



UNIVERSITÀ DEGLI STUDI DI SALERNO

**Dipartimento dell'Ingegneria dell'Informazione ed
Elettronica e Matematica applicata**

Corso di Laurea in *Ingegneria Informatica* / L-8

Reti di Calcolatori

Appunti delle lezioni di *Gennaro Francesco Landi*

Docente: *Diodato Ferraioli*
I Sessione di Laurea A.A. 2023-2024

*"The devil is in the details"
God is in this summary, you're welcome*

Indice

1 Introduzione	4
1.1 Internet	4
1.1.1 Caratteristiche delle reti di calcolatori	5
1.1.2 Che cos'è Internet	5
1.1.3 Architetture a Livelli e Stack di Protocolli Internet	6
2 Architettura di Internet	8
2.1 Il Core	8
2.1.1 Struttura di Internet	9
2.2 Ritardi e Troughput di una rete	10
3 Principi di Applicazioni di Rete	11
4 Il WEB - HTTP	13
4.1 HTTP: HyperText Transfer Protocol	13
4.2 Wireshark	18
5 E-MAIL	19
6 DNS	21
6.1 Un database distribuito e gerarchico	22
6.2 Message format:	24
7 Streaming & Content Distribution Network	26
7.1 Streaming HTTP e DASH	26
8 Livello di trasporto	29
8.1 Relazione tra i livelli di trasporto e di rete	30
8.1.1 Differenza tra UDP e TCP	31
9 Principi di Trasferimento Affidabile dei Dati	33
9.0.1 RDT 1.0: canale affidabile	34
9.0.2 RDT2.0: canale con errori sui bit	34
9.0.3 RDT3.0: canale con perdite di pacchetti ed errori sui bit	38
9.1 Protocolli per il trasferimento dati affidabile con pipeline	38
10 Il protocollo TCP	44
10.0.1 Trasmissione affidabile in TCP	48
10.0.2 Timeout di TCP	49
10.0.3 Approfondimento: Gestione della connessione TCP	51
10.1 Controllo della congestione	51

10.1.1 Come controllare la congestione	53
10.1.2 Finestra di Congestione	54
11 Livello di Rete	58
11.0.1 Cosa c'è all'interno di un router ?	59
11.1 Il Protocollo Internet (IP) : IPv4	60
11.1.1 Come ottenere l'indirizzo di un host: DHCP	63
11.2 NAT, IPv4 & ICMP	66
11.2.1 IPv6	67
11.2.2 ICMP: Internet Control Message Protocol	69
12 Livello di rete: il piano di controllo (control panel)	72
12.1 Routing Algorithms	72
12.2 Come rendere il routing scalabile? Protocolli di instradamento: OSPF e BGP . .	74
12.2.1 Routing intra-AS	75
12.2.2 Routing inter-AS: BGP	77
12.3 Il piano di controllo SDN	79
12.3.1 OpenFlow Protocol	82
13 Il Livello di Collegamento	84
13.1 Servizi offerti dal livello di collegamento	84
13.1.1 Errori di trasmissione	85
13.1.2 Collegamenti di Rete	89
13.2 Indirizzamento al Livello di Collegamento	90
13.2.1 Indirizzi IP vs MAC	93
13.3 Reti LAN cambrate: Ethernet e VLAN	95
13.3.1 Servizio offerto da Ethernet	96
13.3.2 Gestione trasmissioni simultanee	98
13.4 Interconnessione di Switch: LAN	99
13.4.1 Switch VLAN	100
14 Reti Wireless	103
14.1 Caratteristiche dei link wireless	104
14.2 WiFi: 802.11 Wireless LANs	106
14.2.1 802.11: CSMA/CA (Collision Avoidance, MAC protocol)	107
14.2.2 Bluetooth: 802.15.1	112
15 Reti Cellulari 4G e 5G	114
15.1 Stack dei protocolli LTE	117
15.2 Gestione della Mobilità degli Utenti: dal punto di vista del livello di Rete	121
15.2.1 IP Mobile	124
15.2.2 Mobilità in reti 4G	125

Capitolo 1

Introduzione

1.1 Internet

Internet è un'infrastruttura di comunicazioni che consente il trasferimento di dati tra dispositivi collegati alla rete, indipendentemente dalla posizione, dal tipo di dispositivi e dal tipo di dati digitali che si vuole trasmettere.

- **Rete:** insieme di elementi (*nodi*) tra i quali sono definite delle relazioni (*collegamenti*).
- Una **rete di calcolatori** è *multi-purpose*:
 - connette dispositivi programmabili
 - può supportare applicazioni differenti
 - trasporta qualsiasi tipo di dati digitali attraverso un mezzo trasmittente

Un **sistema di comunicazione** è composto da:

1. Messaggio
2. Mittente
3. Destinatario
4. Mezzo trasmittente
5. Protocollo

Per **connessione diretta** si intende un sistema di connessioni adatto solo a reti piccole, in cui il numero di collegamenti e nodi collegati ad una linea è limitato, quindi non è scalabile. **Tipi di connessione diretta**:

- Punto-Punto
- Multi-Punto

Invece sistemi di **connessione indiretta** permettono di creare reti grandi ed economiche utilizzando due tipi di nodi:

- **Nodi terminali:** dispositivi finali che si connettono direttamente agli utenti o ai dispositivi

- **Nodi di commutazione (*switch*)**: dispositivi utilizzati per instradare il traffico dati tra i nodi terminali

Tipi di comunicazione *indiretta*:

- Unidirezionale *simplex*
- Bidirezionale alternata (*half-duplex*) (es. *comunicazioni push-to-talk*)
- Bidirezionale (*full duplex*) (es. *comunicazioni in tempo reale*)

1.1.1 Caratteristiche delle reti di calcolatori

L'insieme di switch costituisce una rete a commutazione, ovvero la soluzione più economica e scalabile che consente un uso efficiente della rete. Le due tecniche di commutazione più diffuse sono:

- **Commutazione a *Circuito***: Stabilisce un percorso dedicato tra mittente e destinatario prima della trasmissione dei dati. Le risorse lungo il percorso sono riservate per l'intera durata della comunicazione.
- **Commutazione a *Pachetto***: I dati vengono suddivisi in pacchetti prima della trasmissione. Ogni pacchetto è indirizzato individualmente e può seguire percorsi diversi attraverso la rete.

Classificazione delle reti in base all'area coperta:

- **PAN (*Personal Area Network*)**: copre lo spazio fisico di una persona
- **LAN (*Local Area Network*)**: copre l'area di un'abitazione, edificio o campus
- **MAN (*Metropolitan Area Network*)**: copre l'area di una città
- **WAN (*Wide Area Network*)**: copre l'area di una regione o nazione

Indipendentemente dal tipo di rete utilizzata, ogni nodo potrà comunicare soltanto con altri nodi connessi alla sua stessa rete, peccò gli *utenti* richiedono un servizio di **comunicazione universale**, indipendente dalla posizione, dal tipo di rete o dal tipo di software utilizzato.

1.1.2 Che cos'è Internet

Internet è l'esempio più conosciuto di inter-rete, descrivibile tramite **i suoi componenti** ed i **servizi che offre**.

I suoi componenti:

- **Hosts (*end-systems*)**: dispositivi collegati alla rete in grado di eseguire elaborazioni, utilizzati dagli utenti
- **Linee di comunicazione**: linee su cui viaggiano i dati, avendo tassi di trasmissione differenti (*bandwidth*)
- **Dispositivi di connessione (*routers e switch*)**: instradano blocchi di dati attraverso la rete per farli arrivare a destinazione.

I router sono i commutatori dell'inter-rete e gli switch commutatori all'interno della rete stessa.

Internet permette la trasmissione di dati sulla rete, se un host vuole inviare un messaggio ad un altro end-system collegato alla rete:

1. Divide il messaggio in blocchi di taglia fissa (*pacchetti*) e aggiunge a ciascuno un'*intestazione* con le informazioni sul destinatario
2. ogni pacchetto inviato sulla rete viaggia in maniera *indipendente* dagli altri.
I **commutatori** utilizzano le informazioni sul destinatario per *instradare* ciascun pacchetto
3. quando i pacchetti arrivano a destinazione vengono riassemblati per ricostruire il messaggio originario

Gli host non accedono direttamente a internet ma utilizzano i servizi di un **ISP** (*Internet Service Provider*), i rapporti sono regolati da dei contratti detti **SLA** (*Service Level Agreement*).

Per garantire l'interoperabilità tra reti differenti, tutti gli ISP devono accordarsi per utilizzare gli stessi protocolli per l'invio, l'inoltro e la ricezione di pacchetti, i principali sono **TCP/IP** che sono definiti come *standard di Internet* in accordo dall'IETF (*Internet Engineering Task Force*). Gli **RFC** (*Request for Comments*) sono documenti che descrivono gli standard di Internet.

Internet può essere vista come una piattaforma che **fornisce servizi alla applicazioni**, dette *applicazioni distribuite* eseguite esclusivamente sugli host.

Un sistema operativo di un host collegato alla rete fornisce un'**API** (*Application Programming Interface*)

1.1.3 Architetture a Livelli e Stack di Protocolli Internet

Per organizzare e comprendere la complessità di Internet, si adotta un'architettura a livelli, dove ogni strato si occupa di aspetti specifici della comunicazione, seguendo linee guida generali definite come **architettura di rete**.

Le *architetture di rete* sono un'astrazione che descrive il processo di comunicazione tra entità, **organizzate gerarchicamente** e con **interfacce** che permettono l'*interazione tra livelli adiacenti*. Questo approccio consente una separazione dei problemi e semplifica il progetto, garantendo l'interoperabilità. *Ogni livello è implementato tramite protocolli che forniscono servizi di comunicazione e interagiscono tra loro attraverso interfacce specifiche*.

I **protocolli**, che definiscono il modo in cui i dati vengono scambiati, sono strutturati in pacchetti contenenti intestazioni di controllo e dati. *Internet adotta il modello TCP/IP*, mentre il **modello OSI** è utilizzato come *riferimento*, ma non è ampiamente implementato. Il modello *OSI* (Open System Interconnection) è descritto in un'**architettura a 7 livelli**:

1. *Applicazione*: permette l'accesso alle risorse di rete
2. *Presentazione*: cambia il formato, cifra e comprime i dati
3. *Sessione*: instaura, gestisce e termina una sessione di comunicazione
4. *Trasporto*: rende affidabile la trasmissione da processo a processo
5. *Rete*: permette di spostare i pacchetti dal nodo sorgente al nodo destinazione
6. *Collegamento*: organizza i bit in frame e permette di effettuare una consegna diretta dei frame

7. *Fisico*: gestisce la trasmissione fisica dei bit sul mezzo di trasmissione

Un *potrocollo* fornisce un servizio di comunicazione allo strato superiore e utilizza i servizi del livello inferiore. Ogni *protocollo* ha due *interfacce*:

1. **l'interfaccia di servizio** per comunicare con lo strato *inferiore/superiore*;
2. **l'interfaccia di livello** (*peer-to-peer*) per comunicare con la controparte (*peer*) del protocollo sull'altro host.

Quando un potrocollo richiede al potroollo di livello inferiore un servizio gli passa una quantità di dati organizzati in pacchetti, dove *ogni pachetto consiste di*:

- **intestazione** (*header*): contenente informazioni di controllo per il protocollo
- **copro** (*body o payload*): contenente i dati ricevuti dal protocollo superiore
- un eventuale **coda** (*trailer*): contenente altre informazioni di controllo

Stratificazione dei protocolli

Considerati assieme, i protocolli dei vari livelli sono detti **pila dei protocolli** (*stack dei protocolli*). L'incapsulamento non è rigido, i protocolli di livelli non adiacenti possono comunicare, ma ci interessiamo dei *servizi* che ogni protocollo offre a quello al livello superiore e dei *servizi* che richiede a quello inferiore (*service model*).

Lo **stack dei protocolli di Internet** è organizzato in *5 livelli*:

1. **Applicazione**: sede delle applicazioni di rete e dei relativi protocolli, per quanto riguarda Internet include protocolli come HTTP, SMTP e FTS oppure protocolli come il DNS. Un protocollo a livello di applicazione scambia pacchetti di informazioni (*messaggi*) con un'altra applicazione;
2. **Trasporto**: trasferisce i messaggi del livello di applicazione tra punti periferici gestiti dalle applicazioni. In Internet troviamo due protocolli di trasporto, *TPC* e *UDP*. Entrambi permettono di trasmettere *segmenti* di dati tramite la rete;
3. **Rete**: si occupa di trasferire i pacchetti a livello di rete, detti **datagrammi**, da un host all'altro. Il protocollo principale a questo livello è l'*IP* (Internet Protocol), che gestisce l'indirizzamento e l'instradamento dei pacchetti;
4. **Collegamento**: questo livello si occupa della comunicazione diretta tra nodi adiacenti sulla stessa rete fisica. Include protocolli che gestiscono l'accesso al mezzo trasmittivo, come Ethernet o Wi-Fi, e si occupa della suddivisione dei dati in frame;
5. **Fisico**: questo livello rappresenta l'hardware effettivo che trasmette i bit sulla rete fisica. Include i componenti come cavi, switch, router, schede di rete e trasmettitori wireless.

Capitolo 2

Architettura di Internet

2.1 Il Core

Il **core** di Internet è un'inter-rete di router collegati tra loro tramite tre elementi chiave:

- **Commutatori di pacchetto**
- **Multiplexing**
- **Inoltro ed instradamento**

Per spostare dati tra diversi hosts tra due diverse reti di accesso, Internet utilizza le **reti a commutazione**, ovvero la potenzialità di una rete di costruire, mantenere e abbattere un collegamento tra i nodi che la compongono.

Esistono due principali tipi di reti a commutazione: la **commutazione di pacchetto** e la **commutazione di circuito**.

Nel caso della **commutazione di pacchetto**, il mittente divide i dati da trasmettere in pacchetti di dimensione fissa, ciascuno con un'intestazione contenente le informazioni sulla destinazione. I router ricevono i pacchetti, leggono l'intestazione e decidono come instradarli verso la destinazione. La maggior parte dei router utilizza una modalità di trasmissione *store-and-forward*, ritrasmettendo un pacchetto solo dopo averlo ricevuto completamente. Tuttavia, la commutazione di pacchetto può causare ritardi di accodamento e perdita di pacchetti in caso di congestione del buffer. Per mitigare questi problemi, i router eseguono complessi protocolli di routing per aggiornare le tabelle di inoltro e adeguare i percorsi dei pacchetti alla situazione della rete.

A differenza della commutazione di pacchetto, la **commutazione di circuito** prevede la creazione di un circuito dedicato tra sorgente e destinazione per l'intera durata della comunicazione. Sebbene questa tecnica garantisca prestazioni costanti e una velocità di trasmissione garantita, riserva la banda anche quando non è utilizzata e richiede costi di setup.

In conclusione, la maggior parte delle reti di calcolatori, compreso il core di Internet, utilizza la commutazione di pacchetto per trasportare dati in modo efficiente e flessibile, anche se la commutazione di circuito può essere adatta per applicazioni che richiedono prestazioni garantite e regolari.

Multiplexing nelle reti a commutazione di circuito

Un circuito all'interno di un collegamento è implementato tramite:

- **multiplexing a divisione di frequenza (FDM)**: lo spettro di frequenza di un collegamento viene suddiviso tra le connessioni stabilite tramite il collegamento. Nello specifico,

il collegamento dedica una banda di frequenza a ciascuna connessione per la durata della connessione stessa.

- **multiplexing a divisione di tempo (*TDM*)**: il tempo viene suddiviso in frame, intervalli di durata fissa, a loro volta ripartiti in un numero fisso di slot temporali. Quando la rete stabilisce una connessione attraverso un collegamento le dedica uno slot di tempo in ogni frame, tali slot sono dedicati unicamente a quella connessione.

2.1.1 Struttura di Internet

Oggi giorno Internet, una rete di reti, è complessa e consiste di dozzine di ISP di primo livello e migliaia di IPS di livello inferiore. Gli ISP si distinguono per la copertura geografica: alcuni coprono continenti ed oceani altri sono ristretti alle regioni. Gli IPS di livello più basso si connettono a quelle di livello superiore e queste a loro volta si connettono tra di loro tramite IXP. Ultimamente i più grandi fornitori di contenuti hanno creato le loro reti private e, quando possibile, si connettono direttamente agli IPS di livello inferiore.

Le reti di accesso

L'architettura di Internet si struttura principalmente in due livelli: i *sistemi terminali* o *periferici*, su cui operano gli utenti, e *il core*, che facilita la comunicazione tra questi dispositivi terminali. Le reti di accesso svolgono il ruolo fondamentale di collegare i dispositivi terminali al core.

I *sistemi periferici*, o *terminali*, sono dispositivi di calcolo connessi a Internet, come PC, smartphone, laptop e altri dispositivi smart. Essi ospitano ed eseguono i programmi applicativi, come browser, software di posta elettronica e social network. **Questi sistemi possono essere distinti in client e server**: i *client* richiedono servizi tramite la rete, mentre i *server* forniscono tali servizi.

Le **reti di accesso** comprendono varie tipologie, come le reti residenziali con accesso *DSL* o *via cavo*, le reti aziendali e istituzionali con accesso *Ethernet* e *WiFi*, le reti cellulari con accesso *3G*, *4G* e *5G* e le *reti satellitari*. La scelta del tipo di rete di accesso dipende dalla larghezza di banda disponibile e se la rete è dedicata o condivisa.

Nel **contesto residenziale**, l'accesso *DSL* viene fornito dall'operatore telefonico attraverso l'utilizzo delle linee telefoniche esistenti. Il *DSLAM* nella centrale riceve il segnale analogico e lo separa in segnali telefonici e dati, consentendo agli utenti di navigare in Internet e telefonare contemporaneamente.

Le *reti di accesso via cavo* utilizzano le *infrastrutture della cable-TV*, con il *cable modem* che svolge un ruolo simile al *modem DSL* e il *CMTS* che funge da *DSLAM*. Queste reti sono ibride fibra-coassiale (HFC), con prestazioni fino a diversi Gbps in downstream e decine di Mbps in upstream.

Negli ultimi anni, le *tecnologie FTTx* (Fiber to the x) stanno diventando sempre più diffuse, offrendo prestazioni superiori basate sulla trasmissione su fibra ottica. Le varianti di FTTx includono FTTC, FTTH, FTTP, FTTH, a seconda di dove termina la fibra ottica.

Le **reti aziendali** utilizzano spesso Ethernet come mezzo di connessione, collegando i dispositivi a switch Ethernet, che a loro volta sono collegati a un router aziendale e all'ISP aziendale.

Le **reti LAN domestiche** combinano l'accesso residenziale *DSL* con la tecnologia *LAN WiFi*, consentendo ai dispositivi di muoversi liberamente e condividere l'accesso a Internet.

Le reti di accesso wireless su scala geografica sfruttano l'infrastruttura della rete cellulare per consentire agli utenti di inviare e ricevere dati durante gli spostamenti. Le reti 3G, 4G e 5G offrono prestazioni sempre migliori, con velocità di trasmissione in costante aumento.

Infine, esistono mezzi trasmissivi guidati, come il doppino di rame intrecciato, il cavo coassiale e la fibra ottica, e mezzi trasmissivi non guidati, come i canali radio terrestri e satellitari, che permettono la propagazione dei segnali su lunghe distanze senza l'uso di mezzi fisici.

2.2 Ritardi e Throughput di una rete

Il funzionamento ottimale di una rete di computer, soprattutto di Internet, dovrebbe consentire la trasmissione di dati tra qualsiasi coppia di dispositivi senza ritardi o perdite di dati. Tuttavia, le leggi fisiche e le limitazioni tecnologiche impongono dei vincoli che influenzano le prestazioni di una rete. Queste *limitazioni* includono il **throughput limitato**, l'introduzione di **ritardi** e la possibilità di **perdere pacchetti** durante la trasmissione. Per gestire efficacemente queste sfide, è fondamentale comprendere i vari tipi di ritardi che possono verificarsi durante la trasmissione dei dati.

I ritardi possono essere categorizzati in diversi tipi, tra cui:

- Ritardo di elaborazione: tempo necessario per leggere l'intestazione del pacchetto e decidere come instradarlo. Nei router ad alta velocità, questo ritardo è nell'ordine di microsecondi o meno.
- Ritardo di accodamento: tempo trascorso in coda in attesa della trasmissione, dipende dal numero di pacchetti presenti in coda e dalla loro dimensione. Può variare da microsecondi a millisecondi a seconda della natura e dell'intensità del traffico.
- Ritardo di trasmissione: tempo necessario per trasmettere tutti i bit del pacchetto sul collegamento, dipende dalla lunghezza del pacchetto e dalla larghezza di banda del collegamento.
- Ritardo di propagazione: tempo necessario affinché ogni bit si propaghi lungo il collegamento e raggiunga la destinazione, dipende dalla velocità di propagazione e dalla lunghezza del mezzo trasmissivo.

Il ritardo end-to-end, ovvero il tempo totale trascorso dalla trasmissione del primo bit del pacchetto alla ricezione dell'ultimo bit da parte del destinatario, è influenzato da una combinazione di questi ritardi.

L'intensità del traffico e la possibilità di congestimento possono anche influenzare i ritardi. Quando l'intensità del traffico supera la capacità di elaborazione della rete, si possono verificare perdite di pacchetti, specialmente se le code dei buffer hanno dimensioni finite.

Il **throughput** è il tasso a cui i bit dati vengono effettivamente trasferiti (*bit trasmessi nell'unità di tempo*). Il **throughput end-to-end** dipende dalle velocità dei collegamenti sia nel core di Internet che nelle reti di accesso del client e del server. Se consideriamo una comunicazione singola, il throughput sarà limitato dalla velocità del collegamento più lento attraversato durante la trasmissione dei dati, che può essere una rete di accesso. Tuttavia, in un contesto reale, molte connessioni condividono i collegamenti nel core di Internet. Con l'aumentare del numero di connessioni, il link condiviso nel core può diventare un collo di bottiglia, riducendo il throughput complessivo. Per mitigare questo problema, è necessario instradare le connessioni su percorsi distinti quando possibile. Questo aiuta a distribuire il carico di traffico e a mantenere prestazioni accettabili per tutte le comunicazioni in corso.

Capitolo 3

Principi di Applicazioni di Rete

Le Applicazioni di rete, conosciute anche come network app, sono programmi progettati per funzionare su macchine diverse e comunicare tra di loro. Queste applicazioni sono la base di molte delle attività online che svolgiamo quotidianamente.

Quando si crea una network app, si scrivono programmi che possono funzionare su diversi sistemi finali e che si comunicano tramite la rete. È importante notare che non è necessario scrivere il software per i nodi centrali della rete; ci si concentra solo sui sistemi finali, il che consente uno sviluppo rapido delle applicazioni e una rapida diffusione.

Le *architetture delle applicazioni possono essere di due tipi principali*:

- **client-server**: ci sono uno o più server che forniscono servizi ai client, che a loro volta richiedono e ricevono tali servizi. I server sono macchine sempre accese con indirizzi IP permanenti, mentre i client possono essere connessi in modo intermittente e avere indirizzi IP dinamici. Non comunicano direttamente tra di loro, ma solo con il server.
- **peer-to-peer (P2P)**: non ci sono server centrali. Le macchine finali comunicano direttamente tra di loro, consentendo una distribuzione più decentralizzata dei servizi. Questo approccio offre una maggiore scalabilità, in quanto ogni nuovo peer porta nuove capacità al servizio. Tuttavia, la gestione di questa rete può essere più complessa, poiché le macchine possono essere connesse in modo intermittente e cambiare spesso indirizzo IP.

I processi comunicanti sono i programmi in esecuzione su una macchina che si scambiano messaggi. Su una stessa macchina, i processi comunicano attraverso meccanismi definiti dal sistema operativo, mentre su macchine diverse si scambiano messaggi tramite la rete. Ci sono processi client, che iniziano la comunicazione, e processi server, che attendono di essere contattati. **Le applicazioni con architettura P2P possono avere sia processi client che server.**

Le *socket* sono un'astrazione utilizzata dai processi per inviare e ricevere messaggi attraverso la rete. *Ogni processo ha una propria socket*, simile a una porta. Il mittente invia i messaggi attraverso la porta, affidandosi all'infrastruttura di trasporto per consegnare il messaggio al processo destinatario.

Per identificare i processi e consentire loro di ricevere messaggi, è necessario un sistema di indirizzamento. Ogni **host ha un unico indirizzo IP**, ma poiché possono essere in esecuzione più processi contemporaneamente, è necessario includere anche il numero di porta associato a ciascun processo.

Protocolli di trasporto su Internet

Le applicazioni hanno requisiti diversi per quanto riguarda i servizi di trasporto. Alcune richiedono un trasferimento affidabile dei dati al 100%, mentre altre possono tollerare una certa

perdita. Alcune applicazioni hanno esigenze rigide in termini di ritardo di trasmissione, mentre altre sono più elastiche. Alcune richiedono un throughput minimo garantito, mentre altre possono funzionare con qualsiasi throughput disponibile.

I protocolli di trasporto su Internet, **TCP e UDP**, forniscono servizi diversi.

- *TCP* offre un trasporto affidabile tra processo mittente e destinatario, controllando il flusso e la congestione, ma non fornisce timing, throughput minimo garantito o sicurezza;
- *UDP*, invece, offre un trasferimento dati non affidabile senza alcun controllo del flusso o della congestione.

Le applicazioni su Internet utilizzano una varietà di protocolli a livello applicazione e di trasporto. Ad esempio, per l'email si utilizza SMTP su TCP, per l'accesso remoto al terminale si utilizza Telnet su TCP, per la navigazione web si utilizza HTTP su TCP e così via.

Per rendere sicure le comunicazioni su TCP, si può utilizzare SSL, che fornisce connessioni cifrate, integrità dei dati e autenticazione degli endpoint, *praticamente fa da livello intermedio tra l'applicazione e il SO, per aggiungere maggiore sicurezza cifrando il contenuto dei pacchetti*. SSL è un protocollo a livello applicazione che le applicazioni possono utilizzare per comunicare tramite TCP in modo sicuro.

Infine, il protocollo a livello applicazione definisce i *tipi di messaggi* scambiati, la *sintassi* e la *semantica* dei messaggi, nonché le *regole per l'invio e la ricezione dei messaggi*. Questi sono elementi fondamentali per il corretto funzionamento delle applicazioni di rete.

Capitolo 4

Il WEB - HTTP

Il **World Wide Web** è una network application che fornisce un servizio per la consegna di documenti (*hyperlinkati*). I documenti trasferiti sono **web pages** che contengono sempre un **base HTML-file** che include diversi **referenced objects** (*file HTML, immagine JPEG, file audio, ecc..*). Ogni oggetto è identificato da un **URL**.

Un **URL** (*Uniform Resource Locator*) è una sequenza di caratteri che identifica univocamente l'indirizzo di una risorsa su una rete di computer ospitata da un server e resa accessibile ai client tramite Internet. Stringa composta di 6 campi, alcuni facoltativi:

1. **protocollo**: protocollo da utilizzare per accedere alla risorsa (HTTP, HTTPS, FTP, MMS);
2. **username:password@ (opzionale)**: credenziali di autenticazione per l'accesso alla risorsa, però è una tecnica pericolosa e non sicura;
3. **host[:port]**: identificatore del server su cui risiede la risorsa, "port" identifica la *porta di servizio* di rete al quale inoltrare la richiesta;
4. **percorso**: pathname nel file system del server che identifica la risorsa, se non è presente il server restituisce un file predefinito;
5. **query string (opzionale)**: consente di passare al server uno o più parametri
6. **fragment (opzionale)**: indica una posizione all'interno della risorsa

URL: *protocollo://[username[:password]@]host[:porta]</percorso>[?querystring][#fragment]*
Esistono tre categorie di pagine WEB: **statiche** (il contenuto è fissato), **dinamiche** (il contenuto viene generato nel momento dell'accesso) e **attive** (viene eseguito un programma sul computer del client).

4.1 HTTP: HyperText Transfer Protocol

Un protocollo a livello di applicazione del Web, con un *architettura client/server*:

- il browser del **client** che richiede, riceve (*tramite protocollo HTTP*) e mostra i web objects;
- il Web **server** invia (*usando il protocollo HTTP*) oggetti in risposta alle richieste.

HTTP usa *TPC*, il client tramite il browser inizia la connessione TPC (crea la socket) con il server sulla porta 80 (*default per il Web*), il server accetta la connessione *TPC* dal client e successivamente vengono scambiati messaggi HTTP utilizzando la connessione *TPC*. Dopo che tutti i messaggi sono scambiati, la connessione *TPC* viene chiusa per liberare risorse sul server e sul client.

Connessioni persistenti e non persistenti

Una scelta progettuale quando di creano connessioni *TPC* è quella di dover scegliere tra:

- **Connessioni non persistenti:** al più un oggetto viene inviato sulla connessione TCP e poi la connessione viene chiusa;
1. Il client HTTP inizia la connessione TCP al server HTTP in ascolto ad un certo URL sulla porta 80
 2. L'HTTP server sull'host dell'url in attesa di connessioni TCP sulla porta 80, "accetta" la connessione e notifica il client
 3. Il client HTTP invia un HTTP **request message** (*contenente l'URL*) nella socket corrispondente alla connessione TPC. Il messaggio indica che il client desidera l'oggetto indicato dal path dell'url
 4. L'HTTP server riceve il request message attraverso la propria socket associata alla connessione, recupera l'oggetto indicato dall'url dalla memoria (centrale o di massa), lo incapsula in un **response message** HTTP che viene inviato al client attraverso la socket
 5. Il processo server HTTP comunica a TCP di chiudere la connessione, questo però non termina la connessione finché non sia certo che il client abbia ricevuto integro il messaggio di risposta
 6. Il client HTTP riceve il messaggio di risposta, **la connessione TPC termina**. Il messaggio indica che l'oggetto incapsulato è un file HTML, il client estrae il file dal messaggio di risposta, esamina il file HTML e trova i riferimenti ad altri oggetti
 7. Vengono quindi ripetuti i primi quattro passi per ciascuno degli oggetti referenziati.

Calcolo del response time: quando il client HTTP vuole iniziare la connessione TCP, invia un piccolo segmento TCP al server e quest'ultimo manda una conferma per mezzo di un piccolo segmento TCP. Infine il client da anch'esso una conferma di ritorno al server (**handshake a tre vie three-way handshake**).

Il **vantaggio principale** delle connessioni HTTP non persistenti è che **i browser possono aprire connessioni TCP parallele** per recuperare gli oggetti referenziati.

Il **RTT** (*Round Trip Time*) è il tempo per un piccolo pachetto per viaggiare dal client al server e poi tornare indietro.

L'*HTTP response time* richiede un *RTT* per inizializzare la connessione TCP, un *RTT* per la richiesta HTTP e per i primi bytes della corrispondente HTTP response, il tempo per la trasmissione dell'intero file. Quindi in totale: **2RTT + file trasmissione time**

- **Connessioni persistenti:** più oggetti possono essere inviati su una singola connessione TPC tra client e server.

Il funzionamento non è molto diverso da quella non persistente ma il server in questo caso lascia la connessione TCP aperta dopo l'invio di una risposta, per cui le richieste

e le risposte successive tra gli stessi client e server possono essere trasmesse sulla stessa connessione. In particolare, non solo il server può inviare un'intera pagina web su una sola connessione TCP permanente, ma può anche spedire allo stesso client più pagine web.

Queste richieste di oggetti possono essere effettuate una di seguito all'altra senza aspettare le risposte delle richieste precedenti (*pipelining*).

In generale, il server HTTP chiude la connessione quando essa rimane inattiva per un dato lasso di tempo configurabile.

Il response time per le connessioni HTTP persistenti è di *un RTT* per ogni oggetto referenziato.

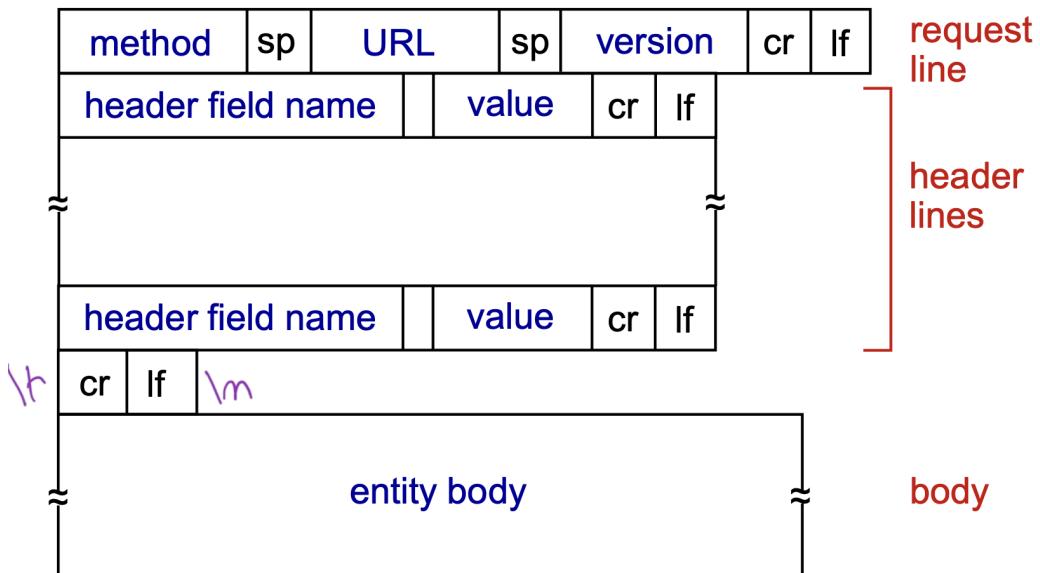


Figura 4.1: HTTP REQUEST MESSAGE: formato generale

Il protocollo HTTP (Hypertext Transfer Protocol) prevede due tipi principali di messaggi: le *richieste* (*request*) e le *risposte* (*response*) ed entrambe contengono **solo caratteri ASCII**.

Una **richiesta HTTP** è composta da una linea di richiesta, seguita da una serie di righe di intestazione (header lines). La linea di richiesta include metodi come GET, POST, HEAD, che indicano l'azione da eseguire, l'URL dell'oggetto richiesto e la versione del protocollo HTTP. Le righe di intestazione forniscono informazioni aggiuntive sulla richiesta, come il tipo di contenuto accettato dal client, le preferenze linguistiche e i meccanismi di controllo della cache.

Ci sono *diversi metodi HTTP*, come **GET**, **POST** e **HEAD**, che vengono utilizzati per scopi diversi. Ad esempio, il metodo **GET** viene utilizzato per richiedere il contenuto di una risorsa specifica (*non fa controllare il body al server*), mentre il metodo **POST** viene utilizzato per inviare dati al server (*fa leggere il body*), ad esempio quando si compila un modulo su una pagina web. **HEAD** chiede al server di inviare solo le informazioni sull'oggetto richiesto, ma non l'oggetto stesso.

In HTTP/1.1 vengono aggiunti altri due metodi (*di default disabilitati ai client*): **put** (carica un file nel corpo della richiesta al path specificato nel campo URL),**delete** (cancella il file specificato nel campo URL).

Come caricare l'input di una form:

Quando viene inviata una richiesta *POST*, i dati dell'input della form vengono solitamente inclusi nel corpo del messaggio HTTP, mentre con il metodo URL, i dati dell'input vengono aggiunti direttamente all'*URL* come parametri di query.

Le righe di intestazione forniscono ulteriori informazioni sulla richiesta, come le preferenze di codifica e lingua del client, nonché i dettagli sulla cache e le informazioni sulla data dell'ultima modifica della risorsa richiesta.

Una risposta HTTP è composta da una linea di stato seguita da righe di intestazione e dai dati richiesti. La linea di stato include il codice di stato HTTP (come 200 OK, 404 Not Found) che indica il risultato della richiesta. Le righe di intestazione forniscono informazioni aggiuntive sulla risposta, come la data di creazione della risorsa, la lunghezza dei dati e il tipo di contenuto.

I **codici di stato HTTP** sono divisi in categorie, come le *risposte informative (1xx)*, le *risposte di successo (2xx)*, i *reindirizzamenti (3xx)*, gli *errori del client (4xx)* e gli *errori del server (5xx)*. Questi codici vengono utilizzati per comunicare lo stato della richiesta al client, come ad esempio se la richiesta è stata completata con successo o se è stato riscontrato un errore.

In sintesi, il protocollo HTTP definisce il formato e il comportamento dei messaggi scambiati tra client e server web, consentendo loro di comunicare in modo efficace e trasferire dati tra loro attraverso la rete.

User-Server State - Cookies

Essendo che i **server HTTP sono privi di stato (stateless)**, molti siti Web usano la tecnologia dei *Cookies*, composta da quattro componenti

1. *coockie header line* dell'HTTP *response message*;
2. *cookie header line* nel successivo HTTP *request message*;
3. un *cookie file* conservato sull'host dell'utente e gestito dal suo browser
4. un *database* di back-end è conservato dal Web server, contenente tutti gli identificativi dei vari client.

Così quando un nuovo client visita un sito, dopo la richiesta HTTP al server del sito, esso creerà un'**unico ID** e un'**entry** del back-end database per quell'*ID*.

I cookies possono essere utilizzati per: authorization, shopping carts, recommendations, user session.

Per mantenere lo "stato" tra le varie transazioni utente, ci sono diverse tecniche. Una di queste è l'**utilizzo di endpoint**, che sono punti di accesso attraverso i quali il client e il server comunicano tra loro. Gli endpoint possono memorizzare lo stato tra le richieste utilizzando i cookie. Quando il server invia una risposta al client, può includere un cookie che verrà memorizzato sul dispositivo dell'utente e inviato nuovamente al server con le richieste successive. In questo modo, il server può identificare e mantenere lo stato dell'utente attraverso le diverse interazioni con il sito web.

In breve, i cookie sono utilizzati per una vasta gamma di scopi pratici nei siti web, ma è importante essere consapevoli del loro potenziale impatto sulla privacy e sulla sicurezza delle informazioni personali degli utenti.

Web Caches (Proxy Server)

I proxy server possono essere configurati dai utenti nel proprio browser per accedere al web attraverso una cache. In questo scenario, quando un browser invia una richiesta HTTP, anziché inviarla direttamente al server di origine, la invia alla cache. Se l'oggetto richiesto è presente nella cache, la cache restituisce direttamente l'oggetto al browser. Altrimenti, la cache inoltra la richiesta al server di origine, memorizza l'oggetto nella cache e quindi lo restituisce al browser.

L'obiettivo dell'utilizzo di una cache è quello di soddisfare le richieste del client senza sovraccaricare il server di origine. La cache svolge un duplice ruolo, agendo sia come client rispetto al server di origine, sia come server rispetto al browser.

Le cache sono tipicamente installate da Internet Service Provider (ISP), università, aziende o provider di servizi residenziali. Esistono diversi motivi per utilizzare le cache web:

- **Riduzione del tempo di risposta per le richieste del client:** Poiché gli oggetti memorizzati nella cache sono disponibili localmente, il tempo di accesso è inferiore rispetto al tempo richiesto per ottenere l'oggetto dal server di origine.
- **Riduzione del traffico sui collegamenti di accesso di un'istituzione:** Utilizzando la cache per memorizzare e servire contenuti comuni, si riduce la necessità di richiedere continuamente gli stessi oggetti dal server di origine, riducendo così il carico sui collegamenti di rete.
- **Aumento della densità di Internet:** Consentendo ai provider di servizi con risorse limitate di fornire contenuti in modo efficiente tramite la cache anziché richiederli ripetutamente dai server di origine.

Per ottimizzare ulteriormente l'uso della cache e ridurre il traffico di rete, viene utilizzato un meccanismo chiamato "**Conditional GET**". In questo scenario, la cache specifica la data dell'ultima copia memorizzata nella richiesta HTTP. Il server di origine, se la copia nella cache è aggiornata, risponderà con un codice di stato "304 Not Modified", indicando che non è necessario trasferire nuovamente l'oggetto.

Evoluzione di HTTP

HTTP/2 è una versione del protocollo HTTP progettata per ridurre la latenza percepita nelle richieste HTTP multioggetto e migliorare la velocità di caricamento delle pagine web.

- **HTTP/1.1:** ha introdotto la possibilità di fare **molteplici GET su una singola connessione TCP**, però il server risponde sempre in-order (**FCFS - Fist Come First Served**) alle richieste GET, e quindi oggetti piccoli che sono in coda devono aspettare la trasmissione degli oggetti grandi prima di loro (*head-of-line (HOL) blocking*)
- **Flessibilità del server:** Sebbene metodi, codici di stato e molti header fields rimangano invariati rispetto ad HTTP 1.1, HTTP/2 permette al server di inviare oggetti in base a una priorità specificata dal client, riducendo il tempo di attesa per i contenuti prioritari.
- **Funzionalità di "push":** HTTP/2 supporta il "push", consentendo al server di inviare oggetti non richiesti dal client ma ritenuti utili per la visualizzazione della pagina, migliorando l'esperienza di navigazione.
- **Mitigazione dell'HOL blocking:** Per affrontare il problema dell'"Head-of-Line" (HOL) blocking, HTTP/2 suddivide ogni messaggio in frame di dimensione definita. I frame dei vari oggetti vengono inviati in modo intersperso, permettendo al client di ricevere e visualizzare i contenuti prioritari mentre attende il completamento del trasferimento degli oggetti meno prioritari.
- **Gestione del framing:** Il framing dei messaggi HTTP/2 avviene tramite un sottolivello del protocollo, in cui il server crea il response message, lo suddivide in frame e li invia tramite un'unica connessione TCP (*i singoli frame devono avere informazioni che consentono la ricomposizione*). Il client ricompone i frame e trasmette il response message al browser.

- **HTTP/3:** Introdotta successivamente, HTTP/3 utilizza *UDP* per la trasmissione dei dati aggiungendo funzionalità di sicurezza e controllo degli errori e della congestione, mirando a migliorare ulteriormente le prestazioni e l'affidabilità del protocollo HTTP.

4.2 Wireshark

Sniffing e Sniffer

- **Cos'è lo sniffing e qual è il ruolo dei prodotti sniffer?**

Lo sniffing è l'attività di intercettazione passiva dei dati in una rete telematica. I prodotti sniffer sono utilizzati per intercettare e memorizzare il traffico di rete e per analizzarlo, sia per scopi legittimi (come individuare problemi di comunicazione o intrusione) che per scopi illeciti (come l'intercettazione fraudolenta di informazioni sensibili).

- **Cosa fa uno sniffer e quali protocolli di comunicazione può analizzare?**

Uno sniffer raccoglie le informazioni che viaggiano lungo una rete e può analizzare il traffico con diversi protocolli di comunicazione, tra cui Ethernet, TCP/IP e IPX.

- **Quali sono le funzioni tipiche degli sniffer?**

Le funzioni tipiche degli sniffer includono il filtraggio e la conversione dei dati e dei pacchetti in una forma leggibile, l'analisi dei difetti di rete, l'analisi delle prestazioni della rete, la ricerca automatica di password e nomi utente, la creazione di log del traffico di rete e la scoperta di intrusioni attraverso l'analisi dei log.

Componenti di un Packet Sniffer

- **Quali sono i requisiti hardware per un packet sniffer?**

La maggior parte dei prodotti lavora con standard network adapters, ma alcuni richiedono hardware speciale che consente di analizzare errori hardware.

- **Qual è il ruolo del capture driver in un packet sniffer?**

Il capture driver è la parte più importante di un packet sniffer. Cattura il traffico di rete dal cavo, lo filtra e memorizza i dati in un buffer.

- **Come funziona il buffer in un packet sniffer?**

Una volta che i pacchetti sono catturati dalla rete, vengono memorizzati in un buffer. Ci sono due modi di cattura: si può catturare finché il buffer non si riempie, oppure si può usare il buffer come un "round robin", dove i dati più nuovi sostituiscono i più vecchi.

Wireshark

- **Qual è lo scopo di Wireshark?**

Wireshark è un packet analyzer utilizzato per mostrare il contenuto dei messaggi inviati e ricevuti dai protocolli a diversi livelli dello stack.

- **Quali sono le funzionalità di Wireshark?**

Wireshark ha una vasta gamma di funzionalità che includono la capacità di analizzare più di 500 protocolli e una ben progettata interfaccia utente. Opera sia su computer che usano Ethernet o WI-FI per connettersi a Internet, sia su quelli che utilizzano protocolli point-to-point come PPP.

Capitolo 5

E-MAIL

La posta elettronica è una delle principali applicazioni (*architettura client/server*) di Internet, un servizio **asincrono**. Conta la presenza di tre componenti principali: **user agents**, **mail servers**, **SMTP** (*Simple Mail Transfer Protocol*).

I **mail servers** costituiscono la parte centrale dell'infrastruttura del servizio di posta elettronica. Forniscono ad ogni *user agent*:

- **mailbox** che gestisce e contiene i messaggi in entrata inviati dagli utenti ad un *user agent*. Un tipico messaggio inizia il proprio viaggio da un user agent, giunge al mail server del mittente e prosegue fino al mail server del destinatario, dove viene depositato nella sua *mailbox*. L'*user agent* per accedere alla propria casella di posta deve essere autenticato dal server che lo ospita tramite nome utente e password.
- **message queue**: *coda di messaggi uscenti* (che devono essere inviati). Il *mail server* si occupa anche di conservare la posta che non è riuscito ad inviare e la trattiene in una coda di messaggi, tendendo di inviarla dopo un intervallo di tempo. Se dopo giorni non riesce ad inviarla, rimuove il messaggio e notifica la mancata consegna al mittente con un messaggio di posta elettronica.

SMTP

SMTP rappresenta il principale protocollo a livello di applicazione per la posta elettronica su Internet. Fa uso del servizio di trasferimento dati affidabile proprio di TCP (*sulla porta 25*) per trasferire le mail del server del mittente a quello del destinatario (*client / server*).

Principali caratteristiche SMTP:

- Una TCP, porta 25;
- **direct transfer**: il server mittente invia direttamente al server ricevente, nessun intermedio;
- *Tre fasi di trasferimento*:
 1. handshaking (greetings)
 2. transfer of messages
 3. closure
- interazione **command/response**:
 - *command*: ASCII text

– *response*: status code

- i messaggi devono essere in ASCII a 7-bit
- Utilizza connessioni TCP persistenti
- Usa "\r\n . \r\n" per definire la fine di un messaggio

Confronto con HTTP:

- HTTP: pull (*dal server prendo l'oggetto*)
SMTP: push (*dal client carico l'oggetto al server*)
- Entrambi hanno un'interazione *command/response* in ASCII e *status code*
- Con HTTP ogni oggetto è incapsulato nel proprio messaggio di risposta ed è necessario eseguire una *GET* per ogetto
- Con SMTP più oggetti sono inviati in un *multipart message*

Formato del messaggio mail:

- *header lines*: to, from, subject (**queste informazioni sono diverse da SMTP MAIL FROM, TO presenti sul protocollo, infatti possono essere falsificati**)
- *una linea bianca*
- il *body* del messaggio scritto in soli caratteri ASCII

Protocolli di accesso mail

Tuttavia, il destinatario di un'e-mail non può utilizzare il protocollo SMTP per prelevare (*pull*) la mail dal **mail server**, dunque dovrà utilizzare un protocollo di tipo **push**.

- **SMTP**: consegna / salvataggio al server del ricevente
- Portocollo di mail access, recupero delle mail dal server:
 - **POP** (*Post Office Protocol*): authorization, download
 - **IMAP** (*Internet Mail Access Protocol*): più funzionalità, inclusa la manipolazione dei messaggi registrati sul server
 - **HTTP** (*per web-mail*): gmail, hotmail

POP, in particolare, è utilizzato per autorizzare e scaricare le email dal server. Durante la fase di autorizzazione, il client invia comandi come "user" e "pass" per dichiarare la propria user-name e password, mentre il server risponde con "+OK" o "-ERR". Successivamente, durante la fase di transazione, il client può eseguire operazioni come elencare i messaggi, recuperarli tramite il loro numero e cancellarli, prima di chiudere la sessione con il comando "quit".

Tuttavia, **POP** presenta limitazioni, come nel caso dell'esempio in cui il destinatario non può rileggere le email se cambia il suo *user agent* di posta. Una soluzione è rappresentata da POP3 in modalità "scarica-e-mantieni", che mantiene copie dei messaggi accessibili da diversi client, ma rimane **STATELESS**. Al contrario, **IMAP** mantiene sempre tutti i messaggi sul server, consentendo agli utenti di organizzarli in cartelle e mantenendo lo stato dell'utente tra le sessioni.

Un'altra modalità di accesso alle email è attraverso i servizi di webmail, che consentono un accesso diretto ai server delle email tramite il protocollo **HTTP**. In questo caso, le email non vengono scaricate sulla macchina del destinatario, permettendo anche l'invio di email attraverso il server SMTP. Questo approccio è comodo e flessibile, consentendo agli utenti di accedere alle proprie email da qualsiasi dispositivo connesso a Internet.

Capitolo 6

DNS

Tutti gli *host* vengono identificati dai cosiddetti **indirizzi IP**, una struttura gerarchica (*perchè leggendo da sinistra a destra si ottengono informazioni sempre più specifiche sulla collocazione dell'host in Internet*) di 4 byte, espressi con un numero decimale compreso tra 0 e 255 separati dai punti.

Dunque, esistono due modi per identificare gli host: il *nome* e l'*indirizzo IP*. Al fine di conciliare i due approcci è necessario l'utilizzo del servizio fornito dal **DNS** (*Domain Name System*) di Internet. DNS è:

1. un **database distribuito** implementato in una gerarchia di **DNS server**;
2. un **protocollo** a *livello di applicazione* (*application-layer protocol*) che consente agli host di interrogare il *database* dei *name servers* per **risolvere** i nomi.

Il protocollo DNS utilizza UDP e la porta 53 e viene comunemente utilizzato da altri protocolli a livello di applicazione (HTTP, SMTP) per tradurre i nomi di host forniti dall'utente in indirizzi IP.

Per esempio l'utente inserisce nella barra degli indirizzi del browser un URL:

1. La stessa macchina su cui è presente il browser utilizza l'API **resolver** per interrogare il client dell'applicazione DNS;
2. Il browser estrae il nome dell'host (**hostname**) dall'URL e invoca la funzione *gethostbyname(hostname)* del resolver;
3. Il resolver DNS invia una query al server DNS per chiedere di tradurre quell'hostname in un indirizzo IP;
4. Quando il resolver DNS riceve la risposta dal server, passa l'indirizzo IP ricevuto al browser;
5. Il browser indica questo indirizzo IP alla socket come indirizzo di destinazione del messaggio.

DNS necessariamente introduce un ritardo aggiuntivo alle applicazioni Internet che lo utilizzano, però spesso l'indirizzo IP desiderato si trova spesso in una cache e questo riduce i ritardi.

Oltre alla traduzione degli hostname in indirizzi IP, DNS mette a disposizione altri importanti servizi:

- **Host aliasing:** un host dal nome complicato può avere uno o più *alias*, sinonimi. Il DNS può essere invocato da un'applicazione per ottenere l'*hostname canonico* di un *alias*, così come l'indirizzo IP dell'host;

- **Mail server aliasing:** un'applicazione di posta può invocare il DNS per ottenere il nome canonico di un sinonimo fornito. Infatti il *record MX* permette al server di posta di una società e al web server di avere hostname (*alias*) identici.
- **Load distribution** (*distribuzione del carico di rete tra server replicati*): i siti con molto traffico vengono replicati su più server, ognuno eseguito su un host diverso con indirizzo IP differente. Nel caso di web server replicati, va dunque associato ad ogni hostname canonico un insieme di indirizzi IP.

Il database DNS contiene questo insieme di indirizzi e quando i clienti effettuano una *query DNS* per un nome associato ad un insieme di indirizzi, il server risponde con l'intero insieme di indirizzi, ma varia l'ordinamento ad ogni risposta.

6.1 Un database distribuito e gerarchico

DNS non può essere centralizzato per vari problemi:

- **One point of failure:** se il DNS server si guasta, Internet collassa;
- **Volume di traffico:** un singolo DNS server dovrebbe gestire tutte le richieste;
- **Distanza dal database centralizzato:** molti client potrebbero trovarsi troppo lontano dal server; **Manutenzione:** il singolo DNS server sarebbe difficile da essere aggiornato frequentemente.

Di conseguenza, il DNS utilizza un gran numero di server, organizzati in maniera gerarchica e distribuiti nel mondo. Nessun DNS server ha le corrispondenze per tutti gli host di Internet ma sono distribuite tra i server. Esistono tre classi principali di DNS server: i **root server**, i

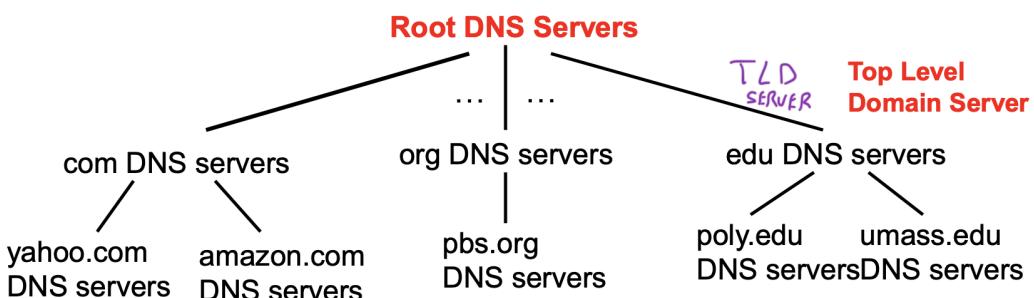


Figura 6.1: DNS: un database distribuito gerarchico

TDL (*Top-Level Domain server*, es: .it, .com) e gli **authoritative server** (*server autoritativi*, se ne occupa l'organizzazione o può essere affidati ad un service provider).

Il **DNS server locale** non appartiene strettamente alla gerarchia di server, ma è comunque centrale nell'architettura DNS. Ciascun *ISP*, come un'università, un'azienda o un *ISP* residenziale, ha un **DNS server locale**, detto anche **default name server**. Quando un host si connette ad un **ISP**, quest'ultimo gli *fornisce un indirizzo IP* tratto da uno o più dei suoi DNS server locali.

Quando un host effettua una richiesta DNS, la *query* viene inviata al DNS server locale che opera da proxy e, se la traduzione non è presente nella cache locale, inoltra la query alla gerarchia dei DNS servers.

Esistono due tipi di *query DNS*:

1. **iterated query**: il server contattato replica con il nome di un altro server da contattare;
2. **recursive query**: affida la risoluzione del nome al name server contattato, però questo metodo pone un pesante carico ai livelli più alti della gerarchia.

Una volta che un server DNS impara un mapping (cioè l'associazione tra un nome e un indirizzo IP), lo memorizza nella sua cache per un certo periodo di tempo. Questo è utile per evitare di dover cercare nuovamente informazioni che sono state recentemente richieste. Tuttavia, è importante notare che queste informazioni potrebbero diventare obsolete se l'host cambia il suo indirizzo IP senza notificarlo a tutti i server DNS interessati. **Se il client vuole essere certo di un'informazione deve chiedere una risposta autoritativa.**

DNS Resource Records

DNS è un database distribuito che mantiene informazioni memorizzate come **resource records RR** format: (*Name*, *Value*, *Type*, *TTL*), *TTL* = time to leave, tempo residuo di vita di un record e determina quando la risorsa va rimossa dalla cache.

- **type=A - Assign**: fornisce la corrispondenza tra hostname e indirizzo IP, quindi *Name* è il nome dell'host e *Value* è il suo indirizzo IP;
- **type=NS - NextServer**: instrada le richieste DNS successive al prossimo DNS che ha più informazioni. Allora, *Name* è un dominio e *Value* è l'hostname del DNS server autoritativo che sa come ottenere gli indirizzi IP degli host del dominio;
- **type=CNAME - CanonicalName**: può fornire agli host richiedenti il nome canonico relativo ad un hostname, dove *Value* è il nome canonico dell'host per il sinonimo *Name*;
- **type=MX - RecordMX**: consente agli hostname dei mail server di avere sinonimi semplici, allora *Value* è il nome canonico di un mail server che ha il sinonimo *Name*.

I messaggi inviati tra i client e i server DNS (**query e reply**) hanno entrambi lo stesso *message format* ed utilizzano **UDP (porta 53)** come protocollo di trasporto predefinito, anche se, in caso di troncamento della richiesta viene utilizzato TCP.

La normale interazione con il Domain Name Server consiste di una richiesta ed una risposta:

- una richiesta DNS consiste di un solo *datagram*;
- La risposta potrebbe superare il limite di 512bytes e richiedere più datagram, un flag indicherà la divisione.

6.2 Message format:

- **header section** (*sezione di intestazione*): i primi 12 byte, a sua volta contiene:
 - **identificatore**: numero di 16bit che identifica la richiesta. Viene copiato nei messaggi di risposta ad una richiesta, consentendo al client di far corrispondere le risposte ricevute con le query inviate.
 - **flags**: 16bit che contengono varie informazioni, come (*query/response*, *risposta ad una richiesta autoritativa*, *iterative/recursive query*).
 - **4 campi** che indicano il numero di occorrenze delle quattro sezioni di *tipo dati* che seguono l'intestazione
- **sezione delle domande**: contiene informazioni sulle richieste che stanno per essere effettuate, include:
 1. un campo **nome** con il nome che sta per essere richiesto;
 2. un campo **tipo** che indica il tipo della domanda sul nome, per esempio type A / MX.
- **sezione delle risposte**: nelle risposte provenienti da DNS server contiene i record di risorsa relativi al nome originariamente richiesto. Una risposta può restituire più **RR**, dato che un hostname può avere più indirizzi IP
- **sezione autoritativa**: contiene i record di altri server autoritativi
- **sezione aggiuntiva**: racchiude altri record utili, per esempio se il campo di risposta relativo ad una richiesta MX contiene un record di risorsa che fornisce l'hostname canonico del server di posta, la sezione aggiuntiva contiene un record di tipo A che fornisce l'indirizzo IP relativo all'hostname canonico del server di posta.

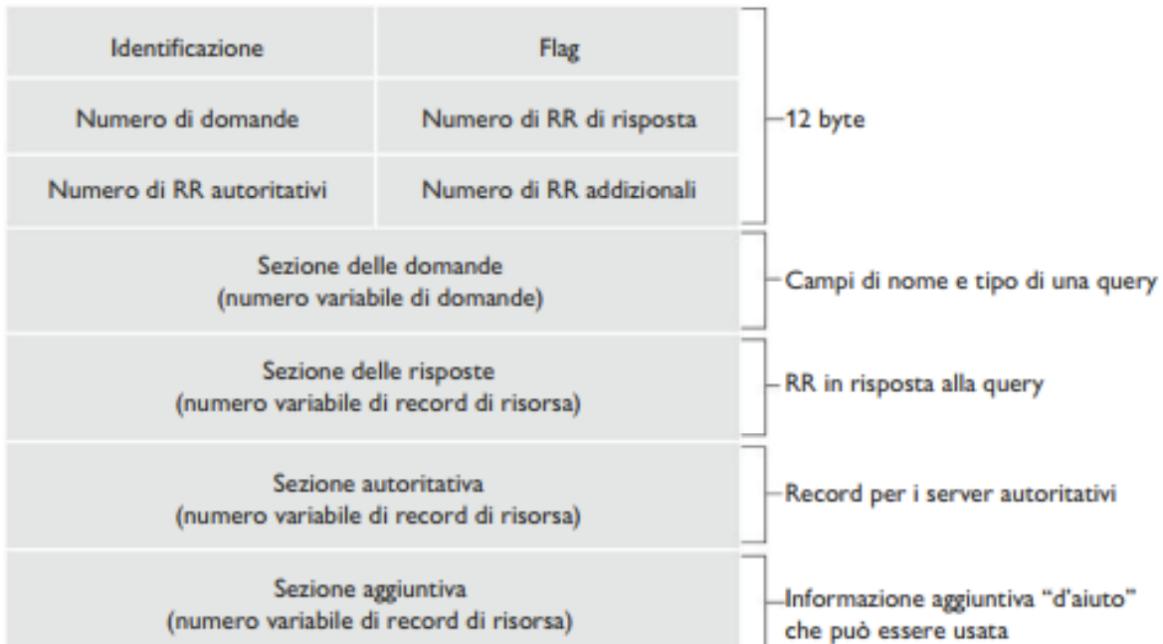


Figura 6.2: Formato dei messaggi DNS

Inserimento di record nel database DNS

Infine, quando si desidera registrare un nome di dominio, è necessario passare attraverso un processo di registrazione gestito da organizzazioni come ICANN. Queste organizzazioni delegano la responsabilità per i domini a **registrar** (*azienda che verifica l'identità del nome di dominio, lo inserisce nel database DNS e richiede una somma di denaro per i servizi*) che gestiscono i nomi di dominio a livello pratico. Una volta registrato un nome di dominio, è possibile specificare quali server DNS autoritativi devono gestirlo, così come altri dettagli come gli alias e gli indirizzi IP associati al dominio.

Capitolo 7

Streaming & Content Distribution Network

Il traffico video consuma la maggior parte della banda di trasmissione. Mandare in streaming una tale mole di traffico a tutti gli utenti è una grande sfida in quanto dovrebbe soddisfare i principi di **scalabilità** (*un singolo data center non funziona*) ed **eterogeneità** (*differenti utenti hanno diverse proprietà, wired vs mobile, bandwidth rich vs bandwidth poor*).

Un **video** è una sequenza di immagini, visualizzate tipicamente ad una velocità costante. Un'immagine non compresa e codificata digitalmente consiste di un array di pixel, ognuno dei quali codificato con un numero di bit per rappresentare luminanza e crominanza. I video possono essere compressi in modo da raggiungere un compromesso tra qualità del video e **bit rate**, gli algoritmi di compressione permettono di comprimere un video in qualsiasi bit rate considerando cambiandone ovviamente la qualità.

Bit-rate: num. bit/sec richiesti, **throughput**: num. bit/sec a disposizione

La **codifica** usa la ridondanza **within** e **between** le immagini per decrementare il numero di bits usati per codificare un'immagine:

- *spaziale* (within)
- *temporale* (between)

7.1 Streaming HTTP e DASH

Sul lato client i byte vengono memorizzati in un buffer dell'applicazione client, quando il numero di byte nel buffer supera una certa soglia fissata, l'applicazione client inizia la riproduzione e periodicamente prende i frame video dal buffer, decomprime e li visualizza all'utente.

Quindi l'applicazione di video streaming consiste nel visualizzare i frame mentre li riceve, memorizzando gli ultimi nel buffer.

Lo *streaming HTTP* presenta un grande **svantaggio**: i client ricevono tutta la stessa versione codificata del video, nonostante abbiano a disposizione una larghezza di banda che può variare tra i dispositivi e nel tempo.

Per superare questo problema, è stato sviluppato un *nuovo tipo di streaming basato su HTTP*, chiamato **DASH** (*Dynamic Adaptive Streaming over Http*), ovvero un *protocollo a livello d'applicazione*.

Dal lato **server**:

- i video vengono divisi in molteplici pezzetti (**chunks**);

- ogni *chunk* registrato è codificato a bit-rate differenti e quindi con diversi livelli di qualità;
 - fornisce un **manifest file**, ovvero gli *URLs* per i differenti pezzetti con i relativi bit-rate.
- dal lato **client**:

- misura periodicamente la banda tra server e client;
- consulta il manifest e richiede un *chunk* alla volta;
- sceglie il *coding rate* massimo sostenibile ($throughput > bit-rate$) data la banda correntemente disponibile;
- può scegliere differenti coding rates a differenti momenti.

Con *DASH* è il client a preoccuparsi di garantire uno streaming fluido in quanto è lui a determinare:

- **dove** richiedere i chunk, può richiederli da un server più vicino o da uno con un'ampiezza di banda disponibile più alta;
- **come, a quale coding rate** richiederlo, decide la qualità in base alla banda disponibile;
- **quando** richiedere un chunk, per evitare overflow o buffer starvation.

L'approccio più diretto per le aziende di streaming video in Internet sarebbe costruire un unico enorme data center, memorizzare in esso tutti i video e mandarli in streaming direttamente. Però quest'approccio porterebbe varie problematiche:

1. Single point of failure
2. Punto di congestione di rete
3. Path lunghi verso i clients distanti
4. Molteplici copie dello stesso video inviate sugli stessi link uscenti, sprecando banda e pagando più volte gli ISP

Per superare queste problematiche, quasi tutte le maggiori aziende di streaming usano le **CDN**, *Content Delivery Networks* (reti di distribuzione di contenuto). Una CDN gestisce server distribuiti in posti diversi, memorizza copie dei video e cerca di dirigere le richieste degli utenti al punto della CDN in grado di offrire il servizio migliore.

Le CDN adottano una delle due politiche di dislocazione dei server:

1. **Enter Deep**: posizionare i CDN server in profondità in quante più reti di accesso possibili. **Vantaggio**: vicino agli utenti; **Svantaggio**: difficile sincronizzare i vari server.
2. **Bring Home**: un numero più piccolo di grandi cluster vicino ai *Point of Presence*, **PoPs** e **IXP**, degli ISP di livello 1. **Vantaggio**: meno problemi di gestione e manutenzione; **Svantaggio**: spesso minore qualità video.

Come funziona la CDN

Quando un browser nell'host di un utente chiede il recupero di una specifico video identificato a un URL, la CDN deve intercettare la richiesta in modo da poter:

- determinare il cluster di server più appropriato per quel cliente a quell'istante;
- dirigere la richiesta del client ad uno dei server di quel cluster.

Molte CDN sfruttano il servizio DNS per intercettare e ridirigere le richieste.

Bob (client) richiede video <http://netcinema.com/6Y7B23V>

- video memorizzato nella CDN a <http://KingCDN.com/NetC6y&B23V>

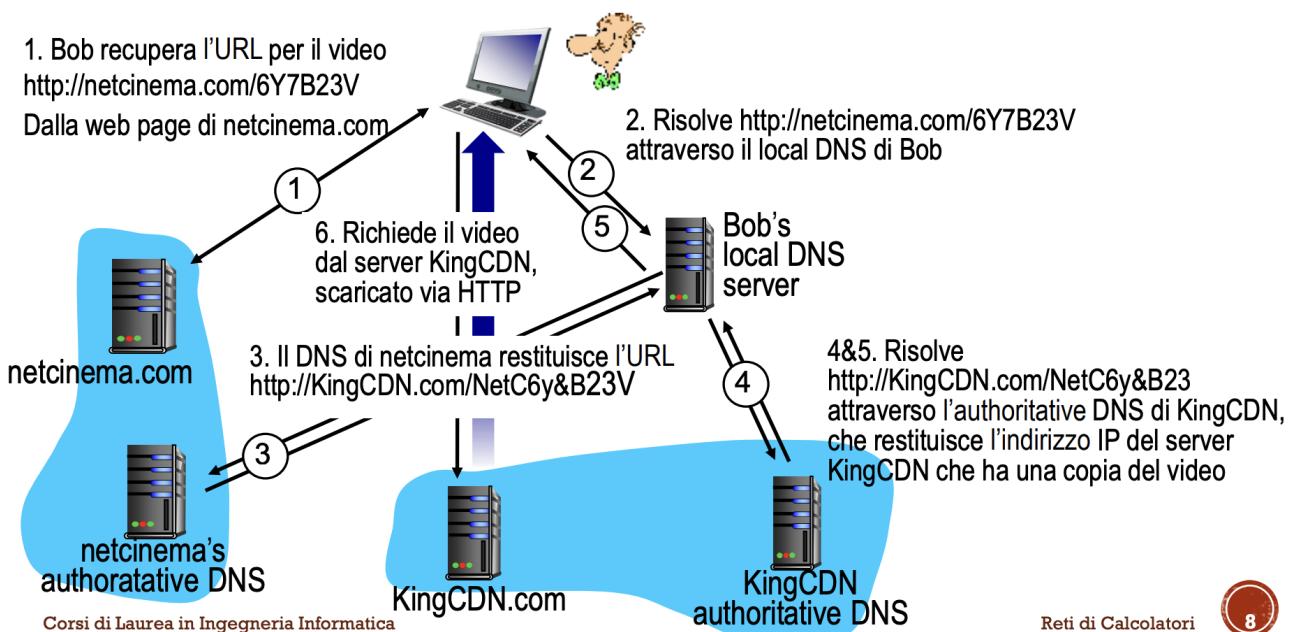


Figura 7.1: Accesso ai contenuti in CDN

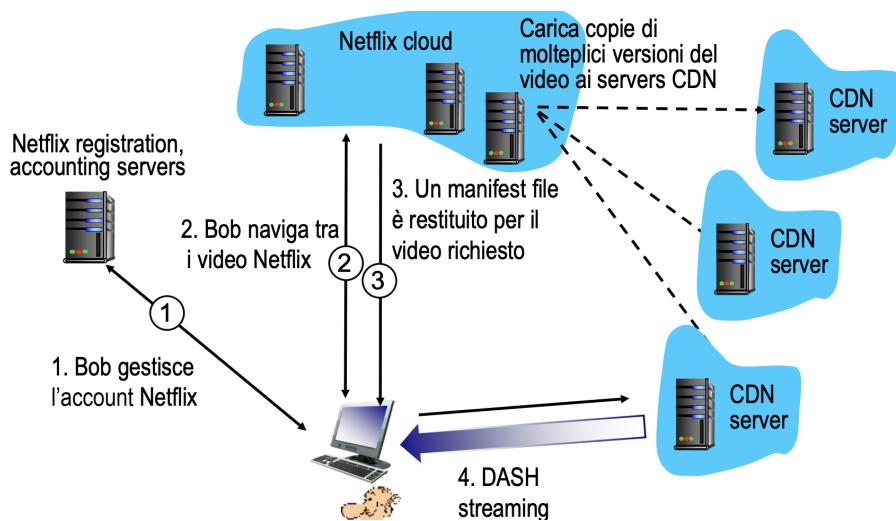


Figura 7.2: Caso di studio: Netflix

Capitolo 8

Livello di trasporto

Posto tra il livello di applicazione e quello di rete, il livello di trasporto constituisce una parte centrale dell'architettura stratificata delle reti e riveste la funzione critica di fornire servizi di comunicazione direttamente ai processi applicativi in esecuzione su host differenti.

Dunque, *il livello di trasporto fornisce una connessione logica tra processi applicativi di host differenti, utilizzando i servizi forniti dai livelli inferiori. I protocolli a livello di*

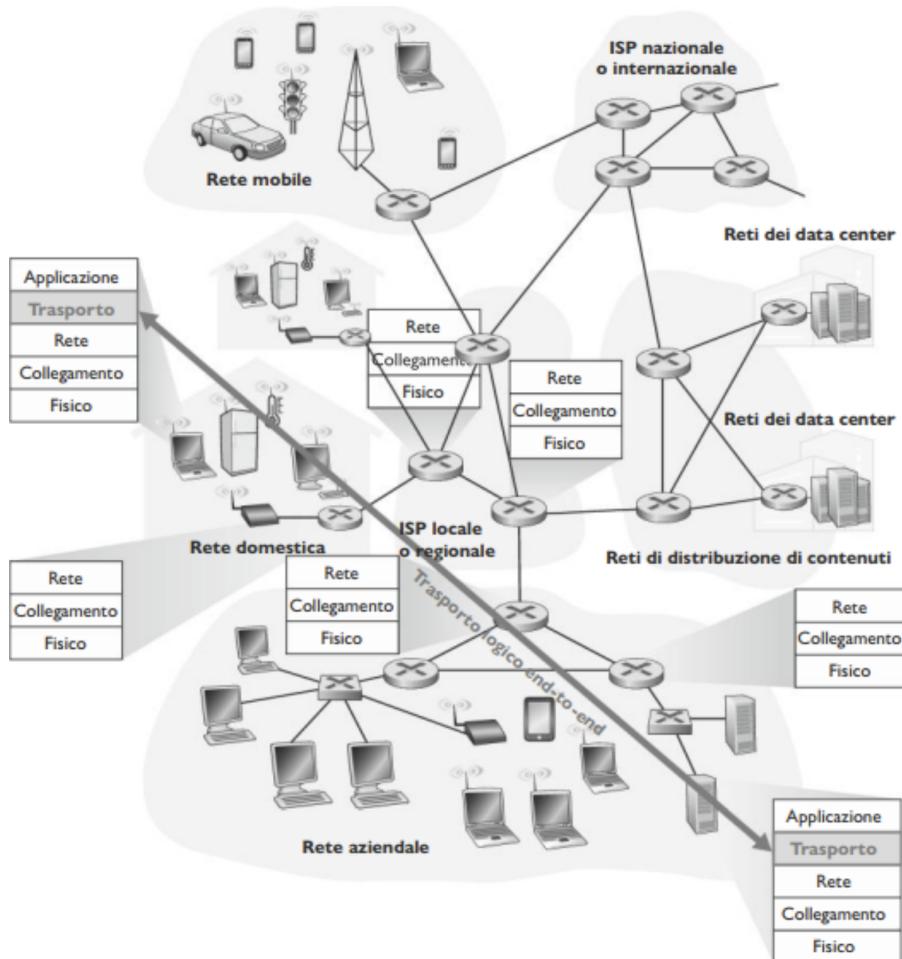


Figura 8.1: Il livello di trasporto fornisce comunicazione logica anziché fisica tra due processi applicativi

trasporto sono implementati nei sistemi periferici, ma non nei router della rete.

- Dal lato mittente, il livello di trasporto converte i messaggi che riceve da un processo applicativo in pachetti a livello di trasporto, detti **segmenti** (*transport-layer segment*).

Questo avviene spezzando i messaggi applicativi in parti più piccole ed aggiungendo a ciascuna di esse un'intestazione di trasporto per creare un segmento. Successivamente il livello di trasporto passa il segmento al livello di rete, dove viene incapsulato all'interno di un pacchetto a livello di rete (*datagramma*) ed inviato a destinazione.

- Dal lato del ricevente, il livello di rete estraе il segmento dal datagramma e lo passa al livello di trasporto. Quest'ultimo elabora il segmento ricevuto e *fornisce i dati ai processi applicativi tramite socket*.
- **Comunicazione processo-processo:** *strato di trasporto - tramite socket / porta*
- **Comunicazione host-host:** *strato di rete - tramite IP*
- **Comunicazione nodo-nodo:** *strato di collegamento*

8.1 Relazione tra i livelli di trasporto e di rete

Internet, o più in generale una rete TCP/IP, mette a disposizione del livello di applicazione due diversi protocolli: **UDP** (*User Datagram Protocol*) che fornisce alle applicazioni un servizio non affidabile e non orientato alla connessione, l'altro è **TCP** (*Transmission Control Protocol*), che offre un servizio affidabile e orientato alla connessione.

Il **protocollo a livello di rete** di Internet si chiama **IP (Internet Protocol)**, fornisce una comunicazione logica tra due host. Il suo modello di servizio prende il nome di **best-effort delivery service**, o semplicemente **best effort**. Non assicura né la consegna né l'integrità dei segmenti né il rispetto dell'ordine originario. Per queste ragioni si dice che IP offre un servizio **non affidabile** (*unreliable service*). *Ciascun host possiede un indirizzo IP*.

Multiplexing e demultiplexing a livello di trasporto

Dunque, i protocolli a livello di trasporto offrono un'estensione del servizio di consegna di IP "*tra sistemi periferici*" a quello di consegna "*tra processi in esecuzione sui sistemi periferici*".

Questo passaggio, da consegna host-to-host a consegna process-to-process viene detto *multiplexing e demultiplexing a livello di trasporto*, ciò come il servizio di trasporto da host a host fornito a livello di rete possa diventare un servizio di trasporto da processo a processo per le applicazioni in esecuzione sugli host.

Nell'host destinatario il livello di trasporto riceve segmenti dal livello di rete e deve consegnare i dati al processo applicativo appropriato in esecuzione nell'host.

- **multiplexing:** il compito di radunare frammenti di dati da diverse socket sull'host di origine ed incapsularne ognuno con intestazioni a livello di trasporto per creare dei segmenti e passarli a livello di rete
- **demultiplexing:** il compito di trasportare i dati dei segmenti a livello di trasporto verso la giusta socket

Formato dei segmenti a livello di trasporto:

- Sappiamo che il multiplexing a livello di trasporto richiede (1) che le socket abbiano identificatori unici e (2) che ciascun segmento presenti campi che indichino la socket a cui va consegnato il segmento.

Numero di porta del mittente e del destinatario (*32 bit, 16 bit per porta*), indica l'applicazione a cui deve rispondere e l'applicazione a cui deve inviare il messaggio.

- altri campi dell'intestazione
- dati dell'applicazione (*payload*)

Gestione dei numeri di porta:

- I numeri di porta sono gestiti dall'**IANA** (*Internet Assigned Number Authority*)
- porte da 0 a 1023, porte ben note assegnate dall'IANA
- porte da 1024 a 49151, possono essere assegnate dall'IANA
- da 49152 a 65535, porte effimere

Multiplexing e Demultiplexing orientati e non alla connessione

- **Non orientati alla connessione (UDP/IP):** una socket UDP viene identificata completamente da una coppia data da un indirizzo IP e di un numero di porta di destinazione.

Di conseguenza due segmenti UDP che presentano diversi indirizzi IP e/o diversi numeri di porta di origine, ma hanno stesso indirizzo IP e stesso numero di porta di destinazione, vengono diretti allo stesso processo di destinazione tramite la medesima socket.

Dunque, un numero di porta identifica un punto di collegamento con un'applicazione (*socket*). Per le **socket UDP** l'interfaccia è una coda di segmenti, una per l'entrata ed una per l'uscita. La socket per una porta è unica, quindi tutti i segmenti provenienti da mittenti differenti o da applicazioni differenti andranno nella stessa coda.

- **Orientati alla connessione (TCP/IP):** a differenza della socket UDP, la socket TCP è identificata da 4 parametri, indirizzo IP di origine, numero di porta di origine, indirizzo IP di destinazione e numero di porta di destinazione.

In particolare, a differenza di UDP, due segmenti TCP in arrivo, aventi indirizzi IP di origine o numeri di porta di origine diversi, vengono diretti a due socket differenti, anche a fronte di indirizzo IP e porta di destinazione uguali, con l'eccezione dei segmenti TCP di *handshaking*.

Invece, una **socket TCP** è costituita da due buffer circolari, nel buffer del mittente sono presenti i byte da spedire lungo la connessione, invece il buffer del destinatario legge i byte in arrivo.

C'è un buffer per ogni connessione, quindi anche una socket per connessione.

8.1.1 Differenza tra UDP e TCP

- **Comunicazione non orientata alla connessione:** Dati spediti senza bisogno di stabilire una connessione prima, i datagrammi sono spediti indipendenti, utilizza il protocollo *UDP*

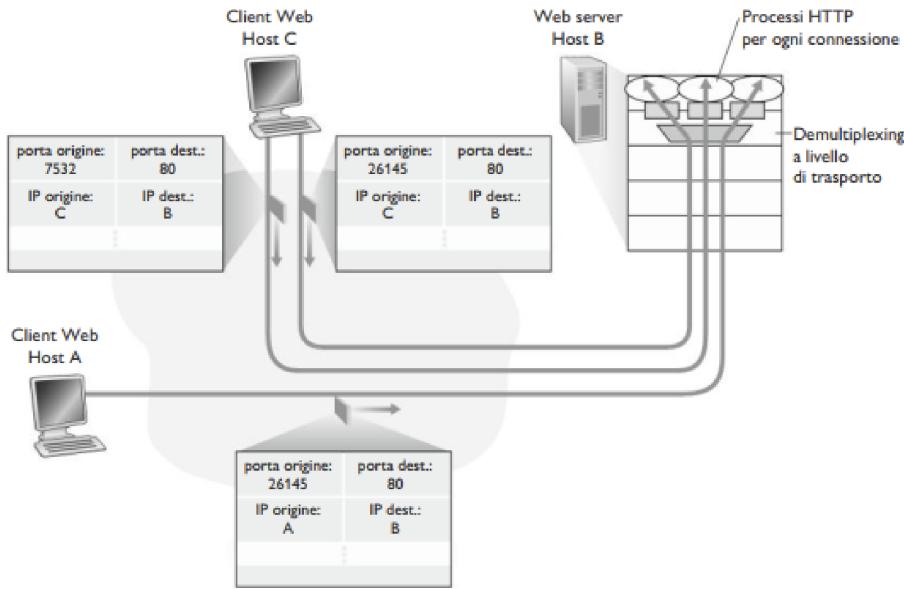


Figura 8.2: Due client che usano lo stesso numero di porta (80) di destinazione per comunicare con la stessa applicazione sul web server

- **Comunicazione orientata alla connessione:** Prima di iniziare la comunicazione bisogna instaurare la connessione (una sorta di circuito digitale tra i due host dove passa un flusso di bit), i datagram appartenenti alla connessione, utilizza protocollo *TCP*

UDP

- UDP fa il minimo che un protocollo di trasporto debba fare, a parte la funzione di multiplexing/demultiplexing non aggiunge nulla ad IP.
- UDP prende i messaggi dal processo applicativo, aggiunge il numero di porta di origine e destinazione, aggiunge altri due piccoli campi (**length** e **checksum**) e passa il segmento al livello di rete. Questo viene incapsulato in un datagramma IP e viene effettuato il tentativo di consegnarlo all'host di destinazione in modalità best-effort.

Il **checksum UDP** serve per il rilevamento degli errori, viene utilizzato per determinare se i bit del segmento UDP sono stati alterati durante il loro trasferimento.

- Dal lato mittente UDP effettua il complemento a 1 della somma di tutte le parole da 16 bit nel segmento, l'eventuale riporto finale viene aggiunto al primo bit.
- Tale risultato viene posto nel campo *Checksum* del segmento UDP.
- In ricezione si sommano le tre parole iniziali ed il *checksum*, se non ci sono errori nel pacchetto, l'addizione darà 1111111111111111, altrimenti se un bit vale 0 sappiamo che è stato introdotto almeno un errore nel pacchetto

Capitolo 9

Principi di Trasferimento Affidabile dei Dati

Il problema dell'implementazione di un trasferimento dati affidabile si verifica non solo a livello di trasporto ma anche a livello di applicazione e collegamento.

Con un canale affidabile a disposizione nessun trasferimenti di bit dei dati è corrotto, o va perduto e tutti i bit sono consegnati nell'ordine di invio.

Il compito di un **protocollo di trasferimento dati affidabile**(RDT - *reliable data transfer protocol*) è l'astrazione del servizio offerto alle entità dei livelli superiori di un'implementazione di un canale di trasmissione affidabile.

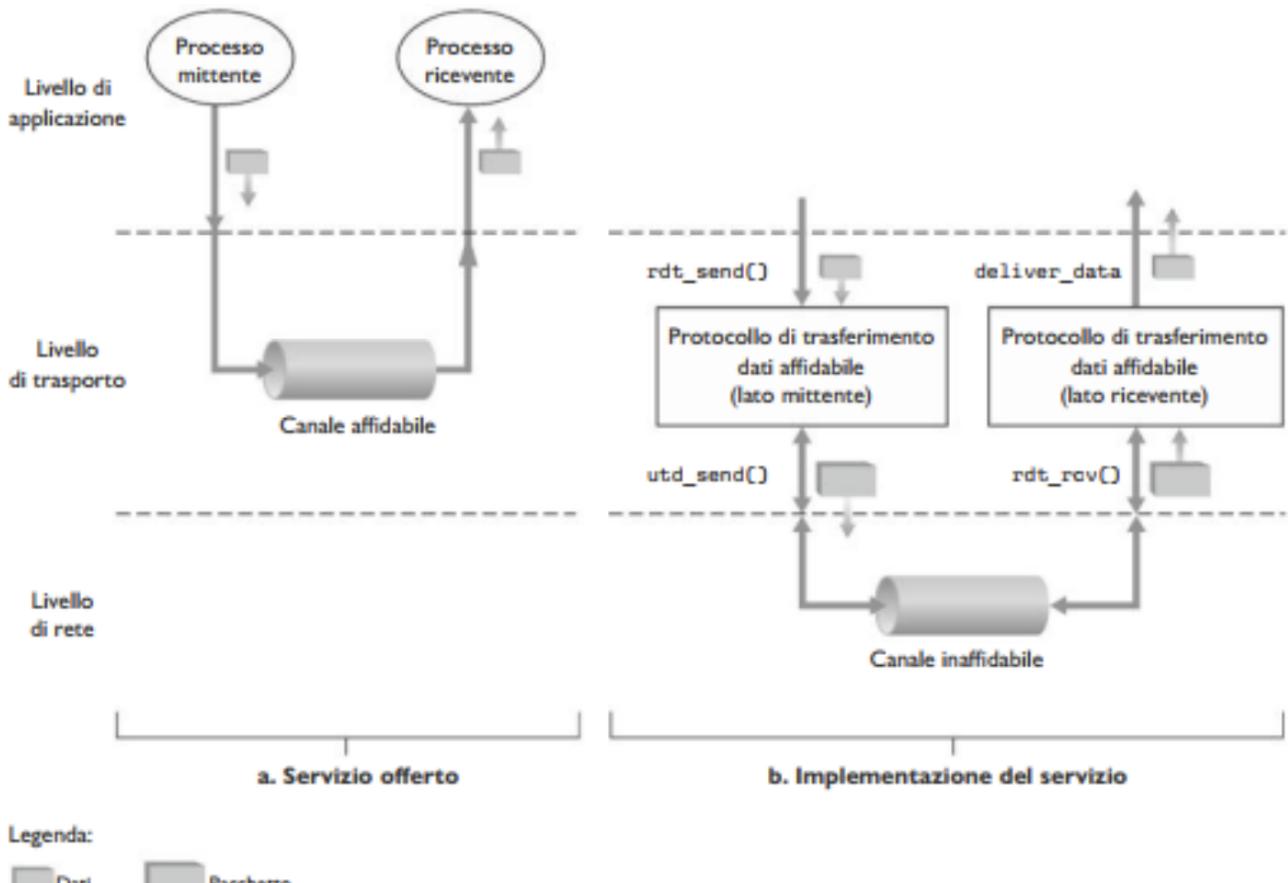


Figura 9.1: Trasferimento dati affidabile: modello di servizio e implementazione

Premesse

- Consideriamo solo il caso di **trasferimento dati unidirezionale**
 - Utilizzeremo **Macchine a stati finiti** (MSF), o *automa*, che permette di descrivere in maniera semplice e concisa il comportamento di molti sistemi.
- Una MSF è rappresentata da un grafo: i nodi sono gli stati del sistema e gli archi rappresentano le transizioni di stato. *Per ogni arco è indicato l'evento che causa la transizione di stato e le azioni prodotte dalla transizione di stato.*
- consideriamo una serie di protocolli rdt sempre più complessi fino ad arrivare a quello più completo.

9.0.1 RDT 1.0: canale affidabile

Caso più semplice in cui il canale sottostante è completamente affidabile.

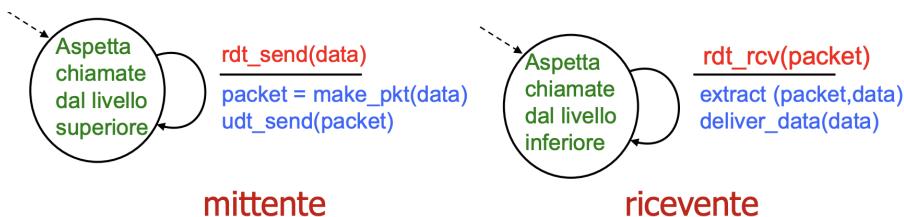


Figura 9.2: rdt1.0: protocollo per un canale completamente affidabile

9.0.2 RDT2.0: canale con errori sui bit

Sono necessari dei messaggi di controllo per notificare al mittente se il dato ricevuto è corretto o corrotto.

I protocolli di trasferimento dati affidabili basati su ritrasmissioni sono noti come **protocolli ARQ** (*Automatic Repeat reQuest*) ed hanno tre funzionalità aggiuntive:

- **Rilevamento dell'errore (error detection)**: i bit verranno raccolti nel campo di checksum nel pacchetto dati *rdt2.0*.
- **Feedback del destinatario (receiver feedback)**: il destinatario invia risposte di notifica positiva (ACK) o negativa (NAK) per indicare il bisogno di ritrasmissione o meno.
- **Ritrasmissione (retransmission)**: un pacchetto ricevuto con errori sarà ritrasmesso dal mittente.

Per questo tipo di comportamento i protocolli *rdt2.0* sono noti come **protocolli stop-and-wait**. Potrebbe succedere che i messaggi di notifica (ACK, NAK) vengono corrotti, il mittente non può sapere se il pacchetto è stato ricevuto correttamente o meno. Per risolvere questo problema, si possono aggiungere campi di checksum ai pacchetti ACK e NAK o richiedere la ritrasmissione della notifica. Tuttavia, la richiesta di ritrasmissione potrebbe innescare un ciclo infinito. Aggiungere bit al checksum potrebbe complicare la soluzione e non risolvere il problema della perdita dei pacchetti.

La soluzione adottata, è quella di **ritrasmettere in caso di ricezione di ACK/NAK alterato**, che tuttavia introduce la possibilità di *pacchetti duplicati*.

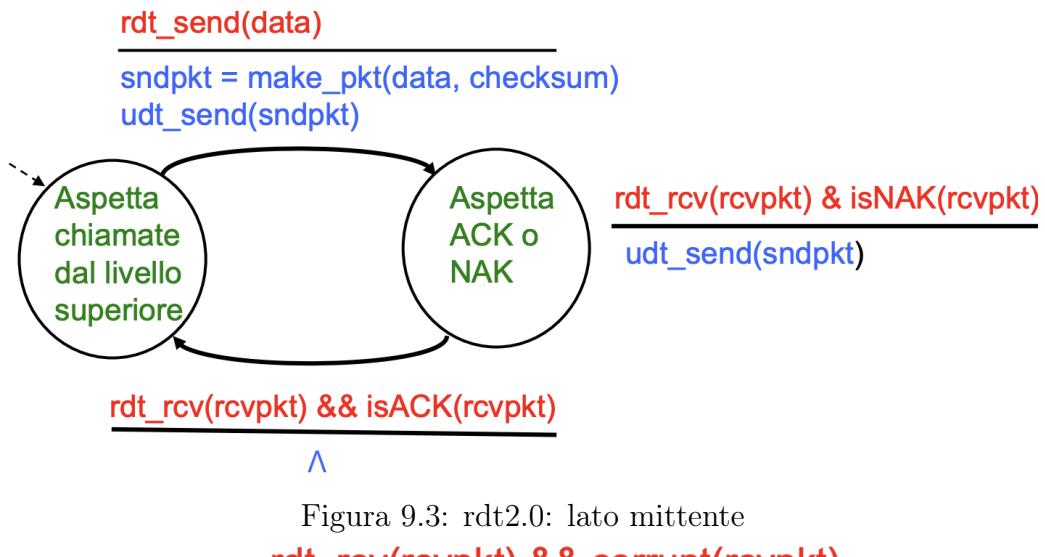


Figura 9.3: rdt2.0: lato mittente

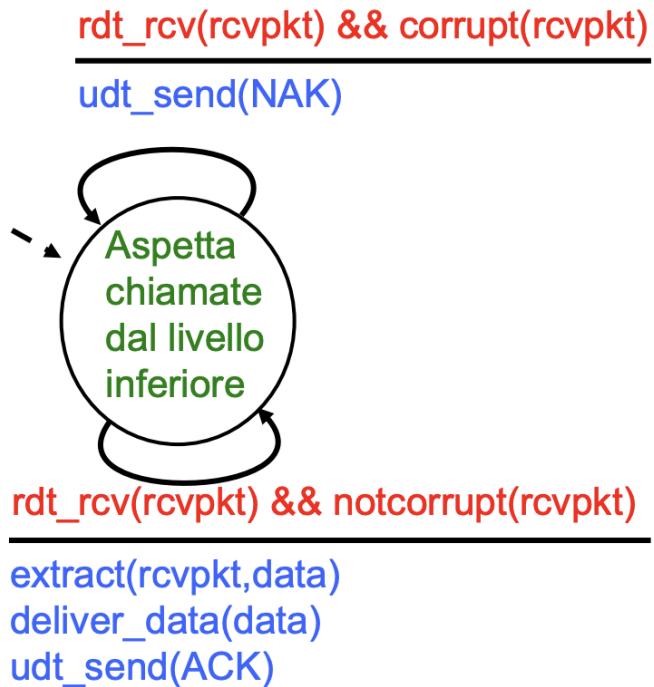


Figura 9.4: rdt2.0: lato ricevente

Per gestire i pacchetti duplicati, si utilizzano i numeri di sequenza, aggiungendo un campo all'intestazione di ogni pacchetto contenente il numero di sequenza. Con un solo bit per i protocolli stop-and-wait, il mittente alterna tra 0 e 1. Se il destinatario riceve un pacchetto con lo stesso numero di sequenza del precedente, lo considera una copia.

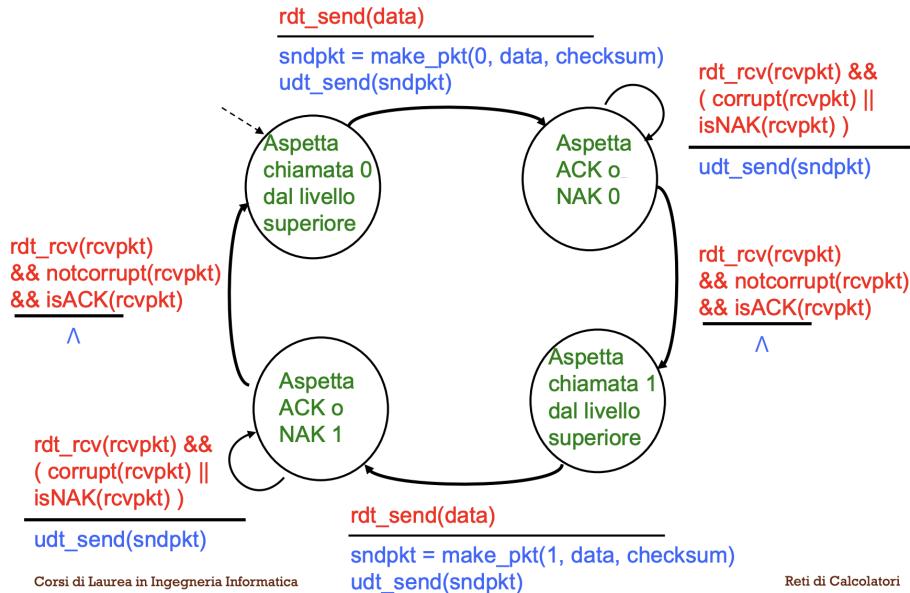


Figura 9.5: rdt2.1: lato ricevente

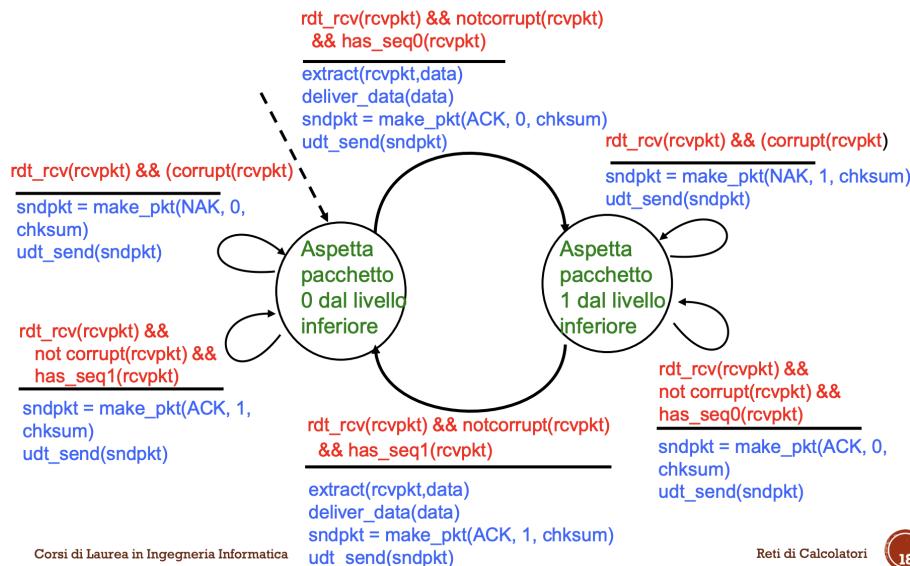


Figura 9.6: rdt2.1: lato ricevente

Modifiche avvenute ad rdt2.1:

- Dal lato del *mittente*:
 - Aggiunto un numero di sequenza per ogni pacchetto
 - Verificare se le notifiche ricevute sono corrette
 - Ogni stato di attesa deve ricordare se dobbiamo aspettare un pacchetto con numero di sequenza 0 o 1

- Dal lato del *ricevente*:

- Deve verificare se il pacchetto ricevuto è un duplicato
- Non sa se l'ultima notifica inviata è stata ricevuta correttamente o meno

Le **NAK** vengono inviate dal destinatario quando riceve un pacchetto alterato. Possiamo ottenere lo stesso risultato inviando un ACK riferito all'ultimo pacchetto ricevuto correttamente.

Il mittente riceve un ACK duplicato e capisce che il pacchetto successivo a quello confermato non è stato ricevuto correttamente.

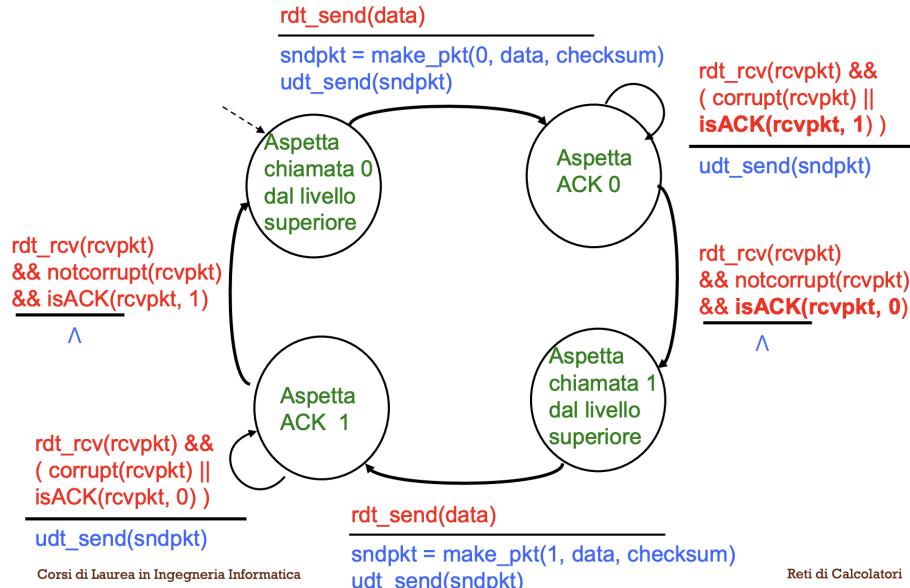


Figura 9.7: rdt2.2: lato ricevente

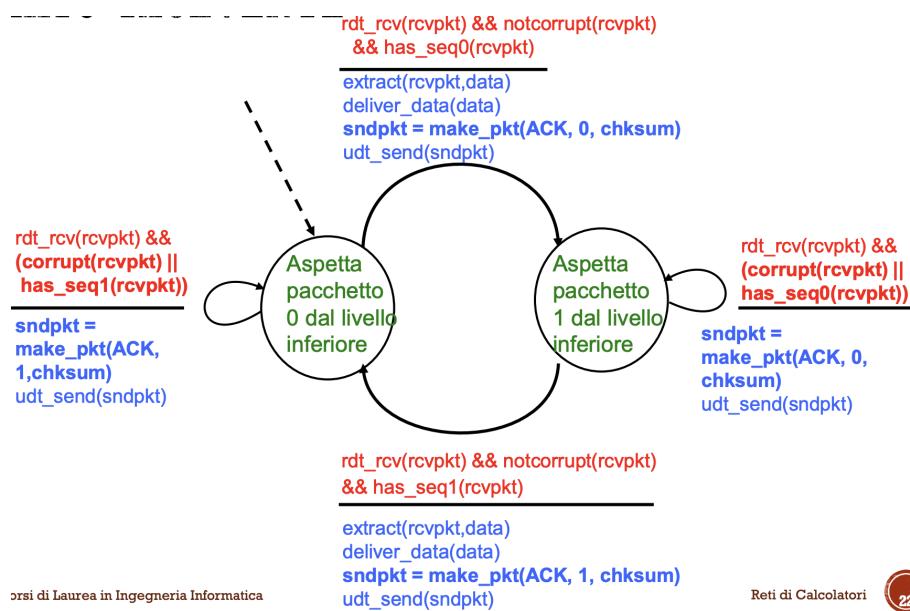


Figura 9.8: rdt2.2: lato ricevente

9.0.3 RDT3.0: canale con perdite di pacchetti ed errori sui bit

Se il canale di trasmissione, oltre a danneggiare i bit, può anche perdere i pacchetti, il protocollo deve preoccuparsi di due aspetti aggiuntivi: *come rilevare lo smarrimento e come agire in questi casi.*

Sarà compito del mittente gestire questa situazione, in quanto è l'unico a sapere se era stato veramente inviato un pacchetto, aggiungendo un *contatore* (*countdown timer*) che indica il periodo di tempo che deve aspettare il messaggio di notifica prima di aver dato quel pacchetto per smarrito.

Potrebbe succedere che il pacchetto arrivi al ricevente dopo la scadenza del *timer*, questo caso comporterebbe l'invio di pacchetti duplicati, problema già risolto da rdt2.2.

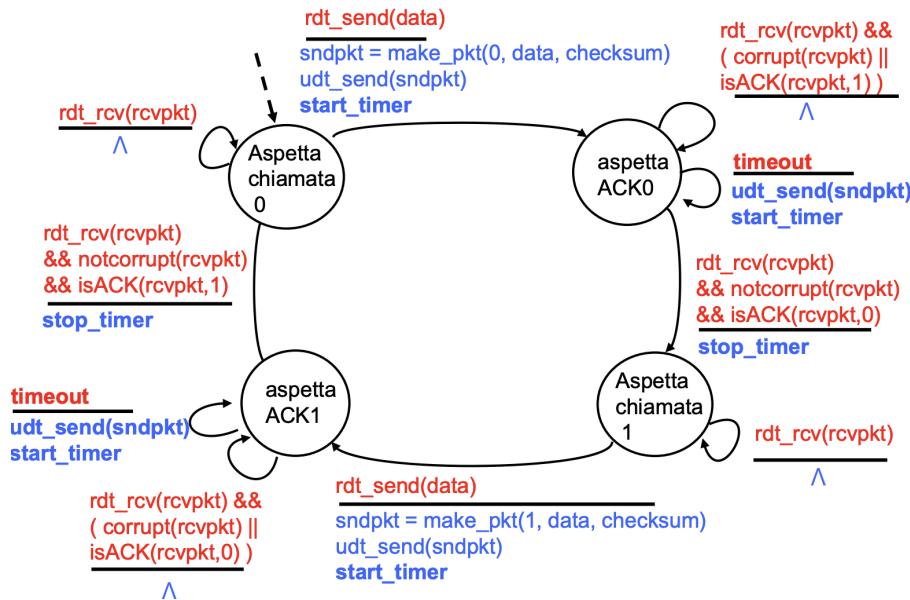


Figura 9.9: rdt3.0: lato mittente

Dunque, gli elementi chiave di un **protocollo di trasferimento dati** sono: *checksum, numeri di sequenza, contatori e pacchetti di notifica positiva e negativa.*

9.1 Protocolli per il trasferimento dati affidabile con pipeline

I protocolli stop-and.wait sono corretti ma particolarmente inefficienti, di conseguenza vengono utilizzati solo su canali molto rumorosi.

Per ovviare a questo problema si utilizzano i **protocolli con pipeline**, i quali consentono al mittente di inviare più pacchetti senza attendere gli *acknowledgment*.

Un protocollo *RDT* con pipeline richiede:

- un buffer per mantenere i pacchetti non ancora riscontrati
- un insieme di numeri di sequenza più ampio per garantire che ogni pacchetto sulla rete sia identificato univocamente.

La quantità di numeri di sequenza necessari ed i requisiti di buffer dipendono dal modo in cui il protocollo di trasferimento dati reagisce ai pacchetti smarriti, alterati o troppo in ritardo. Si possono identificare due approcci di base verso la risoluzione degli errori con pipeline: **Go-Back-N** e **ripetizione selettiva** (*selective repeat*).

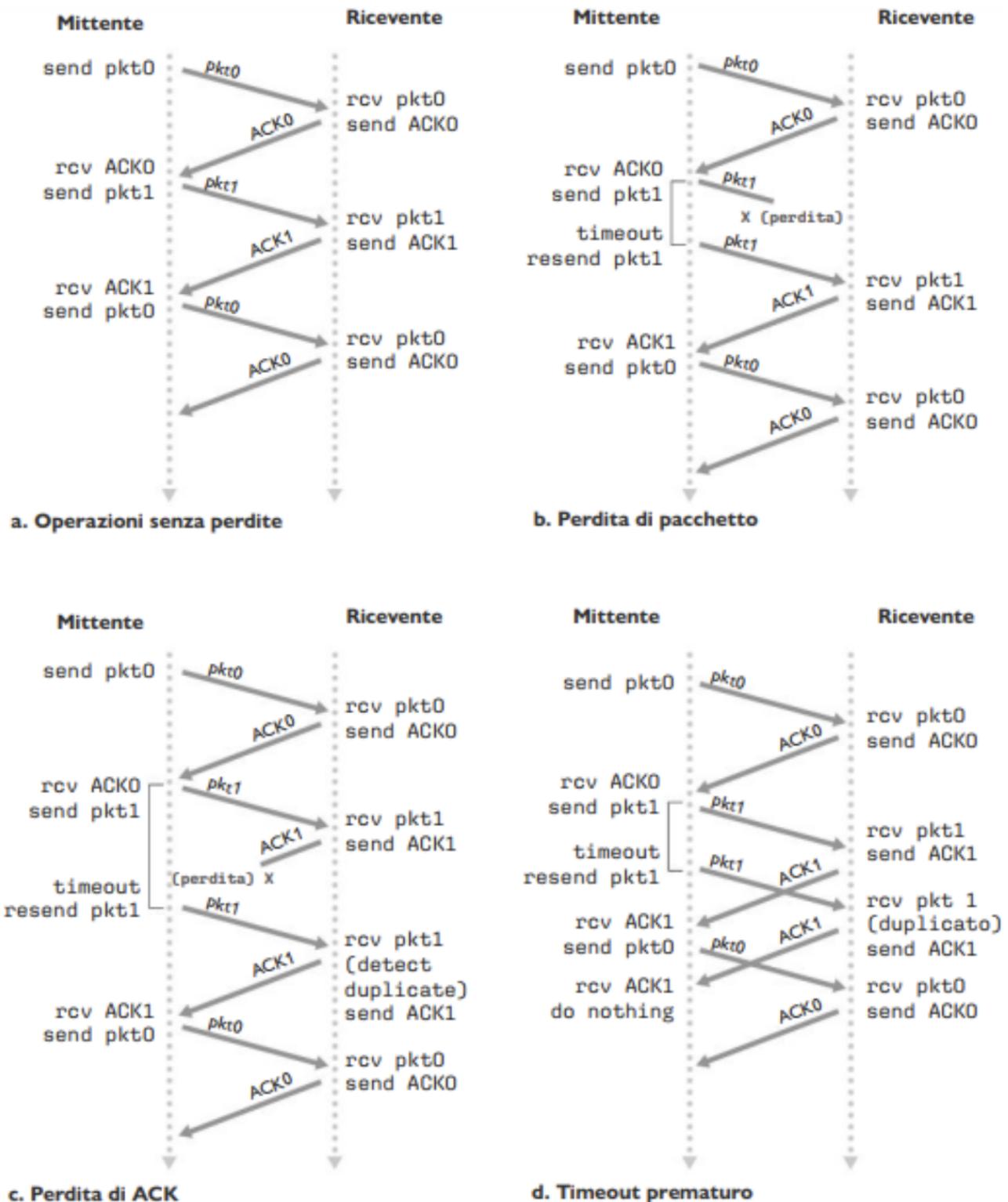


Figura 9.10: rdt3.0: esempi trasmissione

Caratteristiche Go-Back-N (*GBN*):

- Il mittente spedisce fino ad N pacchetti sulla pipeline senza attendere riscontri
- Il ricevente manda ACK cumulativi
Se riceve un pacchetto fuori sequenza lo scarta e non manda ACK
- Il mittente mantiene un timer per il pacchetto spedito da più tempo e non riscontrato
- Quando scade il time-out ritrasmette tutti i pacchetti non ancora riscontrati

Il protocollo *GBN* è un tipo di protocollo a **finestra scorrevole** (*sliding window*):

- Nell'intestazione dei pacchetti c'è un campo **SeqNum** a k bit, i numeri di sequenza vengono assegnati a rotazione, non ci possono essere sulla rete due pacchetti non riscontrati con lo stesso *SeqNum*
- Sia $0, \dots, 2^k - 1$ l'insieme dei numeri di sequenza, indichiamo:
 - **base**: *SeqNum* del pacchetto più vecchio tra quelli non riscontrati
 - **nextSeqNum**: più piccolo *SeqNum* non ancora utilizzato
 - La finestra del mittente è grande N ($N < 2^k$)
 - La finestra del destinatario è grande 1 e non accetta pacchetti fuori ordine

Il flusso di pacchetti da spedire è diviso in quattro parti:

1. Pacchetti spediti e già riscontrati **[0, base-1]**
2. Pacchetti spediti ma non ancora riscontrati **[base, nextSeqNum-1]**
3. Pacchetti che possono essere spediti **[nextSeqNum, base+N-1]**
4. Pacchetti che non possono ancora essere spediti **[base+N, 2 k - 1]**

Il mittente può spedire solo i pacchetti dei gruppi 2 e 3, per questo necessita di una finestra scorrevole dei pacchetti ammissibili.

- Quando il mittente riceve un ACK x la finestra scorre fino ad $x+1$ ($base = x+1$).

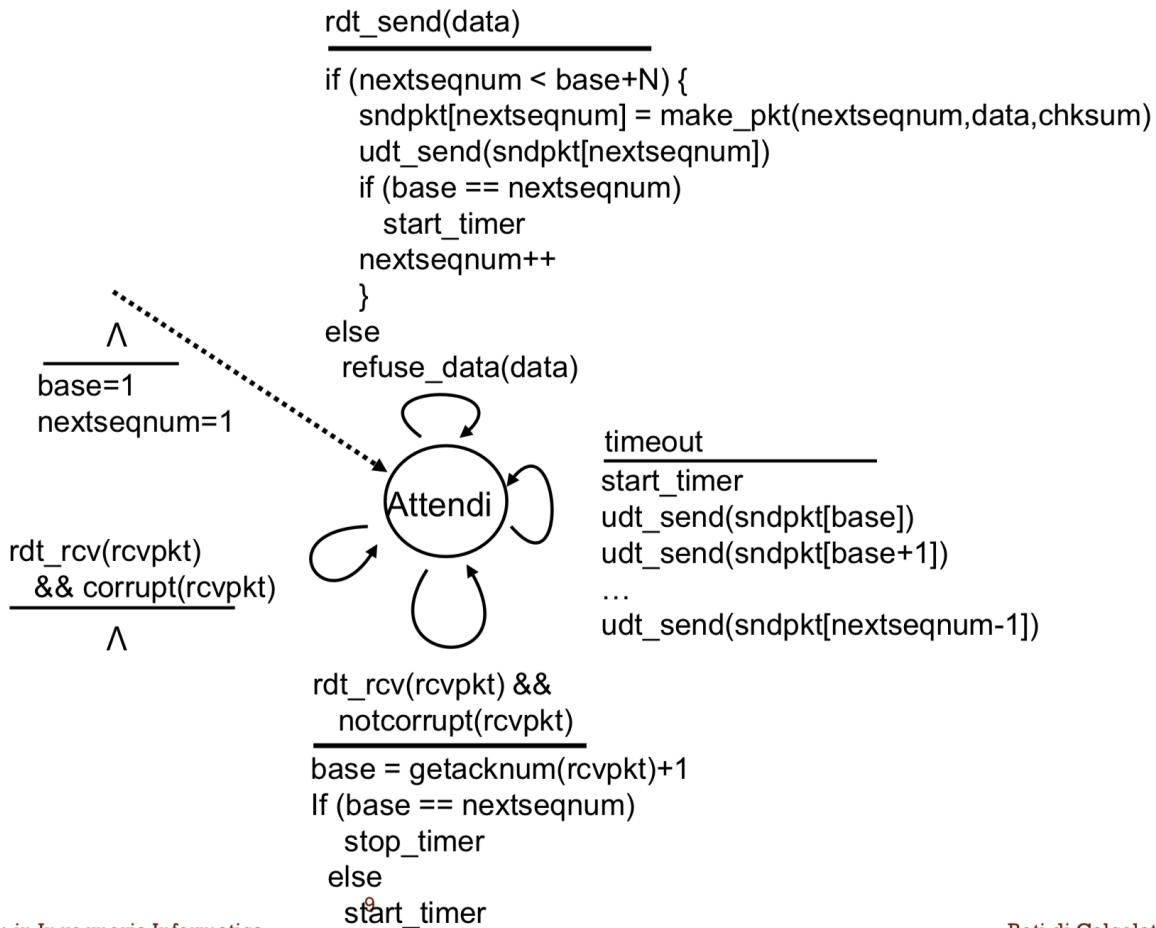
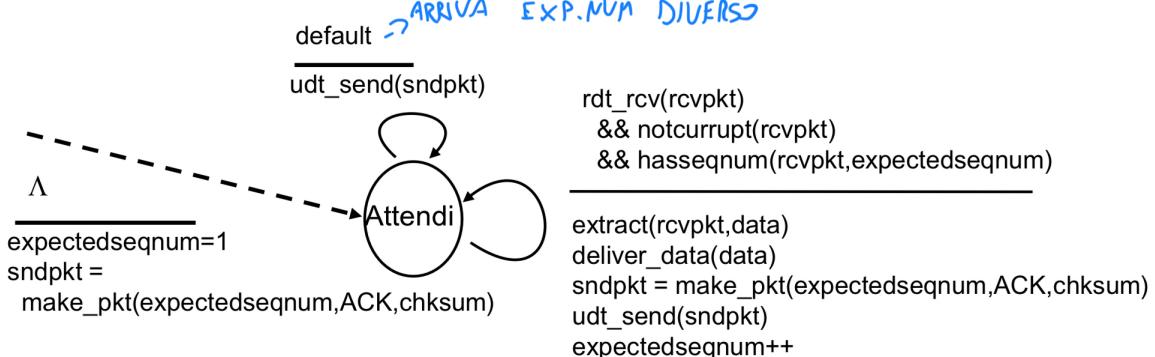


Figura 9.11: Mittente GNB



- Si aspetta di ricevere i pacchetti in ordine
 - Mantiene solo una variabile **expectedseqnum**
- Gli ACK contengono sempre i più alti NumSeq ricevuti in ordine
 - Può generare ACK duplicati
- Se arriva un pacchetto fuori sequenza
 - Il pacchetto viene scartato
 - Il ricevente non ha buffer dove conservare i pacchetti
 - Rispedisce l'ACK per il pacchetto ricevuto in ordine con il NumSeq più alto

Figura 9.12: Ricevente GNB

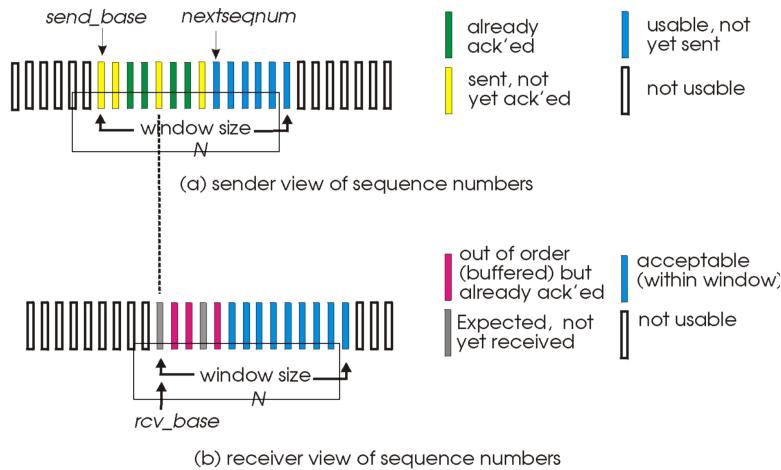


Figura 9.13: Le finestre di mittente e ricevente

Ripetizione Selettiva

GNB è più efficiente di *Stop-and-Wait* ma comunque non è molto performante, in quanto richiede la rirasmisione anche di pacchetti ricevuti correttamente.

I protocolli a ripetizione selettiva evitano tutte le ritrasmissioni non necessarie:

- Il ricevente manda ACK individuali per ogni pacchetto ricevuto correttamente, consente la ricezione di pacchetti fuori ordine
- I pacchetti devono essere mantenuti in un buffer per essere rimessi in ordine e consegnati correttamente al protocollo di livello superiore, *quando viene ricevuto correttamente il pacchetto X vengono passati al livello superiore tutti i dati nel buffer fino ad X*.
- Il mittente rispedisce soltanto i pacchetti per i quali non ha ricevuto ACK: *deve avere un timer per ogni pacchetto, allo scadere del timer il pacchetto viene ritrasmesso*.

Dal lato del mittente:

- *dati dal livello superiore*: se il prossimo seqNum è nella finestra, manda il pkt
- *timeout(n)*: rispedisce pkt n, restart timer
- *ACK(n)* in $[sendbase, sendbase+N]$:
 1. marca pkt n come ricevuto
 2. se n è il più piccolo pkt in attesa di ACK, scorri la finestra fino al prossimo *unACKed seqNum*

Dal lato del ricevente:

- *pkt n* in $[rcvbase, rcbase+N-1]$:
 1. manda ACK(n)
 2. Se out-of-order: buffer
 3. Se in-order: consegna all'applicazione tutti i pkt nel buffer in ordine e avanza la finestra fino al prossimo pkt non ancora ricevuto
- *pkt n* in $[rcvbase-N, rcbase-1]$: ACK(N)
- *altrimenti*: ignora

1. Dati ricevuti dall'alto. Quando si ricevono dati dall'alto, il mittente SR controlla il successivo numero di sequenza disponibile per il pacchetto. Se è all'interno della finestra del mittente, i dati vengono impacchettati e inviati; altrimenti sono salvati nel buffer o restituiti al livello superiore per una successiva ritrasmissione, come in GBN.
2. Timeout. Vengono usati ancora i contatori per cautelarsi contro la perdita di pacchetti. Ora però ogni pacchetto deve avere un proprio timer logico, dato che al timeout sarà ritrasmesso un solo pacchetto. Si può utilizzare un solo contatore hardware per simulare le operazioni di più timer logici [Varghese 1997].
3. ACK ricevuta. Se si riceve un ACK, il mittente SR etichetta tale pacchetto come ricevuto, ammesso che sia nella finestra. Se il numero di sequenza del pacchetto è uguale a <code>send_base</code> , la base della finestra si muove verso il pacchetto che non ha ricevuto acknowledgement con il più piccolo numero di sequenza. Se la finestra si sposta e ci sono pacchetti non trasmessi con numero di sequenza che ora cade all'interno della finestra, questi vengono trasmessi.

Figura 9.14: Eventi e azioni di un mittente SR

1. Il pacchetto con numero di sequenza nell'intervallo $[rcv_base, rcv_base+N-1]$ viene ricevuto correttamente. In questo caso, il pacchetto ricevuto ricade all'interno della finestra del ricevente e al mittente viene restituito un pacchetto di ACK selettivo. Se il pacchetto non era già stato ricevuto viene inserito nel buffer. Se presenta un numero di sequenza uguale alla base della finestra di ricezione (rcv_base nella Figura 3.22), allora questo pacchetto e tutti i pacchetti nel buffer aventi numeri consecutivi (a partire da rcv_base) vengono consegnati al livello superiore. Per un esempio, consideriamo la Figura 3.26. Quando si riceve un pacchetto con numero di sequenza $rcv_base = 2$, è possibile consegnarlo al livello superiore insieme ai pacchetti 3, 4 e 5.
2. Viene ricevuto il pacchetto con numero di sequenza nell'intervallo $[rcv_base-N, rcv_base-1]$. In questo caso si deve generare un ACK, anche se si tratta di un pacchetto che il ricevente ha già riscontrato.
3. Altrimenti si ignora il pacchetto.

Figura 9.15: Eventi e azioni di un ricevente SR

Meccanismo	Uso e commenti
Checksum	Utilizzato per rilevare errori sui bit in un pacchetto trasmesso.
Timer	Serve a far scadere un pacchetto e ritrasmetterlo, forse perché il pacchetto (o il suo ACK) si è smarrito all'interno del canale. I timeout si possono verificare per via dei ritardi anziché degli smarrimenti (timeout prematuro), o quando il pacchetto è stato ricevuto dal destinatario, ma è andato perduto il relativo ACK dal destinatario al mittente. Per questi motivi il destinatario può ricevere copie duplicate di un pacchetto.
Numero di sequenza	Usato per numerare sequenzialmente i pacchetti di dati che fluiscono tra mittente e destinatario. Le discontinuità nei numeri di sequenza di pacchetti ricevuti consentono al destinatario di rilevare i pacchetti persi. I pacchetti con numero di sequenza ripetuto consentono al destinatario di rilevare pacchetti duplicati.
Acknowledgment (ACK)	Utilizzato dal destinatario per comunicare al mittente che un pacchetto o un insieme di pacchetti sono stati ricevuti correttamente. Gli acknowledgement trasporteranno generalmente i numeri di sequenza del pacchetto o dei pacchetti da confermare. A seconda del protocollo, i riscontri possono essere individuali o cumulativi.
Acknowledgment negativo (NAK)	Usato dal destinatario per comunicare al mittente che un pacchetto non è stato ricevuto correttamente. Gli acknowledgement negativi trasporteranno generalmente il numero di sequenza del pacchetto che non è stato ricevuto correttamente.
Finestra e pipelining	Il mittente può essere forzato a inviare soltanto pacchetti con numeri di sequenza che ricadono in un determinato intervallo. Consentendo a più pacchetti di essere trasmessi e non aver ancora ricevuto acknowledgement si può migliorare l'utilizzo del canale rispetto alla modalità operativa stop-and-wait. Vedremo tra breve che l'ampiezza della finestra può essere impostata sulla base della capacità del destinatario di ricevere e memorizzare messaggi in un buffer, su quella del livello di congestione della rete, o su entrambe.

Figura 9.16: Riepilogo dei meccanismi di trasferimento dati affidabile e loro utilizzo

Capitolo 10

Il protocollo TCP

Il protocollo TCP (*Transmission Control Protocol*) è un protocollo del livello di trasporto che fornisce un servizio affidabile ed orientato alla connessione, utilizza molte delle funzionalità implementate dei protocolli ARQ (*Stop-and-Wait* o *Pipeline*): checksum, numeri di sequenza e di ACK, ACK cumulativi, ritrasmissioni, timeout, controllo del flusso, controllo della congestione.

Principali caratteristiche di TCP

TCP è un protocollo:

- **Punto-punto:** un mittente, un destinatario
- **Affidabile, orientato al flusso (*stream oriented*):** può trasmettere flussi di bit senza limite di dimensione, i numeri di sequenza non indicano i pacchetti ma i byte inviati
- **Orientato alla connessione:** handshaking prima dello scambio di dati
- **Full Duplex:**
 - *Flusso bidirezionale nella stessa connessione*
 - *MSS (Maximum Segment Size):* massima dimensione del segmento concordata tra gli end-system
- **Flusso controllato:** Mittente non sovraccarica né il destinatario né la rete
- **Pipelined:** controllo di flusso e congestione definiscono la dimensione della finestra

Ciclo di vita di una connessione TCP

1. **Creazione della connessione:** Entrambi gli endpoint devono accettare di creare la connessione richiesta dal client ed allocano alcune strutture dati e concordano sul valore di alcune variabili
2. **Utilizzo della connessione:** Comunicazione bidirezionale, in ogni direzione viaggiano sia dati che ACK all'interno dello stesso segmento
3. **Chiusura della connessione:** richiesta avviata dal client o dal server, rilascio delle strutture dati allocate

STRUTTURA DEL SEGMENTO TCP

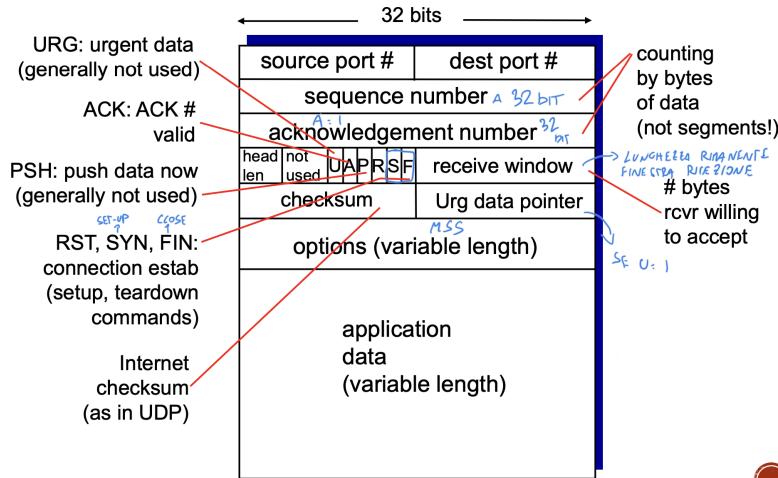


Figura 10.1: Struttura del segmento TCP

Creazione della connessione TCP

Dunque, la connessione viene creata su richiesta del client dopo aver concordato:

1. **creazione della connessione (identificatori di porte + indirizzi IP)**
2. **numeri di sequenza**: ogni endpoint sceglie a caso il valore iniziale di *SeqNum* per evitare che dati in ritardo di una vecchia connessione possano essere confusi con quelli della sua riincarnazione
3. **dimensione dei segmenti (MSS)**: dimensione massima del payload dei segmenti scambiati sulla connessione, ogni endpoint propone il proprio e si sceglie il minimo, *utilizza il campo OPTION dell'intestazione TCP*

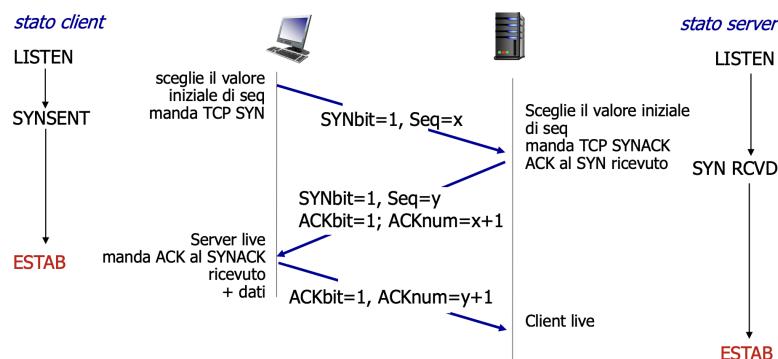


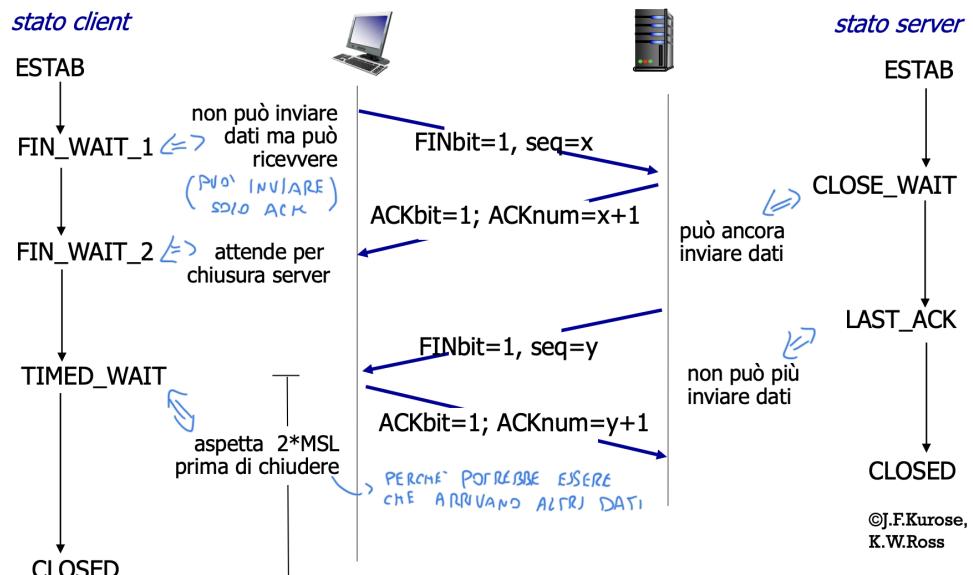
Figura 10.2: 3-way handshake

Chiusura della connessione TCP

- **Connessione chiusa separatamente da client e server**: ognuno può prendere l'iniziativa, possono chiudere in momenti diversi, connessione chiusa definitivamente solo dopo che entrambi hanno chiuso
- **Per chiudere la connessione un endpoint esegue:**

1. Manda un segmento TCP con bit $FIN = 1$ ed aspetta ACK
 2. Risponde al segmento FIN ricevuto con ACK
- I messaggi FIN e ACK possono essere combinati
 - Si deve prevedere la chiusura contemporanea dei due endpoint

4-WAY HANDSHAKE



▪ Maximum Segment Lifetime (default è 2 minuti)

- Tempo per cui un segmento TCP può esistere nell'inter-rete

Figura 10.3: 4-way handshake

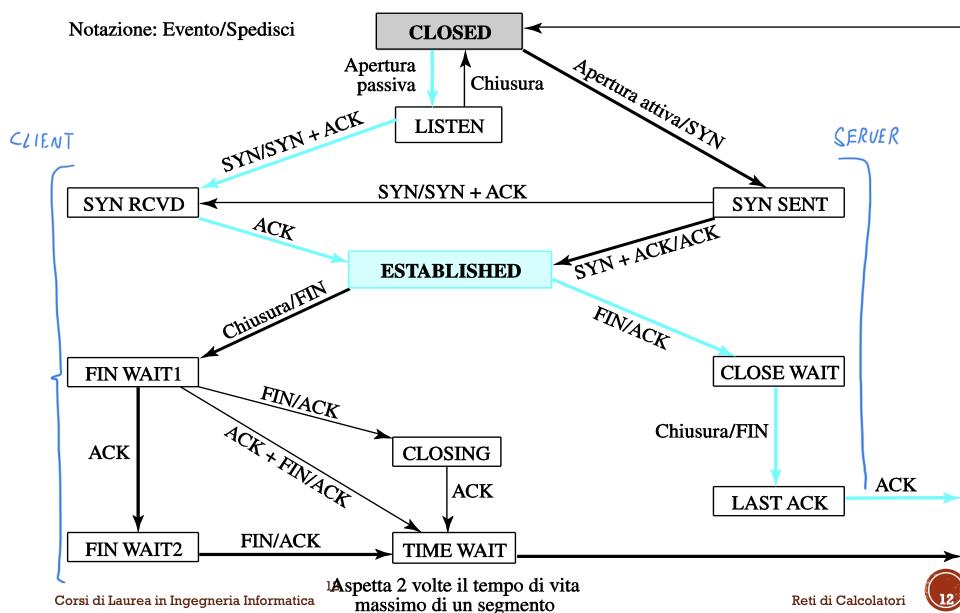


Figura 10.4: MSF di TCP

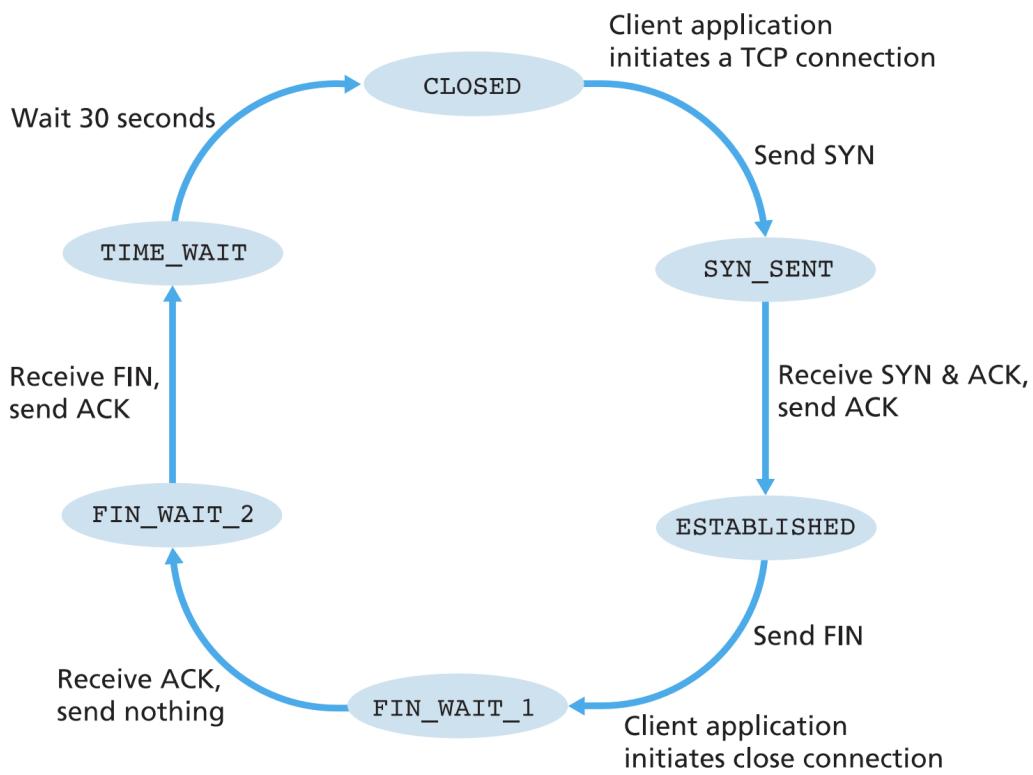


Figura 10.5: TCP LATO CLIENT

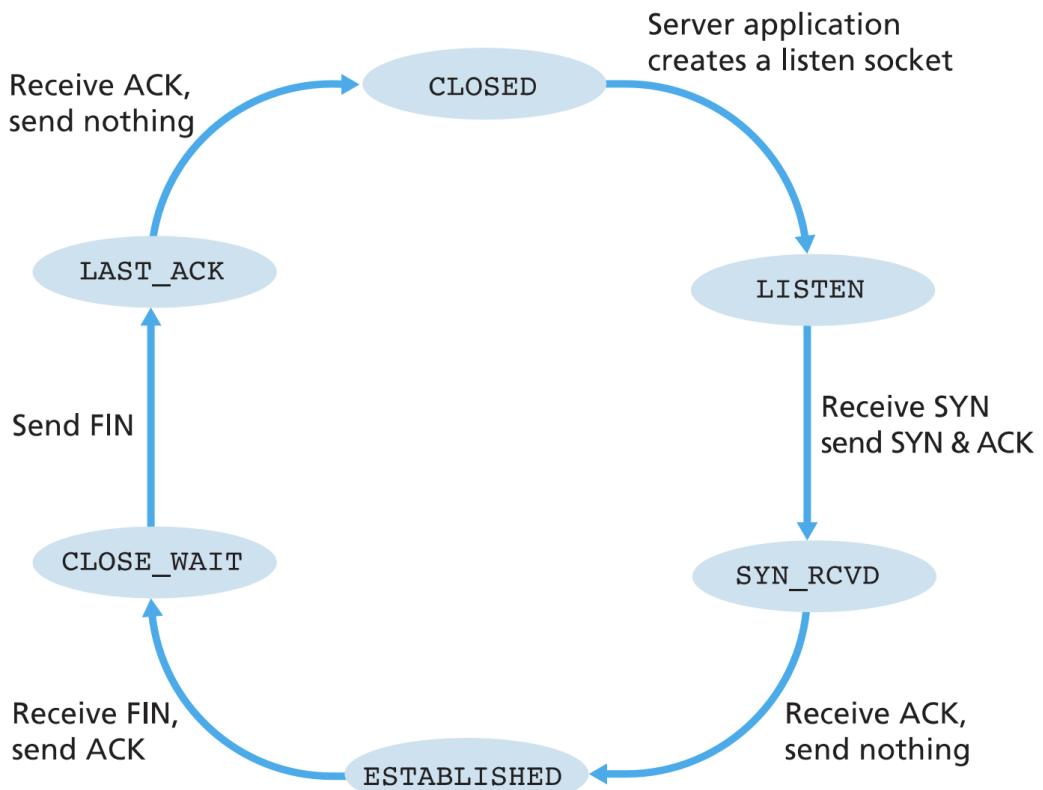


Figura 10.6: TCP LATO SERVER

10.0.1 Trasmissione affidabile in TCP

TCP deve realizzare un trasferimento affidabile (*flusso di bit spedito dal mittente ricostruito al destinatario senza errori, senza buchi o duplicazioni e riordinato correttamente*) dei dati sulla base del *servizio inaffidabile e best-effort di IP*.

Principali caratteristiche:

- **Inizializzazioni:**

- $NextSeqNum = InitialSeqNumber$
- $SendBase = InitialSeqNumber$

- **Ricezione di dati provenienti dal livello applicazione:**

- Crea segmento con dati incapsulati, SeqNum = NextSeqNum e ACKNum
- $NextSeqNum +=$ lunghezza(dati)
- se il timer non è attivo viene attivato

- **Ricezione di ACK(y):**

- Se $y > SendBase$ allora $SendBase = y$
- Se ci sono dati in attesa nel buffer crea nuovo segmento
- Se ci sono segmenti in attesa di ACK fa partire un timer

- **Scadenza del timeout:**

- Aggiorna IntervalTimeout $\leftarrow 2$
- Rispedisci solo il segmento che è da più tempo nel buffer e fa ripartire il timer

TCP non gestisce il destinatario, ma *da raccomandazioni*:

<i>Evento al destinatario</i>	<i>Azione del destinatario TCP</i>
Arrivo di un segmento in ordine con SeqNum atteso. Tutti i dati fino a SeQNum hanno ricevuto ACK	ACK ritardato. Aspetta fino a 500ms per altri segmenti. If non ce ne sono invia ACK
Arrivo di un segmento in ordine con SeqNum atteso. Un segmento ricevuto è in attesa di invio ACK	Manda immediatamente un ACK cumulativo per riscontrare entrambi i segmenti
Arrivo di un segmento fuori ordine. Viene rilevato un buco	Invia immediatamente un ACK duplicato con SeqNum del prossimo byte atteso
Arrivo di un segmento che colma parzialmente il buco <small>Corsi di Laurea in Ingegneria Informatica</small>	Manda immediatamente un ACK se il segmento comincia dall'estremità del buco

Reti di Calcolatori

Ritrasmissione rapida - Fast Retransmit

Molto spesso il timeout può rivelarsi troppo lungo, posso dedurre che un segmento si è perso dalla ricezione di ACK duplicati.

- Se il mittente riceve 3 ACK duplicati, allora rispedisce il segmento in attesa di ACK con NumSeq minimo senza attendere lo scadere del timeout

10.0.2 Timeout di TCP

- $\text{IntervalTimeout} = \text{RTT} + X$
- però RTT varia e scegliamo come X la varianza
- **TCP cerca di stimare RTT dinamicamente**

TCP calcola ***SampleRTT***: tempo trascorso dall'invio del segmento alla ricezione dell'ACK, escludendo dai calcoli il *SampleRTT* dei pacchetti ritrasmessi. In ogni istante c'è un solo segmento per il quale si sta calcolando *SampleRTT*

Calcola una ***Media Mobile Esponenziale Ponderata*** dei SampleRTT:

- $\text{EstimatedRTT} = (1 - a) * \text{EstimatedRTT} + a * \text{SampleRTT}$
- $a = 0.125$
- maggior peso alle misurazioni più recenti
- le variazioni di sample RTT vengono smussate da EstimatedRTT

Oltre alla media dei SampleRTT è importante avere anche una misura della loro variabilità, dunque si calcola la varianza:

- $\text{DevRTT} = (1 - b) * \text{DevRTT} + b * |\text{SampleRTT} - \text{EstimatedRTT}|$
- $b = 0.25$
- indica quanto è attendibile EstimatedRTT rispetto alla situazione reale della rete
- ***TimeoutInterval = EstimatedRTT + 4 * DevRTT***
- Il protocollo parte con TimeoutInterval = 1
- Quando viene ricevuto un ACK per un segmento per cui stiamo calcolando SampleRTT:
 1. viene aggiornato EstimatedRTT
 2. ricalcolato TimeoutInterval
- quando **scade un timeout**:
 - TimeoutInterval viene raddoppiato
 - Al successivo segmento ricevuto si riprende il calcolo normale

TCP è un ibrido tra GoBackN e Ritrasmissione Selettiva

- **Assomiglia a GBN perché:**
 - ACK cumulativi
 - Segmenti ricevuti fuori ordine non riscontrati singolarmente (ma vengono accettati)
 - Il mittente deve conservare solo SendBase (primo byte in attesa di ACK) e NextSeqNum (prossimo byte da spedire)
- **somiglia a RS perché:**
 - Accetta segmenti fuori ordine
 - Allo scadere del timeout (*o con 3 ACK duplicati*) viene ritrasmesso solo un segmento

Controllo del flusso in TCP

Se il mittente invia dati troppo velocemente può saturare il buffer di ricezione del socket, infatti **Il controllo del flusso evita la saturazione del buffer di ricezione del destinatario.**

- L'endsystem limita la velocità del mittente in modo da non saturare il buffer di ricezione della sua socket TCP.
- Il ricevitore comunica al mittente lo spazio disponibile nel suo buffer utilizzando il campo **receive window** dell'intestazione del segmento TCP
- il mittente non può spedire più di **rwnd** byte, dimensione della finestra che cambia dinamicamente
- Il ricevente calcola rwnd e lo manda al mittente
- $rwnd = RcvBuffer - (LastByteRcvd - LastByte Read)$
- Il mittente garantisce che in ogni istante
- $LastByteSent - LastByteAcked \leq rwnd$

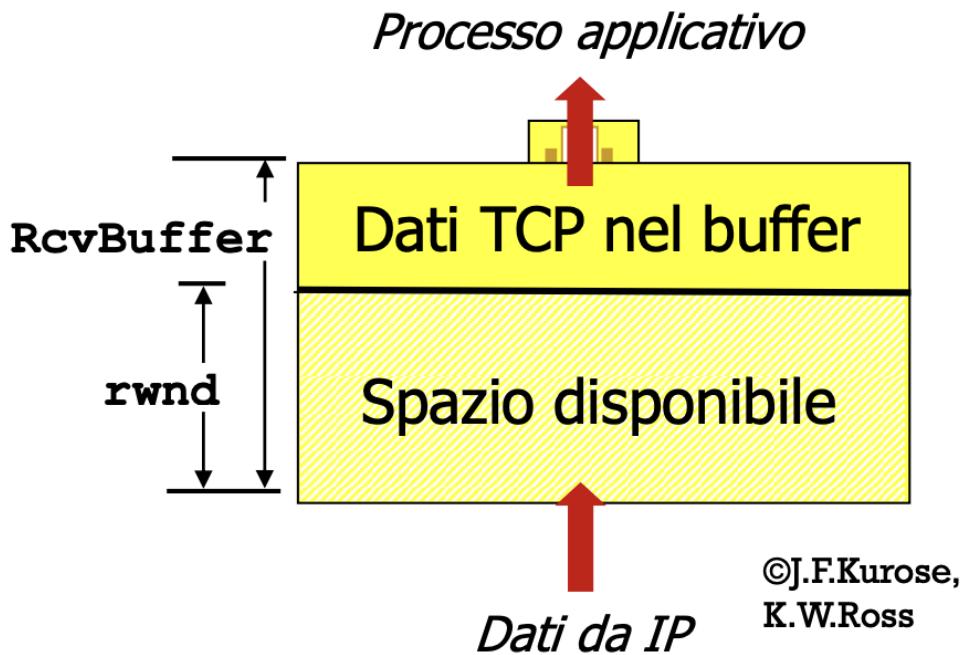


Figura 10.7: Buffer ricevente

Se **rwnd = 0**:

- il protocollo TCP prevede che il mittente continui a mandare segmenti con un solo byte di dati
- Servono solo a sollecitare gli ACK di risposta del destinatario
- Il mittente ha sempre il valore di rwnd aggiornato

10.0.3 Approfondimento: Gestione della connessione TCP

Il processo applicativo client informa il lato client di TCP di voler stabilire una connessione verso un processo nel server. Il TCP nel client quindi procede a stabilire una connessione TCP con il TCP nel server:

- **TCP lato client** invia uno speciale segmento al TCP lato server, detto **segmento SYN**. Questo segmento speciale *non contiene dati a livello applicativo*, ma uno dei bit dell'intestazione del segmento, il bit SYN, è posto a 1. Inoltre il client sceglie a caso un numero di sequenza iniziale (*client_isn*) e lo pone nel campo *numero di sequenza* del segmento SYN iniziale.

Quest'ultimo viene incapsulato in un datagramma IP ed inviato al server. Ci sono stati numerosi studi su come generare casualmente un appropriato *client_isn* al fine di evitare alcuni attacchi alla sicurezza;

- Quando il datagramma IP contenente il segmento *TCP SYN* arriva all'host server (*ammesso che arrivi*), il server **estrae il segmento dal datagramma, alloca i buffer e le variabili TCP alla connessione ed invia un segmento di connessione approvata al client TCP**. Anche questo segmento *non contiene dati a livello applicativo*, ma nella sua intestazione vi sono 3 importanti informazioni:

- bit SYN è posto a 1
- campo ACK assume il valore *client_isn + 1*
- il server sceglie il proprio numero di sequenza iniziale (*server_isn*) e lo pone nel campo *numero di sequenza*

Il segmento di connessione approvata si chiama **segmento SYNACK**;

- Alla ricezione del segmento *SYNACK*, anche il client alloca buffer e variabili alla connessione. L'host client invia quindi al server un altro segmento in risposta al segmento di connessione approvata dal server. Tale operazione viene svolta dal client ponendo il valore *server_isn+1* nel campo *ACK* dell'intestazione del segmento TCP. **Il bit SYN è posto a 0**, dato che la connessione è stata stabilita.

In questo terzo passo dell'handshake a tre vie il campo dati del segmento può contenere indormazioni che vanno dal client verso il server.

Una volta completati questi 3 passi, gli host client e server possono scambiarsi segmenti contenenti dati. In ciascuno di questi futuri segmenti, il bit SYN sarà posto a 0.

Alla fine della connessione TCP, ciascuno dei due processi che partecipano alla connessione deallocano tutte le "risorse" negli hosti (*buffer e variabili*).

10.1 Controllo della congestione

*In una rete di calcolatori si verifica della **congestione** quando il traffico immesso nella rete è vicino o superiore alla capacità della rete.*

Troppe sorgenti stanno inviando troppi dati e troppo velocemente rispetto alle capacità d'inoltro dei router.

Il controllo della congestione è diverso dal controllo del flusso, quest'ultimo è riferito alle capacità del ricevitore, invece *il controllo della congestione è riferito alla capacità della rete*.

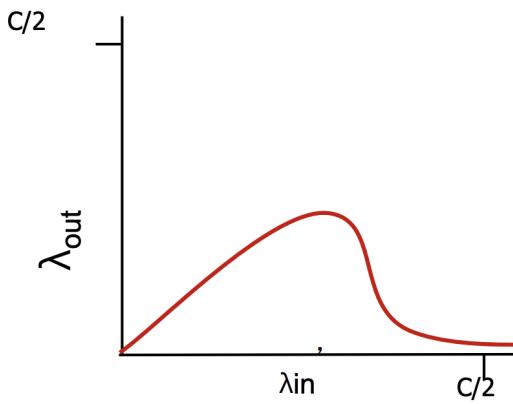


Figura 10.8: Rapporto tra il throughput in e out

Costi della congestione

Le prestazioni dipendono da come si effettuano le ritrasmissioni, però bisogna tener conto che:

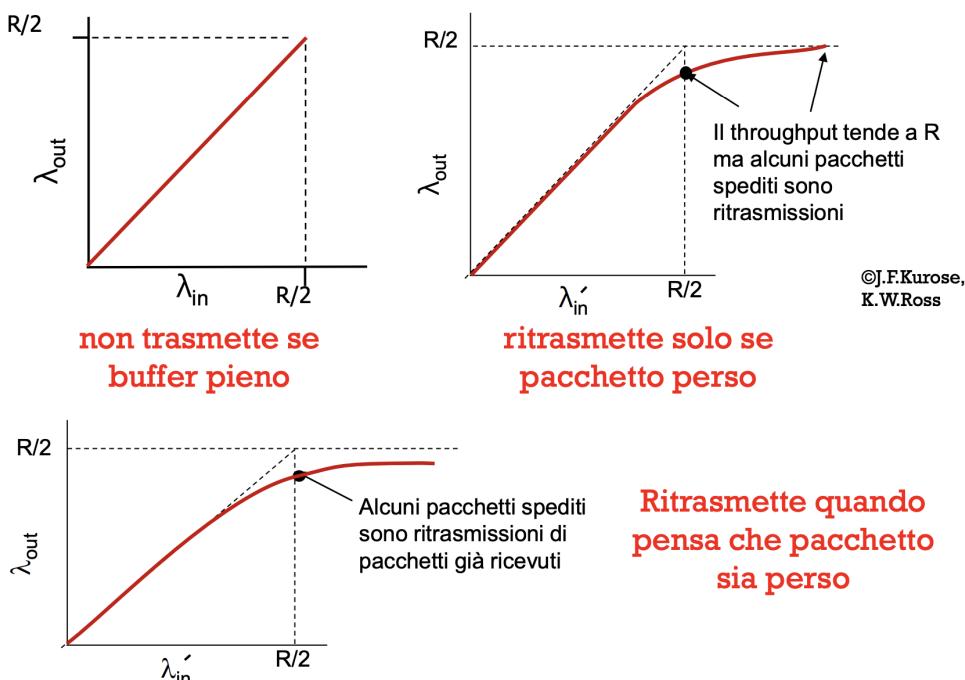
- **casi ideali:**

- *Conoscenza perfetta*: il mittente sa quando il buffer del router è pieno e non trasmette
- *Conoscenza perdita*: il mittente ritrasmette solo quando è certo che il pacchetto sia andato perso

- **Caso reale:**

- *Conoscenza imperfetta*: il mittente ritrasmette quando pensa che il pacchetto sia andato perso

Consideriamo tre alternative diverse:



Quando un pacchetto viene eliminato da un router, tutto il lavoro di instradamento dei router precedenti va sprecato.

10.1.1 Come controllare la congestione

Per controllare la congestione bisogna convincere le sorgenti a ridurre il traffico della rete trasmettendo più lentamente.

Il controllo della congestione può essere:

- **Proattivo** (*feedback - con retroazione*): si fa in modo che la congestione non si verifichi intervenendo sul traffico immesso e con l'eliminazione preventiva di pacchetti dal buffer quando il riempimento supera una certa soglia;
- **Reattivo** (*forward - senza retroazione*): si interviene quando la congestione si è verificata per riportare la situazione sotto controllo, si riconosce la congestione e si cerca di segnalarla alle sorgenti che dovranno ridurre il tasso di trasmissione

Invece, per il **riconoscimento della congestione**:

- *Segnalazione esplicita*: un router quando elimina un pacchetto dalla sua coda avvisa la sorgente dell'eliminazione inviando un *pacchetto ICMP* alla sorgente;
- *Segnalazione implicita*: nessuna notifica per l'eliminazione di pacchetti, l'host vede aumentare i ritardi e scadere i timeout e li interpreta come segnali di congestione

Esistono due *approcci più diffusi al controllo della congestione*:

1. *Controllo end-to-end*:

- Nessun feedback esplicito dalla rete, quindi i *router non sono coinvolti*
- Presenza della congestione dedotta dall'aumento dei ritardi e dalla perdita di pacchetti
- Aprroccio *utilizzato da TCP*

2. *Controllo assistito dalla rete*:

- Router forniscono feedback ai sistemi periferici
- Bit nell'header indicante presenza di congestione
- Nel *protocollo TCP* il router indica alla sorgente il tasso a cui dovrebbe trasmettere

TCP, quindi Internet, utilizza un controllo della congestione end-to-end di tipo *reattivo* e con *segnalazione implicita*:

- I router non forniscono feedback sulla congestione
- Congestione rilevata implicitamente osservando le prestazioni della rete
- Gli host intervengono solo dopo che hanno rilevato la congestione

Come regolare la velocità di trasmissione di una sorgente ?

- Modificando la dimensione della sua finestra di trasmissione
- in presenza di congestione riduciamo altrimenti aumentiamo

Come rilevare la congestione ?

- Scadenza di timeout e ricezione di ACK duplicati

10.1.2 Finestra di Congestione

In ogni istante il mittente TCP mantiene il valore **cwind**:

- Massimo numero di byte che si possono spedire senza provocare congestione
- poichè $\text{IntervalTimeout} \approx RTT$ allora velocità di trasmissione è circa $\frac{cwind}{RTT}$ bps

La dimensione della finestra di spedizione sarà al più $\min(cwind, rwnd)$

- **cwnd**: controllo della congestione, calcolato dal *mittente*
- **rwnd**: controllo del flusso, inviato dal *ricevente* insieme agli ACK

$$\text{LastByteSent} - \text{LastByteAcked} \leq \min(cwind, rwnd)$$

Rilevazione della congestione

Il mittente TCP rileva la presenza di congestione ogni volta che si verifica un "**evento di perdita**":

- Scadenza di timeout
- Ricezione di 3 ACK duplicati

TCP è *auto-temporizzato*:

1. Se gli ACK arrivano con alta frequenza la finestra aumenta velocemente
2. Se gli ACK arrivano con bassa frequenza la finestra aumenta lentamente
3. Se gli ACK non arrivano la finestra diminuisce velocemente

Algoritmo per il controllo della congestione in TCP

Basato sull'idea di *Incremento Additivo / Decremento Moltiplicativo (IADM)*, è composto da tre componenti principali:

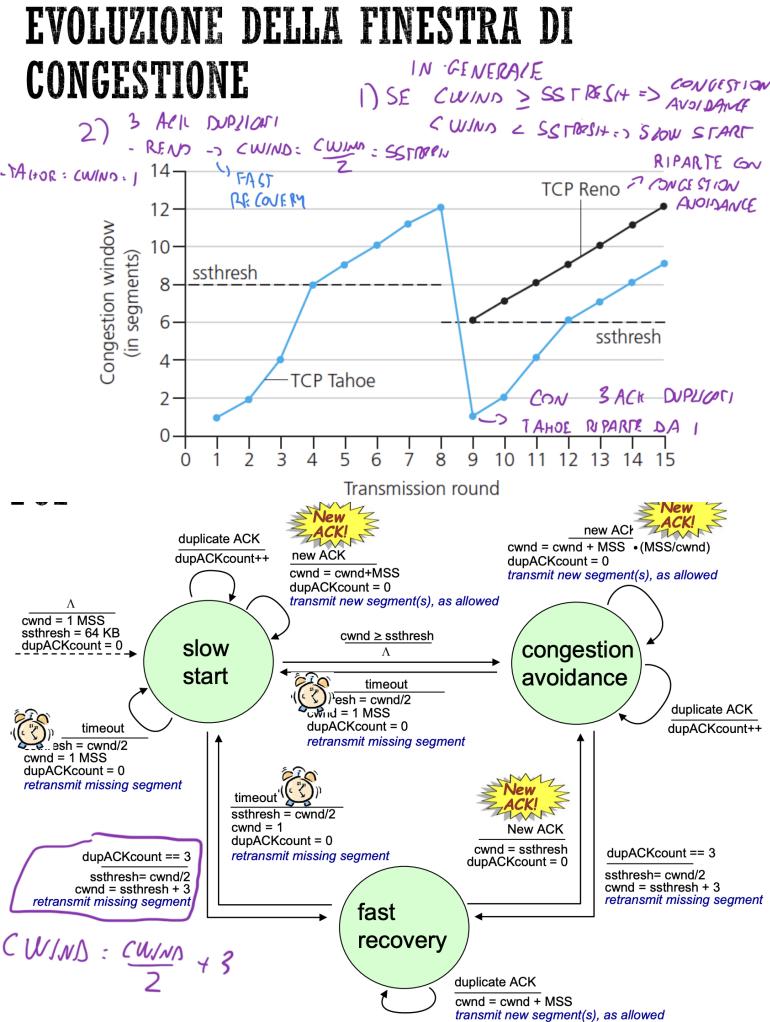
1. **Slow Start**: quando una connessione viene creata parte con una finestra piccola ma cresce velocemente
 - Valore iniziale: $cwnd = 1MSS$
 - Per ogni ACK ricevuto: $cwnd += MSS$
 - Se riceviamo ACK a tutti i segmenti spediti, dopo un tempo pari a RTT abbiamo **raddoppiato cwnd**
2. **Congestion Avoidance**: rispetto a *Slow Start* aumenta cwnd molto più lentamente
 - Ad ogni ACK ricevuto $cwnd += 1 \frac{MSS}{x}$
con x = numero di segmenti attualmente nella finestra
 - Se riceviamo ACK a tutti i segmenti spediti, dopo un tempo pari a RTT abbiamo **aumentato cwnd di 1 MSS**
3. **Fast Recovery - presente in TCP Reno**: un evento di perdita segnalato da 3 ACK duplicati è meno grave della scadenza di un timeout, possiamo dunque utilizzare una riduzione di cwnd meno aggressiva ripartendo da $cwnd = \frac{cwnd}{2} + 3$

Riassunto:

- Due implementazioni di TCP più diffuse:
 - **TCP Tahoe** (*obsoleto*)
 - **TCP Reno** (*con Fast Recovery*)
- In caso di scadenza del timeout:
 - $cwnd = 1MSS$
 - Si riparte con Slow Start
- In caso di 3 ACK duplicati:
 - *Tahoe*:
 - * $cwnd = 1MSS$
 - * Slow start
 - *Reno*:
 - * $cwnd dimezzato$ - Fast Recovery
 - * Si riparte con *Congestion Avoidance*

Per scegliere tra *Slow Start* e *Congestion Avoidace* si utilizza la variabile ***ssthresh*** (**Slow Start threshold**):

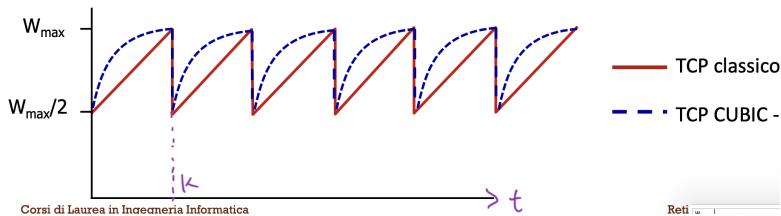
- Inizialmente $ssthresh = 2^{16}byte = 65536byte$
- Quando c'è un evento di perdita, $ssthresh$ viene posto a metà del valore di $cwnd$ prima della perdita
- si confronta $cwnd$ con $ssthresh$:
 1. Se $cwnd < ssthresh \rightarrow$ Slow Start
 2. Se $cwind \geq ssthresh \rightarrow$ Congestion Avoidance



Oggiorno molti server utilizzano **TCP CUBIC**, default in Linux.

L'idea alla base è che lo stato di congestione del collegamento *bottleneck* probabilmente non è cambiato molto, quindi conviene recuperare velocemente la velocità di trasmissione massima precedente al momento in cui si è rilevata la congestione.

In *Congestion Avoidance* la finestra viene incrementata in funzione del cubo della distanza tra l'istante K in cui la finestra TCP raggiungierà la velocità di trasmissione massima e l'istante t attuale.



Invece assumiamo che Congestion Avoidance cresca linearmente, in questo caso:

$$\text{Troughput medio TCP} = \frac{3}{4} \frac{W}{RTT} \frac{\text{byte}}{\text{sec}}$$

- ignorando le fasi di Slow Start e supponendo che il tempo sia diviso in fasi lunghe RTT
- in ogni fase il throughput è circa w/RTT
 - w = dimensione della finestra all'inizio della fase

- durante la fase la dimensione w aumenta al più di 1 MSS
- Sia W = dimensione finestra al verificarsi di un evento di perdita
 - Assumiamo che RTT e W siano costanti per la durata della connessione
 - la velocità di trasmissione varia linearmente tra $\frac{W}{2RTT}$ e $\frac{W}{RTT}$

Controllo di congestione basato sul ritardo

Obiettivo: "mantieni il canale end-to-end pieno, ma non più pieno".

Sono state proposte diverse varianti di Reno per cercare di ridurre ulteriormente la congestione mantenendo un buon throughput: *NewReno*, **Vegas**

L'idea di Vegas è:

- valutare il tasso di trasmissione atteso $\frac{cwnd}{RTT_{min}}$
- se il tasso effettivo è più basso di questo valore, c'è congestione
- diminuire la dimensione della fiestra in maniera lineare

Alternative al controllo della congestione di TCP

- Variante di TCP che permette ai router di segnalare esplicitamente una congestione a mittente e destinatario

Il router utilizza due bit (*ECN*) nell'intestazione IP per segnalare la congestione.

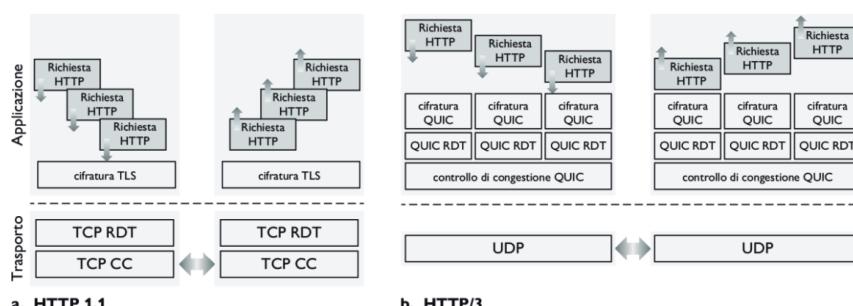
L'indicazione viene ricevuta dal destinatario che la comunica al mittente utilizzando un bit (*ECE*) dell'intestazione del messaggio di ACK

- **QUIC:** protocollo a livello applicativo basato su *UDP* per migliorare le prestazioni di *HTTP*

QUIC è la soluzione se un'applicazione necessita più servizi di quelli offerti da UDP ma non desidera tutte le funzionalità di TCP, quindi implementa i servizi a livello applicativo

PROPRIETÀ DI QUIC

- Orientato alla connessione e sicuro
 - HTTP classico richiede handshake TCP + handshake TLS
 - QUIC usa un unico handshake per entrambi
- Multiplexing di flussi in un'unica connessione
 - Ad ogni richiesta HTTP è assegnato un ID di flusso
 - Una stessa connessione può trasportare flussi diversi
- Trasferimento Dati Affidabile e Controllo della Congestione
 - Il trasferimento dati affidabile è implementato per ogni flusso separatamente
 - La perdita di un pacchetto non blocca/rallenta la ricezione di altri oggetti
 - Controllo della congestione simile a TCP New Reno



Capitolo 11

Livello di Rete

Caratteristiche principali del "Network Layer":

- trasporta segmenti dall'host mittente all'host ricevente
- il lato mittente incapsula segmenti in datagrammi
- il lato ricevente consegna segmenti al livello di trasporto
- i protocolli del network layer sono eseguiti in **ogni** host e router
- router esamina i campi dell'header che lo attraversano

Le due funzioni chiave del network layer:

1. **forwarding** (inoltro): sposta pacchetti dall'input del router all'appropriato output
2. **routing** (instradamento): determina la rotta presa dai pacchetti dalla sorgente alla destinazione (*routing algorithms*)

Il *routing* è l'operazione preliminare al *forwarding* in ogni router.

Quindi, con **inoltro** (*forwarding*) si intende l'azione *locale* con cui il router trasferisce i pacchetti da un'interfaccia di ingresso a quella di uscita. L'operazione avviene nell'ordine di pochi nanosecondi quindi è implementato in *hardware*.

Con **instradamento** (*routing*) si indica il processo *globale* di rete che determina i percorsi dei pacchetti nel loro viaggio *dalla sorgente alla destinazione*. L'instradamento avviene su scale temporali più grandi, nell'ordine di secondi, è quindi implementato in *software*.

Per inoltrare i pacchetti, i router estraggono da uno o più campi dell'intestazione i loro valori che utilizzano come indice nella **tavella di inoltro** (*forwarding table*), un elemento chiave di qualsiasi router. Il risultato indica a quale interfaccia di uscita il pacchetto debba essere diretto.

Il livello di rete può essere diviso in due parti interagenti:

- **Il piano dei dati** (*data plane*): gestisce le operazioni *indipendenti* dagli altri router.
Caratteristiche:
 - locale, tratta le funzionalità dei router
 - determina come i datagrammi che arrivano alla porta di input di un router sono inoltrati alla porta di output
 - funzione di forwarding
- **Il piano di controllo** (*control plane*): progetta i percorsi, non è implementato sul singolo router infatti deve comunicare con altri router. Caratteristiche:

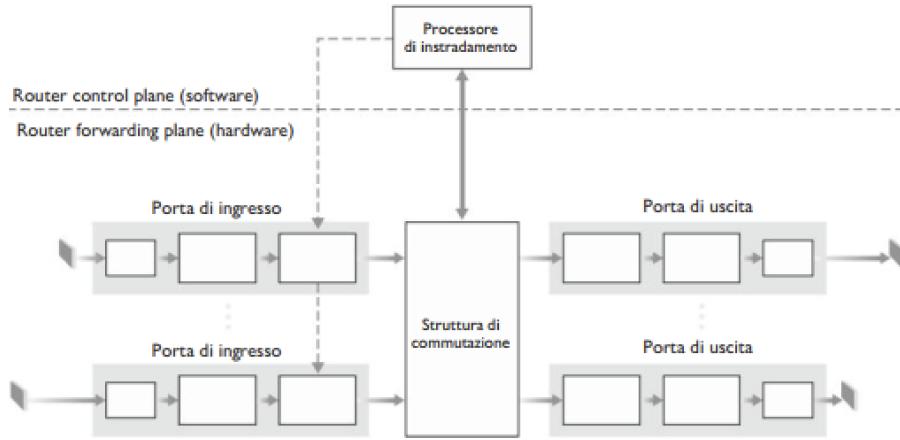


Figura 11.1: Architettura di un router

- logica network-wide
- determina come il datagramma è instradato attraverso i router lungo il path dall'host sorgente all'host destinazione
- Due approcci per il *control-plane*:
 1. **traditional routing algorithms** (*distribuito*): implementati nei router
 2. **software-defined networking** (*SDN* - centralizzato): implementato nei server remoti.

Modello di servizio del livello di rete

Le specifiche sul servizio che il livello di trasporto sono diverse e dipendono dal **modello di rete scelto**. Alcuni servizi che il livello di rete potrebbe offrire sono:

- **Consegna garantita**: assicura che il pacchetto giunga, prima o poi, alla propria destinazione
- **Consegna garantita con ritardo massimo**: non solo garantisce la consegna del pacchetto, ma anche il rispetto di un limite di ritardo specificato (*es. 40ms*)
- **Consegna ordinata**: garantisce che i pacchetti giungano alla destinazione nell'ordine in cui sono stati inviati
- **Banda minima garantita**: questo servizio emula il comportamento di un collegamento trasmissivo con bit rate specificato tra host di invio e di destinazione
- **Servizi di sicurezza**: il livello di rete dell'host sorgente può cifrare tutti i datagrammi inviati e il livello di rete dell'host di destinazione ha il compito di decifrarli

Il *livello di rete* di Internet offre un solo servizio noto come servizio *best-effort*. Con questo servizio non c'è garanzia che i pacchetti vengano ricevuti nell'ordine in cui sono stati inviati, così come non è garantita la loro eventuale consegna.

11.0.1 Cosa c'è all'interno di un router ?

Dalla visione ad alto livello di una generica architettura di router, si possono identificare quattro componenti:

Destination Address Range	Link interface	Nex Hop
11001000 00010111 00010*** *****	0	1.1.1.1
11001000 00010111 00011000 *****	1	2.2.2.2
11001000 00010111 00011*** *****	2	3.3.3.3
otherwise	3	4.4.4.4

Figura 11.2: Architettura di un router

- **Porte di ingresso** (*input port*): svolgono le funzioni a livello fisico di terminazione di un collegamento in ingresso al router. Svolgono anche la cruciale funzione di *ricerca* in modo che il pacchetto inoltrato nella struttura di commutazione del router esca sulla porta di uscita corretta
- **Struttura di commutazione** (*switching fabric*): connette fisicamente le porte di ingresso e quelle di uscita, internamente è contenuta una rete
- **Porte di uscita** (*output port*): memorizzano i pacchetti che provengono dalla struttura di commutazione e li trasmettono sul collegamento in uscita, operando le funzionalità necessarie del livello di collegamento e fisico
- **Processore di instradamento** (*routing processor*): esegue le funzioni del *piano di controllo*. Nel router tradizionale esegue i protocolli di instradamento, gestisce la tabella di inoltro per il router. Nei router *SDN*, il processore di instradamento è responsabile della commutazione con il controller remoto in modo da ricevere le occorrenze della tabella di inoltro ed installarle alle porte di ingresso. Effettua anche le operazioni di gestione di rete.

I router utilizzano i valori dei campi dell'header dei datagrammi per cercare nella tabella di inoltro su quale porta in output devono inoltrare quanto ricevuto sulle porte di input ("*match plus action*"). Studiamo due modi in cui viene effettuato il *forwarding* nel router:

1. **destination-based** forwarding: l'inoltro è basato solo sull'indirizzo IP di destinazione (*inoltro tradizionale*)
2. **generalized** forwarding: l'inoltro è basato sui valori di più campi dell'header

Il caso più semplice di forwarding è il *destination-base forwarding* in cui l'indirizzamento è bastato appunto sull'indirizzo IP di destinazione.

Un'altra tabella semplificata si basa sul **prefix** (*prefisso*) dell'indirizzo di destinazione del pacchetto con una riga della tabella: se c'è corrispondenza, il router inoltra il pacchetto al collegamento associato. Quando si verificano corrispondenze multiple, il router adotta la **regola di corrispondenza a prefisso più lungo** (*longest prefix matching*): quando si cerca una entry nella forwarding table, si usa il prefisso più lungo che corrisponde a quello dell'indirizzo di destinazione.

11.1 Il Protocollo Internet (IP) : IPv4

- **Indirizzi IP**: identificatore di 32 bit assegnato a ciascuna interfaccia dell'host/router. Ogni indirizzo IP si divide in:

- subnet part - primi bit
- host part - ultimi bit
- **CIDR** (*Classes InterDomain Routing*):
 - * la subnet part dell'indirizzo può essere di lunghezza arbitraria
 - * il formato dell'indirizzo è: $a.b.c.d/x$, dove x è il numero di bits nella subnet portion dell'indirizzo
 - * *Favorisce il funzionamento di longest prefix matching*

- **interfaccia**: connessione tra host/router e il collegamento fisico
 - i routers hanno tipicamente molteplici interfacce
 - gli host hanno tipicamente una o due interfacce (es. *wired Ethernet, wireless*)
- **Gli indirizzi IP sono acciociati ad ogni interfaccia**
- **subnet**: Due noti che appartengono alla stessa *subnet* se per comunicare tra di loro non utilizzano un router.
- Cos'è una *subnet*?
 - Le interfacce delle macchine con la stessa subnet part dell'indirizzo IP
 - Possono fisicamente raggiungersi tra di loro **senza l'intervento del router**
- *Come si identifica una subnet ?*

Per determinare le sottoreti si sganciano le interfacce da host e router in maniera tale da creare isole di reti isolate, delimitate dalle interfacce. Ognuna di queste reti isolate viene detta sottorete.

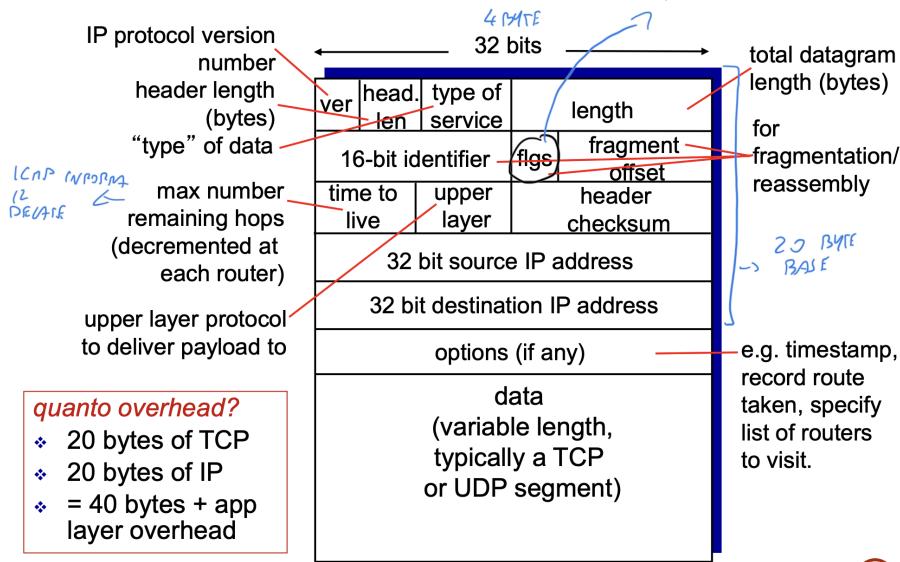
- Interfacce *Ethernet* sono connesse da *Ethernet switches*
- Interfacce *wireless WiFi* sono connesse da *WiFi base station*

Formato dei datagrammi IPv4

I principali campi dei datagrammi IPv4 sono i seguenti:

- **Numero di versione** (*Version Number*): 4 bit che specificano la versione del protocollo IP del datagramma
- **Lunghezza dell'intestazione** (*header length*): la maggior parte dei datagrammi IP non contiene opzioni, pertanto *il tipico datagramma IP ha un'intestazione di 20 bytes*
- **Tipo di servizio** (*TOS - Type of Service*): usati per distinguere diversi tipi di datagrammi (es. *basso ritardo, affidabilità*). Utile anche per distinguere datagrammi un tempo reale
- **Lunghezza del datagramma** (*datagram length*): rappresenta la lunghezza totale del datagramma IP, intestazione più dati. Considerato che questo campo è lungo *16 bit*, la **massima dimensione dei datagrammi IP** è *65.535 byte*, anche se raramente superano i 1500 in modo da non superare la lunghezza massima del campo dati dei frame Ethernet

FORMATO DEL DATAGRAMMA IP



- **Identificatore, flag, offset di frammentazione** (*identifier, flags, fragmentation offset*): questi tre campi servono per la cosiddetta frammentazione
- **Tempo di vita (TTL - Time To Live)**: per assicurare che i datagrammi non restino in circolazione per sempre nella rete. Questo campo viene decrementato di un'unità ogni volta che il datagramma è elaborato da un router, quando raggiunge 0, il datagramma viene scartato
- **Protocollo**: questo campo è usato quando il datagramma raggiunge la destinazione finale. Il valore del campo indica lo specifico protocollo a livello di *trasporto* al quale vanno passati i dati del datagramma
- **Checksum dell'intestazione (header checksum)**: consente ai router di rilevare gli errori sui bit nei datagrammi ricevuti. Il **checksum** deve essere ricalcolato e **aggiornato ad ogni router**.

Perchè TCP/IP effettua la verifica di errore sia a livello di trasporto che di rete ?

- il checksum TCP/UDP è calcolato sull'intero segmento, mentre IP solo sull'intestazione
- TCP/UDP e IP non appartengono necessariamente alla stessa pila di protocolli

- **Indirizzi IP sorgente e destinatario (source and destination IP address)**: quando un host crea un datagramma, inserisce il proprio indirizzo IP nel campo indirizzo IP sorgente e quello di destinazione nel campo indirizzo IP destinazione.

Spesso l'host sorgente determina l'indirizzo di destinazione attraverso una ricerca DNS

- **Opzioni (options)**: campi che consentono di estendere l'intestazione IP
- **Dati (payload)**: il campo dati contiene il segmento a livello di trasporto (*TCP o UDP*) da consegnare alla destinazione. Tuttavia può trasportare anche altri tipi di dati, come i messaggi *ICMP*.

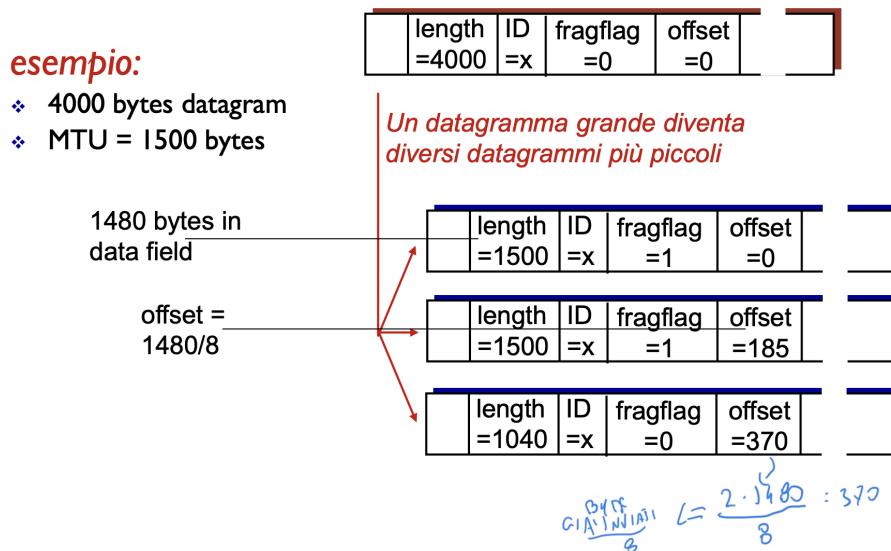
I datagrammi IP hanno 20 bytes di intestazione (*header*), escludendo le opzioni. I datagrammi non frammentati che trasportano segmenti TCP hanno 40 byte complessivi di intestazione: 20 da IP e 20 da TCP, assieme al messaggio di livello di applicazione.

Frammentazione e Riassemblaggio IP

I network links hanno una **MUT** (*max transfer unit*), ovvero la massima dimensione del campo dati in un frame a livello di collegamento.

I datagrammi IP larghi sono quindi frammentati:

- un datagramma diventa diversi datagrammi
- sono "riassemblati" solo alla destinazione finale
- i bits dell'IP header sono usati per identificare e riordinare i frammenti



11.1.1 Come ottenere l'indirizzo di un host: DHCP

Un'organizzazione che ha ottenuto un blocco di indirizzi li può assegnare individualmente alle interfacce di host e router nella propria struttura. Per gli indirizzi delle interfacce dei router, l'amministratore di sistema configura manualmente gli indirizzi IP nel router.

Come fa invece un *host* ad ottenere un indirizzo IP?

- hard-coded da un amministratore di Sistema in un file
- **DHCP** (*Dynamic Host Configuration Protocol*): riceve l'indirizzo dinamicamente da un server. DHCP è un *protocollo applicativo*

DHCP consente ad un host di ottenere un indirizzo IP in modo automatico quando si connette alla rete (*protocollo plug-and-play*). In più:

- Permette il riuso degli indirizzi (*assegna indirizzi solo a macchine che sono connesse*)
- Può rinnovare il "lease" dell'indirizzo che usa
- Supporta utente "mobili" che vogliono connettersi alla rete

Per i nuovi host, il protocollo *DHCP* si articola in quattro punti:

1. **Individuazione del server DHCP:** il primo compito di un host appena collegato è l'identificazione del server DHCP con il quale interagire. Questa operazione è svolta utilizzando un messaggio broadcast **DHCP discover**, che un client invia in un pacchetto UDP attraverso la porta 67 (*67/UDP - DHCP verso i server*).

Il pacchetto UDP è incapsulato in un datagramma IP con l'indirizzo di destinazione broadcast di 255.255.255.255 e l'indirizzo IP sorgente 0.0.0.0, cioè "questo host" (stocazzo). Il datagramma arriverà a tutti i nodi collegati alla sottorete.

2. **Offerta del servizio DHCP:** un server DHCP, che riceve un messaggio di identificazione, risponde al client con un messaggio **DHCP offer** (*datagramma inviato tramite UDP alla porta 68 - DHCP verso client*), che viene inviato in broadcast a tutti i nodi della sottorete.

Dato che in una sottorete possono essere presenti diversi server DHCP, il client dovrebbe poter scegliere tra le diverse "offerte" disponibili. Ciascun messaggio di offerta server contiene:

- l'ID di transazione del messaggio di identificazione ricevuto
- l'indirizzo IP proposto al client
- la maschera di sottorete
- la durata della connessione (*lease time*) dell'indirizzo IP (*ore o giorni*)

3. **Richiesta DHCP:** il client appena collegato sceglierà tra le offerte dei server e risponderà con un messaggio **DHCP request**, che riporta i parametri di configurazione

4. **Conferma DHCP:** il server risponde al messaggio di richiesta DHCP con un messaggio **DHCP ACK**, che conferma i parametri richiesti

Funzionalità di DHCP:

- conferire agli host indirizzi IP automaticamente
- fornire la *network mask* (così da sapere chi poter contattare tramite collegamento)
- fornisce l'indirizzo del router per uscire dalla sottorete (*gateway*)
- fornisce l'indirizzo del suo DNS server locale

Come ottenere un blocco di indirizzi IP

Per ottenere un blocco di indirizzi IP da usare in una sottorete, un amministratore di rete deve innanzitutto contattare il proprio ISP, che potrebbe fornire gli indirizzi attingendo da un blocco più grande che gli stato allocato.

Un provider, per esempio, al quale sia stato allocato un blocco di indirizzi indentificato dalla subnet part, potrebbe dividerlo in altri 8 blocchi contigui e fornirli.

L'**ICANN** è un'autorità globale che ha la *responsabilità ultima* di gestire lo spazio di indirizzi IP e di allocare i blocchi di indirizzi.

Il ruolo di **ICANN**, un'organizzazione senza scopo di lucro, non è solo quello di allocare indirizzi IP ma anche di gestire i root DNS server, ha anche il compito di assegnare e risolvere le dispute sui nomi di dominio.

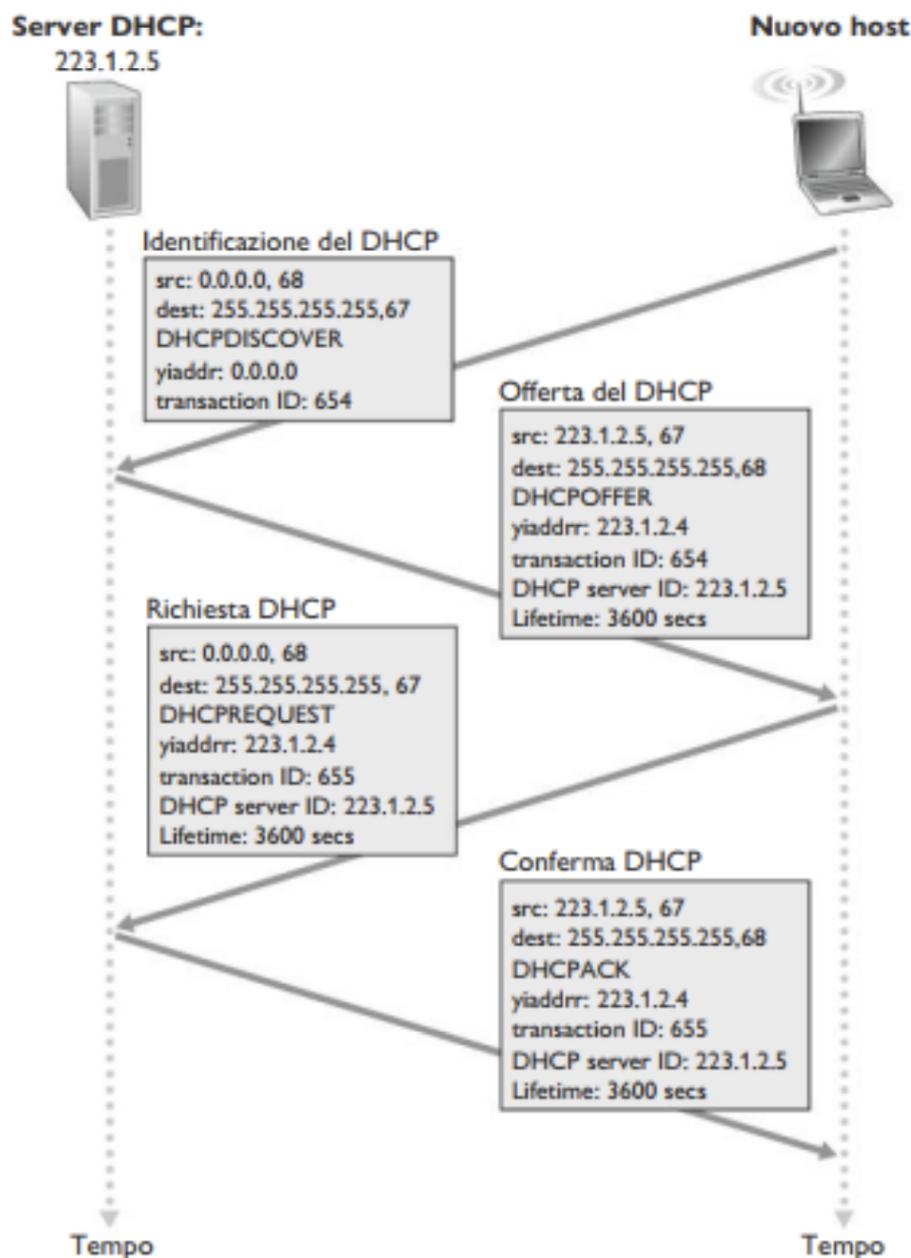


Figura 11.3: Interazione client-server DHCP

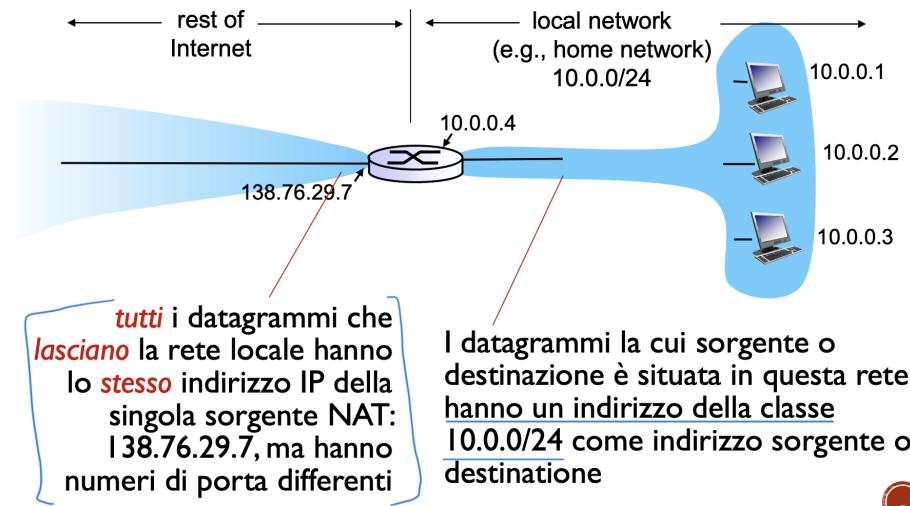


Figura 11.4: NAT: indirizzi IP che hanno senso soltanto all'interno della subnet

11.2 NAT, IPv4 & ICMP

- **NAT (Network Address Translation)**: la tecnica di *aggregazione di indirizzi* che hanno lo stesso *prefisso*.

Motivazioni:

1. Il mondo esterno vede che la rete locale usa un solo indirizzo IP
2. non c'è bisogno di un range di indirizzi dall'ISP, ma solo un indirizzo IP per tutti i devices
3. è possibile cambiare gli indirizzi dei devices nella rete locale senza doverlo notificare al mondo esterno
4. è possibile cambiare l'ISP senza dover cambiare gli indirizzi dei devices nella rete locale
5. i devices nella rete locale non sono visibili ed esplicitamente indirizzabili dal mondo esterno (*security plus*)

- **Implementazione**: una NAT router deve:

1. per i **datagrammi uscenti**: *sostituire* (source IP address, port) di ogni datagramma uscente con (NAT IP address, new port)
2. **ricordare** (*in una NAT translation table*) ogni coppia di traduzioni
3. per i **datagrammi in entrata**: *sostituire* il contrario

- **I problemi di NAT**:

1. i routers dovrebbero solo processare fino al livello 3 (*network*), dovrebbe quindi solo fermarsi a trasportare datagrammi. Invece riscrive il contenuto dei datagrammi cambiando indirizzo IP in base alla porta di destinazione
2. la limitazione degli indirizzi IP (60.000 in una sola LAN) dovrebbe essere risolta da IPv6 invece che da NAT
3. viola l'argomento **end-to-end**, la possibilità che ci sia la NAT deve essere tenuta in conto dai progettisti di app come in P2P applications

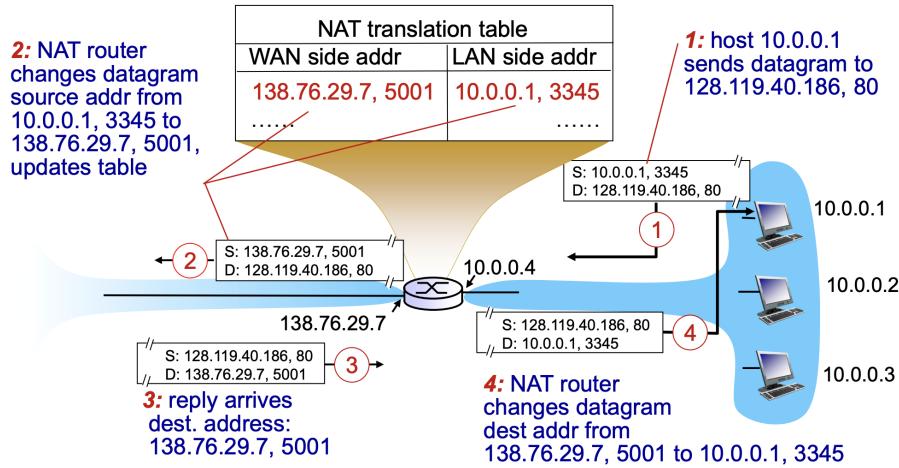


Figura 11.5: NAT translation table

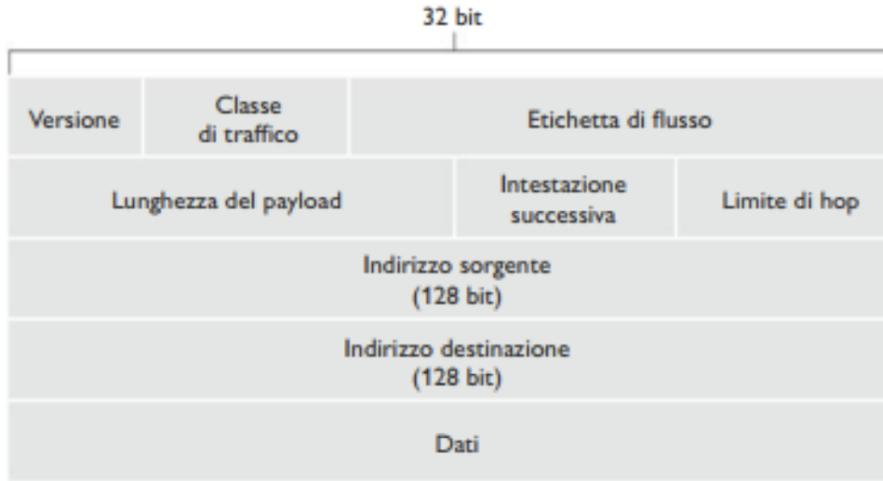


Figura 11.6: Formato dei datagrammi IPv6

4. **NAT traversal**, necessita di altri stratagemmi per poter permettere ad un client di connettersi ad un server che si trova dietro NAT

11.2.1 IPv6

Una prima *motivazione* che spiega alla realizzazione di un successore di IPv4 era legata alla considerazione che lo spazio di indirizzamento IP a 32 bit stava cominciando ad esaurirsi.

Allora i progettisti colsero l'opportunità per apportare migliorie e modifiche rispetto alla precedente versione. Motivazioni addizionali:

- Formato dell'header aiuta a velocizzare le operazioni di processing e forwarding
- L'header cambia per facilitare QoS

Formato dei datagrammi IPv6

Cambiamenti più significativi:

- **Indirizzo esteso**: IPv6 aumenta la dimensione dell'indirizzo IP da 32 a 128 bit.

Oltre agli indirizzi unicast e multicast, IPv6 supporta anche indirizzi anycast, che consentono di consegnare un datagramma ad un host qualsiasi più vicino.

- **Intestazione ottimizzata di 40 byte:** alcuni campi di IPv4 sono stati eliminati o resi opzionali. La risultante intestazione a 40 byte e a lunghezza fissa consente una più rapida elaborazione dei datagrammi IP
- **Etichettatura dei flussi:** IPv6 presenta una definizione di *flusso* (flow) che consente "l'etichettatura di pacchetti che appartengono a flussi particolari per i quali il mittente chiede una gestione speciale", come la trasmissione audio e video.

Campi del datagramma:

- **Versione:** campo a 4 bit che identifica il numero di versione IP
- **Classe di traffico:** campo a 8 bit, simile al campo TOS di IPv4, utilizzato per attribuire priorità a determinati datagrammi
- **Etichetta di flusso:** campo a 20 bit utilizzato per identificare un flusso di datagrammi
- **Lunghezza del payload:** valore a 16 bit trattato come un intero senza segno ed indica il numero di byte nel datagramma IPv6 che seguono l'intestazione fissa di 40 byte
- **Intestazione successiva (next header):** campo che identifica il protocollo a cui verranno consegnati i contenuti (*campo dati*) come TCP/UDP, **stessi valori del campo protocollo dell'intestazione IPv4**
- **Limite di hop:** come TTL
- **Indirizzi sorgente e destinazione:** IPv6 a 128 bit
- **Dati:** payload che viene passato al protocollo specificato nel campo di *intestazione successiva* (next hdr) quando il datagramma IPv6 raggiunge la sua destinazione

IPv4 vs. IPv6:

- **Frammentazione/riassembaggio:** IPv6 non consente né frammentazione né riassemblaggio sui router intermedi, queste operazioni possono essere effettuate soltanto da sorgente o destinazione.
Se un router riceve un datagramma IPv6 che risulta troppo grande, lo elimina e invia al mittente un messaggio d'errore ICMP "Packet Too Big"
- **Checksum dell'intestazione:** dal momento che i protocolli Internet a livello di trasporto e di collegamento calcolano un loro checksum, i progettisti di IPv6 hanno ritenuto questa funzionalità ridondante
- **Opzioni:** il campo opzioni è stato rimosso anche se non è del tutto scomparso. Infatti è una delle possibili "intestazioni successive" (*next header*) a cui punta l'intestazione di IPv6.

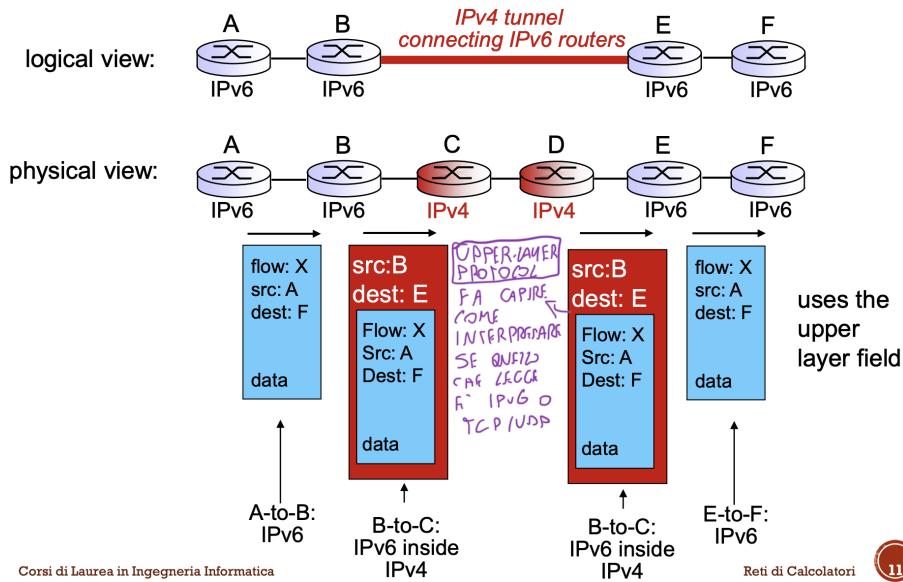


Figura 11.7: tunneling

Passaggio da IPv4 e IPv6

Il problema è che, mentre i nuovi sistemi IPv6 sono *retrocompatibili*, ossia sono in grado d'inviare, instradare e ricevere datagrammi IPv4, i sistemi IPv4 esistenti non sono in grado di gestire datagrammi IPv6.

L'apporoccio alla transizione da IPv4 a IPv6 più diffusamente adottato è noto come **tunneling**. L'idea alla base del tunneling è:

1. supponiamo che due nodi IPv6 vogliano utilizzare datagrammi IPv6, ma siano connessi da un insieme di router intermedi IPv4, che chiameremo **tunnel**;
2. Il Nodo B, al lato di invio del tunnes, prende l'*intero* datagramma IPv6 e lo pone nel campo dati di un datagramma IPv4;
3. Quest'ultimo viene indirizzato al nodo E, al lato di ricezione del tunnes ed inviato al primo nodo del tunnel (C).
4. I router IPv4 intermedi instradano il datagramma IPv4, come farebbero con qualsiasi altro datagramma, ignari che questo consegna un datagramma IPv6 completo;
5. Il nodo IPv6, sul lato ricezione del tunnel, riceverà quindi in datagramma IPv4, determinerà che questo ne conviene uno IPv6 osservando che il valore del campo **numero di protocollo** (*next header in IPv6*) nel pacchetto IPv4 è 41 corrispondente al payload IPv4, lo estrarrà e lo instraderà esattamente come se l'avessa ricevuto da un nodo IPv6 adiacente.

11.2.2 ICMP: Internet Control Message Protocol

Protocollo usato dagli *host* e *router* per comunicare informazioni a livello di rete

- error reporting: unreachable host, network, port, protocol
- echo request/reply

Type	Type	Code	Description
2	0	0	echo reply (ping)
3	0	dest. network unreachable	
3	1	dest host unreachable	
3	2	dest protocol unreachable	
3	3	dest port unreachable	
3	6	dest network unknown	
3	7	dest host unknown	
4	0	source quench (congestion control - not used)	
8	0	echo request (ping)	
9	0	route advertisement	
10	0	router discovery	
11	0	TTL expired	
12	0	bad IP header	

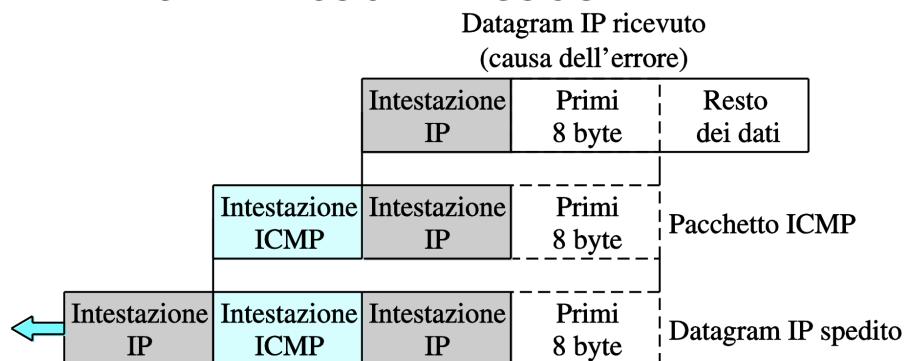


Figura 11.8: Formato del messaggio ICMP

ICMP è un protocollo a livello di rete che utilizza IP per trasportare datagrammi (*non richiede i protocolli di trasporto*).

ICMP message:

1. tipo (Type)
2. codice (Code)
3. intestazione del datagramma IP che ha generato l'errore
4. primi 8 bytes del contenuto (*campo data*) del datagramma IP che ha *causato l'errore* (*prime due righe*).

Successivamente, per essere inviato, ICMP viene incapsulato nel campo data del protocollo IP aggiungendo un'altra intestazione ed inviato alla sorgente. ICMP viene utilizzato molto per il debug. Due applicazioni utilizzate sono:

- **Ping:**

- Utilizza richieste/risposte echo ICMP
- Restituisce richieste in sequenza, numerate e calcola il tempo di andata e ritorno
- Perchè non fare richieste tipo HTTP ? Perchè con PING non ho bisogno di conoscere un'applicazione precisa specificando il numero di porta, ma basta conoscere l'indirizzo IP per sapere se la macchina è accesa.

- **Traceroute:**

Ci offre un modo per conoscere il percorso effettuato da un datagramma tra i router.

- Sfrutta ICMP, inviando dalla sorgente una serie di pacchetti a destinazione con queste caratteristiche:
 - * ogni pacchetto ha come porta di destinazione una impossibile
 - * invia sempre una sequenza di 3 pacchetti UDP
 - * inizia inviando pacchetti con TTL = 1, poi TTL = 2, ecc (*così ogni router del percorso invierà un ICMP message (type 11, code 0 - TTL expired)*)
 - * quando il messaggio ICMP arriva, la sorgente registra l'RTT
 - * *stopping criteria:* continuo così fino a quando non arrivo al destinatario (*facciamo in modo che anche il destinatario mandi ICMP rispondendo con un messaggio ICMP "port unreachable" (type 3, code 3) inserendo nel messaggio dalla sorgente un numero di porta impossibile*)

Il percorso restituito da Trace Route non è per forza il percorso seguito dal prossimo pacchetto perchè le tabelle di inoltro potrebbero cambiare.

Inoltre, il percorso potrebbe pure non essere un vero percorso. Per esempio durante il calcolo le tabelle di inoltro cambiano ed ottengo un percorso impossibile.

Capitolo 12

Livello di rete: il piano di controllo (control panel)

Il *control panel* gestisce la logica di rete globare che controlla non solo come i datagrammi vengono inoltrati tra i router lungo i percorsi *end-to-end* dall'host sorgente all'host destinatario, ma anche come le componenti ed i servizi del livello di rete vengono configurati e gestiti.

12.1 Routing Algorithms

Funzionalità del livello di rete:

- **inoltro** (*data plane*): trasferisci pacchetti da un'interfaccia ad un'altra di un router
- **routing** (*control plane*): determina la rotta intrapresa dai pacchetti dalla sorgente alla destinazione.

Esistono due approcci per strutturare il **control plane**:

1. Controllo **per-router** (tradizionale): *ogni router* ha componenti che eseguono un algoritmo di routing e interagiscono con gli altri router per computare le tabelle di inoltro.
2. Controllo **logicamente centralizzato** (software defined networking - *SDA*): un *controller distinto remoto* interagisce con gli agenti di controllo locale (*CAs*) presenti nei router per computare le tabelle di inoltro

L'**obiettivo** dei *protocolli di routing* è quello di determinare cammini (routes) buoni dall'host mittente all'host ricevente attraverso una rete di routers.

Per formulare i problemi di instradamento si utilizza un **grafo**.

Classificazione degli algoritmi di routing

Gli algoritmi possono essere classificati in base alla conoscenza che i router hanno del grafo: *globali* o *decentralizzati*, oppure in base allo stato del grafo: *statico* (le rotte cambiano lentamente nel tempo) e *dinamico* (le rotte cambiano più frequentemente).

- Algoritmi **link state** (LS):
 1. algoritmi *globali*: tutti i router hanno informazioni complete sulla topologia ed il costo dei link (conoscono il *grafo*).
 2. vengono scambiati messaggi tra tutti i router prima di computare (forwarding table)

3. L'*algoritmo* di **Dijkstra**: computa il costo minimo da una sorgente a tutti gli altri nodi ottenendo la *forwarding table* per quel nodo:

- Per ottenere la topologia della rete ed il costo dei link si invia un "link state broadcast"
- i nodi hanno le stesse info
- algoritmo *iterativo*: dopo k iterazione si ottiene il path migliore per i router raggiungibili in k passaggi

Complessità dell'algoritmo:

- ad ogni iterazione si controllano tutti i nodi per cui il cammino minimo da un router non è ancora noto
- $n(n + 1)/2$ controlli: $O(n^2)$, con implementazioni più efficienti: $O(n \log n)$
- **Sono possibili oscillazioni**: per esempio quando il costo dei link dipende dal traffico trasportato

- Algoritmi **distance vector** (DS):

1. algoritmi *decentralizzati*: i router conoscono solo i vicini a cui sono fisicamente connessi ed il costo di tali link
2. un processo iterativo di computazione, tramite lo scambio di messaggi con i vicini
3. L'*equazione* di **Bellman-Ford**:

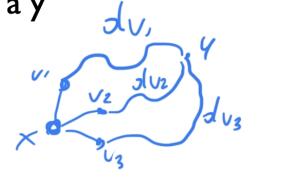
sia

$d_x(y) :=$ il costo del cammino minimo da x a y

allora

$$d_x(y) = \min \{ c(x, v) + d_v(y) \}$$

↓
 costo dal vicino v alla destinazione y
 ↓
 costo da x al vicino v
 ↓
 min tra tutti i vicini v di x



Il nodo che ottiene il minimo è il *next hop* nel cammino minimo che verrà usato nella forwarding table.

- Sia $D_x(y)$ la stima del minor costo da x a y : x conserva il distance vector $D_x = [D_x(y) : y \in N]$
- il nodo x conosce il costo del link a ciascun vicino v : $c(x, v)$
- dopo un tempo prestabilito, ogni nodo invia il proprio distance vector ai vicini
- il nodo x quando riceve un nuovo distance vector dai vicini, aggiorna il proprio distance vector usando l'equazione di *Bellman-Ford*
- se il distance vector di x cambia, allora x invia il nuovo distance vector ai suoi vicini

Confronto tra gli algoritmi LS e DV:

1. *message complexity*:

- **LS**: con n nodi, E links, inviati $O(nE)$ msg. Però la dimensione di msg è piccola in quanto ogni nodo invia solo il suo stato

- **DV**: scambio solo tra i vicini, i msg sono pochi però la dimensione è grande. Msg contengono il vettore delle distanze verso tutti i nodi

2. *velocità di convergenza*:

- **LS**: $O(n^2)$ con $O(nE)$ msg, potrebbe avere oscillazioni
- **DV**: variabili, potrebbe creare routing loops oppure cont-to-infinity problem

3. *robustezza* (cosa succede se un ruoter non funziona bene):

- **LS**: nodo può pubblicizzare i costi di *link* scorretti ed ogni nodo computa soltanto la propria nuova tabella
- **DV**: il nodo può pubblicizzare costi di *cammini* scorretti ma soltanto ai suoi vicini (aggiorna il suo DV ed informa i vicini, ci impiegherà tempo per informare tutti i nodi del grafo). La tabella di ogni nodo è usata da altri nel frattempo che viene notificato l'errore e gli errori si propagano nella rete.

12.2 Come rendere il routing scalabile? Protocolli di instradamento: OSPF e BGP

Lo studio degli algoritmi di instradamento fatto fin'ora si è basato su delle idealizzazioni. Ciascun router era indistinguibile dagli altri nel senso che tutti eseguivano lo stesso algoritmo per calcolare l'intradamento attraverso la rete, tutti i router erano identici ("network flat"). Nella realtà ci sono almeno due problemi:

- **scalabilità**: al crescere del numero di router, il tempo richiesto per calcolare, memorizzare e comunicare le informazioni di instradamento diventa proibitivo;
- **autonomia amministrativa**: Internet è una rete di ISP che desiderano gestire autonomamente i propri router o nascondere all'esterno aspetti dell'organizzazione interna alla rete. L'ideale sarebbe che ciascuno fosse in grado di amministrare la propria rete nel modo desiderato, pur mantenendo la possibilità di connetterla alle reti esterne.

Questi problemi possono essere risolti organizzando i router in **sistemi autonomi** (*AS*, *Autonomous System*), generalmente composti da gruppi di router posti sotto lo stesso controllo amministrativo. A volte *i router ed i collegamenti di un ISP formano un unico AS*, gigante nel caso di ISP tier-1, *altre volte gli ISP partizionano la loro rete in più AS*.

Un *AS* è identificato da un **Numero di Sistema Autonomo (ASN)** univoco che gli viene assegnato da ICANN, esattamente come gli indirizzi IP.

I router di un AS eseguono lo stesso algoritmo di instradamento e gli uni hanno informazioni sugli altri. L'algoritmo di instradamento in esecuzione in un AS è detto **protocollo di instradamento interno al sistema autonomo (intra-AS routing protocol)**.

Dunque, la soluzione per rendere il routing scalabile è quella di *aggregare i routers i regioni note come "autonomous system (AS) (aka domains)"*.

- **intra-AS routing**:

1. routing tra hosts, routers nello stesso AS (network)
2. tutti i routers in un AS devono eseguire lo stesso protocollo intra-domain
3. routers in *differenti* AS possono eseguire *differenti* protocolli di routing intra-domain

4. *gateway* router: ai "confini" del proprio AS, ha collegamenti con routers di altri AS

- **inter-AS routing:**

1. routing tra AS

2. gateways si occupano del routing *inter-domain* (oltre che del routing intra-domain)

Q: perché i routing intra e inter-AS sono differenti?

1. **policy:** inter-AS amministratori vogliono il controllo su quale, quanto e come il traffico è inoltrato sulle proprie reti. Intra-AS singolo amministratore implica no privacy
2. **scalabilità:** routing gerarchico riduce la dimensione delle tabelle ed il traffico generato dagli aggiornamenti
3. **performance:** intra-AS si concentra sulle performance. Inter-AS policy potrebbero influenzare le performance

Protocolli in instradamento principali:

- **IGP** (*Interior Gateway Protocol*) è il protocollo di instradamento che opera all'interno della rete in un singolo ISP;
- **BGP** (*Border Gateway Protocol*) è il protocollo di instradamento che serve ad interconnettere tutte le reti in Internet. Infatti è visto come la colla che tiene insieme Internet.

12.2.1 Routing intra-AS

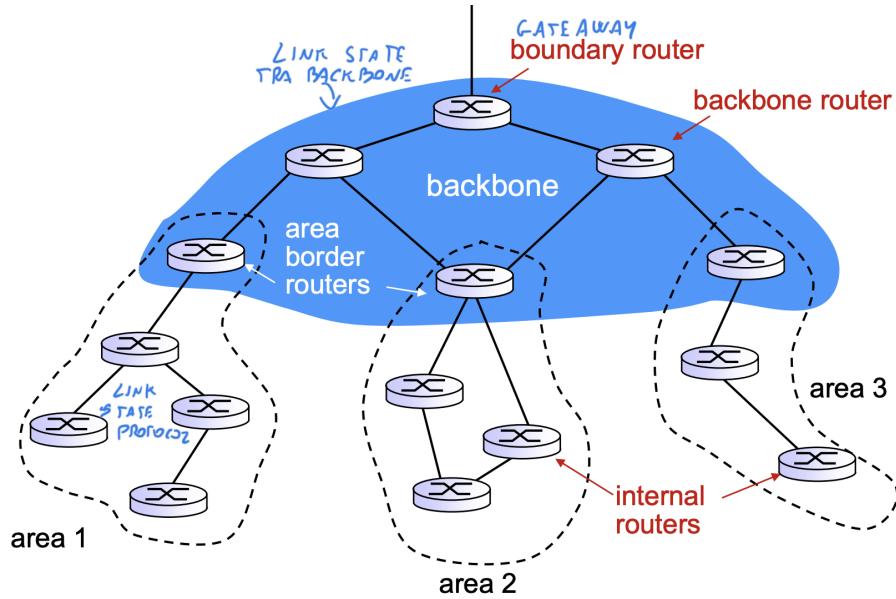
Noto anche come ***Interior Gateway Protocols (IGP)***, i più comuni sono:

- **RIP**: Routing Information Protocol
- **OSPF**: Open Shortest Path First
- **IGRP**: Interior Gateway Routing Protocol

OSPF è un protocollo "open": non proprietario.

- Usa *algoritmi* di routing ***link-state***:

1. disseminazione di pacchetti contenenti lo stato dei link
 2. ogni nodo si costruisce la propria mappa topologica
 3. le rotte sono computate attraverso l'algoritmo di Dijkstra
- il router pubblicizza l'OSPF *link state* con un **flooding** (*inondazione*) verso tutti gli altri nodi nell'**intero** AS. Praticamente, utilizza il *flooding* per inviare in broadcast le informazioni riguardo lo stato dei collegamenti e l'algoritmo di Dijkstra per la determinazione del percorso minimo.
 - In OSPF, ogni qualvolta si verifica un cambiamento nello stato di un collegamento, il router manda informazioni di instradamento via broadcast a *tutti* gli altri router nel sistema autonomo.



- Invia periodicamente lo stato dei collegamenti (almeno ogni 30 minuti) anche se questo non è cambiato.
 1. trasportati da messaggi OSPF encapsulati in datagrammi IP (*next header: 89*)
 2. ogni pacchetto inviato ha un **timestamp** e gli altri router inoltrano un pacchetto solo se ha un timestamp aggiornato
 3. **link state**: *costo* di ognuno dei collegamenti del router, i costi sono definiti dall'amministratore di rete
- **security**: tutti i messaggi di OSPF sono autenticati
- permette di salvare **più paths** con lo stesso costo
- permette per ogni link di definire più costi che possano essere usati per definire path diversi in relazione a differenti **ToS** (*Type of Service*)
- OSPF **gerarchico** in domini di grosse dimensioni, permettendo la scalabilità.

Hierarchical OSPF

- *gerarchia a due livelli*: area locale, backbone.

Link-state advertisements soltanto nell'area. Ogni nodo conosce dettagliatamente solo la *topologia* dell'area e conosce solo l'*area border router* che permette il routing verso altre aree

- *area border routers*: riassumono le distanze verso i router della propria area e le pubblicizzano verso gli altri area border routers
- *backbone routers*: eseguono OSPF solo sulla backbone
- *boundary routers*: si connettono ad altri AS

12.2.2 Routing inter-AS: BGP

Per determinare i percorsi per le coppie sorgente-destinazione che interessano più AS è necessario un *protocollo di instradamento inter-AS* che coordini più AS.

Il **border gateway protocol** (BGP) rappresenta l'attuale standard dei protocolli di instradamento tra AS di Internet.

BGP è forse il più importante protocollo di Internet, insieme ad IP, in quanto è il *collante* che tiene insieme le migliaia di ISP che formano Internet.

Caratteristiche BGP:

- protocollo **path-vector**, fornisce il path ma non il costo come fa distance-vector;
- protocollo decentralizzato ed asincrono

BGP fornisce ad ogni AS un modo per:

- **eBGP**: ottenere dagli AS vicini informazioni su come raggiungere sottoreti diverse.

Inoltre permette anche alle sottoreti di pubblicare la propria esistenza al resto di Internet. Basta che una sottorete urli. "*I am here*" e BGP si assicura che lo sappiano tutti i router di Internet.

- **iBGP**: propagare tali informazioni di raggiungibilità ai router interni all'AS.

Come funziona BGP

- **BGP session**: due router BGP ("peers") si scambiano messaggi BGP su **connessioni TCP semi-permanenti** (porta 179)
- pubblicizzano **path** verso differenti prefissi di rete (protocollo *path-vector*)
 - non specifica i costi del percorso o il numero di router interni che utilizzerà
 - comunica solo se è possibile raggiungere quell'indirizzo oppure no
- Fornisce:
 1. AS-path
 2. destinazione
 3. next-hop

Attributi del path e rotte BGP

Al prefisso di rete pubblicizzato sono associati *attributi* BGP: prefisso + attributi = "path".

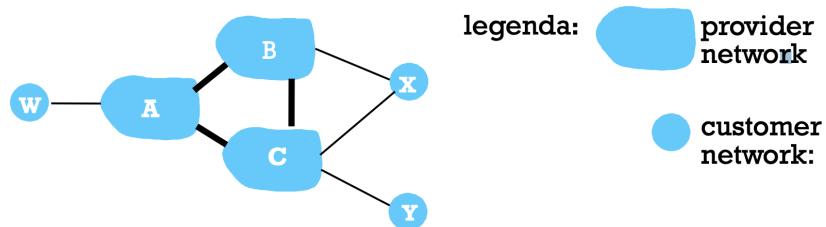
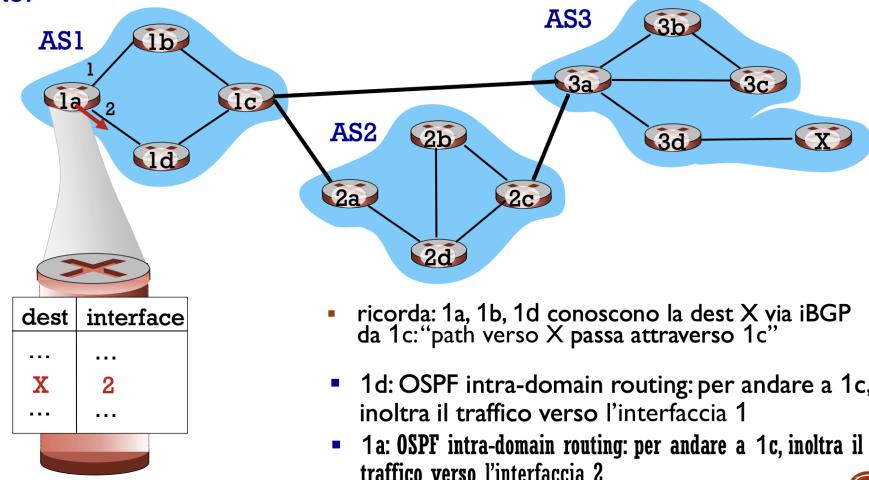
Due attributi importanti:

1. **AS-PATH**: lista degli AS attraversati dalla pubblicizzazione del prefisso di rete
2. **NEXT-HOP**: indica l'indirizzo IP del gateway router del primo AS del path a cui dovranno essere inoltrati i datagrammi

Policy-based routing:

- Il gateway che riceve tali messaggi ha una **import policy** per accettare/declinare delle rotte
- *AS policy* determina anche se **pubblicizzare** rotte ad altri AS vicini

Q: un router come setta nella tabella di inoltro l'entry per un dato prefisso di rete?



Supponi che un ISP vuole trasportare solo traffico da/verso le reti dei propri clienti (non vuole trasportare traffico in transito tra altri ISP)

- A pubblicizza il path Aw a B e a C
- B sceglie di non pubblicizzare BAw a C:
 - B non riceve alcun "guadagno" dalla rotta CBAw, dato che ne C, ne A, ne w sono client di B
 - C non impara il path CBAw
- C userà il path CAw (e non userà B) per raggiungere w

Selezione delle rotte BGP

Un router potrebbe imparare più di una rotta verso un dato prefisso di rete.

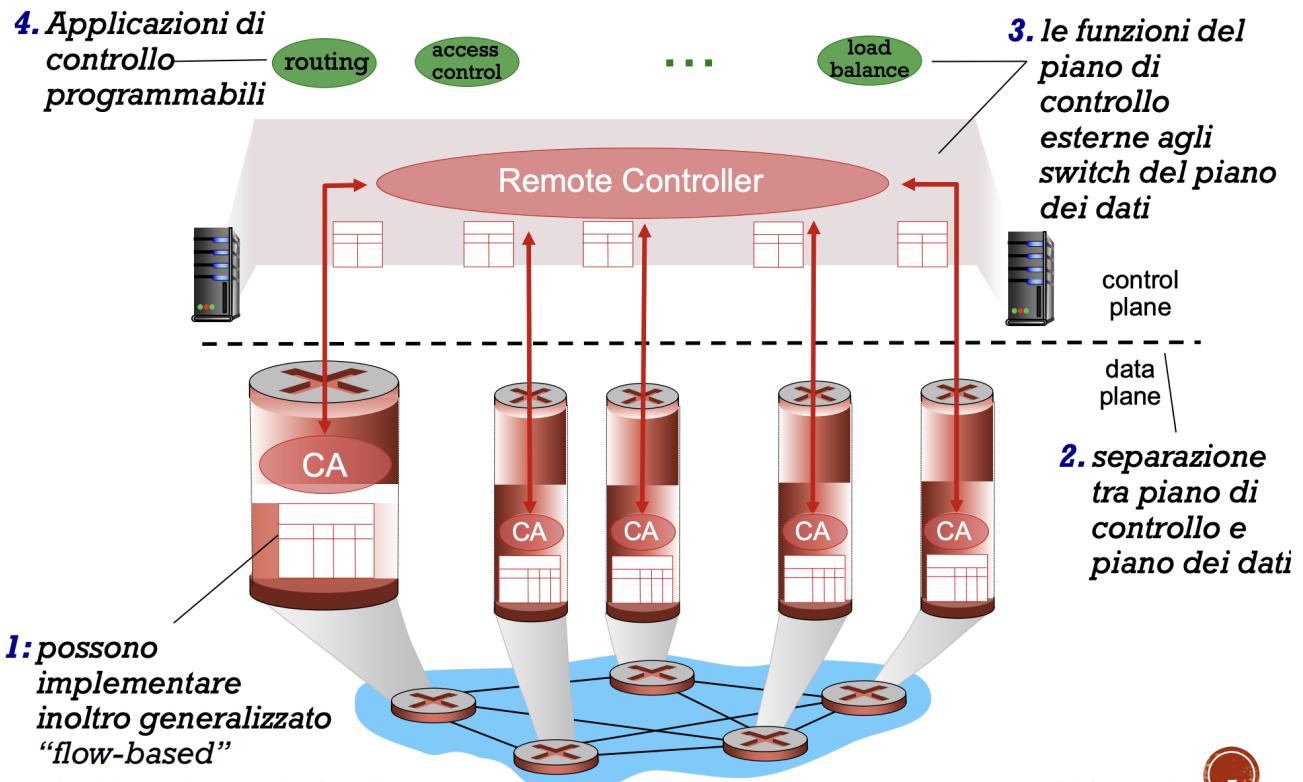
Il router seleziona la rotta basandosi su:

1. policy decision
2. shortest AS-PATH
3. closest NEXT-HOP router: *hot potato routing* (OSPF).
Sceglie il local gateway che ha il minimo costo *intra-AS*
4. criteri addizionali

BGP: politiche di instradamento

Una regola partita seguita dagli ISP commerciali è quella per cui *tutto il traffico che fluisce attraverso la rete dorsale di un ISP deve avere origine e/o destinazione in una sua rete cliente*. X è **dual-homed**: attaccato a due reti.

Policy: X non vuole che il traffico da B a C passi attraverso X



Anycast IP

- DNS root servers hanno molteplici repliche geo-localizzate attorno al mondo, tutte hanno lo stesso indirizzo IP
- Per far rispondere soltanto al *server più vicino*:
 1. DNS pubblica molteplici rotte ad quell'indirizzo IP
 2. BGP e OSPF fanno il resto del lavoro essendo che fanno imparare a tutti i router il *PATH*

12.3 Il piano di controllo SDN

A partire dal 2005 c'è un rinnovato interesse da parte dei progettisti di Internet di ripensare il piano di controllo di rete con il focus su un'implementazione logicamente centralizzata.

Controllo logicamente centralizzato: un controller distinto (tipicamente remoto) interagisce con gli agenti di controllo locale (CA, *Control Agent*) dei router per computare le tabelle di inoltro.

Per controllo **logicamente centralizzato** intendiamo un servizio di controllo dell'istradamento a cui si accede come se fosse un singolo punto centrale di servizio, anche se il servizio probabilmente viene implementato su più server per ragioni di resistenza ai guasti e scalabilità delle prestazioni.

Quattro **caratteristiche fondamentali**:

1. **Inoltro basato sui flussi:** l'inoltro dei pacchetti è effettuato sulla base del valore dei campi dell'intestazione a livelli di *trasporto*, *rete* e *collegamento*.
Questa modalità è in contrasto con l'approccio tradizionale dell'inoltro dei router basato soltanto sull'indirizzo IP di destinazione del datagramma.

Dunque, è compito del piano di controllo SDN calcolare, gestire ed installare le occorrenze della tabella dei flussi in tutti i router dell'AS.

2. Separazione del piano dei dati e del piano di controllo: il piano dei dati consiste in router (hardware) che eseguono le regole "*match-action*" nelle loro tabelle dei flussi.

Il piano di controllo consiste di server e software che determinano e gestiscono le tabelle dei flussi dei router.

3. Funzioni di controllo di rete: *esterne ai router del piano dei dati*.

Dunque, il piano di controllo è implementato via *software* in SDN, al contrario dei router tradizionali il software per il controllo è implementato su server remoti rispetto ai router di rete.

Il piano di controllo consiste quindi in due componenti: *un controller SDN* (Sistema Operativo di rete) e un *insieme di applicazioni di controllo di rete*.

4. Una rete programmabile: la rete diventa appunto programmabile attraverso le applicazioni di controllo di rete che vengono eseguite nel piano di controllo.

Tali applicazioni rappresentano il "cervello" del piano di controllo SDN e usano le API fornite dal controller SDN per specificare e controllare il piano dei dati.

SDN: piano dati e controllo

- Switch del **piano dati** (*network hardware*):

- Switch semplici e veloci che implementano *inoltro generalizzato "flow-based"* in hardware
- Le tabelle di flusso degli switch sono computate e poi installate dal controller
- Esistono API per il controllo dello switch (*es. OpenFlow*)
- Esiste un protocollo per comunicare con il controller (*es. OpenFlow*)

- **Control Plane:**

1. **SDN controller** (*network OS*):

- Mantiene le informazioni di stato della rete
- Interagisce con gli switch di rete via *southbound API*
- Interagisce con le applicazioni di controllo delle reti via *northbound API*. Queste API permettono di leggere/scrivere lo stato della rete e le tabelle dei flussi nel livello di gestione dello stato di rete.
Le applicazioni possono richiedere una notifica quando avviene un cambiamento di stato, in modo che possano intraprendere azioni in risposta.
- Implementato come un sistema distribuito per performance, scalabilità, fault-tolerance e robustezza

2. **Applicazioni di controllo della rete:**

- "cervello" del controllo, implementa le funzionalità di controllo usando i servizi offerti al livello inferiore e le API fornite dall'SDN controller
- **unbundled:** può essere fornito da terze parti, differenti dal produttore del router o dal SDN controller

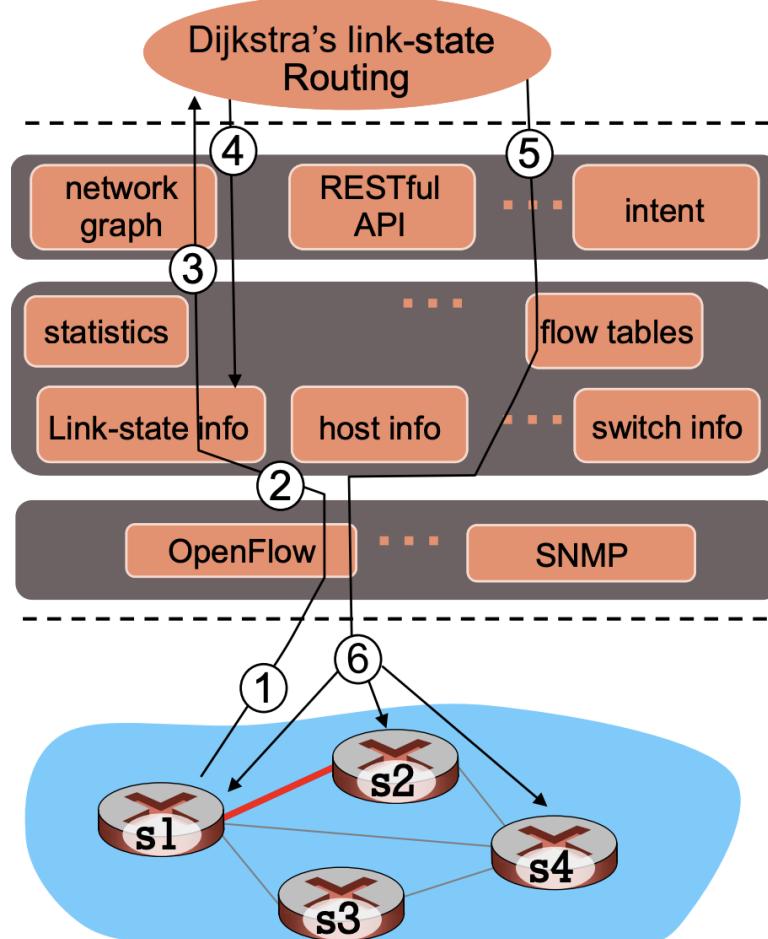
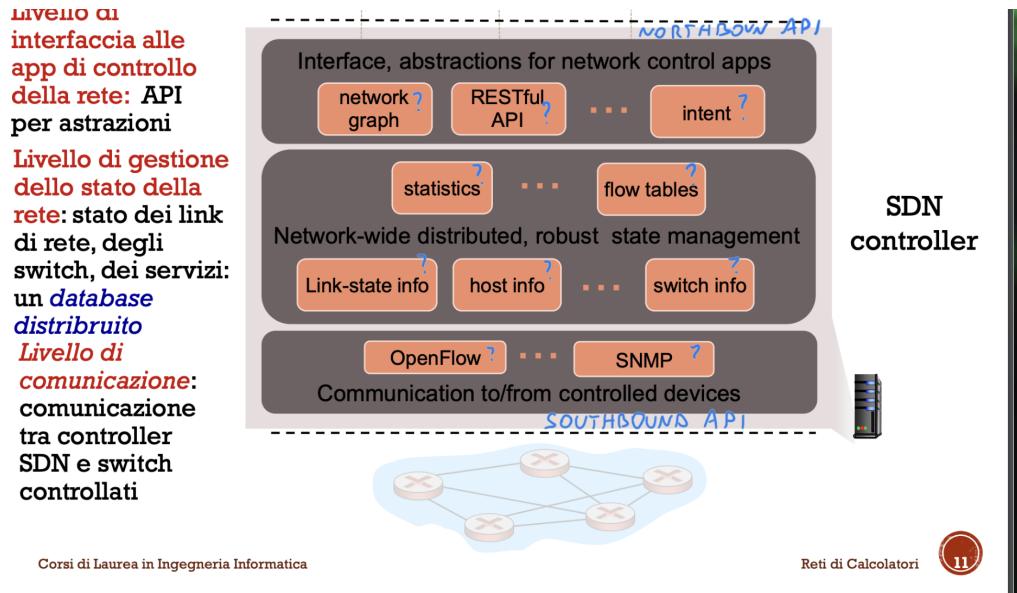


Figura 12.1: esempio di interazione tra le componenti

Componenti di un SDN controller

Analizzando l'immagine:

1. S1, si accorge del fallimento di un link e lo notifica al controller SDN
2. Il controller SDN riceve il messaggio, e aggiorna le info riguardo lo status del link
3. L'applicazione che implementa l'algoritmo di Dijkstra si è precedentemente registrata per essere invocata quando lo stato dei link cambia.
Viene così invocata
4. L'applicazione accede alle informazioni relative al grafo di rete e allo stato dei link del controller e computa così le nuove rotte
5. L'app di routing interagisce con la componente che memorizza le tabelle di flusso del controller SDN che computa e memorizza le nuove tabelle di inoltro
6. Il controller installa le nuove tabelle negli switch che hanno bisogno di essere aggiornati

12.3.1 OpenFlow Protocol

Caratteristiche:

- opera tra controller e switch
- TCP usato per scambiare messaggi, con cifratura opzionale
- tre classi di messaggi OpenFlow:
 1. ***controller-to-switch***, principali messaggi:
 - ***features***: il controller chiede agli switch le proprie caratteristiche e lo switch risponde
 - ***configure***: il controller chiede/sets i parametri di configurazione dello switch
 - ***modify-state***: aggiunge, cancella e modifica le entry nelle tabelle OpenFlow
 - ***packet-out***: il controller può mandare direttamente un pacchetto tramite uno switch da una specifica porta dello switch
 2. ***asynchronous (switch-to-controller)***:
 - ***packet-in***: trasferisce un pacchetto al controller che non corrisponde a nessuna voce della tabella di flusso
 - ***flow-removed***: informa che una entry della tabella di flusso è stata cancellata nello switch
 - ***port status***: informa il controller di un cambio di una porta
 3. ***symmetric (misc)***:
 - ***Hello***: usato durante la fase iniziale di connessione per negoziare la versione del protocollo OpenFlow
 - ***Echo request/reply***: utilizzato per verificare la raggiungibilità tra controller e switch e misurare la latenza della connessione

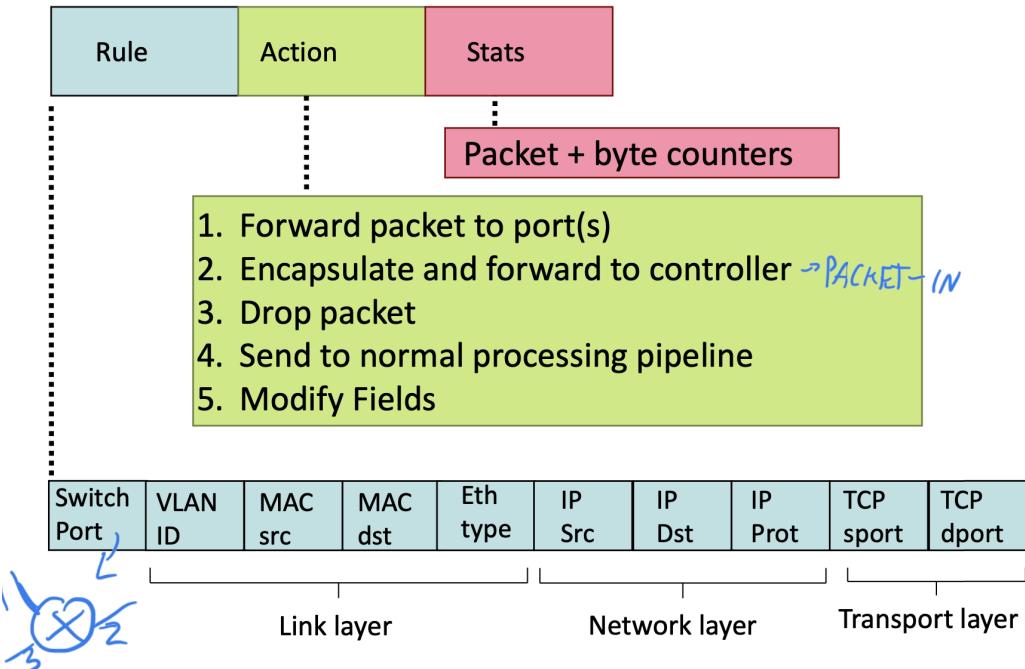


Figura 12.2: OpenFlow: entry delle tabelle

Come opera OpenFlow

Il *flow* è definito dai campi dell'header.

L'**inoltro generalizzato** si basa su semplici regole per la gestione dei pacchetti:

- **Pattern:** match con i valori dei campi dell'header del pacchetto
- **Action for matched packet:** elimina, inoltra o modifica i pacchetti matched o li invia al controller
- **Priority:** disambigua i pacchetti che si sovrappongono
- **Counters:** #bytes e #packets

Vantaggi dell'SDN

1. Algoritmi difficili da implementare in maniera distribuita
2. Un **unbounding** delle applicazioni rispetto all'hardware
3. Inoltro generalizzato: con la stessa architettura è multi-purpose.
Non ho bisogno di altro software per implementare NAT, Firewall o altri software
4. Con SDN si possono applicare al routing algoritmi più personalizzati:
 - Far sì che un pacchetto in un router non venga inviato solo in base alla destinazione ma anche in base al flusso precedente
 - *load balancing*, far trasmettere da un router il traffico tra più path

Capitolo 13

Il Livello di Collegamento

Terminologia:

- **Nodo:** qualunque dispositivo che opera a livello di collegamento;
- **Collegamenti (*link*):** canali di comunicazione, che collegano nodi adiacenti lungo un cammino.
Su ogni *collegamento*, un *nodo* trasmittente incapsula il datagramma in un **frame del livello di collegamento** (*link-layer frame*) e lo trasmette lungo il collegamento stesso.

13.1 Servizi offerti dal livello di collegamento

Sebbene il servizio di base del livello di collegamento sia il trasporto di datagrammi da un nodo a quello adiacente lungo un singolo canale di comunicazione, i dettagli dei servizi forniti possono variare da un protocollo all'altro.

Tra i possibili servizi che possono essere offerti dai protocolli di collegamento sono inclusi:

- **Framing:**
 - *Al mittente:* incapsula i datagrammi in frame e aggiunge ad ogni frame intestazione e coda
 - *Al destinatario:* riconosce l'inizio e la fine del frame, verifica la correttezza, estrae i dati e li passa al livello superiore
- **Indirizzamento:** Indirizzi MAC inseriti nell'intestazione di ogni frame per identificare sorgente e destinazione.
Gli indirizzi MAC sono indirizzi fisici, diversi dagli indirizzi IP
- **Accesso al mezzo:** se un canale è condiviso tra più utenti serve un protocollo che gestisce l'accesso al mezzo.
Specifica quando un nodo può trasmettere ed evita che si verifichino collisioni.
- **Trasmissione affidabile tra nodi adiacenti:** basata su RDT.
Il *vantaggio* di aggiungere affidabilità di trasmissione sta nel fatto che **al livello di collegamento è possibile rilevare gli errori in maniera tempestiva**, evitando che la rete impieghi tante risorse per la ritrasmissione e per riconoscere gli errori.
Dove conviene quindi implementare il *trasporto affidabile* ?

- Al livello di **collegamento** quando errori di trasmissione sono frequenti (es. collegamenti wireless, reti locali)
- Al livello di **trasporto** quando errori di trasmissione sono *rari* (es. Fibra ottica, collegamenti ad alta velocità)
- **Controllo del flusso:** regolazione della velocità di trasmissione del mittente sulle capacità di trasmissione del destinatario
- **Individuazione degli errori sui bit:** errori provocati da distorsione, attenuazione del segnale e rumore.

Quando il destinatario rileva un errore elimina il frame e segnala la perdita alla sorgente.

Questo controllo è fondamentale per la trasmissione affidabile, anche se per esempio *Ethernet* non effettua un controllo sul flusso ma fa comunque un controllo degli errori sui bit
- **Gestione della linea di comunicazione:** half-duplex e full-duplex.

Con *half-duplex*, entrambi i nodi all'estremità di una linea di comunicazione possono trasmettere ma non contemporaneamente.

Le funzioni del livello di collegamento sono implementate in *tutti i nodi*, sia host che router, nelle schede di rete o sui chip tramite una combinazione di hardware, firmware, software.

Possiamo pensarla come diviso in due sottostrati:

- **LCC (Logical Link Control):** software, si occupa della gestione dei link logici tra nodi e della fornitura di interfacce per il livello superiore
- **MAC (Media Access Control):** hardware e firmware, si occupa del controllo dell'accesso al mezzo fisico di trasmissione, gestione degli indirizzi MAC e encapsulamento/decapsulamento dei frame

La **scheda di rete** (*network interface card* e *network adapter*) implementa le funzioni dei livelli *collegamento e fisico*.

Il cuore della scheda è il **controller**:

- Chip dedicato che implementa le funzioni del livello di collegamento
- In genere implementazione hardware

Fino agli anni 90 le schede erano fisicamente separate (*schede PCI*), oggi sono quasi sempre implementate come chip montati direttamente sulla scheda madre.

13.1.1 Errori di trasmissione

Tipi di errori:

- Errori su **singoli bit** (rari)
- Errori a **raffica - burst** (comuni): il checksum non basta per gli errori a raffica, dunque serve un controllo non software implementato direttamente sul chip

L'obiettivo è quello di *rilevare e correggere gli errori*, per farlo servono sicuramente bit aggiuntivi aggiungendo ridondanza, correggere errori è ovviamente più difficile di rilevarli.

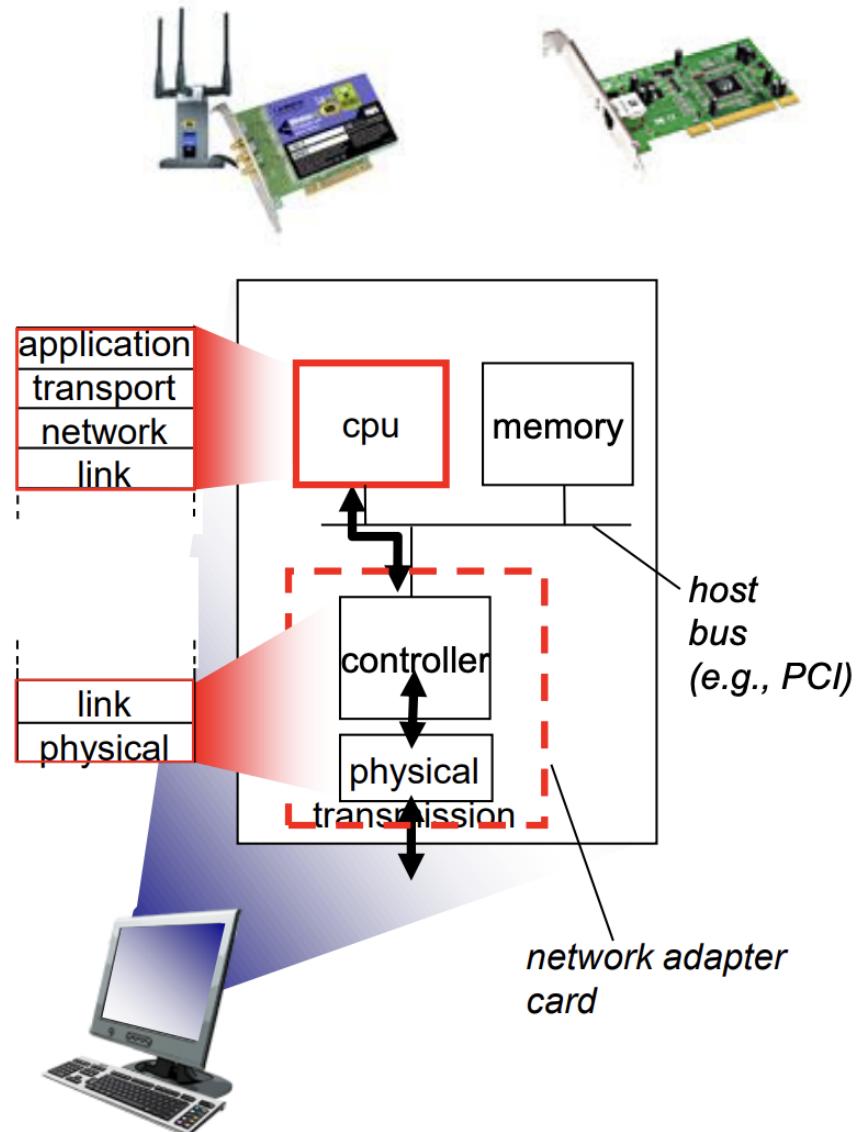
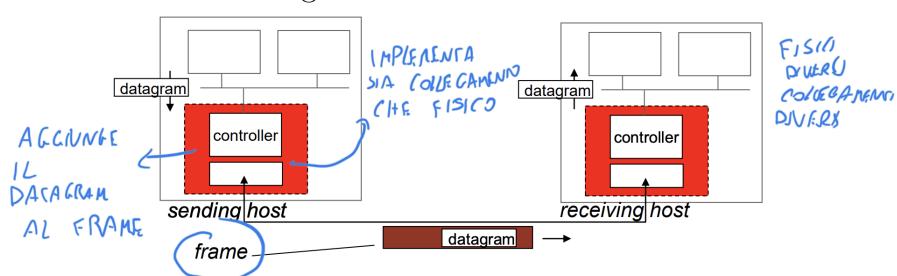


Figura 13.1: Scheda di rete

**Lato mittente**

- Incapsula i datagram nei frame
- Aggiunge i campi per il controllo degli errori, il controllo del flusso, la trasmissione affidabile, ecc.

Lato destinatario

- Recupera i datagram dal frame
- Controlla i campi per il controllo degli errori, il controllo del flusso, la trasmissione affidabile, ecc.

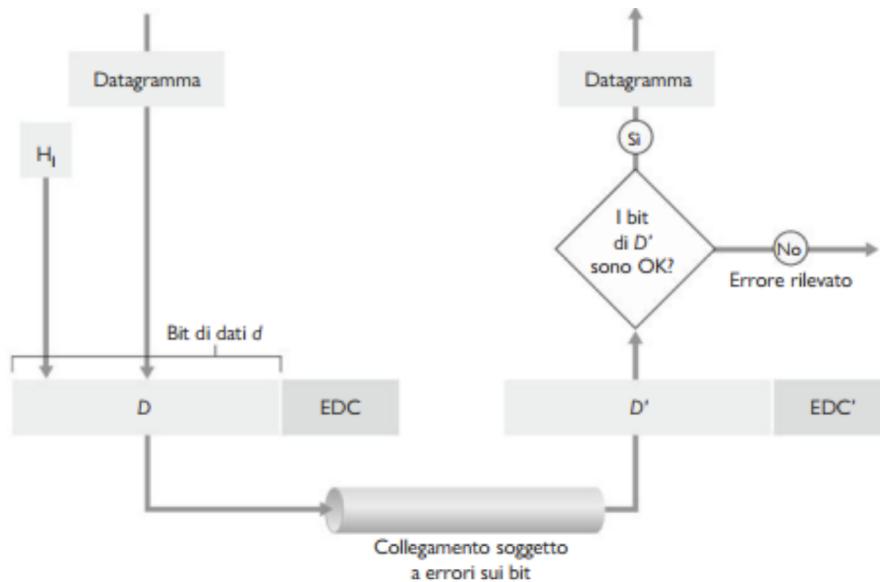
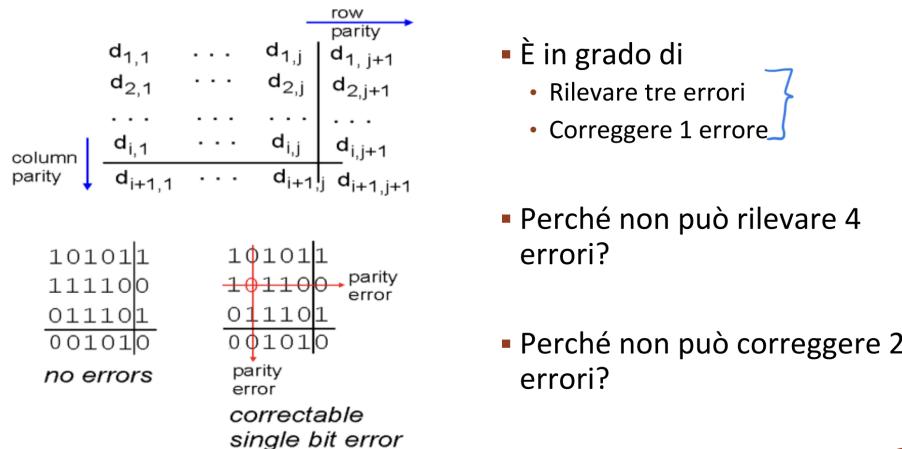


Figura 13.2: Scendario di rilevazione e correzione degli errori



Tecniche di rilevazione e correzione degli errori

Al nodo trasmittente, ad dati D che devono essere protetti da errori vengono aggiunti dei bit detti ***EDC*** (*Error-Detection and Correction*). I dati di D e i bit di EDC sono inviati in un frame al nodo ricevente che legge una sequenza di bit D' e EDC' .

Anche con l'utilizzo di bit di rilevazione degli errori è possibile che ci siano degli **errori non rilevati**.

Esistono tre tecniche per rilevare gli errori nei dati trasmessi:

1. **controllo di parità** (*parity check*): mododimensionale o bidimensionale. Utilizzato all'interno dei calcolatori per rilevare errori di trasmissione e memorizzazione.

La forma più semplice è quella che utilizza un unico bit di parità (*parity bit*):

- Parità pari: il numero degli 1 deve essere pari
 - Parità dispari: il numero degli 1 deve essere dispari

Però può rilevare solo un errore.

Invece, la **parità bidimensionale** è uno strumento più potente: calcoliamo un bit di parità per ogni riga e per ogni colonna

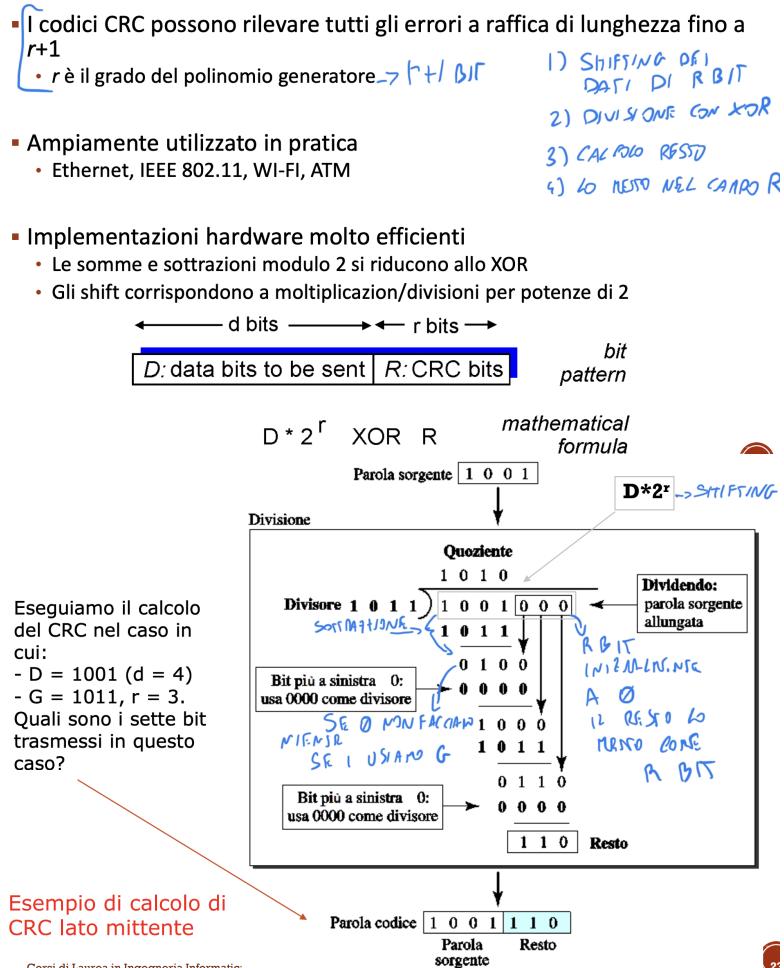


Figura 13.3: Esempio di utilizzo di CRC -lato mittente

2. **checksum**: utilizzate nei protocolli TCP/IP, semplici da implementare in software
3. **controllo a ridondanza ciclica- CRC** (*cyclic redundancy check*): utilizzato nella maggior parte dei protocolli a livello di collegamento, più complesso ma con migliori proprietà di rilevazione. Facile da implementare con *hardware dedicato*.

Tecnica basata sui *codici CRC* (*codici polinomiali*).

- Vedono il messaggio da inviare come il vettore dei coefficienti di un polinomio binario con coefficienti 0/1.
- le operazioni effettuate sul messaggio sono interpretate come operazioni sul polinomio associato
- Mittente e destinatario concordano su un **polinomio generatore G** di grado r , rappresentato quindi da una stringa binaria di $r+1$ bit che inizia con 1.
- Il mittente aggiunge al messaggio D da trasmettere una stringa **R** di r bit in modo tale che il polinomio associato alla stringa (D,R) sia divisibile per **G** (in modulo 2)
- Il *destinatario* quando riceve un pacchetto lo divide per **G**, sempre in **modulo 2**: se il **resto della divisione è 0** allora **non ci sono stati errori**, altrimenti **rileva un errore**

13.1.2 Collegamenti di Rete

Esistono due tipi di collegamenti di rete:

1. **Collegamenti punto a punto:**

- Un nodo trasmittente ed un unico nodo ricevente
- **PPP** (*Point-to-Point Protocol*) per collegamenti dial-up (stabilire una connessione diretta tra due nodi attraverso una linea telefonica)
- **HDLC** (*High-Level Data Link Control*)
- Collegamenti punto-punto tra host e switch in **Ethernet veloce**, nelle reti Ethernet moderne, ogni dispositivo è collegato direttamente ad uno switch permettendo connessioni dedicate ad alta velocità

2. **Collegamenti broadcast (canale condiviso):**

- più nodi trasmittenti e riceventi collegati allo stesso canale di comunicazione
- **Ethernet** originaria (nelle reti Ethernet tradizionali i dispositivi sono collegati ad un unico collegamento di rete condiviso)
- **IEEE 802.11 (WiFi)**
- è necessario, per questo tipo di collegamenti, definire meccanismi di regolazione per evitare interferenze tra le trasmissioni

Con un *singolo canale broadcast* dobbiamo gestire il problema dell'accesso multiplo, tramite **Protocolli ad Accesso Multiplo (MAC)**: algoritmi distribuiti che fissano le modalità con cui i nodi regolano le loro trasmissioni sul canale condiviso (quando un nodo può trasmettere).

Problema: Tutte le informazioni relative alla condivisione del canale devono essere inviate sul canale stesso, non è possibile utilizzare un canale separato per il coordinamento.

Obiettivi di un *protocollo MAC*:

1. Consentire ad un nodo di utilizzare tutta la banda quando è il solo a trasmettere
2. Ripartire equamente la banda tra i trasmittenti quando ci sono più nodi in trasmissione
3. Essere completamente decentralizzato:
 - non c'è un nodo speciale che coordina le trasmissioni
 - non richiede sincronizzazione tra i nodi
4. Essere semplice da implementare dalle schede di rete

Tipologie di *protocolli MAC*:

1. **Protocolli a suddivisione di canale:**

- dividono il canale in sottocanali
- ogni sottocanale allocato in maniera esclusiva ad un nodo

2. **Protocolli ad accesso multiplo:**

- il canale non viene diviso, ogni nodo prova a trasmettere quando ha dati da inviare
- collisioni possibili

- serve un meccanismo per rilevare e gestire le collisioni

3. Protocolli a **rotazione**:

- nodi utilizzano il canale a turno
- il tempo di utilizzo del canale può variare in funzione della quantità di dati da trasmettere

Protocolli MAC a suddivisione di canale

La suddivisione del canale è basata su tecniche di multiplexing e le risorse assegnate ai nodi possono essere: tempo, frequenze, codici.

- **TDMA** (*Time Division Multiple Acces*):

- ogni nodo ha una porzione di tempo (slot) assegnata in cui può trasmettere
- deve aspettare il proprio turno, l'accesso al canale avviene in round
- in ogni round viene inviato un frame
- se il nodo non ha un pacchetto da trasmettere lo slot viene sprecato

- **FDMA** (*Frequency Division Multiple Accesss*):

- Ogni nodo ha un frazione della banda assegnata
- può trasmettere in ogni momento ma la velocità di trasmissione è limitata
- ad ogni nodo è assegnata una banda distina e può trasmettere soltanto in quella banda

- **CDMA** (*Code Division Multiple Access*):

- ogni nodo ha un codice assegnato
- codici scelti in modo che le trasmissioni siano sovrapponibili: elimina le collisioni
- quando il ricevente riceve più segnali contemporaneamente riesce a separarli
- utilizzato per le trasmissioni in rete cellulari

13.2 Indirizzamento al Livello di Collegamento

Quando un nodo deve spedire un frame inizia a trasmettere alla massima velocità di trasmissione del canale e non esiste coordinamento a priori tra i nodi.

Se due o più nodi trasmettono contemporaneamente si verifica una **collisione**: i segnali dei diversi frame si sovrappongono e generano un segnale spazzaturea.

I protocolli **MAC** ad *accesso casuale* devono specificare:

- come si riconoscono le collisioni
- come si risolvono le collisioni

Protocolli MAC ad **accesso casuale**:

- **slotted ALOHA**:

- Quando un nodo ha un frame da trasmettere inizia a trasmettere all'inizio dello slot successivo
- Se non ci sono collisioni la trasmissione termina correttamente
- Se c'è una collisione, il nodo ritrasmette negli slot successivi ogni volta con probabilità p , fino a quando la trasmissione non riesce.

Il protocollo opera sotto le seguenti assunzioni:

- tutti i frame hanno la stessa dimensione
- tempo divisso in slot, ***tempo di uno slot = tempo di trasmissione di un frame***
- nodi iniziano a trasmettere soltanto all'inizio di uno slot, serve un meccanismo di sincronizzazione dei nodi
- se due nodi trasmettono nello stesso slot tutti i nodi rilevano la collisione

Vantaggi:

- se un nodo trasmette può usare tutta la banda del canale
- altamente decentralizzato
- semplice da implementare

Svantaggi:

- slot inutilizzati
- in caso di collisione slot sprecato
- nodi potrebbero essere in grado di rilevare le collisioni prima della fine dello slot per segnalare l'errore
- richiede sincronizzazione dei clock dei nodi

• **ALOHA:**

- Quando un nodo ha un frame da trasmettere inizia a trasmettere immediatamente, evitando la sincronizzazione dei nodi
- aumenta però così la probabilità di collisione perché un frame spedito al tempo t_0 collide con ogni altro frame spedito nell'intervallo $[t_0 - 1, t_0] + 1]$

Vantaggi di ALOHA:

- completa decentralizzazione
- però paga la completa decentralizzazione con l'efficienza pari al **18%**, la metà di *slotted ALOHA*, pari al 37%

I protocolli **ALOHA** adatti solo a reti dove vengono *trasmessi pochi frame*.

• **CSMA, CSMA/CD, CSMA/CA:** Accesso Multiplo con Rilevamento della Portante (Carrier Sense Multiple Access).

Idea chiave:

- prima di iniziare a trasmettere verifica se il canale è libero
- se è libero trasmette

- se è impegnato rimanda la trasmissione
- le collisioni possono comunque verificarsi, il *tempo di propagazione* gioca un ruolo fondamentale nel determinare la probabilità di collisione
- semplice da implementare in reti LAN cablate, in quanto misura la Portante ovvero la potenza del segnale ricevuto e lo confronta con quello trasmesso.
Se la potenza è anomala non è il segnale trasmesso.
- Difficile da implementare in LAN wireless in quanto la potenza del segnale ricevuto è molto più debole di quella del segnale trasmesso

Algoritmo CSMA/CD in *Ethernet*:

1. *NIC* (Network Interface Card - Scheda di rete) riceve un datagram dal livello di rete e lo incapsula in un frame
2. NIC testa il canale:
 - Se libero, trasmette
 - Se occupato, aspetta finché non si libera e poi trasmette
3. Se NIC trasmette l'intero frame senza rilevare collisioni passa a trasmettere il prossimo frame
4. Se NIC rileva una collisione durante la trasmissione del frame, interrompe la trasmissione e manda un segnale spazzatura (*jam sequence*).
Per far riconoscere agli altri nodi la presenza di un errore nella trasmissione, la dimensione minima del frame da trasmettere è di 512 bit (una cifra soglia scelta, in genere questa) e il nodo trasmittente rileva prima di trasmettere i primi 512bit la collisione ed interrompe. In questo modo, tutti i nodi che ottengono un frame di dimensione inferiore di 512 bit lo scartano
5. Dopo un aborto la NIC entra nel *binary exponential backoff*:
 - Dopo la m -ima collisione, NIC sceglie K a caso in $0, 1, 2, 3, \dots, 2^m - 1$.
 - NIC aspetterà $K * T$ ($T = \text{tempo per trasmettere } 512 \text{ bit}$) e poi ritorna al passo 2.

L'efficienza di *CSMA/CD* dipende da:

- traffico
- tempo di trasmissione e propagazione

$$\text{Efficienza} \approx \frac{1}{1 + 5 \frac{d_{prop}}{d_{tr}}}$$

Dunque, per riassumere, i **protocolli MAC** a:

- *partizione del canale* (TDMA, FDMA, CDMA):
 - efficienti con alti carichi, consentono di dividere equamente la banda del canale ed utilizzarlo a pieno
 - inefficienti a bassi carichi, le risorse assegnate a nodi che non devono trasmettere sono sprecate
- *accesso casuale* (ALOHA, slotted ALOHA, CSMA, CSMA/CD, CSMA/CA):
 - efficienti con bassi carichi, ogni nodo utilizza tutto il canale

- inefficiente ad alti carichi, overhead della gestione delle collisioni
- **a rotazione** (Polling, Token Ring, bluetooth): cercano di prendere il meglio tra le due categorie precedenti.
 - no collisioni
 - un singolo nodo può trasmettere per utilizzare tutta la banda

Esempio di protocolli a rotazione:

Polling:

- un nodo master e tanti nodi slave
- il master gestisce tutto il canale invitando a turno gli slave a trasmettere
- *problema* principale, singolo punto di fallimento (**master**) e ritardi dovuti all'invio dei messaggi di *invito a trasmettere*

Token Passing / Token Ring:

- *un token di controllo* passato tra i nodi
 - un nodo trasmettere solo quando ha il controllo del token
 - un nodo mantiene il token per un tempo massimo e poi lo passa al successivo
 - *problema*: singolo punto di fallimento (**token**). Utilizza una topologia ad anello, se si spezza l'anello o si perde il token bisogna rieleggere il candidato che inizia a trasmettere per primo ecc.
- Ritardi dovuti ai passaggi del token

13.2.1 Indirizzi IP vs MAC

Il livello di collegamento si occupa di trasferire frame tra i nodi appartenenti alla stessa rete, la connessione può essere *diretta* (canale punto-punto o broadcast) o *indiretta* (percorso di comunicazione passa attraverso degli switch).

Gli **switch**:

- operano solo al livello 2 (collegamento) ed utilizzano soltanto le informazioni presenti nell'intestazione dei frame
- smistano soltanto frame tra nodi della stessa rete, quindi **il destinatario del frame** sarà sicuramente un nodo della rete
- **sono trasparenti e non hanno un indirizzo**, il mittente *non sa che i suoi frame passeranno per degli switch*

Indirizzi IP:

- indirizzi a 32 bit (IPv4) assegnati alle interfacce di rete a livello di rete, hanno significato in tutto Internet
- sono indirizzi logici assegnati dall'amministratore di rete
 - se un nodo viene spostato in un'altra rete deve cambiare indirizzo IP
 - hanno una struttura gerarchica che favorisce l'instradamento dei datagram

Indirizzi MAC:

- indirizzi utilizzati localmente per identificare ogni nodo fisicamente collegato. In genere indirizzi assegnati all'interfaccia dal produttore e non modificabili.
- Indirizzo permanente in tutto Internet
- Formato dipende dal tipo di rete e dal protocollo del livello di collegamento.
- La maggior parte delle reti LAN utilizzano indirizzi MAC a 48 bit (1A-2B-3C-88-FF-AA)

ARP - Protocollo di Risoluzione degli Indirizzi (IPv4 -> MAC)

Nasce il problema di come indentificare l'indirizzo MAC associato all'indirizzo IP di destinazione:

- **Soluzione statica:** ogni router mantiene una tabella di corrispondenza indirizzi IP - MAC. Ovviamente non scalabile e difficilissima da gestire
- Soluzione **dinamica:** ogni router utilizza un protocollo per calcolare l'indirizzo MAC associato ad un indirizzo IP, *Address Resolution Protocol - ARP*

Tabelle ARP: ogni nodo in una LAN mantiene una *tabella di corrispondenze* con [**Indirizzo IP / Indirizzo MAC / TTL**].

Algoritmo di ARP per invio nella *stessa sottorete*:

1. Se l'indirizzo IPv4 del destinatario è presente nella tabella ARP legge l'indirizzo MAC associato e spedisce il frame a quell'indirizzo
2. se l'indirizzo IPv4 non è presente nella tabella ARP:
 - (a) Invia una richiesta ARP in **broadcast** sulla rete per la risoluzione dell'indirizzo IP (Indirizzo MAC di destinazione FF:FF:FF:FF:FF:FF)
 - (b) tutti i nodi della LAN ricevono la richiesta ARP, solo il destinatario corretto riconosce il proprio indirizzo IP e invia un **ARP reply**. La risposta è inviata all'indirizzo *unicast* del mittente, ovvero all'indirizzo MAC sorgente della richiesta ARP
 - (c) il mittente riceve la risposta ed aggiorna la sua tabella ARP

ARP è un *protocollo plug-and-play*: al momento dell'accensione di un nodo la sua tabella ARP è vuota e viene costruita dinamicamente.

Algorimto di instradamento verso *un nodo posto in una LAN differente*:

1. A (mittente) conosce l'indirizzo IP di B (destinatario), A sa che per raggiungere B il datagramma deve essere inoltrato ad R (gateway router)
2. A crea un datagram con indirizzo IP sorgente A e indirizzo IP destinazione B
3. A utilizza ARP per scoprire l'indirizzo MAC associato all'interfaccia di R nella rete
4. A incapsula il datagram in un fram con indirizzo MAC sorgente A e destinazione R
5. R riceve il frame, estrae il datagram, consulta la sua tabella degli indirizzi ed individua l'interfaccia a cui inviare il datagram, a sua volta lo trasferisce alla sua scheda di rete incapsulando il datagramma in un nuovo frame e lo spedisce alla sottorete 2

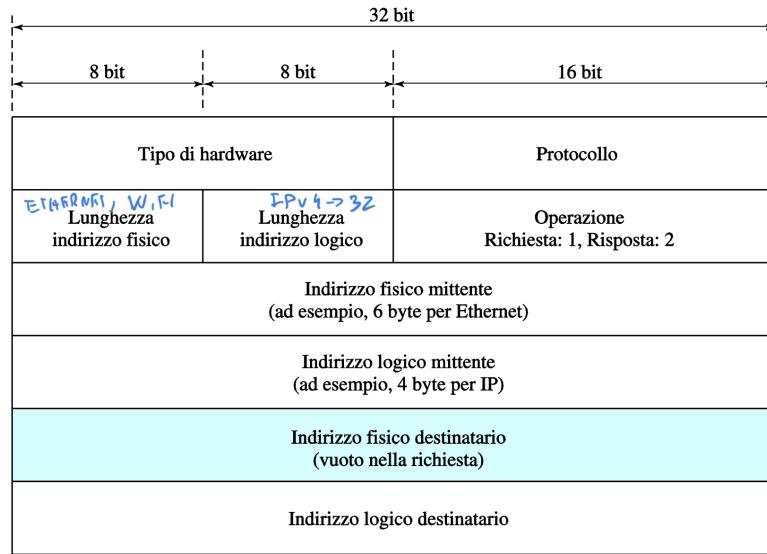
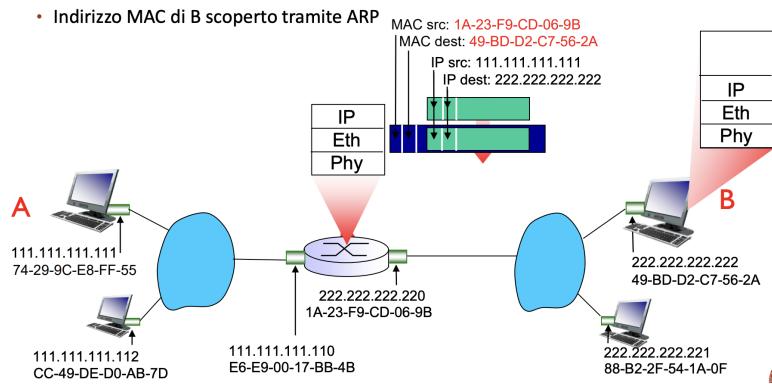


Figura 13.4: Formato messaggi ARP

6. adesso, l'indirizzo MAC di destinazione del frame è davvero quello finale, che il router ha ottenuto tramite ARP



Il protocollo ARP è utilizzato solo per indirizzi IPv4, per IPv6 è usato **NDP** (*Neighbour Discovery Protocol*) basato su ICMPv6.

13.3 Reti LAN cablate: Ethernet e VLAN

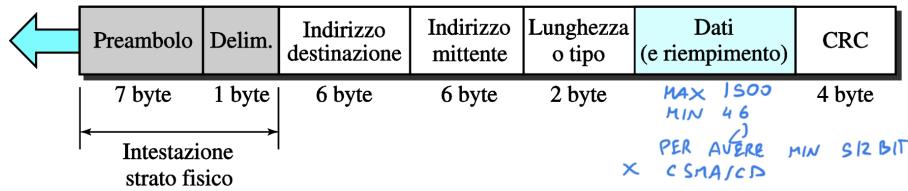
Ethernet è la tecnologia più diffusa nell'ambito delle LAN cablate.

Punti di forza:

- prima tecnologia LAN ad alta velocità sul mercato
- semplice ed economica
- Tassi di trasmissione fino a 1Gbps ed oltre

Ethernet è stata ideata da Bob Metcalfe negli anni 70, utilizzava una *tecnologia a bus*:

- cavo *coassiale* come mezzo trasmittivo condiviso
- accesso multiplto *CSMA/CD*



- il frame viaggia lungo l'*etere*, come viene definito il mezzo trasmisivo, e arrivato al terminatore torna indietro
- il ritardo di trasmissione di un frame deve essere maggiore dell'RTT, richiede una dimensione minima di un frame pari a 512 bit così da interrompere la trasmissione prima di completarla (*jam sequence*)

A metà degli anni 90 fu proposta una nuova versione di Ethernet con **topologia a stella**:

- il dispositivo centrale funge da **hub**, un commutatore "stupido" che deve inviare il frame trasmesso in broadcast a tutti tranne al nodo mittente, simulando il bus
- ogni nodo è collegato all'hub tramite una *connessione punto-punto* con *doppino telefonico*, con una distanza massima di 100 m

All'inizio del 2000 è stata introdotta **Fast Ethernet**:

- L'*hub* centrale è stato sostituito da uno **switch**, un commutatore *store-and-forward*
- lo *switch* conserva il frame ricevuto in un buffer, legge l'indirizzo di destinazione del frame e lo *inoltra solo sull'interfaccia* a cui è collegato il **nodo destinatario**
- si verifica una **collisione** soltanto quando due frame devono essere inoltrati sulla stessa interfaccia

Oggi, tutte le reti Ethernet moderne sono commutate, con una topologia a stella con switch centrale.

Formato dei Frame Ethernet

- **Tipo o Lunghezza**: indica il protocollo di livello superiore (Rete) a cui bisogn consegnare i dati (IP / ARP)
- **Dati**: lunghezza max 1500 byte e lunghezza min = 46 byte, se i dati sono inferiori, il frame viene riempito con *byte dummy*
- **CRC**: campo a 32 bit per la rilevazione degli errori, rileva tutti gli errori in una raffica di 32 bit.

Un *frame alterato* viene **scartato** (*nessun meccanismo di trasferimento affidabile*)

13.3.1 Servizio offerto da Ethernet

Le reti Ethernet forniscono un servizio a livello di collegamento che è:

1. **Senza connessione**: nessun handshake tra le interfacce prima della trasmissione dei frame

2. **Inaffidabile:** il ricevente non invia ACK o NAK (no feedback message) per i frame ricevuti
3. Il protocollo *MAC* utilizzato è **CSMA/CD** senza *slot* con *binary backoff*

Le tecnologia delle reti LAN sono state standardizzate dalla commissione 802 di IEEE: Ethernet è codificato nello standard **802.3**.

Standard IEEE 802.3:

- Ogni tecnologia 802.3 è identificata da una sigla del tipo *XX-YY-ZZ*:
 1. XX identifica la velocità di trasmissione
 2. YY identifica il tipo di trasmissione (banda bassa o banda larga)
 3. ZZ identifica il tipo di mezzo trasmissivo (T = doppino - F,B,S = fibra ottica - 2, 5 = cavo coassiale)

Gigabit Ethernet (Standard 802.3z) è una specifica di Ethernet che consente trasmissione fino a 1Gbps, compatibile con i dispositivi a 10 e 100Mbps. Esistono due modalità:

1. Full-duplex:

- Topologia a stessa con switch centrale
- Lo switch centrale ha un buffer per ogni porta per memorizzare i dati del frame in transito fino alla fine della trasmissione
- *Non usa CSMA/CD* perché **non ci sono collisioni**

2. Half-duplex:

- Topologia a stella con **hub** centrale, utilizza CDMA/CD
- Modalità tradizionale, con lunghezza minima frame di 512 bit e massima distanza tra due nodi di 25 metri;
- Frame esteso con lunghezza minima tra frame di 512 byte e massima distanza di 200 metri.

Permette di spedire più frame di piccole dimensioni nella stessa trasmissione fino ad arrivare ad una lunghezza totale di almeno 512 byte

Nel 2006 è stato definito **10 Gigabit Ethernet**:

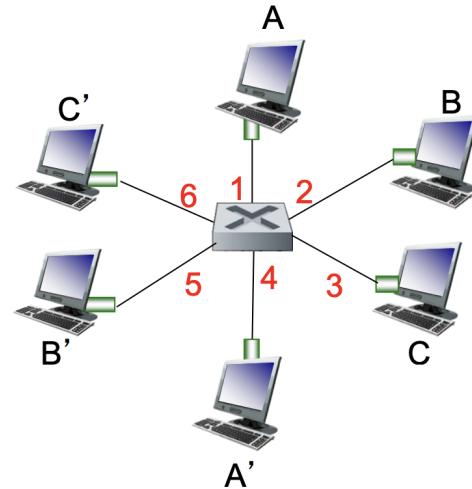
- Standard 802.3ae
- Ideato per interconnettere diverse LAN per creare reti LAN
- velocità di trasmissione pari a 10 Gbps
- Compatibilità solo con full-duplex di Gigabit Ethernet

- Ogni host ha un collegamento full-duplex dedicato verso lo switch
 - Più nodi possono trasmettere contemporaneamente
 - Ogni nodo può trasmettere e ricevere allo stesso tempo

- Lo switch ha un buffer per ogni collegamento
 - Frame da inoltrare sullo stesso collegamento vengono accodati

- Due trasmissioni A-to-A' e B-to-B' possono essere portate a termine senza collisioni

- Utilizza CSMA/CD anche se non ci possono essere collisioni



**Switch con 6 interfacce
(1,2,3,4,5,6)**

Figura 13.5: Gestione trasmissioni simultanee

	Hub	Router	Switch
Isolamento del traffico	No	Sì	Sì
Plug and play	Sì	No	Sì
Instradamento ottimale	No	Sì	No

Figura 13.6: Confronto tra le caratteristiche dei più diffusi dispositivi di interconnessione

13.3.2 Gestione trasmissioni simultanee

In una LAN commutata, il dispositivo di commutazione divide la LAN in segmenti migliorando le prestazioni, riducendo la congestione e aumentando la sicurezza attraverso l'isolamento del traffico. Questa segmentazione è trasparente per i nodi della rete, che non sono consapevoli dell'esistenza di questi segmenti e del dispositivo di commutazione che li gestisce.

Switch vs Hub:

- Un **hub** è un dispositivo *passivo*, ogni frame ricevuto viene inoltrato su tutti gli altri segmenti, simulando il canale broadcast.
Può gestire una sola trasmissione alla volta

- Uno **switch** è un dispositivo *attivo*:
 1. memorizza ed inoltra i frame (store-and-forward)
 2. in base all'indirizzo MAC di destinazione filtra le interfacce su cui inoltrare il frame
 3. utilizza CSMA/CD per trasmettere i frame
 4. gestisce più trasmissioni concorrenti
 5. dispositivo *plug-and-play*: non deve essere configurato, *auto-apprende* dinamicamente come instradare i frame nella rete

Gestione trasmissioni simulatee

Ogni host ha un collegamento full-duplex dedicati verso lo switch:

- più nodi possono trasmettere contemporaneamente
- ogni nodo può trasmettere e ricevere allo stesso tempo

Lo switch ha un buffer per ogni collegamento, i frame da inoltrare sullo stesso collegamento vengono accodati.

Utilizza CSMA/CD anche se non ci possono essere collisioni

Tabelle di commutazione:

- Lo switch svolge le operazioni di filtraggio ed inoltro tramite una tabella di commutazione, simile all'inoltro dei router
- ogni riga delle tabella contiene:
 1. l'indirizzo MAC di un nodo
 2. l'interfaccia dello switch che è collegata al nodo
 3. l'istante in cui la riga è stata inserita nella tabella (*soft state*)
- Non ci sono protocolli di intradamento, la tabella si costruisce dinamicamente

Quando lo switch riceve un frame esegue il seguente algoritmo:

1. memorizza nella tabella di commutazione l'indirizzo MAC sorgente del frame, l'interfaccia da cui è arrivato e l'orario di arrivo
2. cerca nella tabella di commutazione l'indirizzo MAC di destinazione del frame.
3. Se l'indirizzo è presente:
 - se è uguale a quello sorgente del frame, rimuove il frame
 - se è diverso da quello sorgente del frame, inoltra il frame su tutte le interfacce tranne quella da cui proviene il frame
4. se l'indirizzo non è presente: inoltra il frame su tutte le interfacce tranne quella da cui proviene il frame
5. se per un certo intervallo di tempo non riceve frame da un indirizzo sull'interfaccia collegata lo rimuove dalla tabella

Ogni switch **apprende dinamicamente** quali nodi possono essere raggiunti e a quali interfacce sono collegate: legge l'indirizzo MAC sorgente di ogni frame ricevuto e registra la coppia *indirizzo/interfaccia*.

13.4 Interconnessione di Switch: LAN

Gli switch possono essere collegati tra loro per creare *LAN* estese.

Switch vs Router

Entrambi operano *store-and-forward*:

- **Router:** opera a livello 3, di rete (datagrammi IP)
- **Switch:** opera a livello 2, collegamento (frame)

Entrambi utilizzano tabelle di inoltro:

- Router: utilizza algoritmi di instradamento
- Switch: utilizza auto-apprendimento dinamico

Diversa visibilità:

- Router: non sono trasparenti
- Switch: sono trasparenti

Limiti delle LAN organizzate gerarchicamente

Le LAN organizzate gerarchicamente hanno *tre problemi* principali:

1. Mancanza di isolamento dal traffico:

- Il traffico tra nodi della stessa LAN è localizzato nella LAN ma il traffico broadcast attraversa l'intera sottorete, con i messaggi *ARP e DHCP*
- Sarebbe opportuno limitarli per motivi di riservatezza e di efficienza (evitare collisioni)

2. Uso inefficiente degli switch:

- Serve uno switch per ogni LAN, indipendentemente dal numero dei nodi della rete

3. Gestione inefficiente degli utenti:

- Se un nodo deve spostarsi da una LAN ad un'altra bisogna modificare il cablaggio della rete

Per superare questi problemi sono stati introdotti switch che supportano **LAN Virtuali**

13.4.1 Switch VLAN

Gli switch che supportano LAN Virtuali permettono di definire più reti locali virtuali su una singola infrastruttura fisica di rete locale.

- Host appartenenti alla stessa VLAN comunicano come se fossero connessi allo stesso switch
- Host **non appartenenti** alla *stessa VLAN* non possono comunicare tramite lo switch, bisogna passare per un router che collega le due VLAN
- L'assegnazione degli host alle VLAN è fatta in maniera virtuale (*via software*), è possibile modificarla in ogni momento modificando il file di configurazione dello switch.

VLAN Port-Based

L'amministratore di rete divide le porte dello switch in gruppi:

- Ogni gruppo definisce una VLAN
- Il traffico broadcast in una VLAN non viene inoltrato sulle altre VLAN
- Si possono utilizzare al massimo tutte le porte degli switch
- Per spostare un utente da una VLAN all'altra è sufficiente modificare il file di configurazione dello switch

Comunicazione tra nodi di VLAN distinte

L'inoltro di frame tra nodi collegati allo stesso switch ma appartenenti a VLAN distinte deve essere fatto tramite un router, con l'instradamento nell'inter-rete. L'**hardware dello switch si limita a consegnare frame tra porte appartenenti alla stessa VLAN**.

Il router per collegarsi alle diverse VLAN ha anche lui un *collegamento fisico* in una porta specifica (1) dello switch e configurarlo in modo che appartenga ad entrambe le VLAN.

Per fortuna non è più necessario attuare quello dette precedentemente, in quanto, I dispositivi di rete in commercio contengono sia switch che router:

- Il router interno fa gateway tra le diverse VLAN dello switch
- in questo modo non è più necessario un router esterno separato.

VLAN contenenti diversi Switch: VLAN trunking (IEEE 802.1Q)

Se i nodi della stessa LAN sono distribuiti in molti edifici separati, si dovranno interconnettere due switch separati.

Una soluzione potrebbe essere quella di aggiungere un collegamento per ogni VLAN di uno switch sull'altro switch, ma non è possibile in quanto non **scalabile**, in quanto N VLAN richiederebbero N collegamenti.

La soluzione adottata è il **VLAN trunking** (condotto, conduttore), una porta speciale per ogni switch (quella in basso a destra per il primo e quella in alto a sinistra per il secondo) viene configurata come *porta di trunking* per interconnettere i due switch VLAN. La porta di trunking appartiene a tutte le VLAN e i frame inviati a qualunque VLAN vengono inoltrati attraverso il collegamento di trunking all'altro switch.

Però, **come fa uno switch a sapere che un frame che arriva ad una porta di trunking appartiene ad una VLAN particolare?**

IEEE ha definito un *formato esteso* di frame Ethernet nello standard 802.1Q, per frame che attraversano un trunk VLAN.

- Aggiunge un campo di 4 byte (tag VLAN)
- il tag VLAN viene aggiunto dallo switch che trasmette sulla porta di trunking e rimosso dallo switch che lo riceve (*tunnelling*)

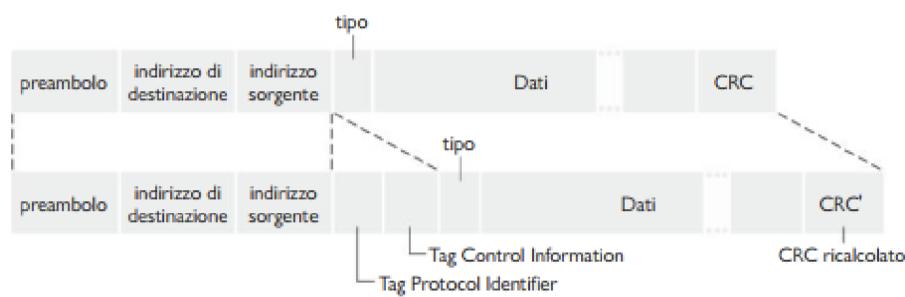


Figura 13.7: Frame Ethernet originale vs. frame Ethernet 802.1Q con etichetta VLAN

Capitolo 14

Reti Wireless

Aspetti importanti:

- **wireless**: comunicazione su canali wireless
- **mobilità**: gestione di utenti mobili (che cambiano il punto di collegamento alla rete)

Elementi di una *Rete Wireless*:

- **wireless hosts**:
 - laptop, smartphone
 - eseguono applicazioni
 - stazionari o mobili (wireless non implica mobilità)
- **stazione base (base station)**:
 - tipicamente connessa ad una rete cablata (wired)
 - responsabile per lo scambio di pacchetti tra rete wired ed host wireless nella sua "area" (punti d'accesso, Access Point 802.11)
- **link wireless**:
 - usato per connettere stazioni wireless alla base station
 - usato anche come collegamento di backbone
 - un protocollo ad accesso multiplo regola l'accesso al collegamento (tipicamente a livello di collegamento si usa la trasmissione affidabile)
 - differiscono per banda e per distanza di trasmissione
- **rete con infrastruttura**:
 - la base station connette gli host mobili alla rete wired
 - *Handover*: quando l'host si sposta dall'area di copertura di una stazione base ad un'altra cambia il suo punto di collegamento con la rete globale
- **reti ad hoc**:
 - non ci sono stazioni base
 - gli host wireless non hanno alcuna infrastruttura a cui connettersi
 - gli host stessi provvedono ai servizi d'intradamento, di assegnazione degli indirizzi, di DNS

	Hop singolo	Hop multipli
modalità infrastruttura (es. AP)	L'host si collega a una stazione base (WiFi, WiMAX, cellulare) che lo collega al resto della rete	Con stazione base, alcuni nodi potrebbero dover fare affidamento ad altri nodi wireless per connettersi alla rete più ampia. Esempi: reti mesh networks e reti di sensori
senza infrastruttura AD HOC	Senza stazione base, uno dei nodi può coordinare la trasmissione degli altri (Bluetooth, reti ad hoc)	Senza stazione base, i nodi possono dover ritrasmettere i messaggi a molti altri nodi per raggiungere la destinazione. Esempi: MANET, VANET

MOBILE Ad hoc NETWORK ↗
 ↙ VEICHLE

Figura 14.1: Tassonomia delle reti wireless

14.1 Caratteristiche dei link wireless

I link wireless si differenziano da quelli wired per:

1. **attenuazione del segnale**: le radiazioni elettromagnetiche (*onde omnidirezionali a bassa frequenza*) si attenuano quando attraversano determinati ostacoli, anche nello spazio libero il segnale si attenua al crescere della distanza per corda (*path loss*)
2. **interferenza da parte di altre sorgenti**: frequenze wireless standard (es 2,4Ghz) condivise da altri dispositivi, anche rumori ambientali causano interferenza
3. **propagazione su più cammini**: una parte delle onde elettromagnetiche si riflette su oggetti e sul terreno, compiendo cammini di diversa distanza tra trasmittente e ricevente, oggetti in movimento possono modificare questi ritardi dinamicamente

Dunque, anche la comunicazione punto-punto diventa più complessa e con un *Bit-Error-Rate* (BER) molto più elevato rispetto alle reti cablate. Di conseguenza, i collegamenti wireless implementano non soltanto il controllo CRC per gli errori, ma anche un **trasferimento affidabile a livello di collegamento** per ritrasmettere i frames corrotti.

L'**SNR** (*Signal-to-Noise Ratio*) indica il rapporto tra la potenza del segnale trasmesso e il rumore, più è grande e più facilita il ricevente nell'estrazione del segnale trasmesso dal rumore di fondo. SNR è tipicamente espressa in dB, unità di misura dagli ingegneri elettronici principalemte per confondere gli informatici.

Bilanciamento di SNR e BER:

- per un dato schema di modulazione, maggiore è SNR e minore sarà il BER (*probabilità che un bit sia corrotto*)
- per un dato SNR, una tenica di modulazione con più elevato tasso di trasmissione dei bit avrà un BER più alto ($BSK < QAM16 < QAM256$)

Aumentare l'SNR oltre una certa soglia non conviene perchè:

1. aumenta esponenzialmente il cosumo di energia ed il consumo della batteria, ma il guadagno il termini di BER è poco significativo

2. aumenta la possibilità di interferenze con dispositivi vicini

I moderni sistemi di trasmissione wireless (*802.11 WiFi e reti cellulari 4G/5G*) scelgono dinamicamente le tecniche di modulazione e di codifica al livello fisico per adattarsi alle condizioni del canale.

Un **maggiore e dinamico bit error rate**, non è l'unica differenza tra i collegamenti wired e wireless. Più mittenti e riceventi wireless creano problemi aggiuntivi (*oltre a quelli legati all'accesso multiplo*):

- **Problema del *terminale nascosto***: oggetti fisici nell'ambiente fanno da ostacoli non permettendo la comunicazione tra due base station A e C anche se questi possono comunicare con B ma causano interferenza
- **Problema del *Fading* (degradazione del segnale)**: quando due stazioni A e C sono posizionati in modo che i loro segnali non sono abbastanza forti per comunicare ma sono abbastanza forti per interferire tra di loro ad una stazione B

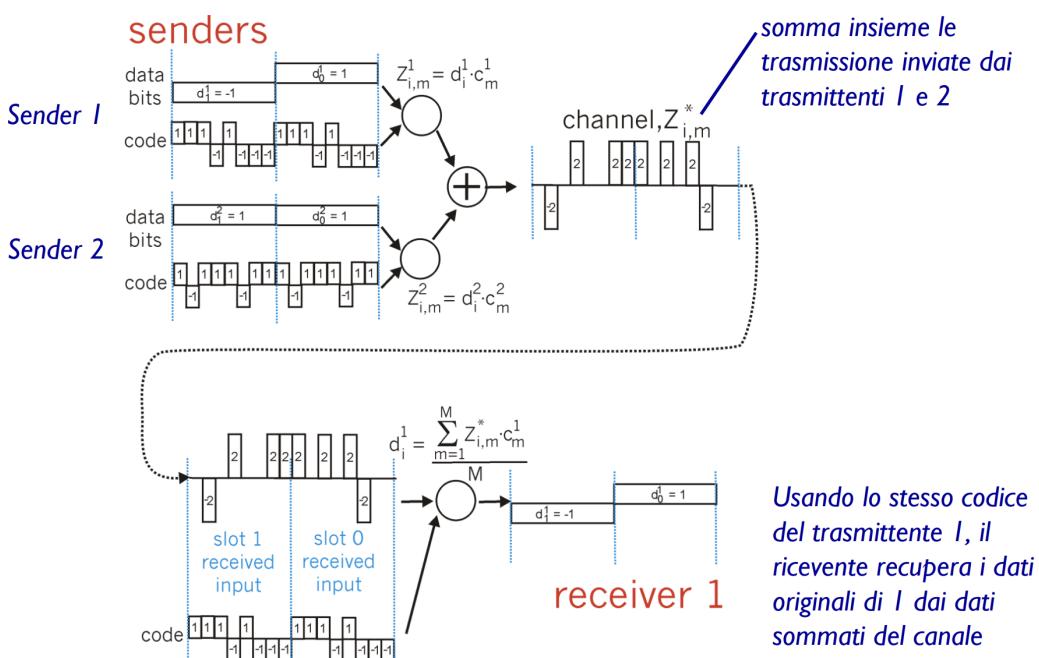
Questi due problemi, rendono l'*accesso multiplo* nei collegamenti wireless più complicato che in quelli wired.

CDMA

CDMA è il protocollo di accesso multiplo al canale condiviso più diffuso nelle reti wireless e nelle tecnologie cellulari.

Un "codice" unico viene assegnato a ciascun utente (*code set partitioning*):

- tutti gli utenti condividono la stessa frequenza, ma ciascun utente ha una propria sequenza "chipping" per codificare i dati
- consente a più utenti di "coesistere" e trasmettere simultaneamente con un'interferenza minima (se i codici sono *ortogonalni*)
- **Segnale codificato** = (dati originali) x (sequenza chipping - CP code)
- **Decodifica**: prodotto interno del segnale codificato e sequenza chipping



IEEE 802.11 standard	Year	Max data rate	Range	Frequency
802.11b	1999	11 Mbps	30 m	2.4 Ghz
802.11g	2003	54 Mbps	30m	2.4 Ghz
802.11n (WiFi 4)	2009	600	70m	2.4, 5 Ghz
802.11ac (WiFi 5)	2013	3.47Gbps	70m	5 Ghz
802.11ax (WiFi 6)	2020	14 Gbps	70m	2.4, 5 Ghz
802.11af	2014	35 – 560 Mbps	1 Km	unused TV bands (54-790 MHz)
802.11ah	2017	347Mbps	1 Km	900 Mhz

- Tutte le versioni utilizzano CSMA/CA per l'accesso multiplo
- Esistono sia versioni con infrastruttura che ad-hoc

Figura 14.2: Versioni dello Standard IEEE 802.11

14.2 WiFi: 802.11 Wireless LANs

- **IEEE 802.11:** WiFi - rete con infrastruttura
- **IEEE 802.12:** Bluetooth - rete-ad-hoc

Architettura delle LAN 802.11:

- Il blocco fondamentale dell'infrastruttura è il **Basic Service Set** (BSS), anche detta "cella".
- Un **BSS** contiene gli host/stazioni wireless che comunicano con la stazione base (base station), ovvero il punto di accesso ad interne (base station = **AP**).
- **Ad hoc network:** le stazioni wireless possono pure unirsi tra di loro per formare un'architettura ad hoc, un network senza un controllo centrale e senza connessioni con l'esterno.

Connessione *on-the-fly*, senza bisogno di connetersi con una struttura centralizzata

Due o più reti di base possono essere collegate in una rete estesa:

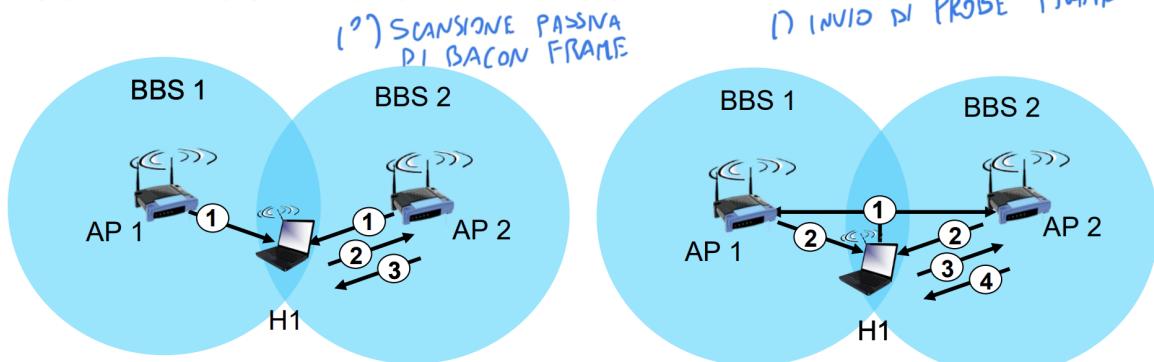
- **Extended Service Set** (ESS)
- collegate da un sistema di distribuzione cablato

802.11: Canali di trasmissione

- 802.11b utilizza lo spettro da 2.4GHz a 2.48GHz diviso in 11 canali parzialmente sovrapposti:
 - due canali non si sovrappongono se distano almeno 4
 - 1, 6, 11 è l'unica terna di canali non sovrapposti su cui è possibile trasmettere contemporaneamente senza interferenze

Quando l'amministratore della rete attiva un AP, gli assegna un identificativo **SSID** (*Service Set Identifier*):

802.11: SCANSIONE PASSIVA/ATTIVA



Scansione passiva:

- (1) Frame beacon inviati dagli AP
- (2) Invio di un frame di richiesta associazione da H1 all'AP selezionato
- (3) Invio di un frame di risposta di associazione dall'AP selezionato a H1

Scansione attiva:

- (1) Frame sonda (probe) di richiesta inviato in broadcast da H1
- (2) Frame sonda di risposta inviato dagli AP
- (3) Invio di un frame di richiesta di associazione da H1 all'AP selezionato
- (4) Invio di un frame di risposta di associazione dall'AP selezionato a H1

- l'amministratore dell'AP scegli il canale utilizzato dall'AP
- reti vicine dovrebbero utilizzare canali di trasmissione non sovrapposti

L'AP segnala la sua presenza trasmettendo periodicamente un **beacon frame**: contiene l'*SSID* ed il *canale di trasmissione dell'AP* ed il suo *indirizzo MAC*.

802.11: Associazioni

Ogni host deve associarsi con un AP:

1. esamina i canali per verificare l'eventuale presenza di beacon frames contenenti il nome dell'AP (SSID) e l'indirizzo MAC ed il canale utilizzato
2. seleziona l'AP con cui associarsi
3. può eseguire una procedura di autenticazione
4. dopo l'associazione tipicamente usa DHCP per ottenere l'indirizzo IP nella sottorete dell'AP

L'operazione di analisi dei canali alla ricerca di AP vien detta **scansione**, può essere *attiva o passiva*.

14.2.1 802.11: CSMA/CA (Collision Avoidance, MAC protocol)

- Problema collisioni: più nodi che trasmettono allo stesso istante
- **CSMA** ascolta il canale prima di trasmettere tramite la rilevazione della portante.
Non si verificano collisioni con la trasmissione in corso

- 802.11: **non rileva le collisioni!**

- difficoltà in ricezione (*sense collision*) durante la trasmissione, a causa della debolezza del segnale ricevuto (*fading*)
- in ogni caso, non potrebbe rilevare tutte le collisioni: problema del *termina nascosto*
- Obiettivo: evitare le collisioni tramite *CSMA/CA*
- poichè il BER è alto, utilizza un **ARQ** (*stop-and-wait*) per fornire un servizio di trasferimento affidabile

Algoritmo alla base di *CSMA/CA*

- **Mittente 802.11:**

1. se percepisce il canale inattivo per **DIFS** (*Distributed Inter-frame Space*) secondi, allora trasmette l'intero frame (no CD)
2. se percepisce il canale occupato, allora:
 - (a) sceglie un valore di ritardo causale usando *binary exponential backoff*
 - (b) decrementa questo valore se il canale viene percepito come inattivo
 - (c) quando il contatore arriva a 0, trasmette l'intero pacchetto
 - (d) se riceve ACK, se ha un altro frame da inviare inizia dal decrementare il valore iniziale precedente dello step 2
 - (e) se non riceve ACK, sceglie un nuovo valore di ritardo casuale superiore a quello precedente

- **Ricevente 802.11:**

1. Se il pacchetto ricevuto è OK:
 - invia un ACK dopo **SIFS** (*Short Inter-frame Spacing*) secondi (necessario a causa del problema del terminale nascosto)
2. Altrimenti lo scarta

Evitare le collisioni: RTS/CTS

Idea: consentendo al mittente di "prenotare" il canale si evitano collisioni anche durante l'invio di lunghi pacchetti di dati. Però è opzionale in quanto è costoso.

- Il mittente inizia a trasmettere un **piccolo** pacchetto **request-to-send RTS** (prenotazione) all'AP usando CSMA.

Potebbero verificarsi collisioni tra i pacchetti RTS ma comunque sono molto piccoli

- AP risponde diffondendo in broadcast il pacchetto **clear-to-sent CTS** in risposta al pacchetto RTS ricevuto
- il pacchetto CTS è ricevuto da tutti i nodi:
 - il mittente invierà il pacchetto
 - le altre stazioni rimanderanno eventuali trasmissioni
- *Evitare completamente le collisioni usando piccoli pacchetti di prenotazione*

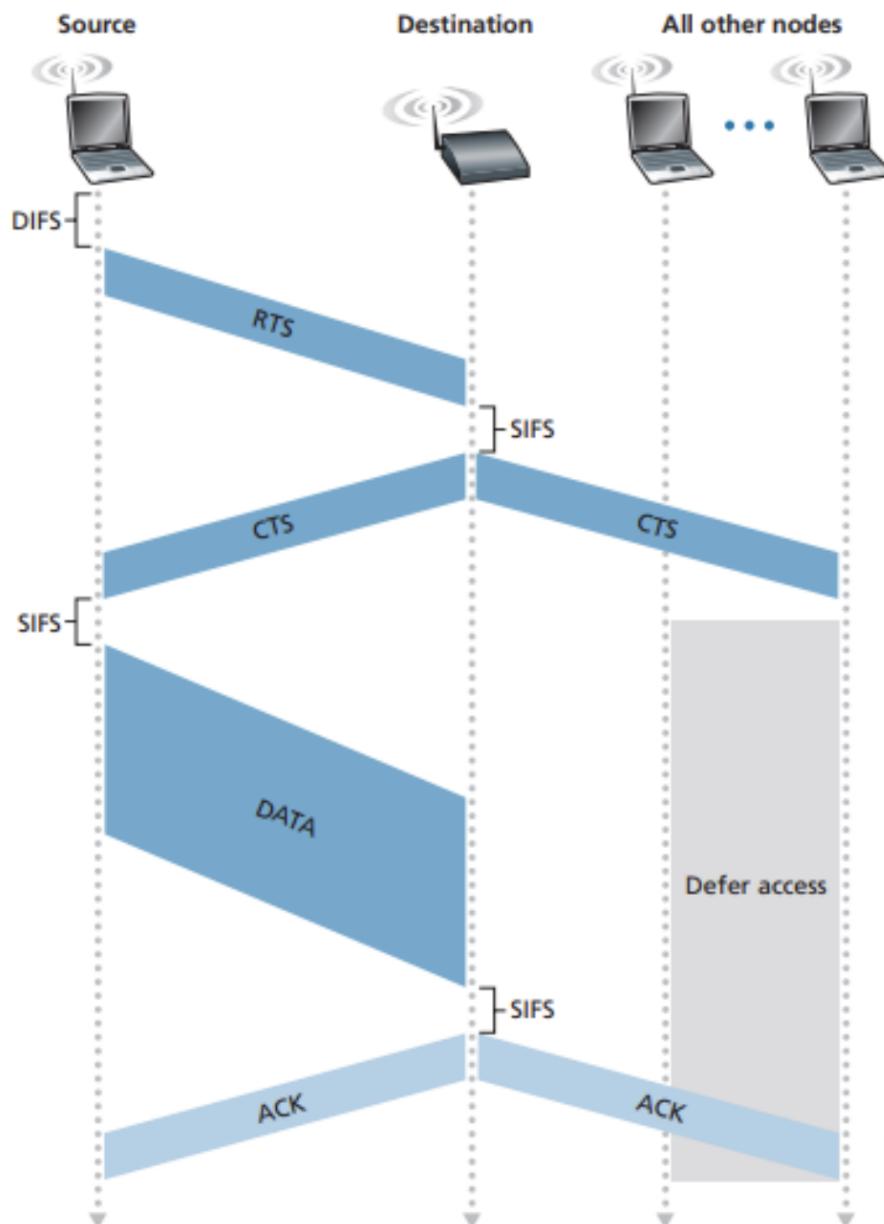


Figura 14.3: Collision Avoidance using RTS/CTS frames

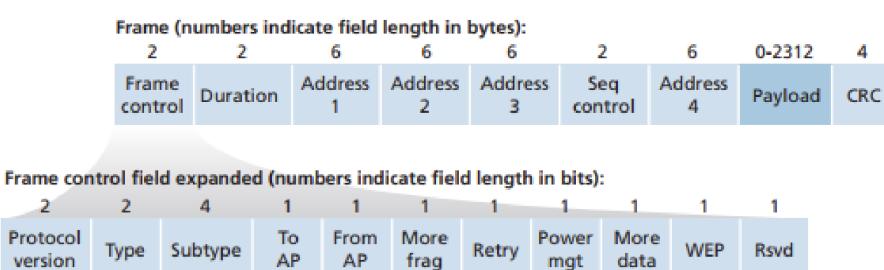


Figura 14.4: The IEEE 802.11 frame

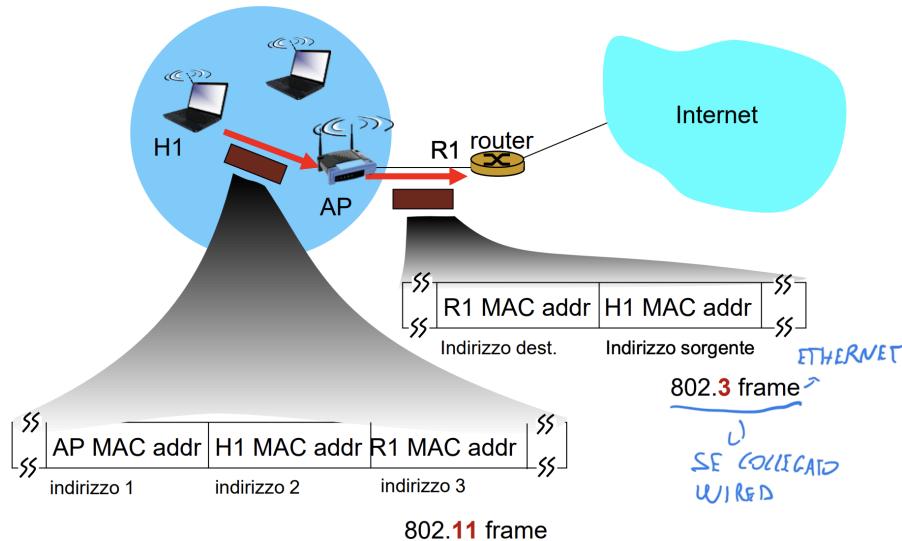


Figura 14.5: Uso degli indirizzi nei frame 802.11

Frame 802.11: indirizzamento

- **Payload and CRC Fields:** tipicamente il Payload è inferiore a 1.500 bytes, includendo un datagramma IP oppure un pacchetto ARP.
Come per il frame Ethernet, include 32-bit di controllo ciclico (CRC).
- **Campi per gli indirizzi:** la differenza maggiore è che ha *quattro* campi per gli indirizzi. Tre sono necessario per interconnettersi, specificamente per spostare il datagramma dal livello di rete ad una stazione wireless attraverso un AP verso l'interfaccia di un router. Il quarto è usato quando le APs inoltrano frames tra di loro in un *ad-hoc mode*.
I primi tre, quelli dedicati per le infrastrutture network, sono:
 - **Indirizzo 2:** indirizzo MAC dell'host wireless o dell'AP che trasmette il frame
 - **Indirizzo 1:** indirizzo MAC dell'host wireless o AP che deve ricevere questo frame
 - **Indirizzo 3:** indirizzo MAC dell'interfaccia router a cui è collegato AP

Nella figura ci sono due *APs*, ognuna delle quali responsabile per un numero di stazioni wireless. Ognuna delle APs ha una connessione diretta con un *router*, che si connette con il resto di Internet.

Ricordiamoci che un AP è un dispositivo a *livello di collegamento*, quindi non riesce a "parlare" IP e non "capisce" gli indirizzi IP.

Consideriamo di spostare un datagramma dall'interfaccia del Router *R1* alla stazione wireless *H1*. Il router non sa che c'è un *AP* tra lui e *H1*, dal punto di vista del router, *H1* è solo un host in uno delle sottoreti connesse al router:

- Il router, conosce l'indirizzo IP di *H1* (dall'indirizzo di destinazione del datagramma), usa ARP per determinare l'indirizzo MAC di *H1*, come in una LAN Ethernet ordinaria.

Dopo aver ottenuto l'indirizzo MAC di *H1*, l'interfaccia del router *R1* incapsula il datagramma in un *frame Ethernet*. L'indirizzo di destinazione di questo frame contiene l'indirizzo MAC di *R1*, e l'indirizzo di destinazione contiene l'indirizzo MAC di *H1*.

- Quanto il frame Ethernet arriva all'AP, lo converte dal frame 802.3 di Ethernet all'802.11 prima di trasmettere il frame all'interno del canale wireless.

L'AP inserisce:

- l'indirizzo MAC di H1 in **indirizzo 1**
- il suo indirizzo MAC in **indirizzo 2**
- l'indirizzo MAC del router R1 in **indirizzo 3**

In questo modo H1 può determinare, dall'indirizzo 3, l'indirizzo MAC dell'interfaccia del router che ha inviato il datagramma all'interno della sottorete.

Consideriamo ora cosa succede quando la stazione wireless *H1* risponde inviando un datagramma all'interfaccia del Router *R1*:

- H1 crea un frame 802.11, inserendo:
 - indirizzo MAC dell'AP in **indirizzo 1**
 - indirizzo MAC suo nell'**indirizzo 2**
 - inserisce l'indirizzo MAC di R1 nell'**indirizzo 3**
- Quando un AP riceve l'indirizzo 802.11, lo converte nel fram Etherent, inserendo:
 - l'**indirizzo sorgente** con l'indirizzo MAC di H1 (*indirizzo 2*)
 - **indirizzo destinazione** con l'indirizzo MAC di R1 (*indirizzo 3*)
 - Così, l'*indirizzo 3* ha permesso all'AP di determinare l'indirizzo MAC di destinazione appropriato quando ha costruito il frame Ethernet

Per riassumere, l'*indirizzo 3* gioca un ruolo fondamentale per l'interconnessione tra il BSS con un collegamento wired LAN.

Altri *campi* frame 802.11:

- **Numero di seq. del frame:** essendo che la stazione che riceve un frame dovrà inviare un ACK, quest'ultimi si possono perdere e la stazione mittente potrebbe ritrasmetterli. Di conseguenza serve per scartare possibili copie del frame ricevuto correttamente.
- **Duration:** sappiamo che il protocollo 802.11 permette ad una stazione wireless di riservare in canale per un periodo di tempo che include il tempo per trasmettere i dataframe ed il tempo per trasmettere gli ACK.

Il valore della durata è inlusso appunto del campo "duration", sia nei frames RTS e CTS

- **Frame Control:** il campo Frame Control contiene molti sottocampi, i più importanti sono:

- **Type e Subtype:** usati per distinguere le *associazioni*, RTS, CTS, ACK, data frames, bacon.
- **To AP / From AP:** usati per definire il significato dei diversi campi "*indirizzi*". Fa distinguere tra network ad-hoc o con infrastruttura
- **WEP:** indica se è utilizzata la crittografia

Mobilità nelle sottoreti

Per aumentare il range fisico di copertura di una LAN wireless, aziende ed università aggiungono più BSSs all'interno della stessa IP subnet.

Questa scelta porta naturalmente al problema della mobilità tra le diverse BSS, però facilmente risolvibile per BSS che fanno parte della stessa sottorete.

Lo **switch** deve sapere a quale AP è associato l'host che si sta spostando:

- Gli switch auto-apprendono quale porta può essere usata per raggiungere l'host
- se cambia BSS, il nuovo AP può inviare un frame Ethernet via broadcast per istruire "subito" lo switch.
- soluzione in caso l'host con cambia l'indirizzo IP

In questo modo si può gestire anche la mobilità tra base station che fanno parte della stessa VLAN.

Funzionalità avanzate di 802.11

- **Adattamento del tasso trasmissivo:** la stazione base ed utente mobile cambiano dinamicamente il proprio tasso trasmissivo (*tecnica di modulazione a livello fisico*) e di conseguenza varia SNR.
 1. SNR cala e BER aumenta quando il nodo si allontana dalla stazione base
 2. se BER diventa troppo elevato, commuta ad un tasso trasmissivo più basso ma con BER inferiore
- **Gestione dell'energia:** ogni nodo alterna fra stati "awake" e "sleep" (non riceve e non invia)

Nodo ad AP: comunica che rimarrà inattivo fino al prossimo bacon frame (circa 100ms):

 - AP sa che non deve trasmettere frame a questo nodo
 - il nodo si riattiva prima del successivo bacon frame

frame bacon: contiene la lista dei nodi i cui frame sono stati memorizzati dall'AP, e i nodi "risvegliati" richiederanno all'AP di inviarli.

Un nodo che non ha frame da inviare o da ricevere può rimanere inattivo il 99% del tempo, con un conseguente significativo risparmio di energia

14.2.2 Bluetooth: 802.15.1

Bluetooth opera in un range molto piccolo (circa 10 metri), a bassa potenza e basso consumo. Per questo motivo, i networks Bluetooth sono considerati **WPANs** (*Wireless Personal Area Network*).

- simula un cavo a bassa frequenza
- trasmissione fino a 4 Mbps
- ad hoc: nessuna infrastruttura
- *master/slave* (detta anche **piconet**);

- Slave attivi (max 7) richiedono al master permesso per inviare
- master accorda i permessi
- fino a 255 slave inattivi
- il maste attiva/disattiva slave

Tecnologie utilizzate:

- **TDM**: 625 microsecondi a slot
- randomized backoff
- polling
- error detection and correction
- trasmissione affidabile via ACKs e NACKs
- **FHSS** (*frequency-hopping spread spectrum*): in ogni slot, si può trasmettere in uno tra 79 canali di frequenza, cambiati in modo pseudo-casuale da uno slot all'altro

Capitolo 15

Reti Cellulari 4G e 5G

Come abbiamo visto, gli APs (Access Point) riescono a coprire un'area molto ristretta ed un host non riesce sicuramente ad associarsi con tutti gli APs che incontra mentre è in mobilità in quanto non c'è una gestione automatica del passaggio tra i vari APs. Di conseguenza, WiFi non basta per gli utenti in movimento.

Invece, le **reti cellulari** garantiscono una copertura molto più ampia del territorio, si sono affermate come la principale tecnologia per garantire un accesso diffuso alla rete con accesso alla banda larga.

L'evoluzione delle reti cellulari:

1. 0G: radiotelefoni mobili dalle dimensioni di una valigetta
2. 1G: telefonia cellulare analogica
3. 2G: telefonia cellulare digitale
4. 3G: telefonia cellulare digitale ad alta velocità
5. LTE (4G): telefonia vocale, dati e multimediale basata su IP "sempre e ovunque" a velocità di trasmissione dati elevate
6. 5G: tassi nell'ordine dei Gbps, bassa latenza ed accesso illimitato. Base per applicazioni di realtà aumentata, telemedicina.

Architettura di una rete cellulare

Una rete cellulare è costituita da diverse **celle** *collegate* ad un'infrastruttura di rete. Infatti il termine "*cellulare*" si riferisce al fatto che le regioni sono coperte da *rete cellulare* sono divise in un numero di aree di copertura geografica, chiamate **celle**.

- Ogni cella contiene una **stazione base** che riceve il segnale e trasmette il segnale che riceve da *diversi utenti mobili*
 - La comunicazione nella cella è wireless
 - La comunicazione tra le base station è wired
- L'area di copertura della cella dipende da:
 - Potenza di trasmissione della stazione base e dei dispositivi
 - Presenza di ostacoli

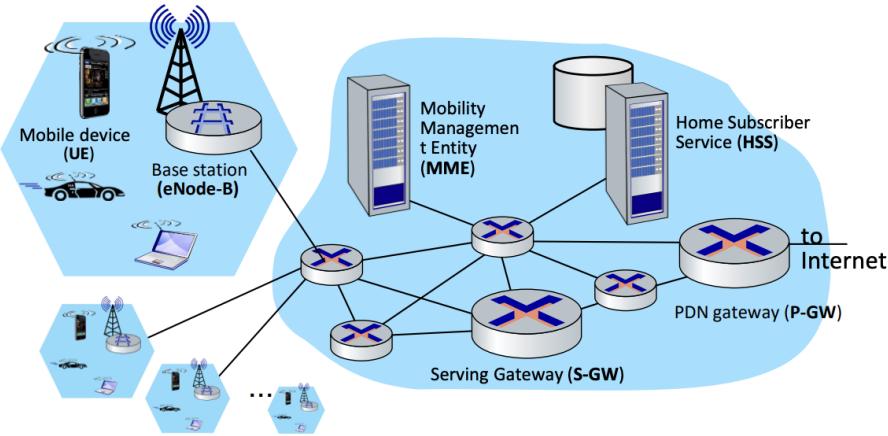


Figura 15.1: Architettura di una Rete Cellulare

- Posizionamento della stazione base

Elementi dell'architettura 4G LTE:

- **Dispositivo mobile:** smartphone, tablet, laptop, IoT.

Il dispositivo mobile in generale implementa tutti i 5 strati dei protocolli di Internet. Sono dotati di indirizzo IP e MAC ed è implementato lo stack TCP/IP.

- I dispositivi mobile sono anche dotati di un indirizzo fisico a 64 bit: **International Mobile Subscriber Identity (IMSI)**, memorizzato sulla scheda **SIM (Subscriber Identity Module)**

L'associazione alla stazione base e la modalità "sleep" sono *simili a WiFi*.

Nel gergo LTE, il dispositivo mobile si chiama *User Equipment (UE)*

- **Stazione Base (BS):** il punto di contatto fra il dispositivo mobile e l'infrastruttura di rete.

1. garantisce le risorse wireless radio della cella
2. gestisce i dispositivi mobili presenti nella cella
3. coordina l'autenticazione dei dispositivi mobili con altri elementi

Simile all'AP di WiFi, ma:

1. ha un ruolo attivo nella mobilità
2. si coordina con stazioni base vicine per ottimizzare l'uso delle frequenze radio

Nel gergo LTE, la base station si chiama *eNode-B*

- **Home Subscriber Service (HSS):** un database che

- opera nel piano di controllo
- memorizza informazioni sui dispositivi mobili per cui la rete dell'HSS è la loro rete di appartenenza (*home network*)
- interagisce con l'*MME* per l'autenticazione dei dispositivi

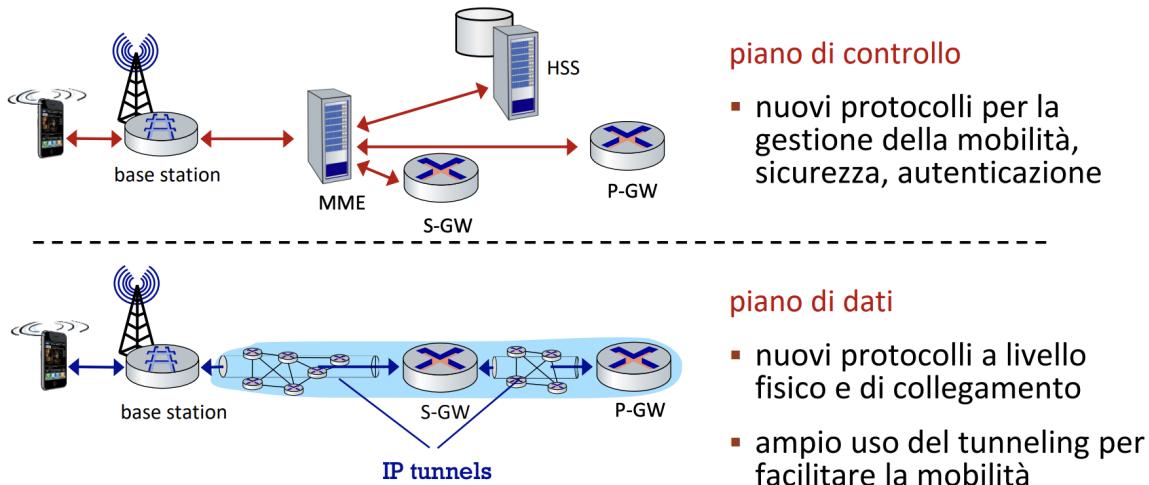


Figura 15.2: LTE: piano dati e piano di controllo

- **Serving Gateway (S-GW), Packet Data Network Gateway (P-GW):** sono dei router che su trovano sul cammino tra il dispositivo mobile ed Internet, possono coincidere sullo stesso nodo
 - **P-GW:** *punto di passaggio* tra la rete cellulare e Internet, *fornisce servizi NAT* e *nasconde al mondo* la *mobilità* dei dispositivi mobili.
Il *PDN Gateway* è l'ultimo elemento LTE che contiene datagrammi che provengono da un device mobile prima di connettersi al resto di Internet.
 - **Mobility Management Entity (MME):** un elemento importante del *piano di controllo*. Insieme all'*HSS* gioca un ruolo fondamentale nell'autentificazione dei dispositivi che vogliono connettersi alla rete.
Inoltre, realizza i *tunnels* sul data path tra i dispositivi mobili e il *PDN Gateway router*, e mantiene le informazioni sulla posizione delle celle dei dispositivi mobili attivi.
- Riassunto:*
- **Autenticazione dei dispositivi** in coordinazione con l'*HSS*
 - **gestione** dei dispositivi mobili in coordinazione con *BSs*:
 - * handover fra celle
 - * cell location tracking
 - gestione dispositivi mobili in "sleep" mode (*paging*)
 - path (tunneling) setup da dispositivi mobili a P-GW

Funzionalità principali dell'MME:

- **Path Setup:** il data path dal device mobile al gateway router dell'operatore mobile proprio (P-GW, Packet Data Network Gateway - PDN Gateway), consiste in:
 1. *primo salto* tra il device e la stazione base ed una concatenamento IP tra la stazione base e il *Serving Gateway* (S-GW) tramite *MME*
 2. ed un *tunnelling* tra Il *Serving Gateway* e il *PDN Gateway*.

LTE Element	Description	Similar WLAN function(s)
Mobile device (UE: User equipment)	End user's IP-capable wireless/mobile device (e.g., smartphone, tablet, laptop)	Host, end-system
Base Station (eNode-B)	Network side of wireless access link into LTE network	Access point (AP), although the LTE base station performs many functions not found in WLANs
The Mobility Management Entity (MME)	Coordinator for mobile device services: authentication, mobility management	Access point (AP), although the MME performs many functions not found in WLANs
Home Subscriber Server (HSS)	Located in a mobile device's home network, providing authentication, access privileges in home and visited networks	No WLAN equivalent
Serving Gateway (S-GW), PDN-Gateway (P-GW)	Routers in a cellular carrier's network, coordinating forwarding to outside of the carrier's network	iBGP and eBGP routers in access ISP network
Radio Access Network	Wireless link between mobile device and a base station	802.11 wireless link between mobile and AP

Figura 15.3: Riassunto architettura LTE

Tutti i tunneling sono sotto il controllo di MME ed sono usati per inoltrare i dati e facilitare la mobilità. Quando un device si sposta, deve essere modificato soltanto il tunnel dal device alla *stazione base*, mentre gli altri tunnel rimangono fissi.

- **Cell location tracking:** come i device si spostano tra le celle, le *base stations* informeranno il MME con la nuova posizione del dispositivo.

Se il dispositivo è in modalità "sleep" ma si sposta nel frattempo, la *base station* non sarà più in grado di rintracciarlo, quindi sarà compito dell'MME a localizzare il dispositivo per il *wakeup*, attraverso un processo noto come **paging**.

- **Similiarità con la rete Internet Cablata:**

1. una rete cellulare globare: una rete di reti
2. diffusione dei protocolli già visti: HTTP, DNS, TCP, UDP, IP, NAT, SDN, separazione di piano di dati e di controllo, Ethernet, tunneling
3. interconnesse alla rete Internet cablata

- **Differenze con la rete Internet cablata:**

1. differenti livelli di collegamento
2. mobilità come servizio di I classe
3. "identità" dell'utente (attraverso SIM card)
4. modello di business: i clienti si rivolgono ad un provider di telefonia:
 - forte nozione di "home network" vs. "roaming" su reti visitate
 - accesso globale, con autenticazione all'infrastruttura di rete

15.1 Stack dei protocolli LTE

Essendo che l'architettura 4G LTE si basa completamente sull'**IP** (*all-IP architecture*), di conseguenza implementa tutti i livelli dell'architettura TCP/IP, quindi ci concentriamo principalmente sul livello di collegamento e fisico.

Il livello di collegamento LTE sul *canale radio utente-stazione base* deve tener conto delle specificità del tipo di collegamento, diviso in tre sottolivelli:

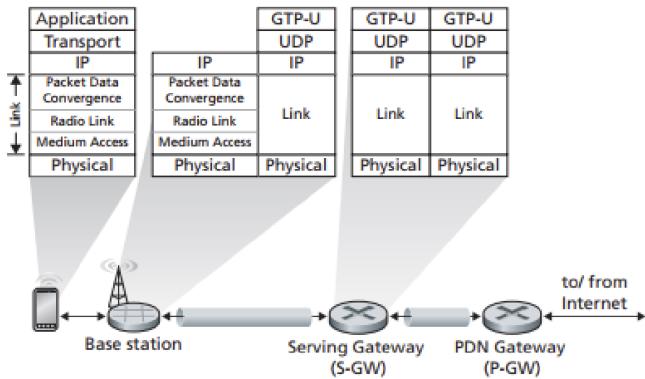


Figura 15.4: LTE data-plane protocol stacks

1. **Packet Data Convergence Protocol (PDCP)**: il sottolivello del livello di collegamento al di sotto di IP.

- compressione dell'intestazione IP per diminuire il numero di bits inviati tramite una connessione wireless
- cifratura e decriptatura del datagramma IP con chiavi concordate tra il dispositivo LTE mobile e MME (Mobility Management Entity) quando il dispositivo si collega la prima volta alla rete

2. **Radio Link Control (RLC)**: due importanti funzioni

- frammentazione/riassembaggio di datagram IP per messaggi che sono troppo grandi per l'invio tramite i protocolli dei sottolivelli del collegamento
- trasmissione affidabile a livello di collegamento (*protocolli ARQ*) basato su ACK/NACK

3. **Medium Access Control (MAC)**:

- permette la schedulazione della trasmissione tramite la richiesta ed uso di slot per la trasmissione radio
- ulteriori funzioni di rilevamento/correzione errori

Tunnelling nei protocolli LTE

Ogni tunnel tra due endpoint ha un identificatore univoco **TEID** (*Tunnel Endpoint IDentifier*). Quando la stazione base riceve datagrammi da un dispositivo mobile, li incapsula usando il protocollo **GPRS**, includendo il *TEID*, ed inviando segmenti UDP al *Serving Gateway* ed ad altri endpoint del tunnel.

Dal lato del ricevitore, la stazione base estrae il datagramma IP del device mobile dal datagramma UDP passato per il tunnel, e invia questo datagramma IP attraverso il salto wireless verso il dispositivo mobile.

Riassunto Tunnelling:

- Datagramma mobile incapsulato usando il *GPRS Tunnelling Protocol (GTP)* e inviato tramite un datagramma UDP a *S-GW*
- *S-GW* reinvia tramite tunnels datagrammi a *P-GW*
- supporto alla mobilità: solo gli estremi della base station cambiano quando gli utenti cambiano BS

LTE Radio Access Network

LTE usa una combinazione di (*FDM*) divisione in frequenza e (*TDM*) nel tempo sul canale *wireless*, noto come **OFDM** (*Orthogonal Frequency Division Multiplexing - Ortogonalni implica minima interferenza tra i canali*).

Ogni dispositivo mobile attivo è allocato ad uno o più slot da 0.5ms e ad una o più frequenze:

- algoritmi di scheduling per gli host non sono standardizzati, dipendono dall'operatore
- si riescono ad avere velocità nell'ordine di 100Mbps per dispositivi mobili

Funzioni LTE aggiuntive: Network Attachment and Power Management

Associazione Network: il modo in cui un dispositivo mobile si associa per la prima volta alla rete.

Il processo si divide in *tre fasi*:

1. **Associazione alla Stazione Base:** la prima fase è simile a come succedeva in 802.11 WiFi, Ethernet, ma un po' diverso.

- (a) BS invia in broadcast segnali multipli di sincronizzazione ogni 5ms su ogni banda di frequenza
- (b) il dispositivo mobile trova il *primo* segnale di sincronizzazione e rimane sulla stessa frequenza fin quando non individua un *secondo* segnale di sincronizzazione sulla stessa frequenza.

Dalle informazioni ottenute dal secondo segnale, il dispositivo mobile potrà:

- ottenere informazioni aggiuntive come la banda del canale, configurazioni del canale e informazioni sull'operatore di rete su quella BS
 - entrare in contatto con multiple BSs
 - (c) Grazie alle informazioni ricevute, il disp. mobile sceglie la BS con cui associarsi, con priorità alla home net
 - (d) ulteriori passi sono necessari per l'autentificazione, set up del piano di dati etc..
2. *Mutua Autentificazione:* la Base Station contatta il MME locale per effettuare l'autentificazione mutua. Controlla il codice IMSI del dispositivo e quando sarà pronto, si potrà configurare il data path *Mobile-device-to-PDN-gateway*
 3. *Mobile-device-to-PDN-gateway:* l'MME contatta il PDN gateway, il quale provvede anche un NAT address al device, contatta anche il Serving Gateway, e la Base Station per stabilire i due tunnels.

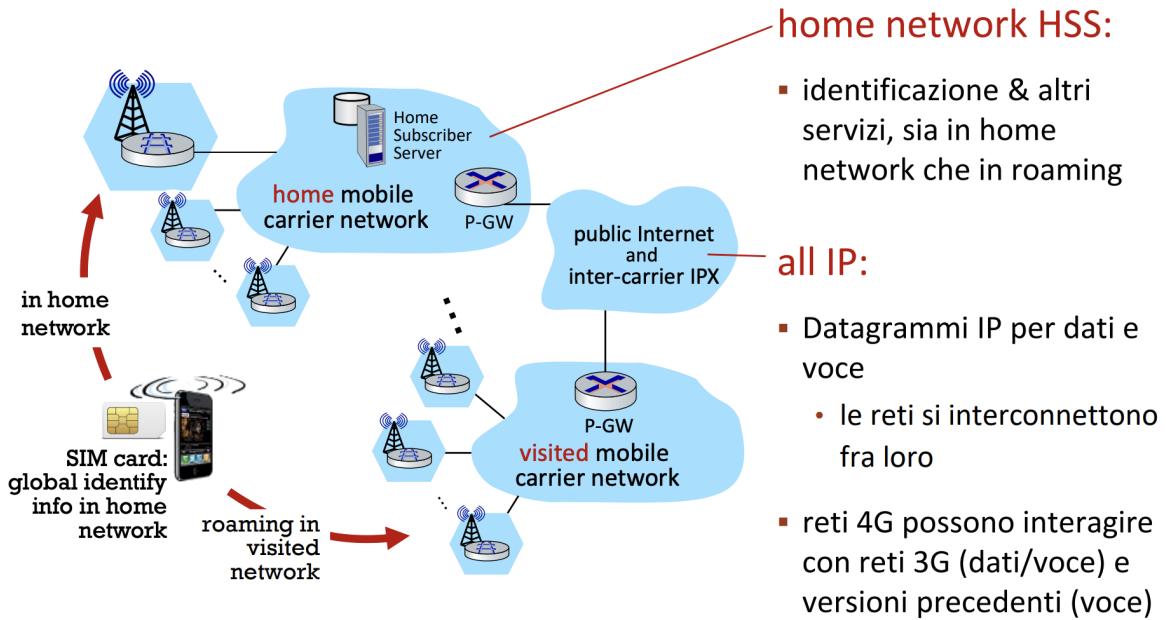
Una volta che il path è completo, il disp. mobile potrà inviare/ricevere datagrammi IP attraverso la Stazione Base.

Gestione dell'energia: un disp. mobile si può mettere in modalità "sleep" quando non è impegnato ad inviare/ricevere dati, simile a WiFi e Bluetooth.

Due modalità di sleep:

- *light sleep:* dopo 100ms di inattività e ri risveglia periodicamente, ogni 100ms, per verificare se ci sono frames in arrivo
- *deep sleep:* dopo 5-10s di inattività

Il disp. mobile potrebbe cambiare cella mentre "dorme", successivamente avrà bisogno di effettuare una nuova associazione (*paging message broadcast dall'MME alle stazioni base vicine alla stazione base dove il dispositivo era precedentemente associato*).



Rete Cellulare Globale

Organizzazione della rete 4G come una rete di reti

5G Cellular Networks

- **Obiettivo:** crescita di 10x in *peak bitrate*, decrescita del 10x di latenza, crescita di 100x in capacità di traffico rispetto a 4G
- **5G NR (New Radio):**
 - due bande di frequenza
 - non compatibile con 4G a livello fisico wireless
 - antenne MIMO (multiple input, multiple output)
- le frequenze FR2 sono "millimiter wave frequency": tassi di trasmissione più alti, ma su distanze più piccole.
 - pico-cells (diametri delle celle): 10-100m, sviluppo denso di nuove BSs
- Architettura della rete 5G simile a 4G
- **Alcune Novità:**
 - **USP (User-Plane Function):** completa separazione dei piani di controllo e dati
 - MME suddiviso in AMF e SMF:
 - * **AMF (Access and Mobility Management Function):** gestione di connessioni di mobilità
 - * **SMF (Session Management Function):** interazione col piano di dati, indirizzamento IP, funzioni DHCP

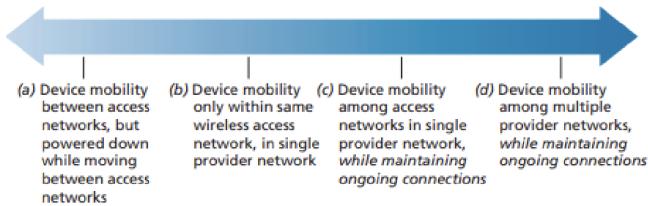


Figura 15.5: Vari livelli di mobilità, dal punto di vista del livello di rete

15.2 Gestione della Mobilità degli Utenti: dal punto di vista del livello di Rete

Vari livelli di mobilità:

1. L'utente è spento ogni volta che cambia rete, assomiglia ad un dispositivo fisso che si sposta. In questo caso funziona come per i dispositivi non mobili
2. il dispositivo è mobile ma rimane nella stessa sottorete. Non è mobile dal punto di vista del livello di rete, inoltre se non cambia nemmeno AP o LTE base station, non è mobile neanche dal punto di vista del livello di collegamento
3. Dal punto di vista del livello di rete, ci interessiamo a questo caso: quando un dispositivo cambia la rete d'accesso mentre continua ad inviare e ricevere datagramma IP, e mentre mantiene una connessione TCP al livello di trasporto.

In questo caso la rete ha bisogno di provvedere un **handover** - un trasferimento di responsabilità per inviare datagrammi da/per un'AP o una base station al dispositivo mobile - quando il dispositivo si muove lungo diverse WLANS o tra celle LTE.

Se l'*handover* occorre tra gli accessi alla rete sotto lo stesso operatore allora viene gestito totalmente dall'operatore stesso.

4. quando un dispositivo si sposta in diverse reti ed il provider non riesce a gestirlo da solo perché si sposta tra le celle di diversi operatori.

Sappiamo che:

- ogni iscritto alla rete mobile ha una "*home network*" (rete domestica, o rete di appartenenza) rispetto al proprio operatore mobile.
- l'HSS (Home Subscriber Service) salva le informazioni di ogni iscritto, includendo il relativo ID globale univoco - IMSI - (*Mobile Subscriber Identity*), informazioni sui servizi a cui ha accesso, le chiavi per la crittografia.

Terminologia:

- **home network** (rete domestica, rete d'appartenenza): indirizzo IP fisso dell'appartamento in cui risiede un nodo mobile
- **indirizzo permanente**: indirizzo della rete domestica, può *sempre* essere usato per raggiungere il dispositivo mobile
- **home agent** (agente domestico): entità che gestisce le funzioni di mobilità per conto del dispositivo mobile.

- **rete visitata** (rete ospitante in cui un dispositivo mobile può trovarsi mentre si sposta): rete in cui risiede attualmente il dispositivo mobile. Il dispositivo è in *roaming* sulla rete visitata
- **agente ospitante**: entità della rete visitata che gestisce la mobilità all'interno di questa
- **indirizzo permanente all'interno della rete visitata**: indirizzo della home network rimane invariato all'interno della rete visitata
- **care-of-address COA**: indirizzo presso la rete visitata
- **corrispondente**: entità che desidera comunicare con il nodo mobile

Un possibile approccio per gestire la mobilità potrebbe essere: **lasciare che la gestiscano i router**. I router rendono pubblico l'indirizzo permanente dei nodi-mobili-residenti mediante il solito scambio di tabelle di inoltro. Ovviamente la soluzione *non è scalabile* per milioni di utenti mobili.

Dunque, **lasciamo che il sistema terminale gestisca la mobilità**: quando un dispositivo mobile visita una rete, è richiesta la coordinazione tra l'*home network* e la *rete visitata*.

L'instradamento dei messaggi può essere:

- *instradamento indiretto*: la comunicazione fra il corrispondente ed il nodo mobile avviene attraverso l'agente domestico e viene poi inoltrata al remoto
- *instradamento indiretto*: il corrispondente ottiene l'indirizzo presso la rete ospitante e potrà poi quindi comunicare direttamente con il nodo mobile

Il dispositivo mobile appena si connette ad una *rete ospitante* procederà alla **registrazione**:

1. Il disp. mobile contatta l'agente ospitante al suo ingresso nella rete visitata
2. L'agente ospitante contatta l'*home network*

In questo modo:

- L'agente ospitante conosce tutti i dati del disp. mobile
- L'agente domestico conosce la localizzazione del disp. mobile

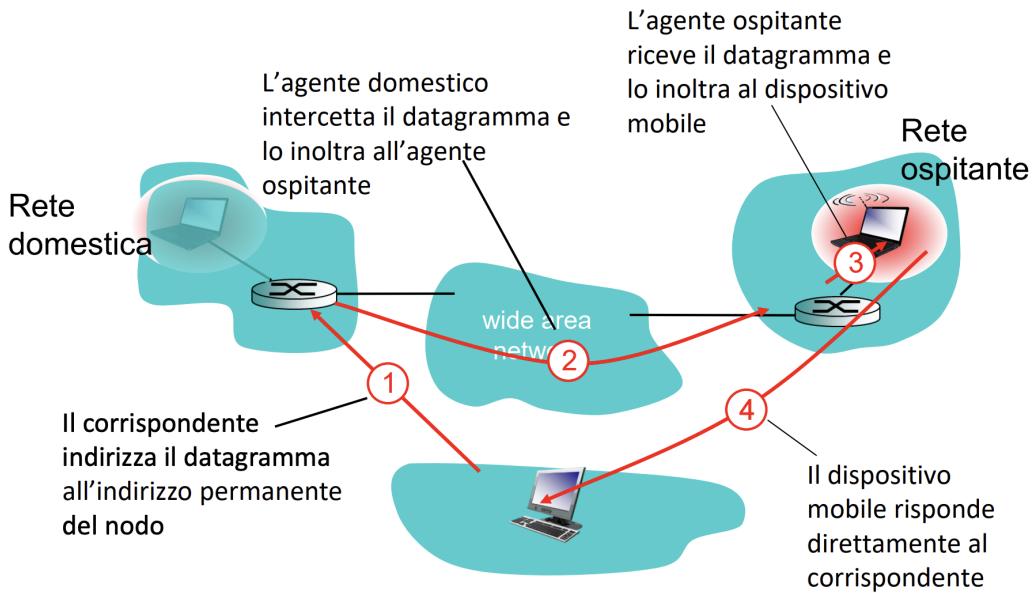
Indirizzamento Indiretto

1. Il *corrispondente* contatta il nodo mobile al suo indirizzo IP permanente. Invia un pacchetto con IP dest: indirizzo permanente
2. L'*home network* intercetta il datagramma, consulta l'HSS, e determina la rete visitata attualmente dal nodo mobile.

A questo punto incapsula il pacchetto ricevuto e lo invia alla *visited network* con IP destinazione: *COA* (core-of-address)

3. Il *visited network* decapsula il datagramma e lo invia ad destinatario originale, il dispositivo mobile

Ora, il dispositivo mobile potrà contattare il corrispondente in due modi:



1. il datagramma potrebbe effettuare il percorso a ritroso
2. potrebbe inviare direttamente il datagramma utilizzando la rete visitata

Problema dell'*indirizzamento indiretto*: *Triangolazione* corrispondente - rete domestica - dispositivo mobile.

Questa *triangolazione* è inefficiente quando il corrispondente ed il dispositivo mobile si trovano all'interno della stessa rete visitata.

*Riassunto, spostamento tra sottoreti distinte con *indirizzamento indiretto*:*

- il disp. mobile si registra presso il nuovo agente ospitante
- il nuovo agente ospitante si registra presso l'agente domestico
- l'agente domestico aggiorna l'indirizzo COA del disp. mobile nell'HSS
- i pacchetti continuano ad essere inoltrati al disp. mobile, ma al nuovo indirizzo COA.

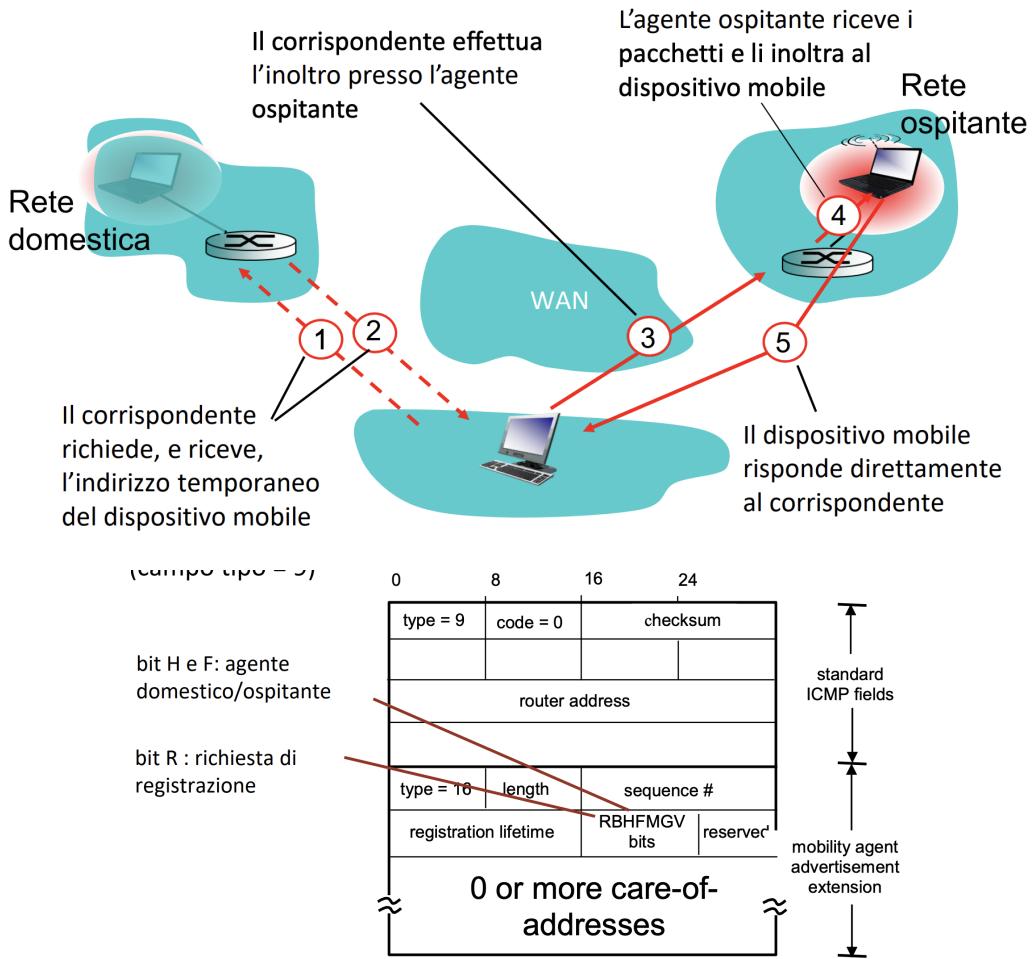
In questo modo spostandosi tra diverse reti, le connessioni in corso possono essere mantenute.

Indirizzamento Diretto

- Risolve il problema della triangolazione
- Però nuovo problema, **non-trasparente al corrispondente**: il corrispondente deve ricevere l'indirizzo COA dall'agente domestico.

Cosa succede se l'utente mobile cambia la rete visitata ?

- L'agente *ospitante d'appoggio*: agente ospitante nella prima rete visitata (a cui è comunicato il nuovo COA dal nuovo agente ospitante)
- i dati vengono sempre inoltrati prima all'agente ospitante d'appoggio
- Quando il dispositivo mobile si sposta in una nuova rete: il nuovo agente ospitante farà in modo di ricevere i dati dal vecchio agente ospitante (**chaining**)



15.2.1 IP Mobile

Versione di IP ideata per il Mobile.

Lo standard è composto da tre parti:

1. Ricerca dell'agente:

- **Avviso all'agente:** un agente (domestico/ospitante) rende noti i suoi servizi inviando periodicamente in broadcast un messaggio ICMP (*campo type = 9*)
- **Richiesta dell'agente:** il nodo mobile invia in broadcast una richiesta ICMP (*campo type = 10*) per ricevere info e l'agente che riceve invierà un avviso in unicast

2. **Registrazione presso l'agente domestico:** la comunicazione dell'indirizzo COA all'agente domestico può avvenire per opera dell'agente ospitante o del nodo mobile. Nel caso di comunicazione da parte dell'agente ospitante:

- (a) Il nodo mobile (dopo aver ricevuto il COA) invia all'agente ospitante un datagramma UDP alla porta 434 contenente il COA scelto, l'indirizzo permanente (*MA*) e indirizzo dell'Agente Domestico (*HA*), tempo di scadenza della registrazione ed identificazione della registrazione a 64 bit
- (b) L'agente ospitante invia un datagramma all'agente domestico (UDP, porta 434) con : COA, MA, HA, formato di incapsulamento richiesto, scadenza di registrazione ed identificazione registrazione

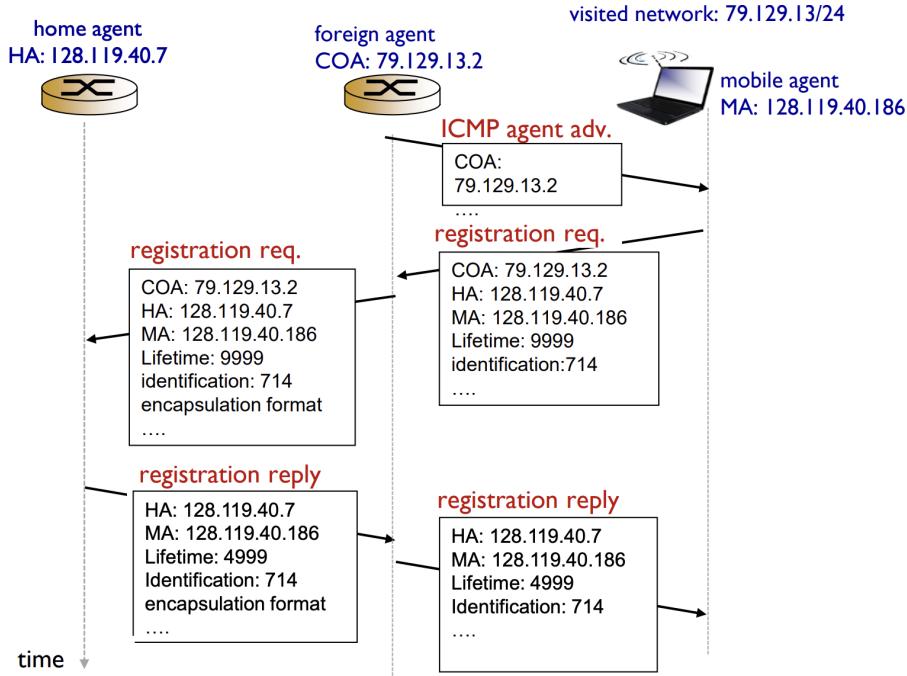


Figura 15.6: Esempio di Registrazione

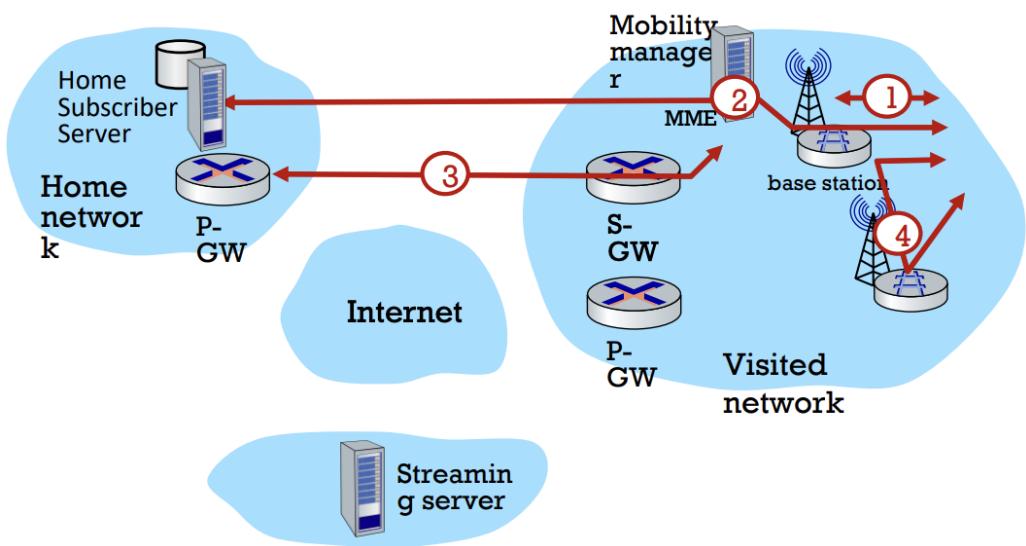
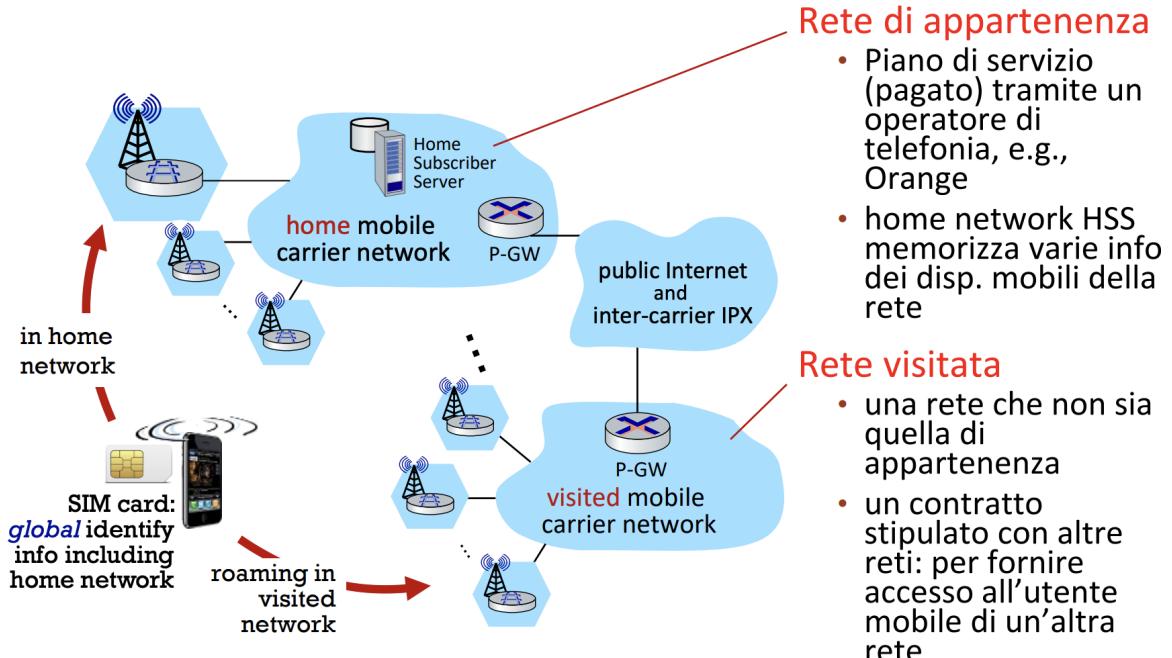
- (c) L'agente domestico invia una risposta di conferma all'agente ospitante (contenente MA, HA, scadeza corrente, identificazione richiesta) e viene inoltrata al nodo mobile dall'agente ospitante.
3. **Instradamento indiretto dei datagrammi**: Mobile IP definisce come i datagrammi devono essere inoltrati ai dispositivi mobili dall'*Home Agent*, includendo regole per l'inoltro di datagrammi, gestione degli errori e varie forme di tunnelling

15.2.2 Mobilità in reti 4G

- associazione con la BS**: Il dispositivo mobile fornisce l'IMSI (*da info su se stesso e la home network*)
- configurazione del piano di controllo**: MME informa HSS della home-network: "Il dispositivo mobile è nella rete visitata"
- configurazione piano di dati**:
 - MME configura tunnels per i dispositivo mobile
 - visited e home networks stabiliscono tunnels da home P-GW a visited S-GW a visited BS.

Sel il dispositivo rimane all'interno della rete ospitante ma cambia BS:

 - il collegamento da P-GW della home network al S-GW della rete visitata rimane fisso
 - cambia il collegamento da S-GW alla nuova BS
 - implementazione del routing indiretto
- mobile handover**: il dispositivo mobile si associa ad un'altra stazione base (BS)

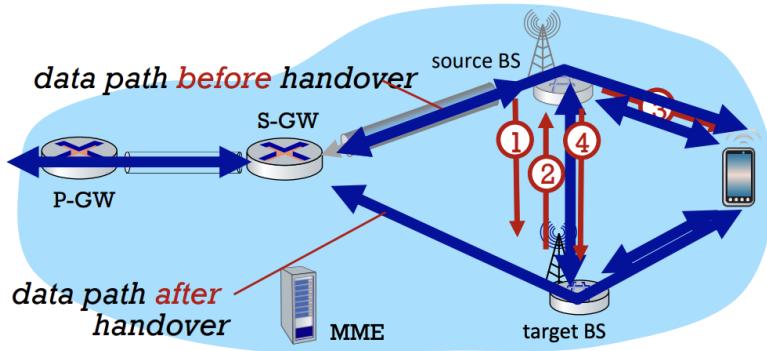


Configurazione del Piano di Controllo - Step 2:

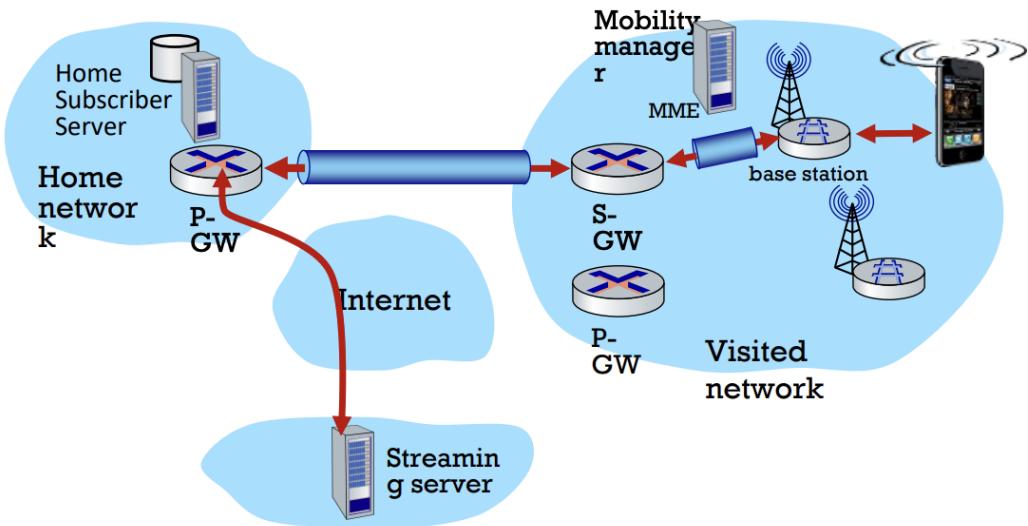
- Il dispositivo mobile comunica con il MME locale attraverso il canale stabilito con BS (*durante la fase di associazione*)
- MME usa le info IMSI per contattare l'HSS della home network:
 - Recupera autenticazione, cifratura, servizi di rete
 - home HSS ora conosce in quale rete visitata si trova il dispositivo
- BS e dispositivo mobile selezionano i parametri per il canale radio (*piano di dati*) fra le due entità

Configurazione del Piano di Dati - Step 3:

- S-GW to BS tunnel:** quando il dispositivo cambia BS, semplicemente cambia Tunnel Endpoint IP

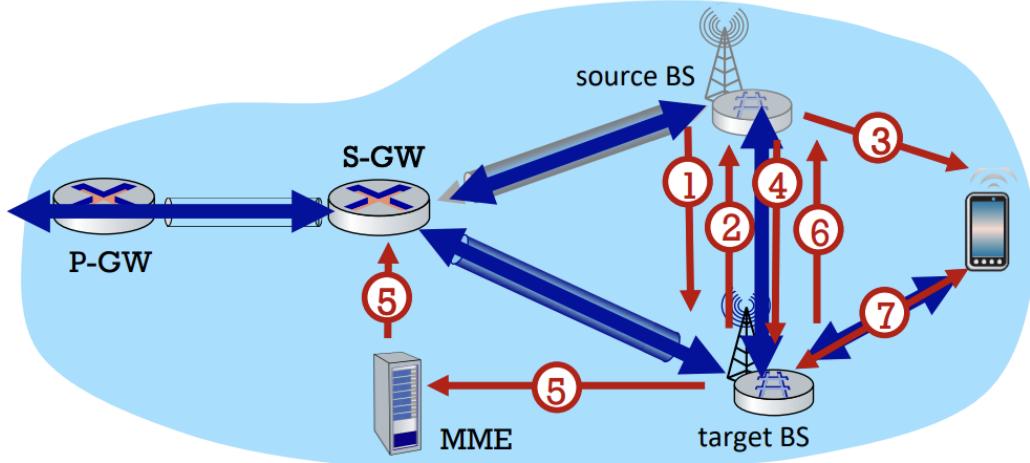


- **S-GW to home P-GW tunnel:** implementazione routing indiretto
- **Tunnelling via GTP (GPRS Tunnelling Protocol):** il datagramma del disp. mobile verso il server streaming è encapsulato usando GTP, dentro UDP, dentro un datagramma



Handover fra BSS nella stessa rete cellulare - Step 4:

1. L'attuale BS (*source*) seleziona la nuova BS (*target*) ed invia un *Handover Request (HR)* messaggio alla futura BS
2. Target BS pre-alloca radio time slots, risponde con un *HR ACK* contenente info per l'accesso del disp. mobile
3. Source BS informa il dispositivo mobile della nuova BS. Il disp. mobile può ora trasmettere tramite la nuova BS - handover sembra completo dal punto di vista del disp. mobile
4. Source BS interrompe l'invio di datagrammi al disp. mobile, e li invia alla nuova BS (*che invia al disp. mobile attraverso canali radio*)
5. Target BS informa MME che è la nuova BS per il disp. mobile, MME instruisce S-GW in modo tale che il nuovo tunnel endpoint sia la nuova BS
6. Target BS invia un ACK a source BS: handover completo, source BS può deallocare le risorse
7. I datagrammi del disp. mobile ora attravessano il nuovo tunnel dalla target BS a S-GW



4G/5G vs Mobile IP

4G/5G

- Rete di appartenenza
- Rete visitata
- Identificatore IMSI
- HHS
- MME
- Base Station (eNode-B)
- Radio Access Network
- Piano Dati: Indirect Routing attraverso la rete di appartenenza, con tunneling fra rete di appartenenza e visitata

IP Mobile

- Rete di appartenenza
- Rete visitata
- Indirizzo IP permanente
- Agente domestico
- Agente ospitante
- Access Point (AP)
- WLAN
- Piano Dati: Indirect Routing attraverso la rete di appartenenza, con tunneling fra reti di appartenenza e visitata