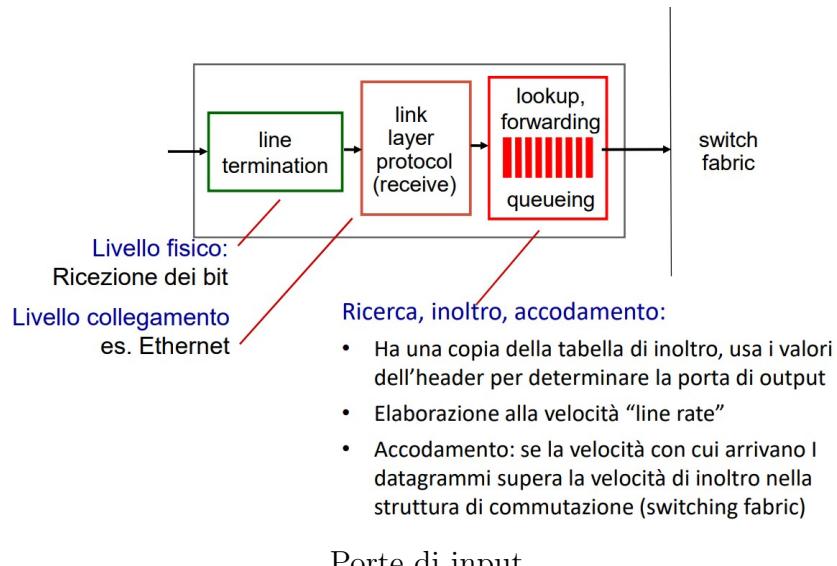
- Riceve dai CA informazioni sui collegamenti e sul traffico
- Invia ai CA i valori da inserire nella tabella di inoltro

Nella nostra discussione circa l'inoltro e l'instradamento abbiamo rappresentato il router come una scatola nera che accetta pacchetti in entrata da una delle porte di input (interfacce), usa una tabella d'inoltro per trovare la porta di output e da questa invia il pacchetto.



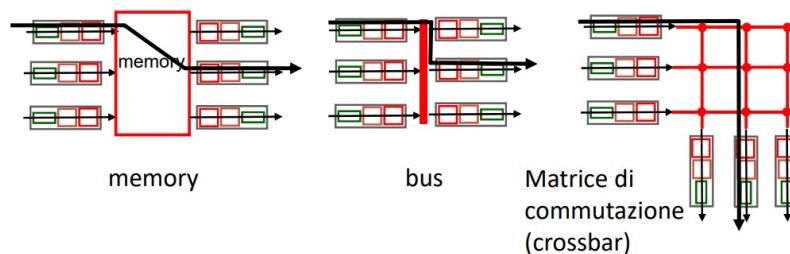
Componenti di un router

Porte di input



Porte di input

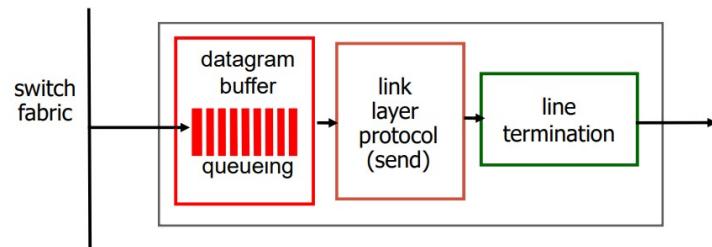
Switching fabric (struttura di commutazione)



Switching fabric

- Trasferisce il pacchetto dal buffer di input al buffer di output appropriato
- Velocità di commutazione: velocità con cui i pacchetti possono essere trasferiti dagli ingressi alle uscite
 - N input: velocità di commutazione auspicabile N volte la velocità di linea

Porte di output



Porte di output

- **Buffering**: richiesto quando i datagrammi arrivano dalla struttura di commutazione ad una velocità maggiore della velocità di trasmissione sul collegamento in uscita
- **Scheduling**: politiche per definire l'ordine di trasmissione dei datagrammi

4.4 Routing unicast

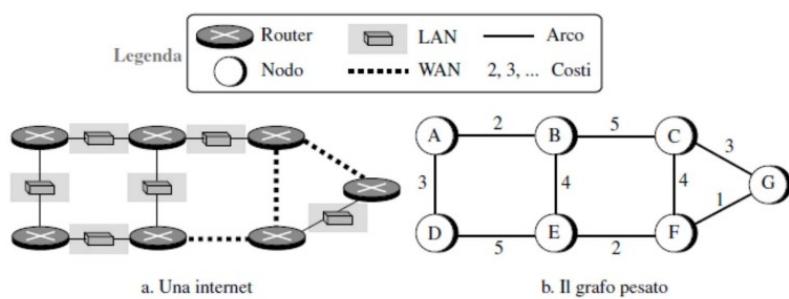
In una rete come Internet lo scopo del livello di rete è quello di consegnare un datagramma dalla sorgente alla destinazione o alle destinazioni. Se un datagramma è destinato ad una sola destinazione (consegna uno-a-uno) si parla di *routing unicast*. Se il datagramma è destinato a numerose destinazioni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

4.4.1 Concetti generali

Nel routing unicast un pacchetto viene instradato, salto dopo salto, dalla sua sorgente alla sua destinazione con l'aiuto delle tabelle d'inoltro. L'host sorgente non ha bisogno di alcuna tabella d'inoltro visto che si limita a consegnare il proprio pacchetto al router di default della sua rete locale. Neanche l'host di destinazione ha bisogno di una tabella d'inoltro, poiché riceve il pacchetto direttamente dal router di default della sua rete locale. Questo significa che solo i router che collegano tra loro le diverse reti hanno bisogno di tabelle d'inoltro. Instradare un pacchetto dalla sua sorgente

alla sua destinazione significa pertanto instradare il pacchetto da un *router sorgente* (il router di default dell'host sorgente) a un *router di destinazione* (il router collegato alla rete di destinazione). Anche se un pacchetto deve passare attraverso i router sorgente e destinazione, la domanda è: quali altri router dovrà attraversare il pacchetto? In altre parole, poiché solitamente ci sono molti percorsi diversi tra sorgente e la destinazione, quale percorso dovrà effettuare il pacchetto?

Per trovare il percorso migliore, una rete di reti (ovvero una internet) può essere modellata per mezzo di un *grafo*. Un grafo in informatica è formato da un insieme di *nodi* collegati da *archi*. Per modellare una internet per mezzo di un grafo, possiamo pensare che ogni router sia un nodo ed ogni rete, tra una coppia di router, sia un arco. Una internet, infatti, è modellata come un *grafo pesato*, in cui ad ogni arco è associato un costo. Se viene usato un grafo pesato per rappresentare un'area geografica, i nodi possono essere le città e gli archi le strade che le collegano; i pesi, in questo caso, sono le distanze tra le città. Nell'instradamento, tuttavia, il costo di un arco ha un'interpretazione diversa a seconda dei diversi protocolli di routing. Per il momento supponiamo che ci sia un costo associato ad ogni arco. Se non c'è un arco che collega due nodi allora il costo associato è infinito.



Una rete e la sua rappresentazione sotto forma di grafo

Instradamento a costo minimo

Quando una internet viene rappresentata come un grafo pesato, uno dei modi per interpretare il percorso *migliore* dal router sorgente al router destinazione è trovare il *minor costo* tra i due. In altre parole il router sorgente sceglie il percorso verso il router destinazione in modo che il costo totale del percorso sia il costo minimo tra tutti i percorsi possibili. Nella figura il percorso migliore tra A ed E è A-B-E, con costo 6. Ciò significa che, per poter instradare i pacchetti utilizzando questa metrica, ogni router deve trovare il percorso a costo minimo tra se stesso e tutti gli altri router.

Alberi di costo minimo

Se in una rete ci sono N router allora ci sono $(N - 1)$ percorsi a costo minimo da ogni router a ogni altro router. Ciò significa che servono $N \cdot (N - 1)$ percorsi a costo minimo per l'intera rete. Quindi, in una rete con solo 10 router ci sono 90 percorsi a costo minimo. Un modo migliore per vedere tutti questi percorsi è combinarli in un *albero di costo minimo*. Un albero di costo minimo è un albero con il router sorgente che fa da radice e che visita tutti gli altri nodi dell'albero seguendo sempre il percorso più corto (meno costoso) tra quelli possibili. In questo modo possiamo avere un solo albero di costo minimo per ogni nodo; avremo quindi N alberi di costo minimo per l'intera rete.

4.4.2 Algoritmi di routing

Routing statico e dinamico

- Nel **routing statico** le righe (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.
- Nel **routing 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 (grandi reti locali private e Internet).

Algoritmi di instradamento globali/decentralizzati

Gli algoritmi di instradamento si possono classificare in:

- **Globali**, se si basano sulla conoscenza della topologia di tutta la rete. Il calcolo può essere fatto in un unico sito (algoritmo centralizzato) o in più nodi, utilizzando informazioni sulla connettività di tutti i nodi e sui costi di tutti i link (es. algoritmo **Link State**).
 - L'algoritmo riceve in ingresso informazioni su tutti i collegamenti tra i nodi e i loro costi.
- **Decentralizzati**, quando nessun nodo conosce la topologia di tutta la rete, ma ha informazioni solo sui nodi e link vicini. Il calcolo del percorso è iterativo e distribuito (es. algoritmo **Distance Vector**).
 - Ogni nodo elabora un vettore di stima dei costi (distanze) verso tutti gli altri nodi nella rete.

- Il cammino a costo minimo viene calcolato in modo distribuito e iterativo scambiandosi informazioni con i nodi vicini.

Routing basato su vettore distanza

Nel distance-vector routing, per prima cosa ogni nodo crea il proprio albero di costo minimo con le informazioni di base che possiede sui suoi soli nodi vicini. Il risultato sono degli alberi incompleti (mancano tutte le informazioni dei nodi che non sono vicini). Questi alberi incompleti vengono a questo punto scambiati tra i vicini per rendere l'albero sempre più completo e rappresentare così l'intera rete. Nel distance-vector routing quello che accade è che ogni router comunica continuamente a tutti i suoi vicini ciò che sa sulla rete (anche se, ovviamente, potrebbe non sapere tutto).

Equazione di Bellman-Ford Il fulcro del routing basato su vettore distanza è la famosa equazione di *Bellman-Ford*. Questa equazione viene usata per trovare il costo minimo (distanza minima) tra un nodo sorgente, x , e un nodo destinazione, y , tramite dei nodi intermedi ($\mathbf{a}, \mathbf{b}, \mathbf{c}, \dots$) dove i costi tra il nodo sorgente e i nodi intermedi e i costi minimi tra i nodi intermedi e la destinazione sono noti. Di seguito mostriamo il caso generale in cui D_{ij} è la distanza minima e c_{ij} è il costo tra il nodo i e il nodo j .

$$D_{xy} = \min\{(c_{xa} + D_{ay}), (c_{xb} + D_{by}), (c_{xc} + D_{cy}), \dots\}$$

Nel distance-vector routing, quello che si vuole fare normalmente è aggiornare un costo minimo esistente con il costo minimo che si ottiene passando attraverso un nodo intermedio, come ad esempio z , se quest'ultimo è minore. In questo caso l'equazione diventa più semplice.

$$D_{xy} = \min\{D_{xy}, (c_{xz} + D_{zy})\}$$

Possiamo dire che l'equazione di Bellman-Ford ci permette di costruire un nuovo percorso a costo minimo rispetto ai percorsi a costo minimo precedentemente stabiliti.

Vettori distanza Il concetto di *vettore distanza* è alla base del distance-vector routing. Un albero a costo minimo è una combinazione di percorsi a costo minimo dalla radice dell'albero verso tutte le destinazioni. Tali percorsi vengono collegati insieme per formare l'albero. Il routing a vettore distanza scinde questi percorsi e crea quello che viene chiamato un *vettore distanza*, cioè un array monodimensionale che rappresenta l'albero.

Da notare che il “*nome*” del vettore distanza definisce la radice, gli *indici* invece definiscono le destinazioni e il *valore* di ogni cella definisce il costo

minimo dalla radice alla destinazione. Un vettore distanza non fornisce il percorso da seguire per giungere alla destinazione (informazione che invece l'albero è in grado di fornire); il vettore riporta solo i costi minimi per le destinazioni.

Sappiamo che un vettore distanza può rappresentare i percorsi a costo minimo di un albero a costo minimo, ma la questione è: in quale modo i nodi all'interno di una rete all'inizio creano il proprio vettore distanza? Ogni nodo della rete, quando viene inizializzato, crea un vettore distanza molto rudimentale con le sole informazioni che il nodo riesce ad ottenere dai propri vicini (i nodi con cui è direttamente collegato). Per far questo, il nodo invia alcuni messaggi di benvenuto attraverso tutte le sue interfacce e scopre l'identità dei vicini e la sua distanza con ciascuno di essi. Quindi crea un semplice vettore distanza inserendo la distanze scoperte nelle celle corrispondenti e lascia il valore delle altre celle ad infinito. Questi vettori rappresentano i percorsi a costo minimo elaborati sulla base delle solo informazioni (limitate) di cui dispone il nodo in questo momento. Quando si è a conoscenza di una sola distanza tra due nodi, questa può essere considerata il minimo. La Figura 4.2 mostra tutti i vettori distanza per la nostra rete. È bene notare che non c'è alcuna necessità che i vettori siano costruiti in modo sincrono, ogni nodo può essere inizializzato in modo del tutto indipendente (asincrono) dagli altri.

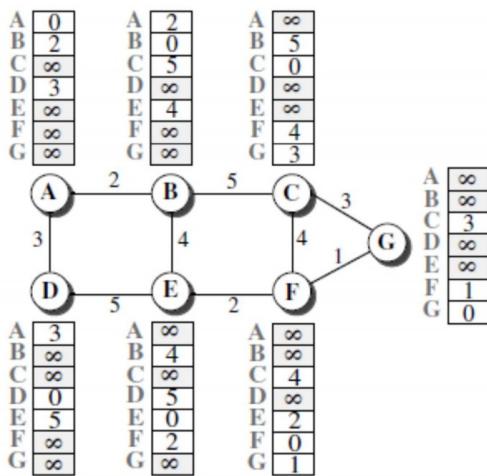
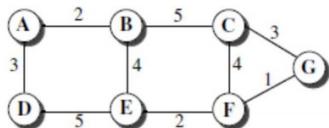


Figura 4.2: Vettori iniziali dei nodi di una rete

Questi vettori rudimentali non permettono veramente ad una rete di inoltrare i pacchetti. Per esempio il nodo A crede di non essere collegato al nodo G in quanto la cella corrispondente mostra un costo infinito. Dopo che ogni nodo ha creato il suo vettore ne invia una copia a tutti i suoi vicini. Quando

un nodo riceve un vettore distanza da un vicino provvede ad aggiornare il suo vettore distanza applicando l'equazione Bellman-Ford (seconda versione). Tuttavia bisogna aggiornare non solo un costo minimo ma N costi minimi, dove N è il numero di nodi nella rete. Se stiamo usando un programma, possiamo farlo con una semplice iterazione; se invece stiamo lavorando su carta, possiamo mostare l'intero vettore invece delle N equazioni separate.



Nuovo B	Vecchio B	A
A 2	A 2	A 0
B 0	B 0	B 2
C 5	C 5	C ∞
D 5	D ∞	D 3
E 4	E 4	E ∞
F ∞	F ∞	F ∞
G ∞	G ∞	G ∞

a. Primo evento: B riceve una copia del vettore di A.

Nuovo B	Vecchio B	E
A 2	A 2	A ∞
B 0	B 0	B 4
C 5	C 5	C ∞
D 5	D 5	D 5
E 4	E 4	E 0
F 6	F ∞	F 2
G ∞	G ∞	G ∞

b. Secondo evento: B riceve una copia del vettore di E..

Figura 4.3: Aggiornamento dei vettori distanza

Ed è esattamente quanto riportiamo nella Figura 4.3. La figura mostra due eventi asincroni, che avvengono uno dopo l'altro, a distanza di qualche tempo. Nel primo evento il nodo A ha inviato il suo vettore al nodo B. Il nodo B aggiorna il suo vettore usando il costo $c_{BA} = 2$. Nel secondo evento E ha mandato il suo vettore al nodo B. Il nodo B aggiorna il suo vettore usando il costo $c_{EA} = 4$.

Grazie al primo evento, il nodo B ha un miglioramento nel suo vettore: il suo costo minimo verso il nodo D è ora cambiato da infinito a 5 (passando attraverso il nodo A). Dopo il secondo evento il nodo B ha un ulteriore miglioramento del suo vettore: il suo costo minimo verso il nodo F è cambiato da infinito a 6 (passando attraverso il nodo E).

Dopo l'aggiornamento di un nodo, esso invia immediatamente il suo vettore aggiornato a tutti i vicini. Anche se i suoi vicini hanno ricevuto il vettore precedente, quello aggiornato può portare a dei miglioramenti.

Algoritmo di routing basato su vettore distanza Finalmente possiamo vedere lo pseudocodice semplificato che descrive l'algoritmo di routing

basato su vettore di distanza. L'algoritmo viene eseguito da ogni nodo in modo indipendente e asincrono.

```

1 Distance_Vector_Routing ()
2 {
3     // Inizializzazione (creazione dei vettori distanza iniziali del nodo)
4     D[me_stesso] = 0
5     for (y = 1 to N)
6     {
7         if (y è un vicino)
8             D[y] = c[me_stesso][y]
9         else
10            D[y] = ∞
11    }
12    spedisci il vettore {D[1], D[2], ..., D[N]} a tutti i vicini
13    // Aggiornamento (usare il vettore ricevuto dal vicino per aggiornare quello locale)
14    repeat (sempre)
15    {
16        wait (un vettore Dw da un vicino w o un qualsiasi cambiamento negli archi)
17        for (y = 1 to N)
18        {
19            D[y] = min [D[y], (c[me_stesso][w] + Dw[y])] // Equazione di Bellman-Ford
20        }   D[y] = minv [ c[me_stesso][v]+Dv[y] ]
21        if (c'è un cambiamento nel vettore)
22            spedisci il vettore {D[1], D[2], ..., D[N]} a tutti i vicini
23    }
24 } // Fine dell'algoritmo distance-vector routing

```

Algoritmo del distance-vector routing

Le righe da 4 a 11 inizializzano il vettore locale del nodo. Le righe da 14 a 23 mostrano come il vettore venga aggiornato dopo aver ricevuto un nuovo vettore da un vicino. Il ciclo *for* (nelle righe da 17 a 20) si occupa di aggiornare tutte le celle del vettore locale in base agli aggiornamenti contenuti nel nuovo vettore che è stato ricevuto. Da notare che il nodo invia il suo vettore due volte: alla riga 12 subito dopo l'inizializzazione e alla riga 22 dopo ogni aggiornamento.

Conteggio all'infinito Un problema con il routing basato sul vettore distanza è che i decrementi di costo (cioè le buone notizie) si diffondono rapidamente, mentre gli aumenti di costo (le cattive notizie) 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 vettore distanza serve avere un po' di tempo. Questo problema è spesso chiamato *conteggio all'infinito*. A volte servono molti aggiornamenti prima che il costo di un collegamento rotto venga registrato come infinito da tutti i router.

Ciclo a due nodi Un esempio di conteggio all’infinito è il problema del ciclo a due nodi. Per comprendere il problema, diamo un’occhiata allo scenario illustrato nella Figura 4.4.



Figura 4.4: Instabilità a due nodi

La figura illustra un sistema formato da tre nodi. Abbiamo mostrato solo le parti della tabella d’inoltro necessarie per la discussione. All’inizio sia il nodo A che il nodo B sanno come raggiungere il nodo X. Ma improvvisamente si guasta il collegamento tra A e X. Di conseguenza, il nodo A modifica la sua tabella. Se A invia immediatamente la sua tabella a B non c’è alcun problema. Tuttavia, se B invia la sua tabella d’inoltro ad A prima di ricevere quella di A, il sistema diventa instabile. Il nodo A riceve l’aggiornamento e, supponendo che B abbia trovato un modo per raggiungere X, aggiorna immediatamente la sua tabella d’inoltro. Ora A invia il suo nuovo aggiornamento a B. Quello che accade è che B è portato a credere che qualcosa sia cambiato vicino ad A e di conseguenza aggiorna la sua tabella d’inoltro. Il costo per raggiungere X aumenta gradualmente fino ad arrivare all’infinito. Ora, finalmente, sia A che B sanno che X non può essere raggiunto. Tuttavia, durante questo periodo di tempo il sistema non è stabile. Il nodo A crede che il percorso per raggiungere X passi da B e il nodo B crede che il percorso per raggiungere X passi attraverso A. Se A riceve un pacchetto destinato X il pacchetto va a B e quindi ritorna ad A. Allo stesso modo, se B riceve un pacchetto destinato ad X, va verso A e torna a B. I pacchetti rimbalzano tra A e B, creando un ciclo a due nodi. Di seguito vengono illustrate alcune soluzioni proposte per risolvere i problemi d’instabilità di questo tipo.

Split horizon (orizzonte spaccato) Una soluzione all’instabilità viene chiamata *split horizon*. Con questa strategia invece di inviare la tabella attraverso ogni interfaccia, ciascun nodo invia solo una parte della sua tabella tramite le sue varie interfacce. Se, secondo tale tabella, il nodo B ritiene che

il percorso ottimale per raggiungere X passi tramite A, allora non deve fornire questa informazione ad A: l'informazione è arrivata ad A che quindi la conosce già. Il fatto che le informazioni vengano prese dal nodo A, vengano modificate e inviate nuovamente al nodo A è ciò che crea confusione. Nel nostro scenario, il nodo B elimina l'ultima riga della sua tabella d'inoltro prima di inviarla ad A. In questo caso il nodo A mantiene il valore di infinito come distanza verso X. Più tardi, quando il nodo A invia la sua tabella d'inoltro a B, anche il nodo B corregge la sua tabella. Il sistema diventa stabile dopo il primo aggiornamento: sia il nodo A che il nodo B sanno che X non è più raggiungibile.

Poisoned reverse (inversione avvelenata) L'utilizzo della strategia split horizon ha un effetto collaterale.

Normalmente il protocollo che implementa l'algoritmo di routing utilizza un timer: se per una certa quantità di tempo non ci sono novità circa un percorso, il nodo lo elimina dalla sua tabella. Quando il nodo B, nello scenario precedente, elimina il percorso verso X dai suoi annunci ad A, il nodo A non può sapere se ciò è dovuto alla strategia split horizon (la sorgente delle informazioni era A) o se dipende dal fatto che B recentemente non ha ricevuto alcuna notizia di X. Nella strategia di inversione avvelenata, B può ancora rendere pubblico il valore per X, ma se la sorgente delle informazioni è A, allora sostituisce la distanza con valore infinito. In questo caso l'infinito viene usato come avvertimento: “Non usare questo valore, quello che so circa questo percorso viene da te”.

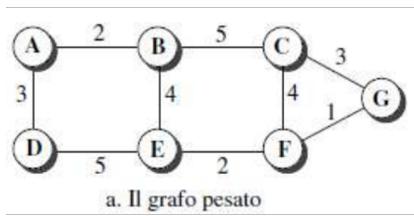
Instabilità a tre nodi L'instabilità a due nodi si può evitare utilizzando la strategia split horizon combinata all'inversione avvelenata. Tuttavia, se l'instabilità è a tre nodi, la stabilità non può essere garantita.

Routing a stato del collegamento

Un algoritmo di routing che deriva direttamente dalla nostra precedente discussione sulla creazione degli alberi a costo minimo e delle tabelle d'inoltro è il *routing a stato del collegamento*, *Link-State (LS) routing*.

Questo metodo utilizza il termine *link-state* (stato del collegamento) per definire le caratteristiche di un collegamento (un arco) che rappresenta una rete, parte di una internet. In questo algoritmo, il costo associato ad un arco definisce lo stato del collegamento. I collegamenti con costi inferiori sono preferibili rispetto a quelli con costi superiori; se il costo di un collegamento è infinito, significa che esso non esiste più o è stato interrotto.

Link-State Database (LSDB) Per creare un albero a costo minimo utilizzando questo metodo ogni nodo deve avere un *mappa* completa della rete, il che significa che deve conoscere lo stato di ciascun collegamento. La raccolta di stati per tutti i collegamenti viene chiamata *Link-State Database (LSDB)*. Il LSDB è unico per l'intera rete, ma ogni nodo deve averne un duplicato per poter essere in grado di creare l'albero di costo minimo. Il LSDB può essere rappresentato per mezzo di un array bidimensionale (matrice) in cui il valore di ogni cella definisce il costo del collegamento corrispondente.

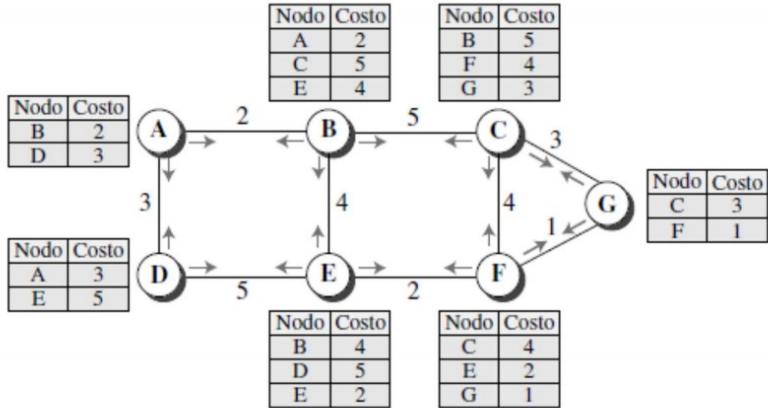


Esempio di un link-state database

	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

b. Il link-state database

Ora il problema è: come può ogni nodo creare la sua copia del LSDB che contenga le informazioni necessarie sull'intera rete? Ciò può essere fatto tramite un procedimento chiamato *flooding* (inondazione). Ogni nodo può inviare dei messaggi di benvenuto a tutti i suoi vicini (i nodi ai quali è collegato direttamente) per raccogliere una coppia di informazioni circa ogni suo vicino: l'identità del nodo e il costo del collegamento. La combinazione di queste due informazioni viene chiamata *pacchetto LS* (LS packet, LSP). L'LSP viene inviato attraverso ogni interfaccia del nodo.



Gli LSP creati ed inviati da ciascun nodo per costruire l'LSDB

Quando un nodo riceve l'LSP da una delle sue interfacce, confronta il nuovo LSP con un'eventuale vecchia copia che potrebbe già avere. Se l'LSP appena arrivato è più vecchio di quello che ha già (e questo è rilevabile confrontando i numeri di sequenza), l'LSP viene scartato. Se è più recente o è il primo ricevuto, il nodo scarta il vecchio LSP (se ce n'è uno) e conserva quello che ha appena ricevuto. A questo punto il nodo invia una copia dell'LSP da ogni sua interfaccia ad esclusione di quella da cui è arrivato il pacchetto. Questo garantisce che il flooding si interrompa da qualche parte nella rete (ad esempio dove un nodo ha una sola interfaccia). Dopo aver ricevuto tutti i nuovi LSP, ogni nodo è in grado di creare la sua copia dell'LSDB globale. Questo LSDB è esattamente uguale per ogni nodo e rappresenta l'intera mappa della rete.

A questo punto possiamo confrontare l'algoritmo di routing a stato del collegamento con quello a vettore distanza. In quest'ultimo, ogni router comunica ai suoi vicini ciò che sa sull'intera rete, mentre nell'algoritmo di routing a stato del collegamento ogni router riferisce all'intera rete ciò che sa dei vicini.

Costruzione degli alberi a costo minimo Per costruire il suo albero a costo minimo utilizzando l'LSDB condiviso, ogni nodo deve eseguire il famoso *algoritmo di Dijkstra*. Questo algoritmo iterativo è composto dai seguenti passi:

1. il nodo sceglie se stesso come radice dell'albero, crea un albero con un singolo nodo e imposta il costo totale di ogni nodo sulla base delle informazioni che trova l'LSDB;

2. il nodo seleziona un altro nodo, tra tutti quelli che non sono presenti nell'albero, in modo che sia il più vicino possibile alla radice, e lo aggiunge all'albero. Dopo che questo nodo è stato aggiunto all'albero, il costo dei nodi non presenti nell'albero deve essere aggiornato in quanto i percorsi potrebbero essere cambiati;
3. il nodo ripete il passaggio 2 finché tutti i nodi non sono stati aggiunti all'albero.

```

1 Initialization of node u:
2    $N' = \{u\}$ 
3   Per tutti i nodi v
4     se v è adiacente a u
5       allora  $D(v) = c(u,v)$ 
6     altrimenti  $D(v) = \infty$ 
7
8 Loop
9   trova w non in  $N'$  tale che  $D(w)$  sia minimo
10  aggiungi w a  $N'$ 
11  aggiorna  $D(v)$  per tutti v adiacenti a w e non in  $N'$  :
12     $D(v) = \min(D(v), D(w) + c(w,v))$ 
13  /* il nuovo costo verso v è il vecchio costo verso v oppure il
costo del cammino minimo noto verso w più il costo da w a v */
14 Finché tutti i nodi sono in  $N'$ 

```



Algoritmo di Dijkstra

Link-state vs Distance vector

<i>Link-state</i>	<i>Distance vector</i>
<i>Complessità dei messaggi</i>	
Con n nodi, E collegamenti, implica l'invio di $O(nE)$ messaggi.	Richiede scambi tra nodi adiacenti. Il tempo di convergenza può variare.
<i>Velocità di convergenza</i>	
$O(n^2)$.	Può convergere lentamente, può presentare cicli d'instradamento, può presentare il problema del conteggio all'infinito.
<i>Robustezza</i>	
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.	Un nodo può comunicare cammini a costo minimo errati a tutte le destinazioni. La tabella di ciascun nodo può essere usata degli altri. Un calcolo errato si può diffondere per l'intera rete.

4.4.3 Protocolli di routing unicast

Struttura di Internet

Prima di parlare dei protocolli di routing unicast dobbiamo comprendere la struttura attuale di Internet. La rete è cambiata, passando da una topologia ad albero, con un'unica dorsale (backbone), a una topologia con dorsali multiple gestite da aziende private. Anche se è piuttosto difficile offrire una visione realistica e generale di Internet oggi, possiamo dire che la sua struttura assomiglia a quanto mostrato nella Figura 4.5.

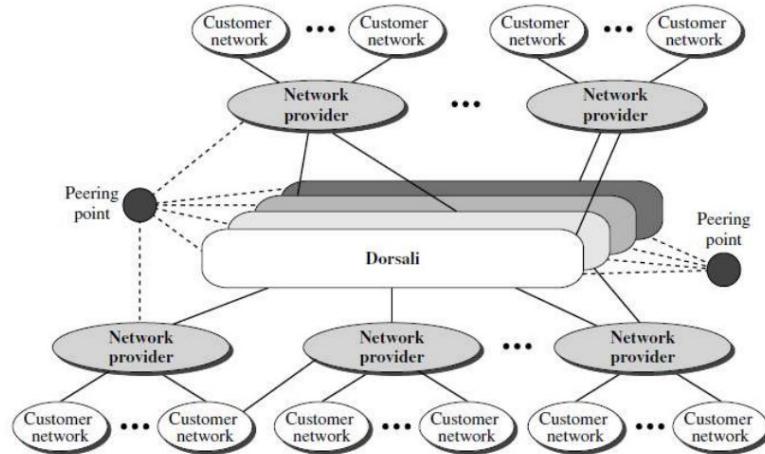


Figura 4.5: Struttura di Internet

Ci sono numerosi dorsali, gestite da società private di telecomunicazioni, che forniscono la connettività globale. Tali dorsali sono connesse tramite alcuni *peering point* che consentono la connettività tra dorsali. A livello inferiore, ci sono dei network provider (fornitori di rete) che utilizzano le dorsali per avere connettività globale ma principalmente forniscono servizi di connettività a clienti Internet. Infine ci sono le customer network (reti dei clienti) che usano i servizi forniti dai network provider. Tutte queste attività (dorsali, network provider o customer network) possono essere definite Internet Service Provider (ISP). In tutti e tre i casi forniscono dei servizi, ma a livelli differenti.

Routing gerarchico Internet oggi è costituita da un numero enorme di reti e di router che le collegano. È ovvio che il routing su Internet non può essere effettuato usando un singolo protocollo, per due ragioni: un problema di scalabilità e un problema amministrativo. Il *problema di scalabilità* è dato dalla dimensione delle tabelle d'inoltro: sono enormi. Quindi, la ricerca di una destinazione in una di queste tabelle farebbe perdere molto tempo e gli aggiornamenti creerebbero una gigantesca quantità di traffico. Il *problema amministrativo* è dato dalla struttura di Internet. Come mostra la Figura 4.5, ogni ISP è un'autorità amministrativa autonoma. L'amministratore deve (e vuole) avere il controllo sul suo sistema. La società proprietaria dell'ISP deve poter utilizzare tutte le sottoreti ed i router di cui ha bisogno, deve poter scegliere di usare router di uno specifico produttore, di voler eseguire un particolare algoritmo di routing che risponde alle esigenze dell'azienda e deve poter imporre una politica specifica sul traffico che passa attraverso l'ISP.

Implementare un routing gerarchico significa considerare ogni ISP come un *sistema autonomo* (*Autonomous System*, AS). Ogni AS può eseguire un protocollo di routing che soddisfa le sue esigenze. A livello di Internet globale, questo non è possibile: è necessario un protocollo di routing globale in grado di unire assieme tutti gli AS. Il protocollo di routing usato all'interno degli AS viene definito *protocollo di routing intra-AS*, *protocollo di routing intra-dominio* (*interdomain routing protocol*), o *Interior Gateway Protocol* (*IGP*); il protocollo di routing globale viene definito *protocollo di routing inter-AS*, *protocollo di routing inter-dominio* (*interdomain routing protocol*) o *Exterior Gateway Protocol* (*EGP*). Ci sono numerosi protocolli di routing intra-dominio, e ogni AS è libero di sceglierne uno, ma dovrebbe essere chiaro che dobbiamo avere un solo protocollo inter-dominio che gestisce il routing tra queste entità. Attualmente i due protocolli di routing intra-dominio più comuni sono RIP e OSPF; l'unico protocollo di routing inter-dominio è il BGP.

I sistemi autonomi Come si è detto in precedenza, ogni ISP è un sistema autonomo quando si tratta di gestire le reti ed i router sotto il suo controllo. Anche se possono esserci AS piccoli, di medie dimensioni e grandi, ad ogni AS viene assegnato un numero identificativo (autonomous number, ASN) dall'ICANN. L'ASN è un numero intero senza segno, di 16 bit, che identifica in modo univoco l'AS. I sistemi autonomi, tuttavia, non sono classificati rispetto alla loro dimensione; vengono classificati a seconda del modo in cui sono connessi agli altri AS. Abbiamo degli AS stub (terminali), degli AS multihomed (a collegamenti multipli) e degli AS di transito. Il tipo influenza il funzionamento del protocollo di routing inter-dominio rispetto all'AS.

- **AS stub.** Un 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 transitino attraverso l'AS. Un buon esempio di AS stub è la rete di una grande azienda che può essere solamente sorgente o destinazione dei dati.
- **AS multihomed.** Un AS multihomed ha più di una connessione con altri AS, ma non consente al traffico dei dati di passare attraverso di esso. Un buon esempio di questo tipo di AS è quello di un'azienda che può usare i servizi di più di un network provider ma come politica impone che gli AS a cui è collegata non possano sfruttare la sua rete per il transito dei loro dati.

- **AS di transito.** Un AS di transito è collegato a vari AS e consente anche il transito del traffico dati. I network provider e le dorsali sono dei buoni esempi di AS di trasito.

Routing Information Protocol (RIP)

Il *Routing Information Protocol (RIP)* è uno dei protocolli di routing intra-dominio più ampiamente usati ed è basato sull'algoritmo di instradamento a vettore distanza che abbiamo descritto in precedenza.

Conto dei salti (hop) In RIP, ogni router a grandi linee implementa l'algoritmo a vettore distanza. Tuttavia, l'algoritmo è stato modificato come descritto in seguito. Per prima cosa, siccome un router in un AS deve sapere come inoltrare un pacchetto alle diverse sottoreti presenti nell'AS, i router RIP rendono noto il costo per raggiungere le diverse sottoreti invece dei costi per raggiungere i nodi del grafo. In altre parole, il costo viene definito tra un router e la rete nella quale si trova l'host di destinazione. Secondo, la gestione dei costi viene gestita in modo molto semplice ed indipendentemente da fattori quali le prestazioni dei router e dei collegamenti, come ad esempio: ritardi, ampiezza della banda, ecc. Quindi, il costo viene definito come il numero di salti (hop), cioè il numero di sottoreti, che un pacchetto deve visitare per andare dal router sorgente all'host di destinazione. Da notare che la rete nel quale si trova l'host sorgente non viene considerata in questo calcolo. Questo perché l'host sorgente non usa alcuna tabella d'inoltro: il pacchetto viene semplicemente consegnato al router di default. La Figura 4.6 mostra un esempio di quanto descritto finora. In RIP il costo massimo di un percorso è di 15 hop, il che significa che il valore 16 rappresenta l'infinito (nessuna connessione). Per tale ragione è possibile usare RIP in sistemi autonomi nei quali il diametro dell'AS non supera i 15 hop.

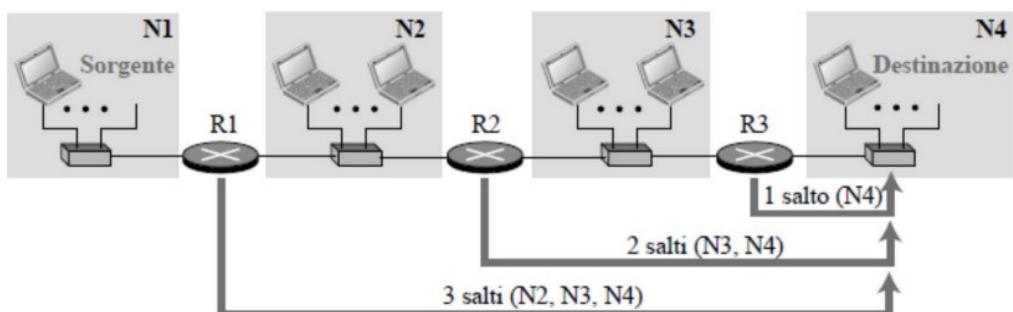


Figura 4.6: Conto degli hop in RIP

Tabelle d'inoltro L'algoritmo a vettore distanza di cui abbiamo parlato nella sezione precedente ha lo scopo principale di scambiare vettori distanza tra nodi vicini. Invece, nel caso dei router di un AS il loro obiettivo è quello di costruire delle tabelle d'inoltro per far giungere i pacchetti alla loro rete di destinazione.

In RIP, una tabella d'inoltro è formata da tre colonne: nella prima c'è l'indirizzo della rete di destinazione, nella seconda l'indirizzo del prossimo router al quale il pacchetto deve essere inoltrato e nella terza il costo (numero di hop) per raggiungere la rete di destinazione. La Figura 4.7 mostra le tre tabelle d'inoltro per i router della Figura 4.6. Si noti che la prima e la terza colonna insieme contengono le stesse informazioni di un vettore distanza, ma con la differenza che in questo caso il costo mostra il numero di salti fino alla rete di destinazione.

Tabella d'inoltro per R1			Tabella d'inoltro per R2			Tabella d'inoltro per R3		
Rete di destinazione	Prossimo router	Costo (in hop)	Rete di destinazione	Prossimo router	Costo (in hop)	Rete di destinazione	Prossimo router	Costo (in hop)
N1	—	1	N1	R1	2	N1	R2	3
N2	—	1	N2	—	1	N2	R2	2
N3	R2	2	N3	—	1	N3	—	1
N4	R2	3	N4	R3	2	N4	—	1

Figura 4.7: Tabelle d'inoltro

Anche se una tabella d'inoltro RIP definisce solo il prossimo router (nella seconda colonna), questa informazione è sufficiente per ricostruire l'intero albero minimo. Ad esempio R1 definisce che il prossimo router per il percorso verso N4 è R2; R2 definisce che il prossimo router per N4 è R3; R3 definisce che non c'è alcun router successivo per tale percorso. Quindi l'albero è R1→R2→R3→N4.

Ci si potrebbe chiedere qual è l'uso della terza colonna della tabella d'inoltro. La terza colonna non è necessaria per inoltrare il pacchetto, ma serve per aggiornare la tabella d'inoltro quando ci sono modifiche del percorso.

Implementazione di RIP Il RIP è implementato per mezzo di un processo demone (sempre attivo) che ascolta sulla porta nota 520 UDP. Nel caso di UNIX BSD tale processo viene chiamato **routed** (come abbreviazione di route daemon). Questo significa che anche se RIP è un protocollo di routing, usato per consentire a IP di instradare i datagrammi attraverso gli AS, i messaggi RIP vengono incapsulati all'interno di datagrammi utente UDP, che a loro volta sono incapsulati in datagrammi IP. In altre parole, RIP lavora a

livello di applicazione ma crea delle tabelle d'inoltro per IP (che si trova a livello di rete).

Gli aggiornamenti vengono inviati ogni 30 secondi (circa) o se la tabella d'inoltro cambia.

Algoritmo RIP Il RIP implementa l'algoritmo d'intradamento a vettore distanza di cui abbiamo parlato nella sezione precedente. Tuttavia sono necessari alcuni cambiamenti all'algoritmo per permettere a un router di aggiornare la sua tabella d'inoltro:

- Invece di inviare solamente vettori distanza, un router deve inviare l'intero contenuto della sua tabella d'inoltro all'interno di un aggiornamento.
- Il destinatario aggiunge un hop a ciascun costo e modifica il campo "router successivo" inserendo l'indirizzo del router mittente. Ogni percorso nella tabella d'inoltro modificata viene definito *percorso ricevuto* e ogni percorso nella vecchia tabella d'inoltro viene chiamato *vecchio percorso*. Il router ricevente seleziona i vecchi percorsi come nuovi ad eccezione dei seguenti tre casi:
 1. se il percorso ricevuto non esiste nella vecchia tabella d'inoltro allora va aggiunto;
 2. se il costo del percorso ricevuto è inferiore al costo di quello vecchio, il percorso ricevuto va selezionato come nuovo percorso;
 3. se il costo del percorso ricevuto è maggiore rispetto a quello del vecchio percorso, ma il valore del prossimo router è lo stesso in entrambi i percorsi, allora il percorso ricevuto va selezionato come nuovo. Questo è il caso in cui il percorso era stato reso noto dallo stesso router in passato, ma ora la situazione è cambiata.
- La nuova tabella d'inoltro deve essere ordinata in base al percorso di destinazione (in ordine di lunghezza del prefisso).

Open Shortest Path First (OSPF)

L'*Open Shortest Path First* è anch'esso un protocollo di routing intra-dominio come il RIP, ma si basa sull'intradamento a stato del collegamento che abbiamo descritto in precedenza. L'OSPF è un protocollo *aperto*, il che significa che la sua specifica è un documento pubblico.

Metrica Nell'OSPF, come nel RIP, il costo per raggiungere una destinazione dall'host si calcola dal router sorgente alla rete di destinazione. Tuttavia, ad ogni collegamento (rete attraversata) può venire assegnato un peso a seconda del suo throughput, tempo di round-trip, affidabilità, ecc. Un amministratore, eventualmente, può anche decidere di usare come metrica il numero di hop. La Figura 4.8 dà l'idea del computo del costo da un router fino alla rete dell'host di destinazione. Possiamo confrontare questa figura con quella relativa al RIP (Figura 4.6).

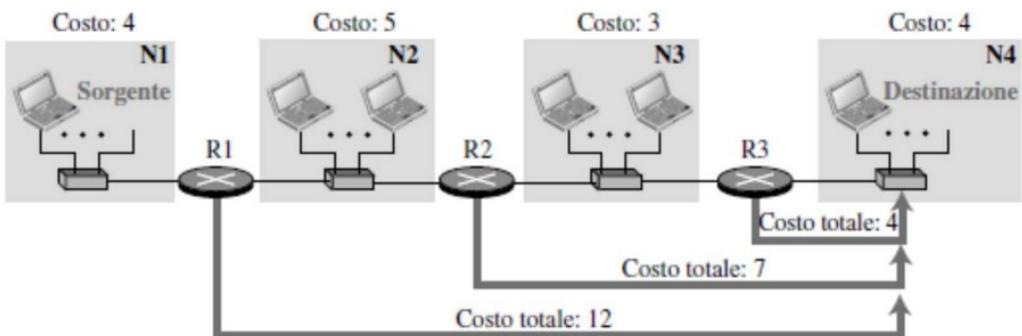


Figura 4.8: Utilizzo dei costi come metrica in OSPF

Tabelle d'inoltro Ogni router OSPF crea una tabella d'inoltro dopo aver trovato l'albero a percorso minimo tra se stesso e la destinazione usando l'algoritmo di Dijkstra. La Figura 4.9 mostra una tabella d'inoltro per il semplice AS della Figura 4.8.

Tabella d'inoltro per R2		
Rete di destinazione	Prossimo router	Costo
N1	R1	9
N2	—	5
N3	—	3
N4	R3	7

Figura 4.9: Tabella d'inoltro dell'OSPF

Confrontando le tabelle d'inoltro per l'OSPF e il RIP nello stesso AS, troviamo che l'unica differenza sono i valori di costo. In altre parole se usiamo il conteggio dei salti per l'OSPF allora le tabelle saranno esattamente identiche. Il motivo è che entrambi i protocolli usano gli alberi a percorso minimo per definire il percorso migliore da una sorgente alla destinazione.

Aree 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, la costruzione degli alberi a percorso minimo nell'OSPF richiede che tutti i router inondino (flooding) l'intero AS con i loro LSP per creare LSDB globale. In altre parole, tutti i router devono inviare il loro LSP a tutti gli altri router dell'AS. Anche se ciò non è un problema negli AS piccoli, potrebbe creare un enorme volume di traffico in AS di medie o grandi dimensioni. Per evitare ciò, l'AS può essere diviso in settori più piccoli chiamati *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 è il sistema autonomo e il secondo è l'area al suo interno.

Tuttavia, ogni router che si trova in un'area ha bisogno di conoscere lo stato dei collegamenti non solo dell'area ma anche delle altre. Per tale ragione una delle aree dell'AS viene definita *area dorsale (backbone area)*, il cui compito è proprio quello di collegare tra loro le varie aree. I router nell'area dorsale hanno la responsabilità di raccogliere le informazioni delle varie aree e comunicarle. In questo modo un router che si trova in un'area può ricevere tutti gli LSP generati nelle altre aree. Per implementare la comunicazione, ogni area è contraddistinta da un identificatore, e l'identificatore della dorsale è lo zero. La Figura 4.10 illustra un AS e le sue aree.

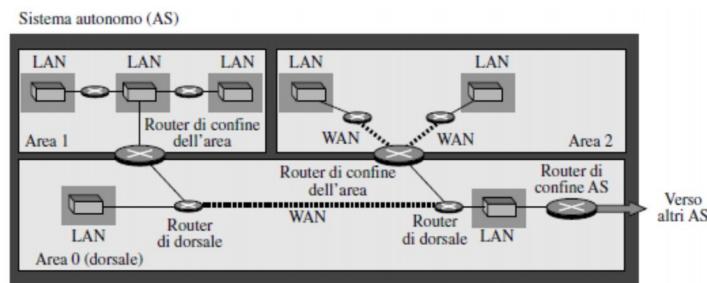
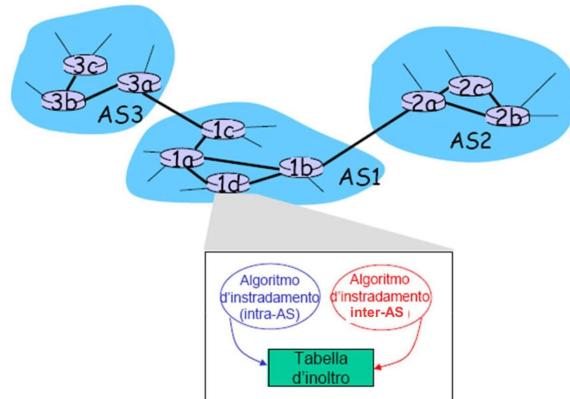


Figura 4.10: Caption

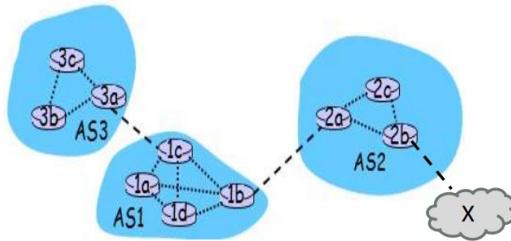
Implementazione dell'OSPF L'OSPF è implementato come applicazione che sfrutta la comunicazione a livello di rete usando direttamente il protocollo IP. I datagrammi IP che incapsulano un messaggio OSPF hanno il valore del campo protocollo impostato a 89. Questo significa che, anche se OSPF è un protocollo di routing usato per permettere l'instradamento dei datagrammi all'interno di un AS, esso stesso è incapsulato direttamente all'interno di datagrammi IP.

Sistemi autonomi interconnessi

Ogni AS nella figura usa uno dei due protocolli d'instradamento intra-domini che abbiamo visto (RIP o l'OSPF). 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 fare ciò viene usato il protocollo BGP4, l'**unico** protocollo INTER-AS usato in Internet, che permette a coppie di router di scambiarsi informazioni su connessioni TCP.



Sistemi autonomi interconnessi



- AS1 scopre (grazie a INTER-AS routing protocol) che una sottorete X è raggiungibile via AS2 (a cui AS1 è collegato mediante il gateway 1b)
- AS1 propaga (con INTER-AS routing protocol) tale informazione al suo interno
- Un router R (es. 1d in figura) di AS1 riceve l'informazione “rete X raggiungibile via AS2” e aggiorna (se necessario) la tabella di inoltro

Per permettere ad ogni router di instradare correttamente i pacchetti, qualsiasi sia la destinazione, è necessario installare su tutti i router di con-

fine (border router) dell'AS una variante del BGP chiamata *BGP esterno* (*external BGP, eBGP*). Tutti i router (non solamente quelli di confine) dovranno invece usare la seconda variante del BGP, chiamata *BGP interno* (*internal BGP, iBGP*). Questo significa che i router di confine devono eseguire ben tre protocolli d'instradamento (intra-dominio, eBGP e iBGP) e che tutti gli altri router ne eseguono due (intra-dominio e iBGP).

- Aggregazione indirizzi: destinazioni rappresentate da prefissi CIDR
- Pubblicizzare un prefisso significa impegnarsi a instradare pacchetti destinati a reti in quel prefisso
- Distribuzione di informazioni su raggiungibilità: un gateway riceve informazioni da gateway di altri AS su sessione eBGP e distribuisce informazioni ai router interni della propria rete su sessioni iBGP. Altri gateway dell'AS possono ri-pubblicizzare info con eBGP.
- Advertisement (ADV) BGP: “route” = prefisso + attributi. I due attributi più importanti sono
 - AS_PATH: sequenza degli AS attraversati nel path pubblicizzato dall'advertisement. Usato per scartare advertisement già ricevuti e scegliere tra più percorsi per lo stesso prefisso
 - NEXT_HOP: indica l'indirizzo IP del primo router lungo un percorso annunciato (al di fuori dell'AS che riceve l'annuncio) a un dato prefisso di rete
- Politiche di importazione: quando un gateway riceve un ADV, usa tali politiche per accettare/rifiutare ADV
- Scelta delle rotte: un router può ricevere più di una rottura per lo stesso prefisso. Sequenza di regole (principale):
 1. Attributo di “preferenza locale” (LOCAL-PREF scelta da amministratore o impostato dai router dell'AS) → vengono selezionati quelli coi valori più alti
 2. Shortest AS-PATH
 3. Closest NEXT-HOP interface (“hot potato routing”)

4.5 IP versione 6

Nell'ultima parte di questo capitolo, si parlerà del protocollo IP di nuova generazione: IPv6. L'esaurimento degli indirizzi e alcune limitazioni di IPv4, già nei primi anni '90, hanno portato allo sviluppo di una nuova versione del protocollo. Questa nuova versione, chiamata *Internet Protocol versione 6* (IPv6) o *IP di nuova generazione* (*IP new generation, IPng*) è nata con lo scopo di aumentare lo spazio degli indirizzi rispetto a IPv4, ridisegnare il formato dei datagrammi IP e allo stesso tempo rivedere anche alcuni protocolli ausiliari come ICMP. È interessante sapere che IPv5 era una proposta basata sul modello OSI ma non si è mai concretizzata. Quanto segue mostra i principali cambiamenti nel protocollo IPv6.

- **Spazio degli indirizzi di dimensione maggiore.** Un indirizzo IPv6 è lungo 128 bit. Paragonato ai 32 bit dell'IPv4, è un enorme aumento (2^{96} volte) dello spazio degli indirizzi.
- **Miglior formato dell'intestazione.** IPv6 utilizza un nuovo formato per l'intestazione (header) dei datagrammi. Nell'intestazione IPv6 le opzioni sono separate dall'intestazione di base e inserite, quando necessario, tra l'intestazione di base e i dati. Ciò semplifica e velocizza il processo di routing visto che la maggior parte delle opzioni non è necessario che sia controllata dai router.
- **Nuove opzioni.** IPv6 supporta nuove opzioni per consentire l'implementazione di funzionalità aggiuntive.
- **Possibilità di estensione.** L'IPv6 è progettato per consentire l'estensione del protocollo se richiesto dall'introduzione di nuove tecnologie o applicazioni.
- **Supporto per l'allocazione delle risorse.** In IPv6 è stato rimosso il campo tipo del servizio (Type Of Service, TOS) che è presente in IPv4. 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. Il meccanismo, se fosse supportato dai router, potrebbe migliorare la gestione del traffico audio o video in tempo reale.
- **Maggiore sicurezza.** Le opzioni relative alla crittografia ed autenticazione permettono di migliorare la riservatezza e verificare l'integrità dei datagrammi.

L'adozione di IPv6 è stata lenta. Il motivo principale che ha portato al suo sviluppo, cioè il progressivo esaurimento degli indirizzi di IPv4, è stato rallentato grazie a tre rimedi a breve termine: l'indirizzamento senza classi, l'uso del protocollo DHCP e soprattutto la diffusione del NAT. Il fatto che si tratti di soluzioni a breve termine, che non risolvono a fondo il problema, e l'uso sempre più ampio di nuovi servizi (per esempio, l'uso del protocollo IP in mobilità, la telefonia basata su IP ecc.) prima o poi richiederanno la sostituzione totale di IPv4 con IPv6. In passato si prevedeva che questa migrazione sarebbe avvenuta entro il 2010, cosa che in effetti non è successa. Le previsioni attuali parlano invece del 2020, ma probabilmente si andrà ben oltre.

4.5.1 Formato dei datagrammi IPv6

Il formato dei datagrammi IPv6 è mostrato nella Figura 4.11. Ciascun datagramma è composto da un'intestazione di base (base header) seguita dai dati (payload). L'intestazione di base occupa 40 byte, mentre il payload può arrivare fino a 65.535 byte. Segue la descrizione dei diversi campi.



Figura 4.11: Formato dei datagrammi IPv6

- **Versione (version).** Il campo versione (4 bit) definisce il numero di versione del protocollo IP. Per l'IPv6, il valore è 6.
- **Classe di traffico (traffic class).** Il campo classe di traffico (8 bit) è utilizzato per distinguere il tipo di servizio di trasferimento a seconda della tipologia di dati contenuti nel payload. Esso sostituisce il campo *tipo di servizio* (Type Of Service, TOS) che è presente in IPv4.
- **Etichetta di flusso (flow label).** L'etichetta di flusso (20 bit) dovrebbe permettere una nuova gestione dei dati sotto forma di flusso di datagrammi, rispetto ai singoli datagrammi indipendenti come accade in IPv4.
- **Lunghezza del payload (payload length).** Il campo lunghezza del payload (2 byte) definisce la lunghezza del datagramma IP escludendo

l'intestazione. Si noti che in IPv4 ci sono due campi legati dalla lunghezza: lunghezza dell'intestazione e lunghezza totale. Invece in IPv6, la lunghezza dell'intestazione di base è fissa (40 byte), quindi è necessario indicare solamente la lunghezza del payload.

- **Prossima intestazione (next header).** La prossima intestazione è un campo (8 bit) che definisce il tipo della prima intestazione estesa (extension header), se presente, o il tipo dei dati che seguono l'intestazione di base all'interno del payload. Questo campo è simile al campo "protocollo" che è presente in IPv4.
- **Hop limit (numero massimo di salti).** Il campo hop limit ha la stessa funzione del campo TTL in IPv4.
- **Indirizzo sorgente e indirizzo destinazione (source and destination address).** I campi indirizzo sorgente e destinazione sono indirizzi Internet da 16 byte (128 bit) che identificano la sorgente originale del datagramma e la sua destinazione.
- **Dati (payload).** Se confrontato con IPv4, il campo payload in IPv6 ha una forma e un significato differenti.

Il campo dati in IPv6 può contenere una combinazione di zero o più intestazioni estese (usate per rappresentare le varie opzioni) seguite dai dati provenienti da altri protocolli (UDP, TCP e così via). In IPv6, le opzioni, che fanno parte dell'intestazione in IPv4, sono implementate sotto forma di intestazioni estese. Il campo dati può contenere tante intestazioni quante ne sono necessarie. Ogni intestazione è formata da due campi obbligatori, intestazione successiva (next header) e lunghezza (length), seguite dalle informazioni specifiche delle varie opzioni. Si noti l'uso che viene fatto del campo "prossima intestazione" (next header): definisce il tipo della prossima intestazione quando si riferisce a un'opzione oppure identifica il protocollo incapsulato nel payload del datagramma.

Concetto di flusso e priorità in IPv6

Il protocollo IP era inizialmente progettato come protocollo senza connessione. Tuttavia, soprattutto a causa di alcune tecnologie di livello inferiore, c'è una certa tendenza ad usare IP come protocollo orientato alla connessione. Per questo motivo, nella versione 6 è stata aggiunta un'etichetta di flusso nell'intestazione che permette questa forma di utilizzo. Ovviamente questo è possibile solamente quando i router offrono il supporto per questa modalità particolare di gestione dei datagrammi.

In un router, un flusso non è altro che una sequenza di datagrammi che hanno le stesse caratteristiche (come ad esempio il percorso, l'uso delle stesse risorse, gli stessi requisiti di sicurezza ecc.). Un router che supporta la gestione delle etichette di flusso possiede una tabella delle etichette di flusso. Questa tabella ha una voce per ogni etichetta di flusso attiva, ognuna di queste voci identifica i requisiti di servizio richiesti dal quel particolare flusso. In questo modo, quando un router riceve un datagramma, consulta la tabella delle etichette di flusso per trovare la voce corrispondente al flusso indicato nell'intestazione del datagramma ed identifica quindi i suoi requisiti. Ovviamente l'etichetta di flusso indicata nel datagramma non specifica direttamente quali sono questi requisiti; queste informazioni dovranno essere propagate ai router per mezzo di altre modalità come ad esempio l'uso di opzioni oppure altri protocolli.

Nella sua forma più semplice un'etichetta di flusso può essere utilizzata per accelerare l'elaborazione dei datagrammi da parte del router. Quando un router riceve un datagramma, invece di consultare la tabella d'inoltro e dover eseguire un algoritmo d'instradamento per determinare l'indirizzo del salto successivo, può semplicemente accedere alla tabella delle etichette di flusso per trovare il salto successivo.

Nella sua forma più sofisticata, un'etichetta di flusso può essere usata per fornire un miglior supporto alla trasmissione di audio e video in tempo reale. Le applicazioni multimediali richiedono risorse come un'elevata ampiezza di banda, grandi buffer, potenza di calcolo e così via. Un'applicazione di questo tipo potrebbe riservare in anticipo le risorse di rete necessarie per garantire che i dati non siano ritardati per mancanza di risorse.

Frammentazione e riassemblaggio

In IPv6 la frammentazione e il riassemblaggio esistono ancora ma rispetto ad IPv4 c'è una differenza fondamentale. I datagrammi IPv6 possono essere frammentati solo dalla sorgente e non dai router lungo il percorso. Ovviamente, anche in questo caso, il riassemblaggio avviene solamente alla destinazione. La frammentazione dei datagrammi nei router non è consentita per velocizzare l'elaborazione da parte del router; è bene ricordare che la frammentazione di un datagramma richiede un'elaborazione piuttosto complessa visto che comporta la modifica di vari campi dell'intestazione. In IPv6, l'host sorgente può verificare la dimensione del datagramma e verificare se frammentarlo o meno. Quando un router riceve un datagramma, controlla la sua dimensione e lo scarta nel caso sia maggiore rispetto a quanto consentito dalla MTU della rete in cui deve inoltrarlo. Inoltre, il router invierà un mes-

saggio d'errore ICMPv6 (di tipo: pacchetto troppo grande) alla sorgente del datagramma per informarla dell'accaduto.

4.5.2 Passaggio da IPv4 a IPv6

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. La prima soluzione che viene in mente, la più banale, è definire un giorno di passaggio nel quale tutti gli host e router smetteranno di usare IPv4 ed inizieranno ad usare solo IPv6. Ovviamente la transizione non può essere fatta in questo modo. A causa dell'enorme numero di sistemi in rete la migrazione non può avvenire improvvisamente. Ci sarà bisogno di una quantità di tempo considerevole prima che tutti i sistemi siano aggiornati. La transizione dovrà essere studiata accuratamente per evitare problemi tra il vecchio sistema e quello nuovo. L'IETF ha studiato tre strategie che possono essere implementate in questa fase: il dual stack (doppia pila di protocolli), il tunnelling e la traduzione dell'intestazione (header translation).

Dual stack

La strategia prevede che tutti gli host, durante la transizione, abbiano una doppia pila di protocolli per la comunicazione in rete. In altre parole, ciascun host (o router) deve eseguire IPv4 e IPv6 simultaneamente fino a quando tutta la rete non sarà passata ad IPv6. Questa strategia è rappresentata in Figura 4.12.

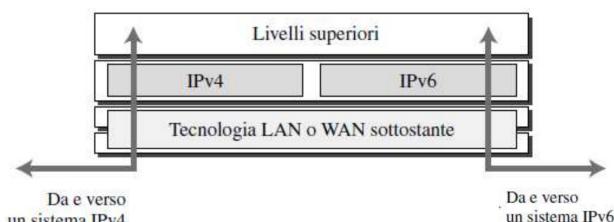
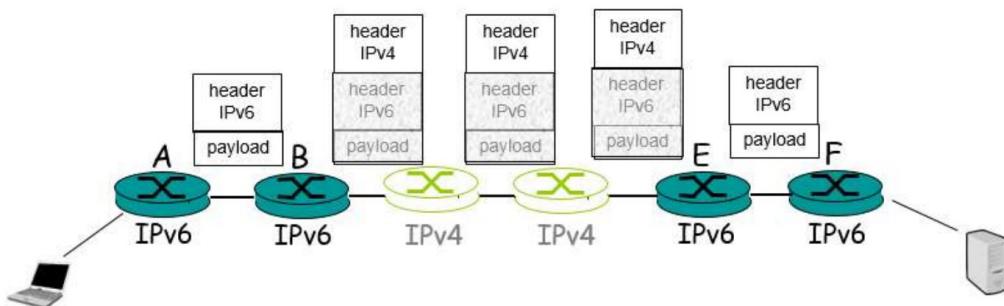


Figura 4.12: Dual stack

Per determinare quale versione utilizzare quando si invia un pacchetto a una destinazione, l'host sorgente interroga il DNS. Se il DNS restituisce un indirizzo IPv4 allora l'host invia un pacchetto IPv4. Se il DNS restituisce un indirizzo IPv6 allora l'host invia un pacchetto IPv6.

Tunneling Il tunneling è una tecnica utilizzata quando due host che usano IPv6 vogliono comunicare tra loro ma i datagrammi devono attraversare una regione della rete che usa solo IPv4. Per passare attraverso questa regione i datagrammi devono necessariamente avere un indirizzo IPv4. Una volta superata la tratta IPv4, il datagramma IPv6 potrà essere nuovamente estratto. È come se i datagrammi IPv6 viaggiassero all'interno di un tunnel tra due isole IPv6. Per rendere esplicito il fatto che il datagramma IPv4 sta trasportando un datagramma IPv6, il valore del protocollo nell'intestazione del datagramma IPv4 viene impostato a 41.



Strategia di tunneling

Traduzione dell'intestazione La traduzione dell'intestazione (header translation) sarà necessaria quando la maggior parte di Internet sarà migrata ad IPv6, con alcuni sistemi residui ancora basati su IPv4. Supponiamo che il mittente intenda (o debba) usare IPv6, ma che il ricevente non sia in grado di supportarlo. In questo caso il meccanismo di tunneling non funziona. L'unica alternativa è quella di effettuare una traduzione completa dell'intestazione del datagramma, in modo di convertirla da IPv6 a IPv4, prima che il datagramma arrivi a destinazione.

4.6 Approfondimenti

4.6.1 Checksum

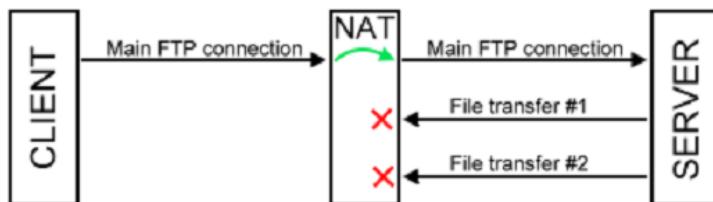
Per quale motivo UDP e TCP si preoccupano di controllare l'integrità di informazioni di livello network se anche IP usa un suo checksum?

1. Ragioni "storiche"
2. Checksum UDP e TCP controlla anche il payload ed è una checksum end-to-end
3. UDP e TCP potrebbero non usare IP come servizio di rete

4.6.2 NAT: problemi

FTP

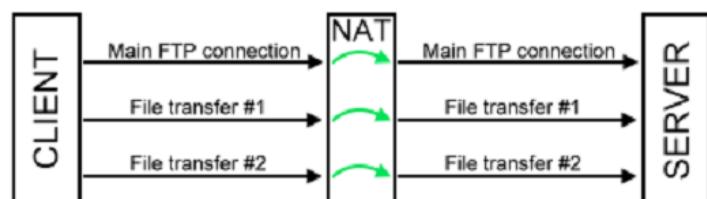
Cosa succede se un client FTP sotto NAT invia RETR e il router NAT del client FTP non accetta richieste di connessioni TCP?



FTP active mode transfers through a NAT router

Poiché la connessione principale è in uscita, il firewall NAT consente di stabilire questa connessione, ma quando il server tenta di riconnettersi al client viene bloccato dal firewall.

La tecnica chiamata "modalità passiva" o PASV è stata introdotta per ridurre questo problema. In questo schema le connessioni vengono sempre effettuate dal client al server e non viceversa.



FTP passive mode transfers through a NAT router

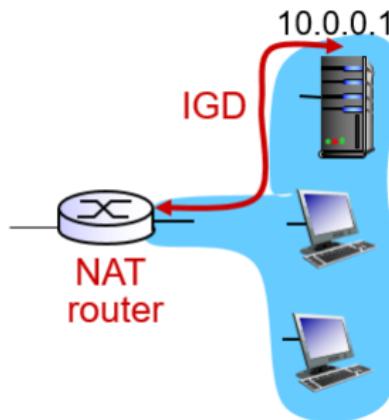
Questo è il motivo per cui la modalità passiva è generalmente preferibile quando sono coinvolti firewall NAT.

P2P

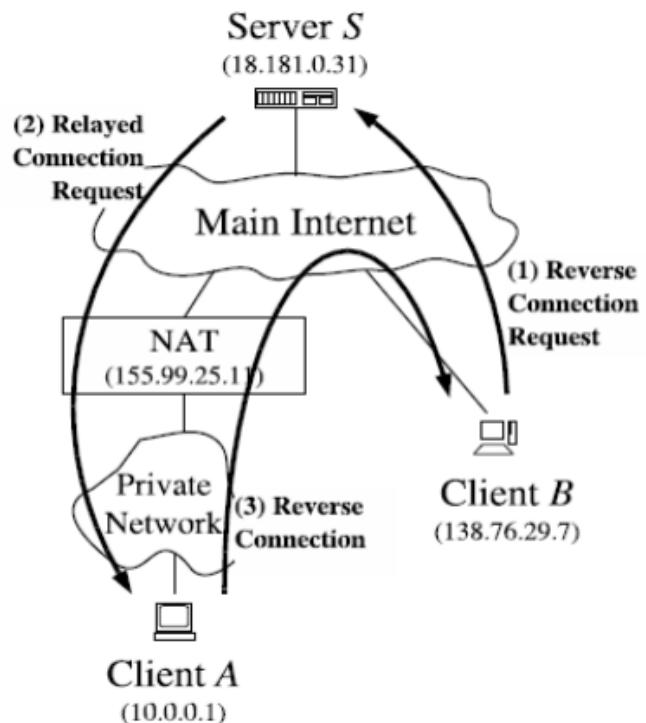
In una applicazione P2P come può un peer chiedere di utilizzare una certa porta per poter essere contattato?

In un'applicazione P2P, qualsiasi Peer A partecipante dovrebbe essere in grado di avviare una connessione TCP a qualsiasi altro Peer B partecipante. L'essenza del problema è che **se il peer B è dietro un NAT, non può agire come un server e accettare connessioni TCP**.

1. Configurare staticamente il NAT per inoltrare le richieste di connessione in entrata su una determinata porta al tuo server.
2. Protocollo Internet Gateway Device (IGD). Consente all'host con NAT di aggiungere/rimuovere mappature delle porte (con tempi di lease) (ad esempio, automatizzare la configurazione statica della mappa delle porte NAT).



3. **Connection reversal.** Questo problema NAT può essere aggirato se il Peer B non è dietro un NAT. In questo caso, il Peer B può prima contattare il Peer A tramite un Peer intermedio (Server S nella figura), che non è dietro un NAT e con il quale A ha stabilito una connessione TCP in corso. Il peer B può quindi chiedere al peer A, tramite il server S, di avviare una connessione TCP direttamente al peer B. Una volta stabilita la connessione TCP P2P diretta tra i peer A e B, i due peer possono scambiarsi messaggi o file.



Connection reversal

Capitolo 5

Livello di collegamento

Nei Capitoli 2, 3 e 4 si è parlato del livello di applicazione, del livello di trasporto e del livello di rete. In questo capitolo parleremo invece del livello di collegamento. La pila dei protocolli TCP/IP non definisce alcun protocollo a livello di collegamento o fisico, in quanto questi sono argomenti di competenza delle singole reti che costituiscono Internet. È evidente come sul mercato ormai vi siano un gran numero di protocolli e tecnologie per il livello di collegamento.

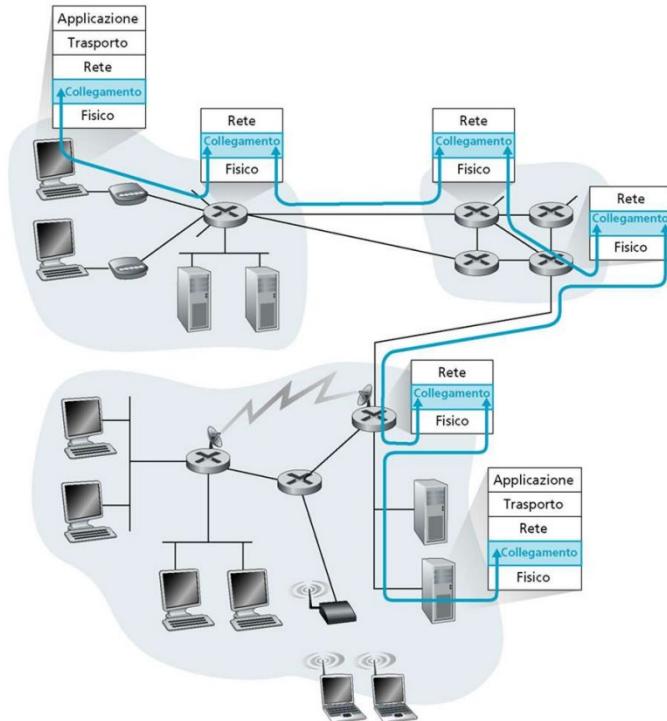
5.1 Introduzione

Nel Capitolo 4 si è visto che la comunicazione a livello di rete avviene tra host. Un datagramma è un'unità dati che può essere frammentata e riassorbita ma che in generale viene inviata da un host, che si trova da qualche parte nel mondo, ad un altro host, che si trova in qualche altra zona. Internet non è altro che una combinazione di reti unite assieme da dispositivi di collegamento (per esempio, router e switch). Se un datagramma deve viaggiare da un host ad un altro, deve quindi passare necessariamente attraverso tali reti.

5.1.1 Nodi e collegamenti

Anche se la comunicazione nei livelli di applicazione, trasporto e rete è end-to-end (cioè avviene, dal punto di vista logico, dall'host iniziale a quello finale), quella a livello di collegamento è invece da nodo a nodo. Come si è visto nei capitoli precedenti, un'unità di dati (cioè un pacchetto) trasmessa da un certo punto della rete per arrivare fino alla sua destinazione deve attraversare molte altre reti (LAN e WAN) collegate per mezzo di router. È consuetudine

riferirsi ai router e agli host alle due estremità come a *nodi* e alle reti nel mezzo come a *collegamenti*.



Comunicazione a livello di collegamento

5.1.2 Due tipi di collegamento

I nodi all'interno delle reti sono fisicamente collegati da un mezzo trasmittivo, come un cavo o l'aria. Compito del livello di collegamento è controllare come viene usato tale mezzo. Possiamo avere un livello di collegamento che utilizza l'intera capacità di un mezzo oppure solo una parte della sua capacità. In altre parole, possiamo avere un *collegamento punto-punto* o un *collegamento broadcast*. In un collegamento punto-punto, il collegamento è dedicato a due soli dispositivi (mittente e ricevente). Viceversa, in un collegamento broadcast, il collegamento è condiviso tra varie coppie di dispositivi. Per esempio, quando due amici usano il telefono di casa per chiacchierare, stanno usando un collegamento punto-punto. Quando gli stessi due amici usano i loro telefoni cellulari, stanno usando un collegamento broadcast (visto che l'aria viene condivisa tra vari utilizzatori di cellulari).

5.1.3 Due sotto-livelli

Per comprendere meglio le funzionalità ed i servizi forniti dal livello di collegamento possiamo dividere questo livello in due sotto-livelli: il *Data-Link Control (DLC)* ed il *controllo dell'accesso al mezzo trasmisivo (Media Access Control, MAC)*. Il sotto-livello DLC si occupa di tutte le questioni comuni sia ai collegamenti punto-punto che a quelli broadcast. Il sotto-livello MAC si occupa invece solo degli aspetti specifici dei canali broadcast

5.1.4 Servizi offerti

Framing

Trasmettere i dati, a livello fisico, significa inviare i bit sotto forma di segnali dalla sorgente alla destinazione. Il livello fisico fornisce la necessaria sincronizzazione in modo che il mittente e il ricevente utilizzino la stessa durata, e temporizzazione, per i bit.

Il livello collegamento, inoltre, ha la necessità di raggruppare i bit all'interno di frame, questo per fare in modo che sia possibile stabilire un ordine tra i bit e distinguerli gli uni dagli altri. Il sistema postale che tutti conosciamo implementa una sorta di framing. Il semplice fatto di inserire una lettera in una busta separa un'informazione da un'altra: la busta funge da delimitatore tra le varie lettere. Inoltre, ogni busta riporta l'indirizzo del mittente e quello del ricevente. Entrambi questi indirizzi sono necessari visto che il sistema postale è un servizio di consegna multi-a-molti.

Il *framing*, a livello di collegamento, ha il compito di separare i vari messaggi durante la trasmissione da una sorgente a una destinazione. Opera aggiungendo a singoli frame sia l'indirizzo del mittente che quello della destinazione. L'indirizzo di destinazione indica dove deve andare il pacchetto, quello del mittente serve invece al ricevente per poter generare un riscontro dell'avvenuta ricezione (*acknowledgment*).

Anche se un intero messaggio (ad esempio di livello applicazione) potrebbe essere inserito in un unico frame, normalmente questo non avviene. Una ragione è che, in presenza di messaggi grandi, il frame diventerebbe molto grande, rendendo inefficienti i controlli del flusso e degli errori. Inoltre, il trasporto di un messaggio in un frame di grandi dimensioni avrebbe come effetto collaterale che anche l'errore su un singolo bit provocherebbe la necessità di ritrasmettere l'intero frame. Quando invece il messaggio viene suddiviso in frame più piccoli, l'errore su un singolo bit coinvolge solamente quel piccolo frame (che dovrà essere ritrasmesso).

Consegna affidabile

- È considerata non necessaria nei collegamenti che presentano un basso numero di errori sui bit (fibra ottica, cavo coassiale e doppino intrecciato).
- È spesso utilizzata nei collegamenti soggetti a elevati tassi di errori (es.: collegamenti wireless).

Controllo di flusso

- Evita che il nodo trasmittente saturi quello ricevente.

Rilevazione degli errori

- Gli errori sono causati dall'attenuazione del segnale e da rumore elettromagnetico.
- Il nodo ricevente individua la presenza di errori: è possibile grazie all'inserimento, da parte del nodo trasmittente, di bit di controllo di errore all'interno del frame.

Correzione degli errori

- Il nodo ricevente determina anche il punto in cui si è verificato l'errore e lo corregge

5.2 Indirizzamento a livello di collegamento

Il prossimo argomento da affrontare sono gli indirizzi di livello collegamento. Nel capitolo 4 si è parlato degli indirizzi IP come identificatori del livello di rete, identificatori che individuano con esattezza i punti di Internet dove sono connessi gli host sorgente e destinazione. Tuttavia, in una rete senza connessione come Internet non è possibile far sì che un datagramma raggiunga la sua destinazione solamente usando gli indirizzi IP. La ragione è che ogni datagramma in Internet, dallo stesso host sorgente allo stesso host destinazione, può prendere un percorso diverso. Gli indirizzi IP sorgente e destinazione definiscono le due estremità della rete, ma non dicono attraverso quali collegamenti deve passare il datagramma.

È bene anche ricordare che gli indirizzi IP in un datagramma non dovrebbero essere modificati durante il trasferimento. Se in un datagramma cambia

l'indirizzo IP di destinazione, il pacchetto non raggiungerà la sua destinazione; se cambia quello della sorgente, l'host di destinazione o i router non potranno comunicare la risposta alla sorgente o riportare eventuali errori.

Quanto sopra indica che, in una rete senza connessione, è necessario avere un ulteriore meccanismo di indirizzamento: gli indirizzi di livello collegamento. Un *indirizzo di livello collegamento* (indirizzo di collegamento, link-layer address, link address) è spesso chiamato anche indirizzo fisico oppure indirizzo MAC (MAC address).

I collegamenti sono ovviamente controllati dal livello di collegamento e utilizzano indirizzi di questo livello. Quando un datagramma passa dal livello di rete al livello di collegamento, esso viene racchiuso all'interno di un frame e viene aggiunta un'intestazione che contiene due indirizzi di livello collegamento (sorgente e destinazione del frame). Questi due indirizzi vengono cambiati ogni volta che il frame passa da un collegamento a un altro.

5.2.1 Address Resolution Protocol (ARP)

Assumiamo che su Internet ci sia un nodo che intende comunicare con un altro nodo e che questa coppia di nodi sia separata da un certo numero di router. Il nodo mittente ovviamente conosce l'indirizzo IP della destinazione ed inoltre è a conoscenza del router di default che si trova all'interno della sua rete locale. Ogni router, tranne l'ultimo, nel percorso tra sorgente e destinazione ricava l'indirizzo IP del router successivo utilizzando la tabella d'inoltro. L'ultimo router conosce l'indirizzo IP dell'host di destinazione. Tuttavia, l'indirizzo IP del nodo successivo non è sufficiente per la trasmissione di un frame attraverso un collegamento: è necessario avere l'indirizzo di collegamento del nodo successivo.

L'*Address Resolution Protocol (ARP)* serve proprio per questo scopo. Il protocollo ARP è uno dei protocolli ausiliari definiti a livello di rete, come mostrato nella Figura 5.1

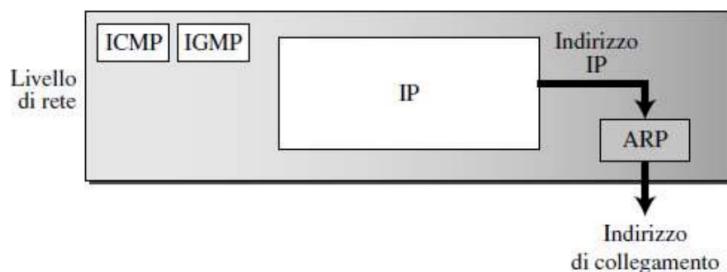


Figura 5.1: Posizione dell'ARP nella pila TCP/IP.

Appartiene al livello di rete, ma abbiamo deciso di rimandare la sua analisi fino ad ora proprio perché il suo scopo è associare gli indirizzi IP agli indirizzi di collegamento. L'ARP accetta in input un indirizzo IP, individua l'indirizzo di collegamento corrispondente e lo passa al livello di collegamento.

Ogni volta che un host o un router deve trovare l'indirizzo di collegamento di un altro host o router che si trova all'interno della sua rete, invia un pacchetto di richiesta ARP che comprende l'indirizzo di collegamento, l'indirizzo IP del mittente e l'indirizzo IP del ricevente. Siccome il mittente non conosce ancora l'indirizzo di collegamento del ricevente, la richiesta viene trasmessa usando un broadcast a livello di collegamento. L'invio broadcast è ottenuto per mezzo del corrispondente indirizzo di broadcast (vedi Figura 5.2), in modo che la richiesta raggiunga tutti i nodi all'interno della rete locale.

Ogni host o router nella rete locale riceve ed elabora il pacchetto di richiesta ARP, ma solo il ricevente designato riconosce il suo indirizzo IP e restituisce un pacchetto di risposta ARP che contiene i suoi indirizzi IP e di collegamento. Il pacchetto di risposta viene inviato in modalità unicast direttamente al nodo che ha inviato la richiesta.

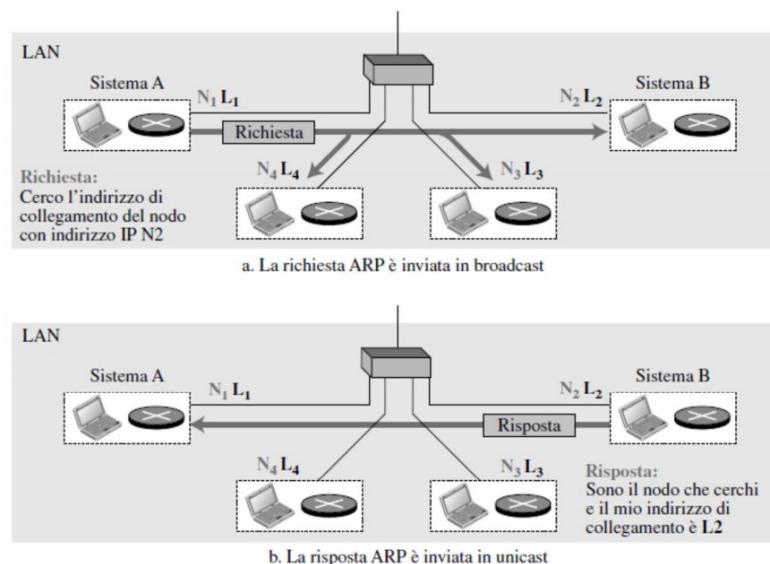


Figura 5.2: Funzionamento dell'ARP

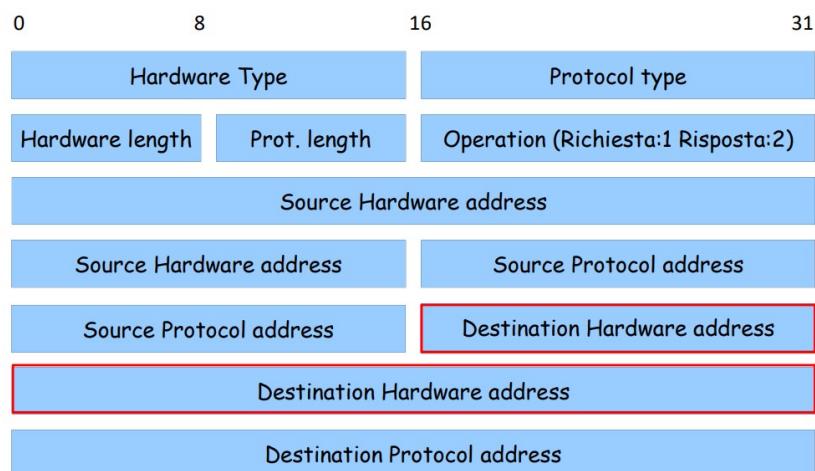
Tabella ARP

- Ogni nodo IP (host, router) nella LAN ha una tabella ARP.
- Tabella ARP: contiene la corrispondenza tra indirizzi IP e MAC.
 - <Indirizzo IP; Indirizzo MAC; TTL>
- TTL (tempo di vita): valore che indica quando bisognerà eliminare una data voce nella tabella (il tempo di vita tipico è di 20 min)
- Poiché ARP risolve gli indirizzi IP solo per i nodi della stessa LAN, le tabelle ARP contengono la corrispondenza fra indirizzi IP e indirizzi MAC per i nodi della stessa sottorete.
- La tabella non contiene necessariamente le corrispondenze per tutti i nodi della sottorete.

Indirizzo IP	Indirizzo LAN	TTL
222.222.222.221	88-B2-2F-54-1A-0F	13:45:00
222.222.222.223	5C-66-AB-90-75-B1	13:52:00

Tabella ARP

Formato del pacchetto ARP



Formato del pacchetto ARP.

Il campo *hardware type* definisce il tipo del protocollo di livello collegamento che viene usato, ad Ethernet viene assegnato il tipo 1. Il campo *protocol type* definisce il protocollo del livello di rete: il codice del protocollo IPv4 è $(0800)_{16}$.

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 nel nostro esempio). Entrambi i campi, per i motivi discussi in precedenza, sono di lunghezza variabile (lunghezze definite per mezzo di *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.

Forwarding diretto/indiretto

Forwarding diretto: vedi Figura 5.2

Forwarding indiretto: invio di un datagramma con mittente IP di A e destinatario IP di B. Ipotesi:

- A conosce l'indirizzo IP di B
 - A conosce l'indirizzo IP del primo router → DHCP
 - A conosce l'indirizzo MAC di R → ARP
1. A crea un datagramma IP con IP sorgente A, destinazione B
 - (a) A crea un frame con indirizzo MAC destinazione il MAC address di R, il frame incapsula il datagramma IP (A-to-B)
 2. frame inviato da A a R
 3. frame ricevuto da R, decapsulato, il datagramma passa al livello IP
 - (a) R estrae il datagramma IP dal frame Ethernet, e vede che la sua destinazione è B.
 - (b) R usa ARP per ottenere l'indirizzo MAC di B
 - (c) R crea un frame con destinazione il MAC address di B
 4. R inoltra il datagramma con IP sorgente A, destinazione B

5.3 LAN cablate: protocollo Ethernet

5.3.1 Progetto IEEE 802

Abbiamo iniziato a parlare del protocollo Ethernet e delle sue varie generazioni. Prima di approfondire in dettaglio le sue caratteristiche è però necessario parlare brevemente dello standard IEEE che viene spesso citato nei documenti tecnici. Nel 1985, la IEEE Computer Society (Institute of Electrical and Electronics Engineers) iniziò un progetto, chiamato *Progetto 802*, con l’obiettivo di definire uno standard per l’interconnessione tra dispositivi di produttori differenti. Il Progetto 802 non aveva lo scopo di sostituire una qualsiasi parte del modello OSI o la pila TCP/IP, ma piuttosto intendeva definire chiaramente le funzioni specifiche del livello fisico e di collegamento dei protocolli LAN. Questo lavoro ha portato alla definizione dell’IEEE Standard 802.

5.3.2 Ethernet Standard

In questa parte del testo si fa riferimento alla tecnologia Ethernet originale con velocità di trasferimento dei dati a 10 Mbps (Ethernet Standard). Nonostante l’evoluzione di Ethernet abbia portato a molti cambiamenti nei meccanismi utilizzati, alcune caratteristiche sono rimaste invariate nel corso dell’evoluzione. Parliamo quindi della versione standard per comprendere meglio le versioni successive.

Formato dei frame

Il frame Ethernet contiene sei campi, come mostrato nella Figura 5.3.

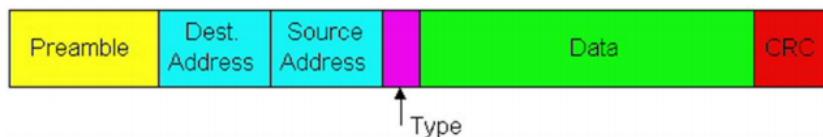


Figura 5.3: Struttura dei frame Ethernet.

- **Preamble (preambolo).** I pacchetti Ethernet iniziano con un campo di otto byte: sette hanno i bit 10101010 e l’ultimo è 10101011. Servono per “attivare” gli adattatori dei riceventi e sincronizzare i loro orologi con quello del trasmittente.
- **Destination Address (DA, Indirizzo di destinazione).** Il campo (6 byte quindi 48) contiene l’indirizzo di collegamento della stazione

o delle stazioni destinatarie del frame. Quando la stazione ricevente trova in questo campo il suo stesso indirizzo di livello collegamento (unicast), l'indirizzo multicast di un gruppo di cui fa parte, o l'indirizzo broadcast, estraе i dati dal frame e li passa al protocollo di livello superiore (definito dal valore del campo Type).

- **Source Address (SA, Indirizzo della sorgente)**. Anche questo campo è di 6 byte e contiene l'indirizzo di collegamento del mittente del frame.
- **Type (Type)**. Questo campo definisce il protocollo di livello superiore del pacchetto incapsulato all'interno del frame. Il protocollo può essere IP, ARP, e così via. In altre parole, ha lo stesso scopo del campo protocollo nell'intestazione dei datagrammi e del numero di porta in un segmento TCP o in un datagramma utente (UDP). È quindi necessario per il multiplexing e il demultiplexing a livello di collegamento.
- **Data and padding (Dati)**. Questo campo trasporta i dati incapsulati nel frame dai protocolli di livello superiore. Ha un minimo di 46 byte e un massimo di 1500 byte. Se i dati che provengono dal livello superiore superano i 1500 byte, essi devono essere frammentati e quindi racchiusi in più frame. Se sono meno di 46 byte, è necessario aggiungere tanti 0 quanti sono necessari per arrivare alla dimensione minima (*padding*). Una volta decapsulati i dati contenuti nel frame vengono consegnati così come sono al protocollo di livello superiore, quindi senza rimuovere il padding (gli 0 aggiuntivi). Ciò significa che la rimozione o l'aggiunta del padding è responsabilità del livello superiore. Questo obbliga il protocollo di livello superiore a conoscere la vera lunghezza dei dati inviati, in caso contrario non sarebbe in grado di distinguere i suoi dati dagli 0 aggiuntivi. Proprio per questo motivo, nell'intestazione dei datagrammi è presente un campo che definisce la dimensione dei dati contenuti nel payload del datagramma.
- **CRC**. L'ultimo campo contiene informazioni per il rilevamento errori. Il codice di ridondanza contenuto in questo campo viene calcolato sui campi indirizzo, tipo e dati. Se il ricevente lo calcola e constata che non è 0, allora il frame è stato alterato durante la trasmissione, e quindi il ricevente si limita a scartarlo.

Servizio senza connessione e inaffidabile

Ethernet fornisce un servizio senza connessione, questo significa che ogni frame inviato è indipendente dal frame precedente e dal successivo. In Ethernet

non c'è alcuna fase di apertura o chiusura della connessione, 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. Può accadere che il mittente sommerga il ricevente di frame, con il risultato che alcuni di questi vengano scartati. Nel caso in cui un frame venga scartato, 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 di 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. Se un frame viene danneggiato durante la trasmissione e il ricevente riscontra il danneggiamento (grazie al codice di ridondanza contenuto nell'intestazione del frame), il ricevente si limiterà a scartare il frame, senza alcun'altra azione. È dovere dei protocolli di livello superiore riscontrare la perdita dei dati e porvi rimedio.

Indirizzo

Tutte le stazioni che fanno parte di una rete Ethernet (ad esempio un computer, una postazione di lavoro o una stampante) sono dotate di una *Network Interface Card (NIC)*, comunemente detta *scheda di rete*. La NIC viene installata nella stazione e fornisce un suo indirizzo di livello collegamento. Gli indirizzi Ethernet sono composti da 6 byte (48 bit) e vengono normalmente scritti in notazione esadecimale, con i due punti a dividere i vari byte. Ad esempio quello che segue è un indirizzo MAC Ethernet

4A:30:10:21:10:1A

Come viene garantita l'univocità?

- IEEE definisce ed assegna i primi 24 bit (*OUI – Organization Unique Identifier*), mentre i rimanenti 24 bit vengono gestiti dalle aziende ed assegnati a livello locale.
- Quando una società vuole costruire adattatori, compra un blocco di spazio di indirizzi (univocità degli indirizzi).

Indirizzi unicast, multicast e broadcast

Un indirizzo sorgente è sempre un *indirizzo unicast*, poiché il frame proviene da una sola stazione. Tuttavia, l'indirizzo di destinazione può essere *unicast*, *multicast* o *broadcast*. Se il bit meno significativo del primo byte nell'indirizzo di destinazione è 0 allora l'indirizzo è unicast, in caso contrario è multicast.

L'indirizzo broadcast è un caso particolare dell'indirizzo multicast: i destinatari sono tutte le stazioni che si trovano all'interno della LAN. Un indirizzo di destinazione broadcast è formato solamente da 1 (cioè 48 bit impostati a 1).

Distinzione tra trasmissione unicast, multicast e broadcast

La tecnologia Ethernet Standard utilizza un cavo coassiale (topologia bus) o una serie di doppini intrecciati (cioè coppie di cavi di rame intrecciati). La topologia della rete prevede un hub al centro della stella.

Nell'Ethernet Standard la trasmissione è sempre broadcast, indipendente dal fatto che si tratti di frame con destinazione unicast, multicast o broadcast. Nella topologia a bus, quando la stazione A invia un frame alla stazione B, tutte le stazioni lo ricevono. Nella topologia a stella, quando la stazione A invia un frame alla stazione B, lo riceverà l'hub. Siccome gli hub sono apparati del tutto passivi, non verificano l'indirizzo di destinazione del frame, ma si limitano a rigenerare il segnale dei bit e a reinviare il frame a tutte le stazioni, ad eccezione di quella da cui l'hanno ricevuto (in questo caso A).

Unicast, multicast e broadcast sono trattati diversamente dai riceventi:

- in una trasmissione unicast, tutte le stazioni ricevono il frame. Il destinatario designato lo memorizza e lo gestisce, tutte le altre stazioni lo scartano;
- in una trasmissione multicast, tutte le stazioni ricevono il frame. Quelle che appartengono al gruppo multicast lo memorizzano e lo gestiscono, tutte le altre stazioni lo scartano;
- in una trasmissione broadcast, tutte le stazioni (eccetto il mittente) ricevono il frame, lo memorizzano e lo gestiscono.

5.3.3 Fast Ethernet

- 100 Mbps
- Mantiene intatti formato e dimensione min/max frame

- Dimensione minima immutata e trasmissione 10 volte più veloce → rete più corta, due soluzioni
 - Topologia a bus → hub passivo con topologia a stella e dimensione massima rete 250m (anziché 2500m)
 - Usare un commutatore (switch) a livello link, dotato di buffer, e una connessione full-duplex per ogni host → **no collisioni**

5.3.4 Gigabit Ethernet (802.3z)

- $1000 \text{ Mbps} = 1 \text{ Gbps}$
- Mantiene invariata lunghezza min/max frame
- Switch al centro della stella, tutti i nodi collegati ai rami della stella → **no collisioni**

5.4 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. A questo scopo vengono usati i dispositivi di interconnessione, dispositivi che possono operare a diversi livelli della pila dei protocolli Internet. In questo testo si vedranno tre tipi di *dispositivi di interconnessione*: i repeater (ripetitori) o hub, gli switch di livello collegamento (detti anche switch di livello 2) e i router (detti anche switch di livello 3). I repeater e gli hub operano solo al primo livello, il più basso, della pila dei protocolli. Gli switch di livello collegamento operano ai primi due livelli, ed infine, i router operano sui primi tre livelli della pila.

5.4.1 Repeater e hub

Un *repeater* è un dispositivo che opera soltanto a livello fisico. I segnali che trasportano informazioni all'interno di una rete possono viaggiare per una distanza ben definita prima che l'attenuazione del segnale metta a repentaglio l'integrità dei dati. Un repeater riceve un segnale e, prima che esso diventi troppo debole o danneggiato, *rigenera* e *risincronizza* la sequenza di bit originale. Successivamente il repeater invia il segnale rigenerato. In passato, quando le LAN Ethernet utilizzavano la topologia a bus, si usava un repeater per collegare tra loro due segmenti di una LAN e quindi superare i limiti di

lunghezza del cavo coassiale. Oggi le LAN Ethernet usano la topologia stella, e in questo caso il repeater è un dispositivo multi-porta (spesso chiamato *hub*) che può essere utilizzato per l’interconnessione delle varie stazioni collegate e allo stesso tempo fungere da ripetitore. La Figura 5.4 mostra che quando un frame proveniente dalla stazione A e destinato alla stazione B arriva all’hub, il segnale che rappresenta il frame viene rigenerato per evitare la presenza di segnali di disturbo possano danneggiare il frame. Inoltre, l’hub inoltra il frame attraverso tutte le sue porte in uscita eccetto quella da cui ha ricevuto il segnale. In altre parole, il frame viene trasmesso in broadcast. Tutte le stazioni nella LAN ricevono il frame, ma soltanto la stazione B lo memorizza mentre tutte le altre lo scartano. La Figura 5.4 mostra il ruolo di un repeater o di un hub in una LAN commutata.

La figura mostra chiaramente che un hub non ha alcuna capacità di filtraggio, non è abbastanza intelligente da capire attraverso quale porta il frame debba essere inviato.

Un repeater non ha capacità di filtraggio.

Gli hub ed i repeater sono dispositivi che operano a livello fisico. Essi quindi non hanno indirizzi di livello collegamento e non effettuano alcuna verifica sugli indirizzi di collegamento presenti nell’intestazione dei frame che ricevono. Rigenerano solamente i frame ricevuti e li inviano attraverso ogni porta.

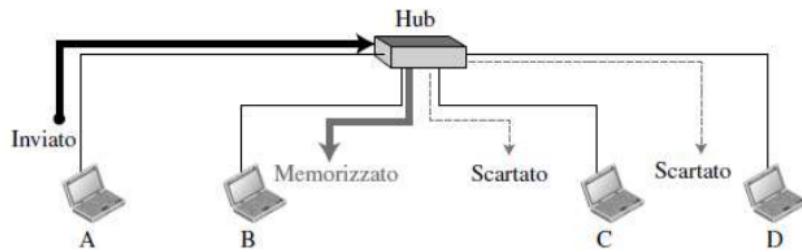


Figura 5.4: Repeater o hub

5.4.2 Switch di livello collegamento

Uno *switch di livello collegamento* (commutatore di livello collegamento) opera sia a livello fisico che a livello di collegamento. Come dispositivo di livello fisico rigenera il segnale che riceve. Inoltre, come dispositivo di livello collegamento, è in grado di verificare gli indirizzi MAC (sorgente e destinazione) contenuti nel frame.

Filtraggio

È normale chiedersi qual è la differenza di funzionalità tra uno switch e un hub. Il primo ha capacità di *filtraggio* (*filtering*), è in grado di verificare l'indirizzo di destinazione del frame e decidere da quale porta in uscita deve essere inviato il frame.

Uno switch ha una tabella che viene utilizzata per il filtraggio.

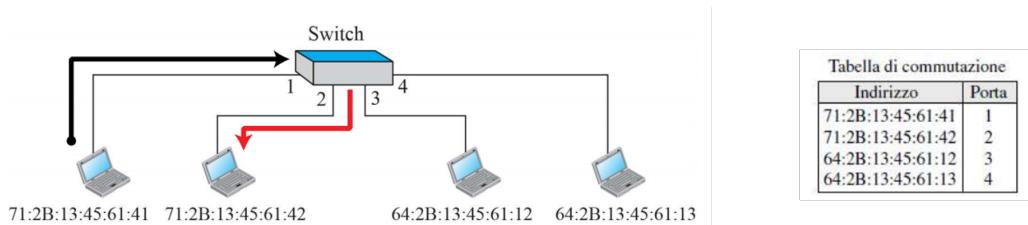


Figura 5.5: Switch di livello collegamento

Vediamo un esempio, nella Figura 5.5 c'è una LAN con quattro stazioni connesse a uno switch. Se un frame destinato alla stazione 71:2B:13:45:61:42 arriva alla porta 1, lo switch consulta la sua tabella per individuare la porta da cui inviarlo.

Secondo la sua tabella, i frame destinati a 71:2B:13:45:61:42 devono essere inviati solo attraverso la porta 2 e quindi non è necessario inoltrare il frame attraverso altre porte.

Uno switch non modifica gli indirizzi MAC contenuti nell'intestazione dei frame.

Switch trasparenti

Uno *switch trasparente* è un commutatore di cui le stazioni presenti nella rete sono completamente ignare dell'esistenza del dispositivo di rete. Se uno switch viene aggiunto o eliminato dalla rete, non è necessaria alcuna operazione di riconfigurazione delle stazioni. Secondo quanto specificato nello standard IEEE 802.1, una rete dotata di switch trasparenti deve soddisfare tre criteri:

- deve essere possibile inoltrare i frame da una stazione ad un'altra;
- la tabella d'inoltro deve essere creata automaticamente analizzando i frame che vengono spediti nella rete;
- devono essere evitati cicli nella rete.

Inoltro Nel caso di uno switch trasparente l'inoltro dei frame non subisce alcuna modifica, è esattamente come descritto in precedenza.

Apprendimento Inizialmente gli switch operavano usando delle tabelle statiche di commutazione. Questo significa che l'amministratore di rete era costretto ad inserire manualmente ciascuna voce della tabella durante la configurazione dello switch. Nonostante il processo fosse abbastanza semplice, non era molto comodo. Infatti, se una stazione veniva aggiunta o cancellata, la tabella doveva essere modificata manualmente. La stessa cosa doveva essere fatta se l'indirizzo MAC di una stazione cambiava, un evento tutt'altro che raro (ad esempio per il cambio di una scheda di rete).

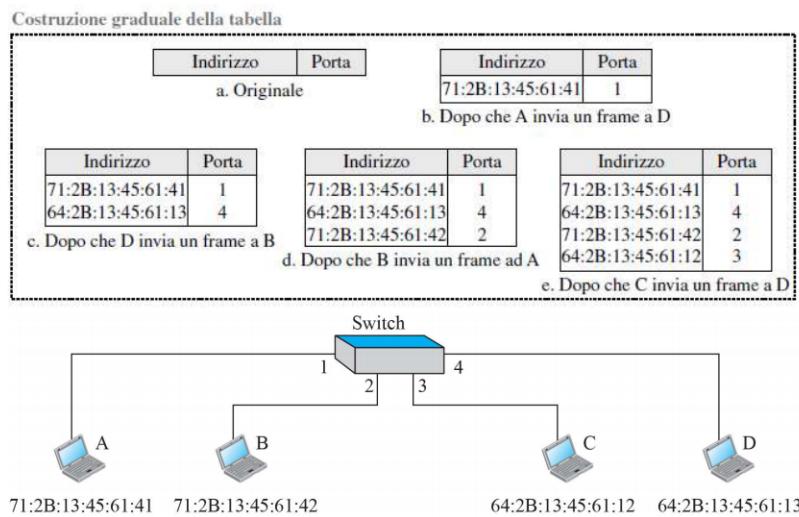


Figura 5.6: Switch con auto-apprendimento

La soluzione, molto migliore della tabella statica, è una tabella dinamica che associa automaticamente gli indirizzi MAC alle porte (interfacce). Per creare una tabella dinamica, occorre uno switch che 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. Vediamo più in dettaglio questo procedimento utilizzando l'esempio della Figura 5.6.

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

invia 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 immediatamente 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.

C'è da notare che il processo di apprendimento potrebbe richiedere molto tempo. Ad esempio, se una stazione non trasmette alcun frame essa non avrà mai una voce nella tabella, ma si tratta di una situazione estremamente improbabile.

5.4.3 Router

Si è già parlato di router nel Capitolo 4, in questo capitolo l'obiettivo è quello di confrontare il loro operato con gli switch e gli hub. Un *router* è un dispositivo che opera su tre livelli: livello fisico, livello di collegamento e livello di rete. Come dispositivo a livello fisico, rigenera il segnale che riceve. Come dispositivo a livello di collegamento, il router verifica gli indirizzi MAC (sorgente e destinazione) contenuti nel frame. Come dispositivo a livello di rete, verifica gli indirizzi di questo livello (nel caso di Internet gli indirizzi IP).

Un router è un dispositivo che opera su tre livelli (fisico, collegamento e rete).

I router servono per collegare tra di loro le reti, sono dispositivi di interconnessione (internetworking): collegano reti diverse per ottenere una rete di reti (una internet). Ci sono tre grandi differenze tra un router e un repeater o uno switch:

1. un router ha un indirizzo fisico (indirizzo MAC) e logico (indirizzo IP) per ognuna delle sue interfacce;

2. 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;
3. i router cambiano l'indirizzo di collegamento del frame (sia la sorgente che la destinazione) quando li inoltrano.

I router cambiano gli indirizzi di livello collegamento dei frame.

Capitolo 6

Sniffer & Portscanner

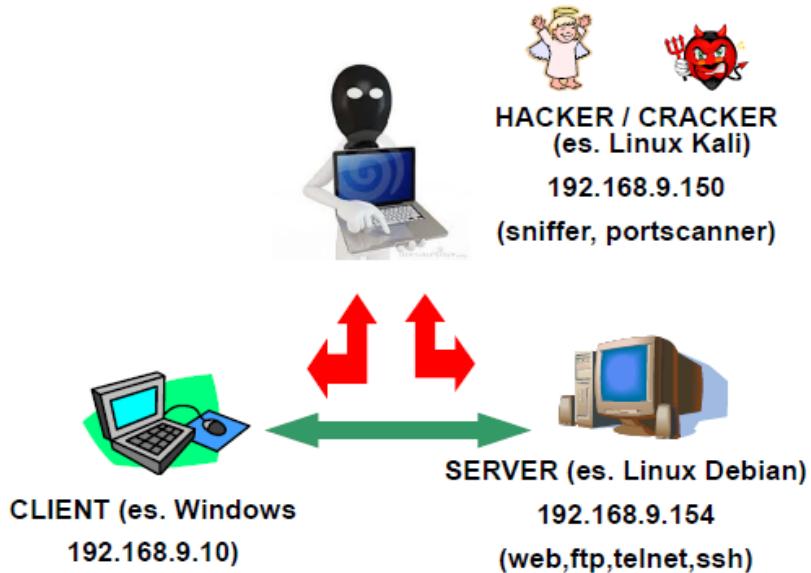
6.1 Introduzione

Gli **Sniffer** & **Port scanner** sono argomenti interessanti ed importanti nell'ambito delle applicazioni di rete. Tuttavia, sono argomenti complessi e solitamente vengono trattati in seminari o corsi sulla sicurezza informatica. Ambiti di utilizzo:

- *lecito*;
- *illecito*.

Gli sniffer e i portscanner sono tipi di strumenti diametralmente opposti:

- **passivi** (*sniffer*)
- **attivi** (*port scanner*)



Situazione tipica

Li vediamo in ambito ***lecito*** (didattico, debug, risoluzione problemi, errori configurazione, ecc...). Esempi “reali e/o pratici”:

- sniffer: src/dst uguale server LAN laboratorio.
- sniffer: bittorrent e consumo banda notebook.
- sniffer/portscan: trojan o malware redirezione traffico uscita (web) o rete nascosta (botnet).
- portscan: installare un nuovo dispositivo (servizi attivi).
- sniffer/portscan: identificazione virus (es. Conficker).

Attenzione a cosa si fa!!

- Non si può attivare uno sniffer in una rete senza autorizzazione
 - privacy, raccolta informazioni, normativa sul lavoro, ecc...
- Non si può effettuare portscan liberamente contro qualsiasi obiettivo senza autorizzazione
 - regolamento ISP, controlli & IDS, risvolti legali, ecc...

6.2 Sniffer

Storia

I programmi per il network tracing sono noti dalla fine degli anni '80. A quel tempo gli analizzatori per scopi commerciali non erano disponibili; il più famoso era il programma Sniffer, sviluppato da Network General. Il termine sniffing risale a tale programma.

Sulle macchine Unix il programma tcpdump è stato sviluppato da Van Jacobson, Leers e McCanne alla fine degli anni '80, questo programma e la libreria libpcap possono essere visti come i nonni di Wireshark.

All'inizio degli anni '90 erano disponibili molti analizzatori di pacchetti commerciali, la maggior parte di essi erano costosi e integrati nell'hardware.

La situazione è cambiata alla fine degli anni '90 con lo sviluppo di "Ethereal" di Gerald Combs, questo programma è stato costruito sopra libpcap e la libreria GIMP Tool Kit (GTK), questo ha portato un analizzatore gratuito a molti sistemi operativi diversi.

6.2.1 Cosa è uno sniffer?

Uno *sniffer* è un software per leggere ed analizzare il traffico di rete che arriva ad una certa macchina; si deve avere accesso alla rete locale.

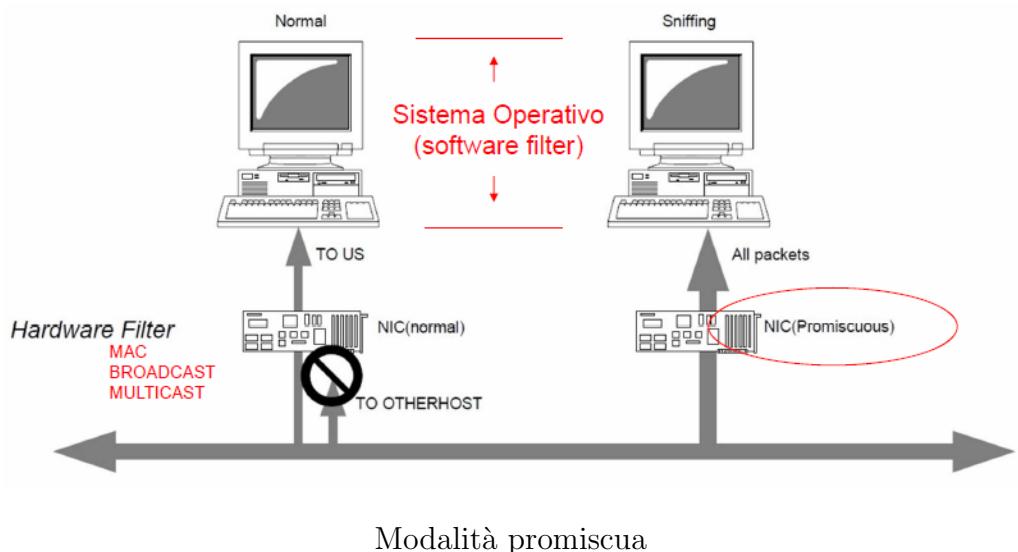
Opera direttamente al livello di collegamento fisico della rete ethernet, quindi ai livelli 1 (fisico) e 2 (data link). È utile per scopi:

- **le citi**: didattici, test, debug e monitoraggio (NIC guasta).
- **il le citi**: traffico non autorizzato, username/password, rootkit & backdoor.

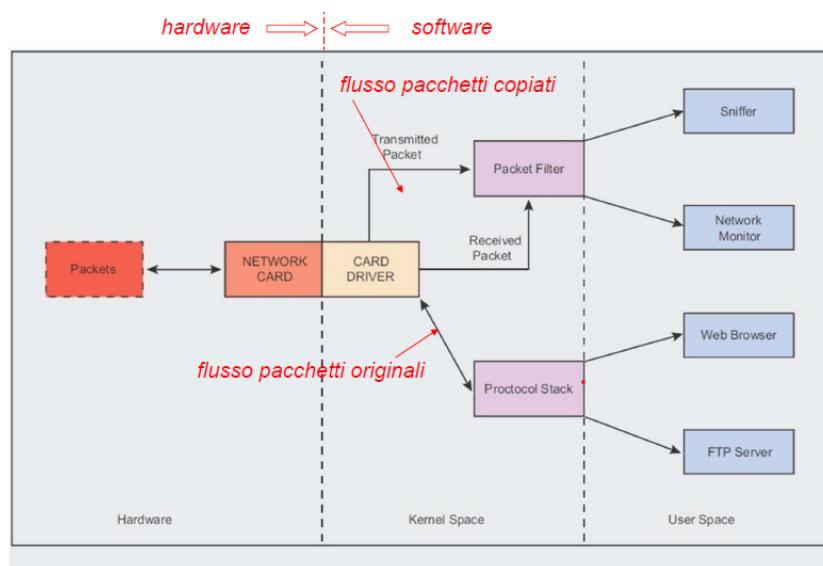
Si imposta l'interfaccia di rete in **modalità promiscua** (cattura cioè anche il traffico non diretto alla propria scheda di rete):

- ifconfig eth0 promisc
- ifconfig eth0 -promisc (ritorno modo funzionamento normale).

Deve essere eseguito con i privilegi di root (o amministratore).



Modalità promiscua



Elementi coinvolti nel processo di cattura

L'attività di sniffing (causa la copia dei pacchetti) è quindi onerosa dal punto di vista computazionale:

- CPU
- memoria RAM
- spazio disco

- conseguente rallentamento attività di rete

Dipende dalla quantità di traffico da analizzare:

- possibile perdita di pacchetti (NIC o driver inadatti)
- Large Receive Offload (lro), Generic Receive Offload (gro)
 - NIC riassembra i pacchetti prima di passarli al kernel
 - problemi con alcuni sniffer evoluti (snort)

Nell'ambito dello *sniffing*, abbiamo dei problemi in ambienti di rete con gli **switch**. Uno **sniffer** deve poter “vedere” tutto il traffico che ci interessa.

Switch Ethernet

Ha un certo numero di porte Ethernet. Il traffico è veicolato **esclusivamente** fra le porte a cui sono connessi i dispositivi che lo generano e che devono riceverlo:

- Sicurezza (proprio contro gli sniffer)
- Prestazioni (traffico esclusivamente tra mittente e destinatario)

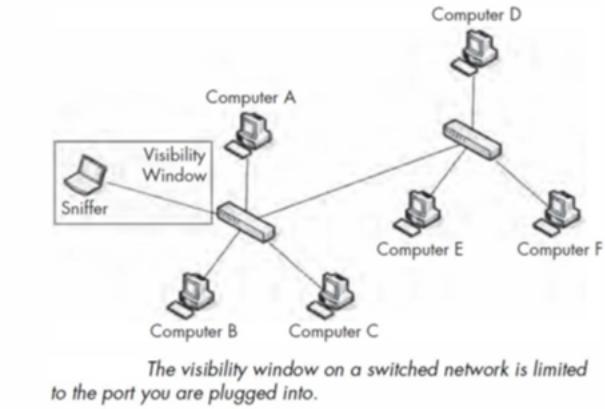
Mano a mano che gli host generano traffico lo switch memorizza le associazioni “MAC address source” → “porta”.

Instrada i frame Ethernet esclusivamente verso la porta sulla quale lo switch “vede” il MAC address di destinazione. Replica del frame su tutte le porte (comportamento hub):

- se il MAC address è quello di broadcast;
- se non è a lui noto.

... in una LAN “**switchata**” esistono diverse soluzioni:

- porta **MONITOR** o **SPAN** (*Switch Port Analyzer*) dove replicare il traffico delle porte interessate o tutto il traffico.
- un “vecchio” **HUB** (*per reti non troppo veloci*).
- dispositivi **TAP** (*Test Access Port*).
- **BRIDGE software**.
- **ARP cache Poisoning** (!!)



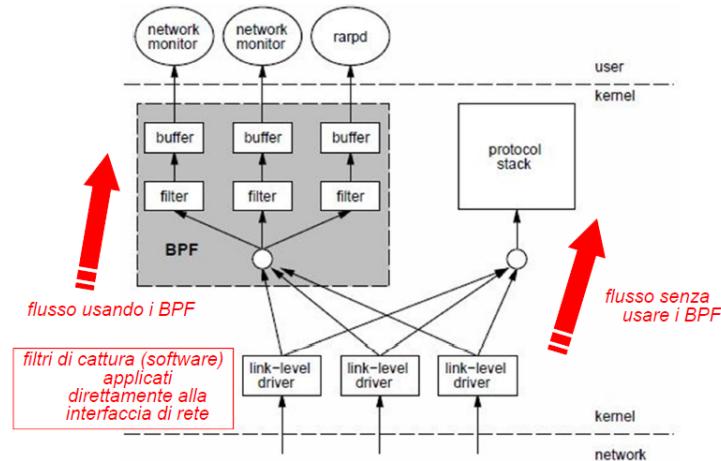
6.2.2 Come lavora uno sniffer?

Il processo di packet-sniffing coinvolge hardware (NIC) & software. Imposta la NIC in promiscuous mode; tutto il traffico viene catturato e passato al sistema operativo e alle applicazioni specifiche. Il funzionamento prevede 3 passaggi:

- *Raccolta*: sono raccolti i dati binary grezzi (raw).
- *Conversione*: sono convertiti in forma leggibile (interpretabile), anche se solo a basso livello; molti tool a linea di comando si fermano qui (tcpdump).
- *Analisi*: analisi approfondita ed elaborata dei dati raccolti; alcuni software evoluti agevolano l'analisi da parte dell'utente (wireshark).

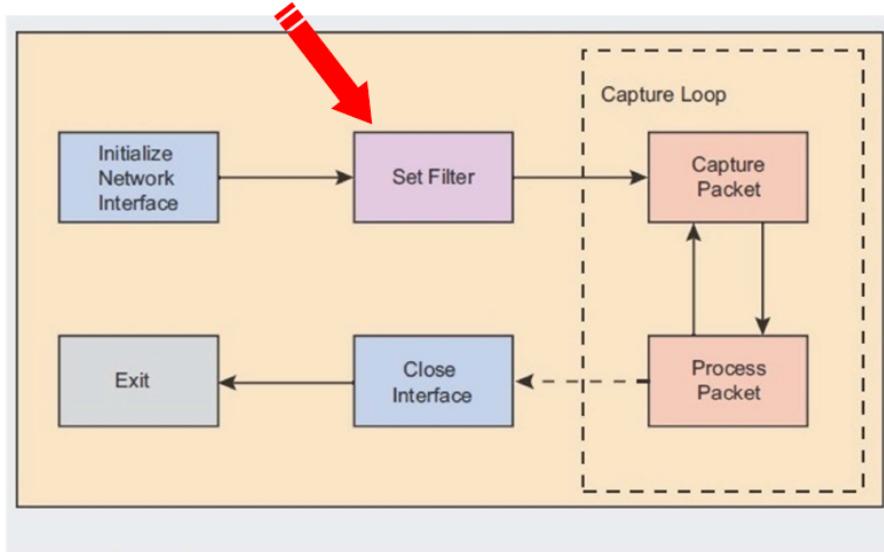
In ambiente GNU/Linux si usano solitamente le librerie **libpcap** (*Packet Capture*). In ambiente MS Windows le *classiche* librerie **winpcap** o le *nuove* **npcap**.

Si sfruttano le funzionalità chiamate **BPF** (Berkeley Packet Filter); permettono di specificare delle espressioni come filtro (*poter selezionare solo il traffico di interesse*). Tali funzionalità sono fornite direttamente dalla libreria libpcap con sintassi generica (*standard de-facto*); utilizzate con tutti gli sniffer (e anche molti IDS) che usano libpcap. Il file del traffico catturato (in formato .pcap) può quindi essere analizzato con software e strumenti diversi (tcpdump, wireshark, ecc). Il formato .pcap è di fatto uno standard.



BPF Overview

i pacchetti sono scartati o raccolti (e solo in questo caso elaborati) in base al filtro BPF, prima di essere passati al kernel e alle applicazioni



Normal program flow of a pcap application

Esempi di sniffer

In ambiente GNU/Linux:

- TCPDUMP

- suite WIRESHARK (TSHARK versione command line, DUMPCAP, EDITCAP)
- IPTRAF (network monitor)
- XPLICO (network forensic analysis tool)
- NGREP
- ETTERCAP (MiTM), NAST
- DSNIFF (password sniffer), P0F (fingerprint passivo)
- TCPKILL
- ETHERAPE (network monitor)
- NTOP (network monitor)
- SNORT (Network IDS)

In ambiente Microsoft Windows:

- suite WIRESHARK (TSHARK versione command line, DUMPCAP, EDITCAP)
- WINDUMP (versione tcpdump per MS Windows)
- SATORI (fingerprint passivo)
- CAIN & ABEL
- Microsoft MESSAGE ANALYZER (ex Microsoft NETWORK MONITOR): sfrutta driver a basso livello del Sistema Operativo MS Windows, spesso usato per sniffer WiFi → **dismesso/deprecato**
- RAWCAP, SMARTSNIFF (localhost,loopback)

6.2.3 TCPDUMP

- In pratica è una interfaccia a riga di comando della libreria libpcap.
- Permette di catturare il traffico di rete in tempo reale, di salvarlo su file (per una analisi successiva) e di rileggere file .pcap raccolti anche con software diversi.

Opzione	Descrizione
-i iface	specifica un'interfaccia su cui ascoltare.
-w file	scrive i pacchetti letti sul file file .
-r file	legge i pacchetti dal file file .
-c count	legge esattamente count pacchetti.
-n	non effettua la risoluzione degli indirizzi.
-p	non porta l'interfaccia in modo promiscuo.
-t	non stampa la temporizzazione dei pacchetti.
-v	aumenta le informazioni stampate a video.
-q	diminuisce le informazioni stampate a video.
-e	stampa anche i dati relativi al protocollo di collegamento fisico (il MAC address).
-F file	usa il contenuto di file come filtro.

BPF (Berkley Packet Filter)

Composti da quelle che si chiamano “*primitive*”; espressioni elementari che permettono di identificare una certa classe di traffico di rete.

$$\text{primitiva} = \text{qualificatore} + \text{identificatore}$$

Qualificatori:

- *di tipo*: host, net, port.
- *di direzione*: src, dst, src or dst, src and dst.
- *di protocollo*: ether, ip, ip6, arp, rarp, tcp, udp, icmp, ...

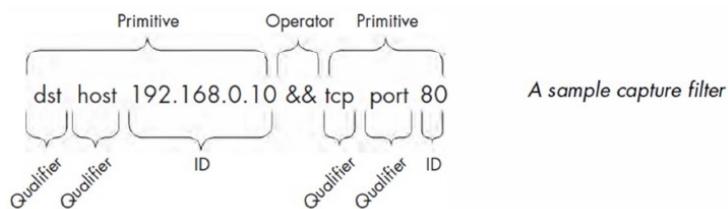
TCPDUMP converte e compila “al volo” (*runtime*) la sintassi BPF ad alto livello (primitive a livello utente) nel codice BPF a basso livello per la “pseudo-machine” (tcpdump -d).

Fondamentali in base a quantità di traffico (limitano dati da elaborare)

- Security Onion
- NetSniff-NG (*sniffer evoluto*)

Qualifier	Description	Examples
Type	Identifies what the ID name or number refers to	host, net, port
Dir	Specifies a transfer direction to or from the ID name or number	src, dst
Proto	Restricts the match to a particular protocol	ether, ip, tcp, udp, http, ftp

The BPF Qualifiers



Struttura di output di TCPDUMP

L'output tcpdump fornisce per impostazione predefinita alcune informazioni di base su ciascun pacchetto. La formattazione di questo output varierà in base ai protocolli in uso, ma i formati più comuni sono:

- TCP:
[Timestamp] [Layer 3 Protocol] [Source IP].[Source Port]>[Destination IP].[Destination Port]: [TCP Flags], [TCP Sequence Number], [TCP Acknowledgment Number], [TCP Windows Size], [Data Length]
- UDP:
[Timestamp] [Layer 3 Protocol] [Source IP].[Source Port]>[Destination IP].[Destination Port]: [Layer 4 Protocol], [Data Length]

Puoi forzare tcpdump a fornire più informazioni in questa riga di riepilogo aggiungendo il tag -v per aumentarne la verbosità.

6.2.4 Wireshark

Estremamente potente, dotato di interfaccia grafica che ne facilita l'utilizzo. Dispone di “protocol dissector”; permettono di effettuare una “dissezione” (*analisi*) dettagliata del traffico catturato. Permette di ricostruire il flusso di

dati (stream) così da poter individuare i dati “effettivamente” catturati al livello di applicazione (email, pagine web, password, ecc...).

Crea statistiche e grafici dai dati elaborati. È possibile rileggere il traffico catturato in formato .pcap anche con strumenti diversi (es tcpdump).

Permette di usare i **filtri di cattura** BPF. Possiede dei **filtri di visualizzazione** per agevolare analisi.

È una suite di strumenti, esiste una versione testuale da linea di comando chiamato TSHARK.

6.3 Port Scanner

Software in grado di rilevare informazioni su un host (o una intera rete) e quali servizi sono attivi. Scansiona le porte (TCP e UDP) per identificare quali di esse sono utilizzate da un servizio. Può “forzare” a provocare certi tipi di risposta, al fine di raccogliere informazioni utili.

È uno strumento indispensabile per verificare in maniera effettiva la *propria* rete. È uno strumento di tipo attivo (a differenza di uno sniffer).

In generale un port scanner genera traffico e “confusione” in una rete, quindi è potenzialmente identificabile. Usato per scopi **offensivi** (ricerca servizi vulnerabili da attaccare), **indispensabile** per scopi di **difesa** (ricerca servizi vulnerabili o inutili da chiudere).

Portsweep

Tecnica di scansione simile al portscanning. Si effettua una scansione di uno o più host (o di una intera rete) alla ricerca della disponibilità di un singolo servizio (e/o una vulnerabilità) concentrandosi però su una singola porta.

Il portscannig, invece, effettua una scansione su diverse porte di diversi host (o una intera rete). Esempio: effettuare un portsweep degli host della nostra LAN sulla porta 3389 per individuarne versioni con problemi di sicurezza. Le finalità sono le stesse del portscannig:

- diagnostica
- attacco

Individuazione/difesa

La pratica del portscan è decisamente invasiva e può risultare “pericolosa”; può essere identificata come un vero e proprio attacco informatico!!

La maggior parte dei sistemi di difesa prevedono dei metodi per identificare; SNORT (uno dei Network IDS più utilizzati) prevede sia un preprocessore (sfportscan) che delle regole costruite “ad-hoc” al fine di identificare i portscan.

6.3.1 NMAP

È il più utilizzato per eseguire portscanning (comunque ne esistono altri). Estremamente flessibile ed ottimizzato:

- esegue portscan più o meno invasivi e/o più o meno identificabili.
- rileva la presenza e tipologia di firewall.
- identifica caratteristiche e versione del sistema operativo e dei servizi remoti (fingerprint attivo).
- presenta report completi e dettagliati.

Prevede un proprio linguaggio di scripting (Nmap Scripting Engine - NSE) per automatizzare e migliorare le analisi. Scansiona intere reti in tempi brevissimi. Dotato anche di interfaccia grafica (ZENMAP).

Funzionamento

Per usufruire di tutte le sue funzionalità, si deve eseguire come amministratore (l'unico che può “forgiare” pacchetti ad-hoc). Si basa sul tipo di risposte ricevute previste dagli standard dei protocolli di rete.

Crea ad-hoc delle richieste (non sempre “formalmente” corrette) per generare delle risposte previste e definite. Sfrutta le peculiarità dei protocolli di rete (si addentra nei “meandri” del loro funzionamento).

Dalle risposte ricevute (e dal confronto con i suoi db interni) identifica tutta una serie di caratteristiche del sistema bersaglio.

Per default (*escluso il Connect Scan -sT*) usa pacchetti minimali e non invasivi, quindi senza dati (vuoti):

- il portscanning di solito “non fa danni” (a meno di bug del servizio remoto...); è la pratica del portscanning ritenuuta dannosa/offensiva!

Cerca in questo modo di non lasciare tracce:

- nei log dei servizi remoti.
- nei log del sistema operativo remoto:

- un pacchetto “strano” generato da nmap può causare un errore nello stack di rete remoto che può essere identificato: “*May 15 11:28:57 server proftpd[3324] server: Fatal: unable to open incoming connection: Transport endpoint is not connected*”
- è comunque impossibile prevedere tutte le possibili implementazioni dei servizi remoti (specialmente nel caso di servizi in UDP, dove i controlli sono demandati alla applicazione), quindi a volte qualche traccia risulta ...

Firewall

- DROP: il pacchetto ricevuto viene silenziosamente scartato (non viene inviata nessuna risposta al client).
- REJECT: il pacchetto ricevuto viene scartato ma viene prodotta una risposta verso il client:
 - TCP: invio pacchetto RST.
 - UDP: invio pacchetto ICMP “Port Unreachable”.
 - possibilità di personalizzare il tipo di risposta mediante la direttiva `reject-with TIPO_PACCHETTO` (`icmp-net-unreachable`, `icmp-portunreachable`, `icmp-proto-unreachable`, ecc...).

Portscan di default

- Come utente non privilegiato:CONNECT SCAN (-sT)
- Come utente amministratore (root): SYN SCAN (chiamato anche semi apertura o half-open) (-sS)
- A seguito della scansione, le porte possono risultare:
 - OPEN: servizio attivo sulla porta ed accessibile.
 - CLOSED: nessun servizio sulla porta.
 - FILTERED: può essere attivo un servizio sulla porta ma la comunicazione è bloccata e non si è in grado di capire se OPEN o CLOSED.
 - UNFILTERED: nel solo caso di TCP ACK Scan (-sA).
 - OPEN— FILTERED: situazione indeterminata (nel caso di scan UDP, IP, protocol, FIN, NULL e Xmas).

- CLOSED—FILTERED: situazione indeterminata (nel solo caso di TCP Idle Scan -sI).

Connect scan (-sT)

Effettua una serie di normali connessioni TCP (SYN→, ← SYN/ACK, ACK→). Funzionamento:

- porta aperta: si riceve una risposta dal demone attivo con successiva chiusura della connessione; porta OPEN.
- porta chiusa (senza firewall): si riceve in risposta una connessione abortita (invio di RST in risposta a connessione su porta chiusa, come previsto dallo stack TCP/IP); porta CLOSED.
- con firewall (DROP): non si riceve risposta, i pacchetti sono scartati dal firewall, quindi si ha un time out; porta FILTERED.

Questa modalità è facilmente individuabile dai log, causa apertura/chiusura di porte senza traffico effettivo. La dimensione del pacchetto SYN è quella normale; ci sono dati contenuti in “options” [mss, wscale, ecc...]

SYN scan (-sS)

È la modalità utilizzata di default (chiamata anche *stealth mode*). Esegue la prima parte dell’apertura (semi-apertura) della connessione TCP (non completa la connessione iniziale, manca ACK finale dal client). Invio di un pacchetto SYN ***creato da nmap***:

- porta aperta: si riceve in risposta SYN+ACK; OPEN
- porta chiusa: si riceve in risposta un RST/ACK; CLOSED
- Firewall (DROP): nessuna risposta; FILTERED

Non serve concludere la connessione da parte di nmap!

Se la porta è aperta e ricevo un SYN+ACK dal target, lo stack TCP del sistema dove si esegue nmap risponde con un RST poiché il SYN+ACK ricevuto dal target non corrisponde a nessun SYN iniziale iniziato da un processo del sistema.

NMAP usa ciò che si chiama “*RAW SOCKET*”. Crea artificialmente il SYN, invece di usare la systemcall *connect()* del kernel (quindi, a differenza di -sT). Il pacchetto SYN (RAW SOCKET) creato da nmap è diverso (più piccolo) di un normale SYN; mancano dati in options (è presente solo [mss]).

Non completando la connessione, il target in questo modo non si accorge di niente; questa scansione non lascia tracce sul target. Rilevabile però dalla maggior parte dei tool di sorveglianza (SNORT, Scanlogd, synlogger, ecc) e può essere bloccato da firewall che filtrano le nuove connessioni (SYN).

Version detection (-sV)

Tenta una connessione ai servizi trovati per determinarne la versione, in base alle risposte di questi ultimi:

- banner ricevuto dal demone del servizio.
- altre informazioni distintive del servizio in ascolto.

Confronta le risposte con l'elenco presente nel file /usr/share/map/nmap-service-probes.

Nmap tenta la connessione ed aspetta circa 5 secondi una qualche risposta (tempo di rispondere se il servizio fosse impegnato). Può essere “ingannato” per eludere proprio i portscan:

- modificando banner magari con quello di una versione senza bug.
- cambiando la porta di default (telnet sulla porta 22).

Nei log, questa scansione può lasciare tracce:

- se è attivo sshd, in auth.log ritroviamo qualcosa tipo:

```
"server sshd[2022]: Did not receive identification string  
from 192.168.9.151"
```

- se è attivo un smtp server (es postfix), in mail.log ritroviamo qualcosa tipo:

- “Apr 19 14:12:56 server postfix/smtpd[2463]: connect from unknown[192.168.9.151]”
- “Apr 19 14:12:56 server postfix/smtpd[2463]: lost connection after CONNECT from unknown[192.168.9.151]”
- “Apr 19 14:12:56 server postfix/smtpd[2463]: disconnect from unknown[192.168.9.151]”

Potrebbe mandare in crash il servizio remoto (*se non progettato correttamente...*). Generalmente viene individuata anche dai sistemi IDS.

TCP/IP Fingerprint (-O)

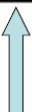
Analizzando le risposte ad una serie di pacchetti di prova, si cercano di identificare caratteristiche dello stack TCP/IP del TARGET, basandosi su una serie di risposte note (/usr/share/nmap/nmap-os-db).

Identifica il sistema operativo, in base alla TCP/IP fingerprint caratteristica di ogni OS. Identifica la predicitività dei numeri di sequenza dello stack TCP (misura la difficoltà di creare pacchetti ad-hoc per inserirsi in una connessione).

Identifica uptime del sistema.

Common Passive Fingerprinting Values			
Protocol Header	Field	Default Value	Operating System
IP	Initial Time to Live	64	NMap, BSD, Mac OS 10, Linux
		128	Novell, Windows
		255	Cisco IOS, Palm OS, Solaris
IP	Don't Fragment Flag	Set	BSD, Mac OS 10, Linux, Novell, Windows, Palm OS, Solaris
		Not set	Nmap, Cisco IOS
TCP	Max Segment Size	0	Nmap
		1440	Windows, Novell
		1460	BSD, Mac OS 10, Linux, Solaris
TCP	Window Size	1024–4096	Nmap
		65535	BSD, Mac OS 10
		2920–5840	Linux
		16384	Novell
		4128	Cisco IOS
		24820	Solaris
		Variable	Windows
TCP	SackOK	Set	Linux, Windows, OpenBSD
		Not set	Nmap, FreeBSD, Mac OS 10, Novell, Cisco IOS, Solaris

Protocol Header	Field	Packet 1 Value	Packet 2 Value
IP	Initial Time to Live	128	64
IP	Don't Fragment Flag	Set	Set
TCP	Max Segment Size	1440 Bytes	1460 Bytes
TCP	Window Size	64240 Bytes	2920 Bytes
TCP	SackOK	Set	Set

 "probabile"
Microsoft Windows

 "probabile"
Linux

Esempio di OS Fingerprint a confronto

Curiosità

È possibile riconoscere (con ragionevole certezza) il sistema operativo remoto molto banalmente osservando le risposte ad un semplice ping. Osservando il campo TTL della risposta:

- Sistemi Microsoft: TTL inizia da 128
- Sistemi Linux: TTL inizia da 64
- Sistemi BSD: TTL inizia da 128

Non è casuale che i sistemi Microsoft e BSD forniscano le stesse tempistiche: Microsoft, ha costruito il proprio stack TCP/IP riprendendolo a piene mani dai sistemi BSD...

Capitolo 7

Cenni di Sicurezza di Rete

7.1 Introduzione

Nel pieno dell'era digitale, si ha la necessità di memorizzare informazioni relative a ogni aspetto della nostra vita. L'informazione è divenuta una risorsa preziosa che deve essere protetta da qualsiasi possibile attacco. In particolare, è necessario proteggerla da accessi non autorizzati (*riservatezza* o *segretezza*) e da modifiche non autorizzate (*integrità*), al fine di poterla rendere disponibile quando necessario alle entità autorizzate (*accessibilità*).

Negli ultimi trent'anni le reti hanno creato una rivoluzione nell'uso dell'informazione, che risulta ora largamente distribuita: mediante le reti è possibile inviare e prelevare informazioni a distanza. In questo nuovo contesto i tre requisiti menzionati in precedenza - riservatezza, integrità e accessibilità - non sono cambiati, ma hanno acquisito nuove dimensioni. Non è più sufficiente garantire la riservatezza delle informazioni archiviate su qualche supporto ma deve essere possibile mantenerne la riservatezza anche quando vengono trasmesse da un computer all'altro. Pertanto è necessario garantire la sicurezza sia nel trasferimento dei dati sia della loro memorizzazione e conservazione.

7.1.1 Obiettivi della sicurezza

Riservatezza

La *riservatezza* (segretezza o confidenzialità) è probabilmente l'aspetto più evidente della sicurezza delle informazioni: è ovviamente necessario proteggere le informazioni considerate confidenziali. Un'organizzazione deve cautelarsi da possibili azioni ostili che mettano a rischio la riservatezza delle proprie informazioni, riservatezza che non è solamente relativa all'*archiviazione* delle informazioni ma anche alla loro *trasmissione*. Quando si trasmettono delle

informazioni riservate con lo scopo di memorizzarle su un computer remoto o quando si prelevano le informazioni da un computer remoto, è necessario garantirne la protezione.

Integrità

Le informazioni richiedono aggiornamenti continui. Se un cliente deposita o preleva denaro dal proprio conto in banca, il saldo deve essere immediatamente aggiornato. Garantire l'*integrità* significa assicurare che le modifiche possano essere apportate esclusivamente dalle entità autorizzate e solo rispettando le procedure previste.

Accessibilità

Il terzo aspetto relativo alla sicurezza delle informazioni è l'*accessibilità*. L'informazione prodotta e archiviata da un'organizzazione deve essere resa disponibile alle entità autorizzate. Qualsiasi informazione diviene inutile se non è accessibile. Deve inoltre poter essere costantemente aggiornata, quindi sempre accessibile alle entità preposte. L'inaccessibilità dell'informazione è tanto dannosa per un'organizzazione quanto per la perdita di riservatezza o integrità delle informazioni.

7.2 Sicurezza della comunicazione

Una comunicazione in rete tra due entità può essere minata da vari tipi di attacchi, che possono essere raggruppati in base all'obiettivo dell'attacco.

7.2.1 Attacchi

Attacchi alla riservatezza

In generale vi sono due tipi di attacchi che possono compromettere la riservatezza dell'informazione: l'*intercettazione* e l'*analisi del traffico*.

Intercettazione L'*intercettazione* si riferisce all'accesso non autorizzato ai dati o alla loro intercettazione durante la trasmissione. Per esempio un file trasferito tramite Internet potrebbe contenere informazioni confidenziali: un'entità non autorizzata potrebbe intercettarne la trasmissione e utilizzarne i contenuti per i propri scopi. Per contrastare l'intercettazione, i dati possono essere resi incomprensibili all'intercettatore utilizzando le tecniche di *cifratura* che verranno discusse in seguito.

Attacchi all'integrità

L'integrità dei dati può essere compromessa da vari tipi di attacchi: *modifica*, *masquerading*, *ripetizione* e *repudiation*.

Modifica In questo caso l'attaccante modifica per i propri scopi le informazioni alle quali ha avuto accesso. Questo può avvenire per esempio quando un cliente invia un messaggio alla propria banca per disporre una transazione e l'attaccante intercetta il messaggio e modifica la transazione a proprio beneficio. Si noti che in certi casi è sufficiente che l'attaccante elimini o semplicemente ritardi il messaggio per danneggiare il sistema o per trarne benefici personali.

Masquerading Si parla di *masquerading* o *spoofing* quando l'attaccante impersona (finge di essere) qualcun altro. Un noto attacco di masquerading è l'IP spoofing, in cui l'attaccante invia un messaggio a un computer indicando che il messaggio arriva da un'entità autorizzata. L'attaccante recupera prima l'indirizzo IP di un'entità autorizzata e poi modifica l'intestazione del pacchetto che invia inserendo l'indirizzo IP di quella entità invece che il proprio. In questo modo l'attaccante può impersonare un'entità autorizzata e ottenere diritti di accesso o redirezionare la vittima su un sito camuffato per carpire informazioni riservate.

Ripetizione Nell'*attacco a ripetizione* l'attaccante ottiene una copia del messaggio inviato da un utente per replicarlo in un momento successivo. Per esempio una persona invia alla propria banca una richiesta di pagamento a favore dell'attaccante che ha svolto un lavoro legittimo; questi intercetta il messaggio e ne invia una replica per ottenere un doppio pagamento dalla banca.

Repudiation Questo tipo di attacco differisce dai precedenti perché viene sferrato da una delle parti in comunicazione, il mittente o il destinatario: il mittente di un messaggio potrebbe successivamente negare di averlo inviato o il destinatario di un messaggio potrebbe successivamente negare di averlo ricevuto. Un esempio di *repudiation da parte del mittente* potrebbe essere il caso del cliente di una banca che ha chiesto di trasferire del denaro a una terza parte, ma nega in seguito di avere richiesto l'operazione. Un esempio di *repudiation da parte del destinatario* potrebbe essere il caso in cui una persona acquista un prodotto da un'azienda pagandone regolarmente il corrispettivo, ma l'azienda nega successivamente di aver incassato il denaro chiedendo di ripetere il versamento.

Attacchi all'accessibilità

Il più noto attacco all'accessibilità è il *denial of service* (negazione del servizio).

Denial of service Il *denial of service* è una tipologia di attacco particolarmente diffusa, che può rallentare o interrompere il servizio di un sistema. Per raggiungere lo scopo l'attaccante ha a disposizione diverse strategie. Può inviare un numero di richieste fittizie talmente elevato da far crollare il sistema sotto il peso del traffico. Può intercettare ed eliminare le risposte del server a un client, facendo credere a quest'ultimo che il server non sia operativo. Può anche intercettare le richieste dei client, inducendoli a inviare più richieste e sovraccaricare il sistema.

7.2.2 Cifratura a chiave simmetrica

La cifratura a chiave simmetrica garantisce la riservatezza dei dati. Nella cifratura a chiave simmetrica si utilizza la medesima chiave sia per l'operazione di cifratura che per quella di decifratura; la chiave può essere utilizzata per la comunicazione bidirezionale. Per tali motivi questo tipo di cifratura, illustrata nella Figura 7.1, viene chiamata *simmetrica*.

Le tecniche di cifratura a chiave simmetrica sono anche chiamate a chiave segreta.



Figura 7.1: L'idea generale della cifratura a chiave simmetrica

Nella Figura 7.1 un'entità, Gaia, invia un messaggio a un'altra entità, Gabriele, su un canale insicuro; grazie alla crittografia un avversario, Eva, non può comprendere i contenuti del messaggio ascoltando semplicemente il canale.

Il messaggio originale da Gaia a Gabriele è chiamato *testo in chiaro (plaintext)*, il messaggio inviato attraverso il canale è chiamato *testo cifrato (ciphertext)*. Per ottenere il testo cifrato dal testo in chiaro, Gaia utilizza un

algoritmo di cifratura (o algoritmo crittografico) e una *chiave segreta* condivisa. Per ottenere il testo in chiaro dal testo cifrato Gabriele utilizza un *algoritmo di decifratura* e la medesima chiave segreta. La chiave è costituita da un insieme di valori (numeri) utilizzati dai due algoritmi.

Algoritmi crittografici a sostituzione

Un cifrario a sostituzione sostituisce un simbolo del testo in chiaro con un altro simbolo. Se i simboli del testo in chiaro sono caratteri, si rimpiazza un carattere con un altro. Gli algoritmi crittografici a sostituzione possono essere classificati in *monoalfabetici* o *polialfabetici*.

- **Cifrari monoalfabetici:** ogni carattere (o simbolo) nel testo in chiaro viene sempre sostituito dallo stesso carattere (o simbolo) nel testo cifrato indipendentemente dalla posizione nel testo.
- **Cifrari polialfabetici:** le diverse occorrenze dello stesso carattere possono corrispondere a un sostituto differente: la relazione fra un carattere nel testo in chiaro e il carattere corrispondente nel testo cifrato è di uno-a-molti. I cifrari polialfabetici hanno il vantaggio di nascondere la frequenza delle singole lettere per violare il testo cifrato.

Cifrari moderni a chiave simmetrica

I cifrari a chiave simmetrica tradizionali operano a livello di caratteri, ma l'avvento dei computer rende preferibile l'impiego di cifrari orientati al bit.

Cifrari moderni a blocchi Un cifrario simmetrico moderno a blocchi cifra un blocco di n bit di testo in chiaro o decifra un blocco di n bit di testo cifrato. In un cifrario di questo tipo un blocco di testo cifrato dipende dall'intero blocco di testo in chiaro. Gli algoritimi di cifratura e decifratura usano una chiave di k bit e devono essere l'uno l'inverso dell'altro.

- **DES: Data Encryption Standard** → chiave di cifratura 56 bit
- **AES: Advanced Encryption Standard**
 - 2001 standard, sostituisce DES
 - Elabora dati in blocchi di 128 bit, chiave di 128, 192, o 256 bit

Problemi

- Come si accordano A e B su chiave da usare? (soprattutto se non si incontrano mai?)
- Segretezza chiave inversamente proporzionale a quanto viene usata.

7.2.3 Cifratura a chiave asimmetrica

Nel caso della crittografia simmetrica l'informazione segreta (la chiave) deve essere condivisa fra due persone. Nel caso della crittografia asimmetrica l'informazione segreta è personale (non condivisa): ciascuna persona crea e conserva privatamente la propria informazione riservata.

Mentre la crittografia simmetrica si basa sulla sostituzione e sulla permutazione di simboli (caratteri o bit), la crittografia asimmetrica si basa sull'applicazione di funzioni matematiche a numeri. Nella crittografia simmetrica il testo in chiaro e il testo cifrato sono considerati una combinazione di simboli, la cifratura e la decifratura permutano questi simboli o li costituiscono con altri. Nella crittografia asimmetrica, invece, il testo in chiaro e il testo cifrato sono considerati numeri; la cifratura e la decifratura sono funzioni matematiche applicate a numeri che consentono di otterne altri.

La crittografia a chiave asimmetrica impiega due chiavi separate: una privata e una pubblica.

L'idea generale

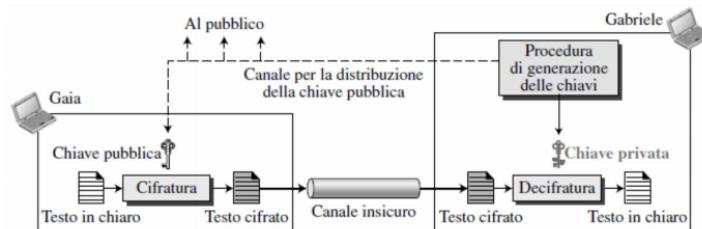


Figura 7.2: L'idea generale della crittografia assimmetrica

La Figura 7.2 schematizza l'idea generale della crittografia assimmetrica nel caso della cifratura. La figura evidenzia alcuni fatti importanti. In primo luogo pone in risalto la natura assimmetrica del cifrario. La responsabilità di garantire la sicurezza ricade principalmente sul destinatario (Gabriele in questo caso) che deve creare due chiavi: una privata e una pubblica. Gabriele ha il

compito di distribuire la propria chiave pubblica alla comunità, per esempio tramite un canale di distribuzione della chiave pubblica. Sebbene questo canale non debba assicurare la riservatezza, deve garantire autenticazione e integrità. Eva non deve poter pubblicizzare la propria chiave pubblica alla comunità sostenendo che si tratti della chiave pubblica di Gabriele.

In secondo luogo la crittografia asimmetrica implica che Gabriele e Gaia non possano utilizzare la stessa coppia di chiavi per la comunicazione bidezionale; ciascun membro della comunità deve creare la propria coppia di chiavi pubblica e privata. La Figura 7.2 illustra come Gaia possa utilizzare la chiave pubblica di Gabriele per inviare messaggi crittografati a Gabriele; se Gabriele volesse rispondere, Gaia dovrebbe stabilire la propria coppia di chiavi pubblica e privata.

Infine, la crittografia asimmetrica comporta che Gabriele abbia bisogno di una sola chiave privata per poter ricevere comunicazioni da qualsiasi membro della comunità, mentre Gaia deve utilizzare n chiavi pubbliche per poter comunicare con n membri della comunità, una per ciascuna di essi. In altre parole Gaia deve utilizzare un *key ring* (mazzo di chiavi).

Lo svantaggio della cifratura asimmetrica è l'efficienza, per questa ragione viene usata generalmente per messaggi di dimensione ridotta. È importante sapere che per usufruire di tutte le funzionalità di sicurezza più avanzate, sono necessarie entrambe le tecniche di crittografia, che sono complementari fra loro.

Esempio di cifratura asimmetrica: RSA.

7.2.4 Message digest

Vi sono casi in cui pur non essendo richiesta la riservatezza, risulta di fondamentale importanza l'*integrità*: il messaggio originale non deve poter essere modificato. Gaia potrebbe per esempio scrivere 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.

Un modo per preservare l'integrità è quella di inviare insieme al messaggio un codice generato dal mittente e dipendente dal messaggio. Questo codice viene elaborato con un particolare algoritmo chiamato *funzione hash crittografica*. Questa funzione crea un'immagine compressa del messaggio. Gabriele, per verificare che il messaggio ricevuto non sia stato modificato, applica la funzione hash crittografica al messaggio ricevuto e confronta il nuovo digest con quello inviato da Gaia: se risultano identici Gabriele ha la garanzia che il messaggio originale non è stato modificato. La Figura 7.3 illustra questa tecnica. Ovviamente non deve essere possibile modificare il digest del messaggio.

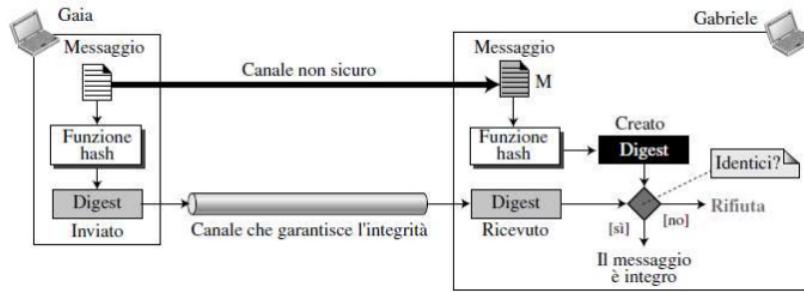


Figura 7.3: Messaggio e relativo digest

Si noti che il digest dipende dal messaggio ma viene inviato separatamente, per cui il messaggio viaggia in chiaro e non è preservata la confidenzialità, che invece richiede l'applicazione di meccanismi di crittografia.

Funzione hash: riceve un messaggio di lunghezza arbitraria e produce un digest di lunghezza fissa. Esempi: MD4, MD5, SHA...

7.2.5 Message Authentication Code

Il digest può essere utilizzato per verificare l'integrità di un messaggio - ovvero per verificare che il messaggio non sia stato modificato. Per garantire anche l'*autenticazione* del messaggio - ovvero garantire che Gaia, non qualcun altro, abbia originato il messaggio - è necessario includere nel processo un'informazione segreta condivisa da Gaia e Gabriele (che Eva non possiede): si deve creare un codice MAC (*Message Authentication Code*). La Figura 7.4 illustra l'idea.

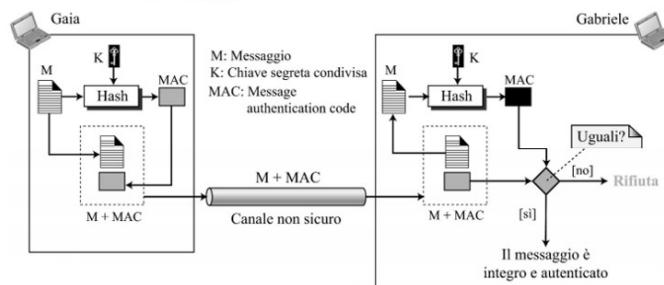


Figura 7.4: Codice di autenticazione di messaggio (MAC)

Gaia e Gabriele condividono la chiave segreta **K**. Gaia crea un MAC applicando la funzione hash alla concatenazione della chiave segreta e del

messaggio $h(K + M)$. Gaia invia il messaggio e il relativo MAC a Gabriele tramite il canale insicuro. Gabriele 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.

Si noti che in questo caso non è necessario usare due canali: sia il messaggio sia il MAC possono essere inviati sullo stesso canale insicuro. Eva può vedere il messaggio ma non può formularne uno nuovo per sostituire il precedente poiché non possiede la chiave segreta condivisa da Gaia e Gabriele. Eva non può creare lo stesso MAC ottenuto da Gaia.

7.2.6 Firma digitale

Un altro modo per garantire l'integrità e l'autenticazione di un messaggio è la *firma digitale*. Il MAC utilizza una chiave segreta per proteggere il digest, mentre la firma digitale utilizza una coppia di chiavi pubblica-privata.

Quando Gaia invia un messaggio a Gabriele, questi può voler verificare l'autenticità del mittente, ovvero può richiedere garanzie che il messaggio provenga da Gaia e non da Eva: in questo caso può quindi chiedere a Gaia di firmare elettronicamente il messaggio. In altri termini la firma elettronica può provare l'autenticità di Gaia come mittente del messaggio. Questo tipo di firma viene chiamata *firma digitale*.

Funzionamento

Gaia firma il messaggio M cifrandolo con la sua chiave privata. La cifratura del messaggio M costituisce proprio la firma S di Gaia (solo Gaia conosce la sua chiave privata). A questo punto Gaia invia sia il messaggio M sia la firma S a Gabriele, il quale verifica l'autenticità del messaggio applicando la chiave pubblica di Gaia. Se il messaggio che Gabriele ottiene è uguale a quello inviato da Gaia allora il messaggio è autentico.

Si noti che quando il documento è firmato, chiunque, Gabriele incluso, può verificarne la firma poiché la chiave pubblica di Gaia è di dominio pubblico. Gaia non deve invece utilizzare la propria chiave pubblica per firmare il documento altrimenti la propria firma potrebbe essere forgiata da chiunque.

È possibile utilizzare una chiave segreta (simmetrica) sia per firmare che per verificare una firma? La risposta è negativa per varie ragioni. Per prima cosa una chiave segreta è nota solo a due entità (Gabriele e Gaia nell'esempio). Quindi se Gaia dovesse firmare un altro documento da inviare a Filippo, dovrebbe utilizzare un'altra chiave segreta. Secondo creare una chiave segreta per una sessione comporta l'autenticazione, che a sua volta utilizza la firma

digitale: si cadrebbe così in un circolo vizioso. Terzo, Gabriele potrebbe utilizzare la chiave segreta condivisa con Gaia per firmare un documento da inviare a Filippo e sostenere che il documento proviene da Gaia.

Si dovrebbe fare una distinzione fra l'impiego delle chiavi pubblica e privata nella firma digitale rispetto al loro impiego nei sistemi crittografici per garantire la riservatezza. In quest'ultimo caso vengono utilizzate le chiavi pubbliche e private del destinatario: il mittente utilizza una chiave pubblica del destinatario per crittografare, quest'ultimo utilizza la propria chiave privata per decifrare. Nella firma digitale vengono invece impiegate le chiavi pubblica e privata del mittente: quest'ultimo utilizza la propria chiave privata per firmare il messaggio, il destinatario utilizza la chiave pubblica del mittente per verificare l'autenticità del messaggio.

Non-repudiation Se Gaia firmasse un messaggio ma successivamente negasse di averlo fatto, Gabriele potrebbe provare il contrario? Con lo schema considerato fino a questo momento Gabriele potrebbe trovarsi in difficoltà nel garantire l'autenticità a distanza di tempo. Dovrebbe conservare la firma in archivio e successivamente utilizzare la chiave pubblica di Gaia per creare il messaggio originale provando che il messaggio in archivio e il messaggio così creato sono identici. Questo non è in realtà fattibile poiché Gaia potrebbe nel frattempo aver sostituito le proprie chiavi, e potrebbe sostenere che l'archivio contenente la firma non è autentico.

Una possibile soluzione consiste nel fare intervenire una terza parte fidata, che faccia da tramite da Gaia e Gabriele al fine di garantire che i messaggi inviati non possano essere disconosciuti dal mittente.

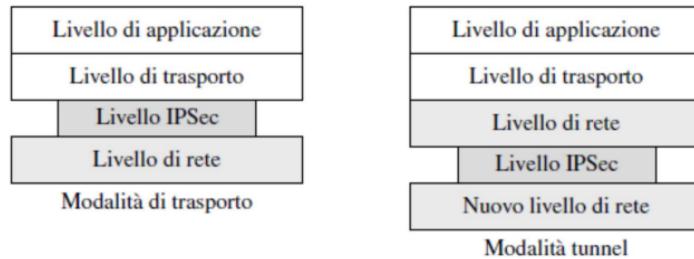
7.3 Comunicazione sicure nello stack protocolare

7.3.1 Sicurezza al livello rete: IPsec

È necessario garantire la sicurezza anche al livello di rete perché ci sono programmi applicativi che non implementano meccanismi di sicurezza o che sfruttano servizi di UDP. Inoltre numerose applicazioni, tra cui i protocolli di routing, usano direttamente i servizi di IP. Tutte queste applicazioni richiedono dunque servizi di sicurezza a livello rete.

IPSec (o IP Security) è un insieme di protocolli sviluppato dall'IETF (Internet Engineering Task Force) per fornire garanzie di sicurezza ai pacchetti di livello rete. IPSec serve ad autenticare e rendere confidenziali i pacchetti

del protocollo IP. IPSec può operare in due modi: in *modalità trasporto* o in *modalità tunnel*.



Modalità trasporto Nella modalità trasporto, IPSec protegge ciò che viene fornito dal livello trasporto al livello rete. In altre parole, questa modalità protegge i dati incapsulati nei pacchetti di livello rete, come rappresentato nella Figura 7.5.

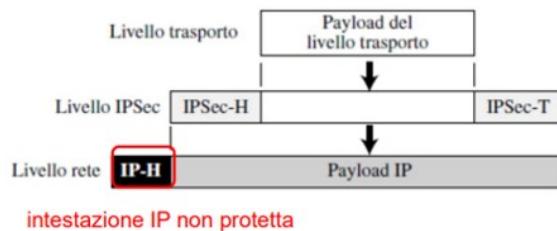


Figura 7.5: IPSec in modalità trasporto.

Si noti che la protezione è solo sui dati, ovvero sul payload dal livello trasporto: l'intestazione del pacchetto IP non viene protetta. L'intestazione (e il trailer) di IPSec vengono aggiunti ai dati provenienti dal livello trasporto prima dell'intestazione IP.

La modalità trasporto viene normalmente utilizzata quando occorre proteggere i dati scambiati fra mittente e destinatario (end-to-end). L'host mittente sfrutta il protocollo IPSec per autenticare e/o cifrare i dati consegnati dal livello trasporto. L'host destinatario sfrutta il protocollo IPSec per verificare l'autenticità dei dati e/o decifrarli, per poi consegnarli al livello trasporto.

Modalità tunnel Nella modalità tunnel IPSec permette di proteggere l'intero pacchetto IP. Con questa modalità l'intero pacchetto IP, inclusa l'intestazione, viene incapsulato in un pacchetto IPSec. Il tutto deve essere poi

inserito in un pacchetto IP: verrà quindi aggiunta una nuova intestazione IP, come mostrato nella Figura 7.6.

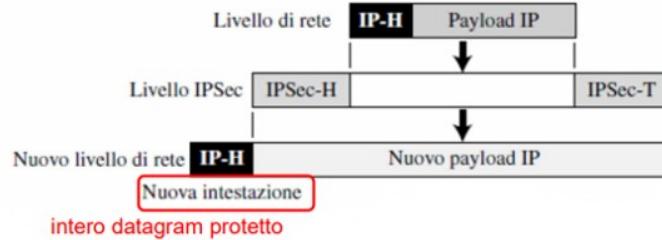


Figura 7.6: IPSec in modalità tunnel.

La nuova intestazione è diversa dall'intestazione IP iniziale. La modalità tunnel viene solitamente utilizzata fra due router, fra un host e un router o fra un router e un host. L'intero pacchetto originale viene protetto da intrusioni fra il mittente e il destinatario, come se il pacchetto viaggiasse in un tunnel immaginario.

Protocollo ESP

ESP (Encapsulating Security Protocol) è un protocollo di IPSec che fornisce autenticazione di sorgente, integrità e riservatezza. Il protocollo ESP aggiunge un'intestazione e un trailer, come mostrato nella Figura 7.7.

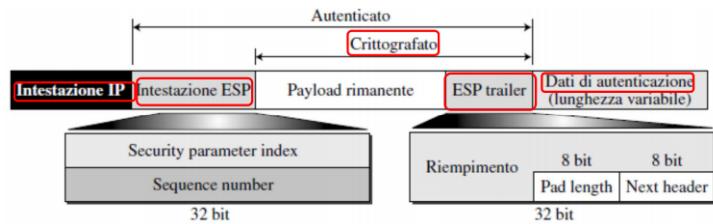


Figura 7.7: Protocollo ESP

Si noti che i dati di autenticazione ESP sono aggiunti in coda al pacchetto, per facilitarne il calcolo. Quando un datagramma IP trasporta un pacchetto ESP, il valore del campo protocollo dell'intestazione IP è 50. Il campo next header nel trailer di ESP specifica il protocollo originale (il tipo di pacchetto contenuto dal datagramma IP, per esempio TCP o UDP). Ecco come avviene 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;
5. i dati di autenticazione vengono aggiunti al termine del trailer ESP;
6. viene aggiunta un'intestazione IP dopo averne impostato il valore del campo protocollo a 50.

Reti private virtuali (VPN)

Una delle applicazioni di IPSec è nelle *reti private virtuali*. Una rete privata virtuale (Virtual Private Network, o VPN) è una tecnologia sempre più utilizzata dalle aziende che adottano Internet per le comunicazioni interne ed esterne, ma che richiedono riservatezza nelle comunicazioni. Una VPN è una rete che è privata ma virtuale: privata perché garantisce la riservatezza all'organizzazione, virtuale poiché non utilizza realmente dei collegamenti WAN privati. La rete è fisicamente pubblica ma virtualmente privata. La Figura 7.8 illustra il concetto di rete privata virtuale.

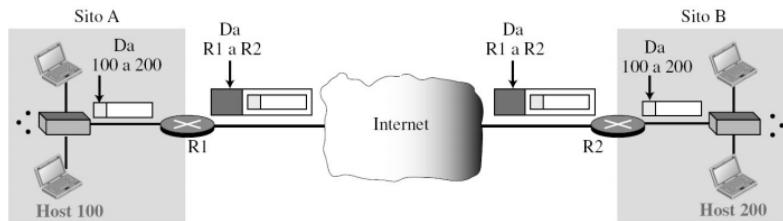


Figura 7.8: Una rete privata virtuale (VPN)

I router R1 e R2 usano la tecnologia VPN per garantire la riservatezza all'organizzazione. La tecnologia VPN impiega il protocollo ESP di IPSec in modalità tunnel: il datagramma privato, inclusa la sua intestazione, viene incapsulato in un pacchetto ESP. Il router al confine del sito mittente usa nel nuovo datagramma il proprio indirizzo IP e l'indirizzo IP del router nel sito destinatario. La rete pubblica (Internet) è responsabile del trasporto del pacchetto da R1 e R2. Eventuali osservatori esterni non possono decifrare i contenuti del pacchetto e nemmeno gli indirizzi mittente e destinatario. La decifratura avviene in R2, che identifica l'indirizzo di destinazione del pacchetto e lo inoltra.