



Università degli Studi di Verona
Dipartimento di Informatica
A.A. 2018-2019

RIASSUNTO DEL CORSO DI “RETI DI CALCOLATORI”

Created by *Davide Zampieri*
Based on slides provided by *prof. Damiano Carra*

❖ [Indice degli argomenti](#)

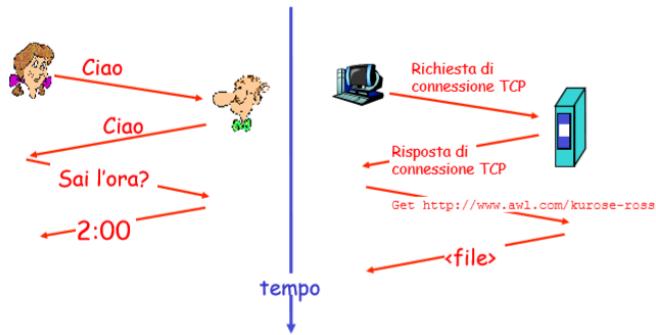
- *Introduzione alle reti (da pagina 3):*
 - Commutazione di circuito e di pacchetto
 - Modelli di riferimento ISO-OSI e TCP-IP
- *Il livello applicazione (da pagina 8):*
 - Il modello client/server
 - World Wide Web (HTTP)
 - File Transfer Protocol (FTP)
 - Posta elettronica (SMTP, POP3 e IMAP)
 - Domain Name Service (DNS)
- *Il livello di trasporto (da pagina 16):*
 - Scopi e servizi
 - Il protocollo TCP (3-way handshake, controllo di flusso, ritrasmissione, controllo di congestione)
 - Il protocollo UDP
- *Il livello di rete (da pagina 23):*
 - Il protocollo IP (formato del pacchetto IP, indirizzi IP, spazio di indirizzamento, CIDR)
 - Il protocollo ICMP
 - Il protocollo DHCP
 - Gli algoritmi di routing (distance vector, link state)
 - Network Address Translation (NAT)
 - IPv6
- *Il livello data link (da pagina 42):*
 - Framing
 - Accesso al canale condiviso
 - Il sottolivello MAC (ALOHA, CSMA, CSMA/CD)
 - Bridge, switch e LAN
 - Risoluzione degli indirizzi (protocollo ARP)
 - Wireless LAN (CSMA/CA)
- *Domande sulla teoria (da pagina 53)*
- *Procedimenti per risolvere gli esercizi (da pagina 58)*

INTRODUZIONE

Cos'è un protocollo?

Un protocollo è un insieme di regole che definiscono il formato e l'ordine dei messaggi scambiati tra due o più entità in comunicazione.

Protocollo umano e protocollo di rete



Ai confini della rete.

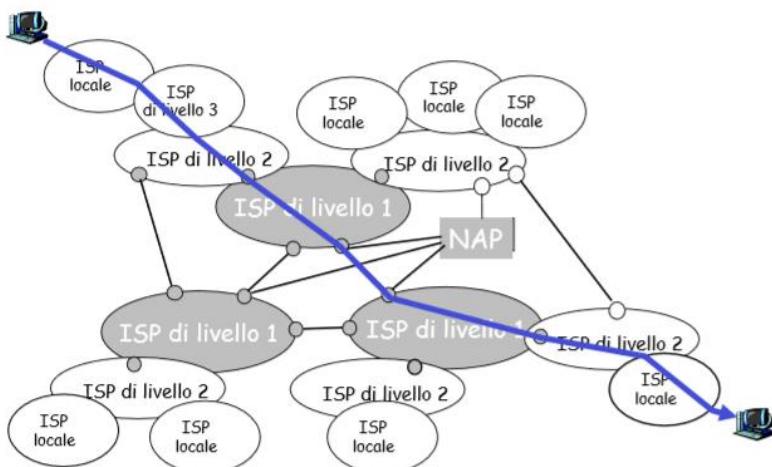
- *Sistemi terminali (host)*: qualunque dispositivo che fa girare programmi applicativi.
- *Architettura client/server*: l'host client chiede e riceve dal server un servizio in esecuzione su un altro terminale (es. browser, e-mail).
- *Architettura peer to peer*: uso limitato (di solito inesistente) di server dedicati (es. Skype, Torrent).

Reti d'accesso e mezzi fisici.

- Esistono vari modi per collegare sistemi terminali e router esterni: reti di accesso residenziale; reti di accesso aziendale; reti di accesso mobile. Nel caso di accesso wireless, una rete condivisa (e senza fili) collega i sistemi terminali al router.
- Il router gestisce i pacchetti tra la rete interna e la rete esterna.

La rete delle reti.

Internet ha una struttura gerarchica. Al centro vi si trovano gli *ISP (Internet Service Provider)* di *livello 1*, che comunicano tra di loro come "pari" e hanno una copertura internazionale. Poi ci sono ISP più piccoli (nazionali o distrettuali) detti *ISP di livello 2*, i quali pagano l'ISP di livello 1 al fine di ottenere connettività per il resto della rete. Infine, ci sono gli *ISP di livello 3* e gli *ISP locali*, che sono quelli che si trovano più vicino ai terminali.



Il nucleo della rete.

I dati possono essere trasferiti attraverso la rete tramite...

- *Commutazione di circuito*: i dati hanno un circuito dedicato per l'intera durata della sessione; le risorse punto-punto sono riservate alla "chiamata"; le risorse di rete sono suddivise in pezzi.
- *Commutazione di pacchetto*: i pacchetti condividono le risorse di rete; ciascun pacchetto utilizza completamente il canale; le risorse vengono usate a seconda delle necessità; ci possono essere problemi dovuti all'eccessiva congestione come ritardi e perdita di pacchetti.

Store and forward.

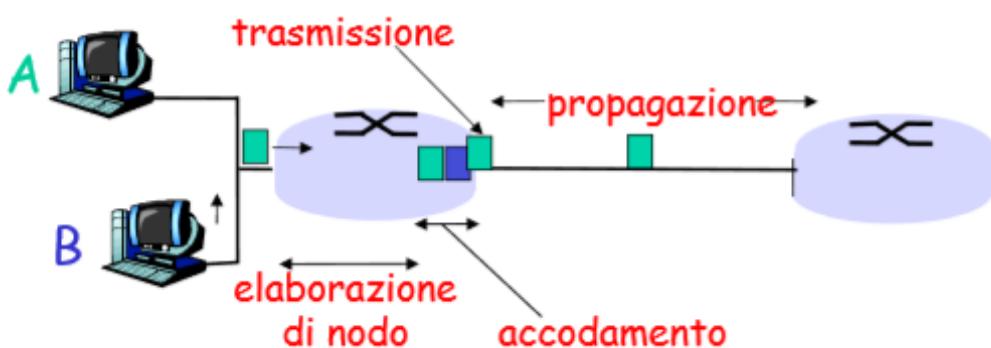
L'intero pacchetto deve arrivare al router prima che questo lo trasmetta sul link successivo. Per fare ciò, il pacchetto viene suddiviso in due parti. La prima parte del pacchetto viene inviata subito; la seconda, invece, viene inviata solamente quando la prima ha raggiunto il successivo router/host. In questo modo, teoricamente, il ritardo diventa nullo.

Come si verificano ritardi e perdite?

- Un *ritardo* si verifica quando i pacchetti in arrivo non vengono smaltiti in fretta, e quindi si accodano nel buffer del router.
- Una *perdita* si verifica quando non ci sono buffer liberi, facendo sì che i pacchetti in arrivo vengano scartati.

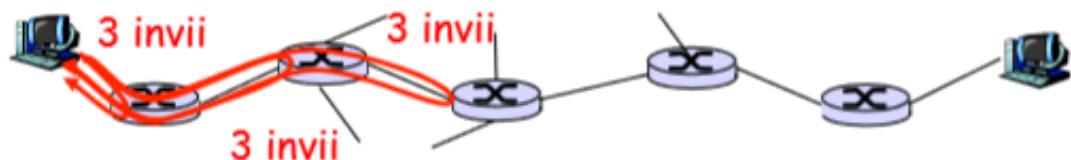
Cause di ritardo per i pacchetti.

- *Ritardo di elaborazione del nodo*: causato dal controllo di errori sui bit e dalla determinazione del canale di uscita.
- *Ritardo di accodamento*: causato dall'attesa di trasmissione e dal livello di congestione del router.
- *Ritardo di trasmissione*: dato dal rapporto tra la lunghezza del pacchetto (L) e la frequenza di trasmissione del collegamento (R).
- *Ritardo di propagazione*: dato dal rapporto tra la lunghezza del collegamento fisico (d) e la velocità di propagazione del collegamento (s).



Traceroute.

È un programma diagnostico che fornisce una misura del ritardo dalla sorgente ai router lungo il percorso.

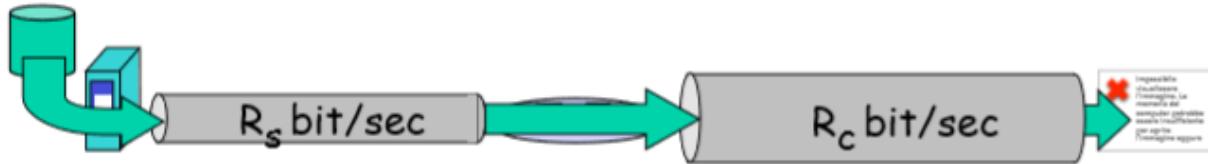


Perdita di pacchetti.

Siccome il buffer (coda) del router ha capacità finita, quando un pacchetto trova la coda piena viene scartato. A questo punto il pacchetto perso può essere ritrasmesso dal nodo precedente (o dal terminale che lo ha generato) o non essere ritrasmesso affatto.

Throughput.

È la frequenza (bit/unità di tempo) alla quale i bit vengono trasferiti tra mittente e ricevente. Il throughput medio end to end è vincolato dal collo di bottiglia, ovvero il collegamento che ha la frequenza minore (R_s).



Sicurezza di rete.

Nel campo della sicurezza di rete, si trovano diversi tipi di attacchi:

- *Trojan*, che è una parte nascosta di un software utile.
- *Virus*, che proviene da un oggetto ricevuto (e mandato in esecuzione) ed è auto-replicante, cioè si propaga da solo ad altri host e utenti.
- *Worm*, che proviene da un oggetto passivamente ricevuto che si auto-esegue ed è auto-replicante.
- *DoS (Denial of Service)*, quando gli attaccanti fanno sì che le risorse (server, ampiezza di banda) non siano più disponibili al traffico legittimo sovraccaricandole di traffico artefatto.
- *Analisi dei pacchetti (packet sniffing)*, quando un'interfaccia di rete legge e registra tutti i pacchetti che l'attraversano.
- *IP spoofing*, quando si inviano pacchetti con indirizzo sorgente falso.
- *Record-and-playback*, quando i dati sensibili vengono sniffati per poi essere utilizzati in un secondo tempo.

MODELLO A STRATI

Modello di comunicazione.

Quando un sistema vuole scambiare informazioni con un altro sistema, nasce il problema della *comunicazione*, che racchiude in sé due sotto-problemi: il *linguaggio utilizzato* e la *modalità di scambio*. Nel nostro *modello* chiameremo *comunicazione logica* il linguaggio e *comunicazione fisica* la modalità di scambio. A seconda del mezzo (canale di comunicazione) utilizzato, si hanno caratteristiche di comunicazione fisica differenti, mentre la comunicazione logica rimane inalterata.

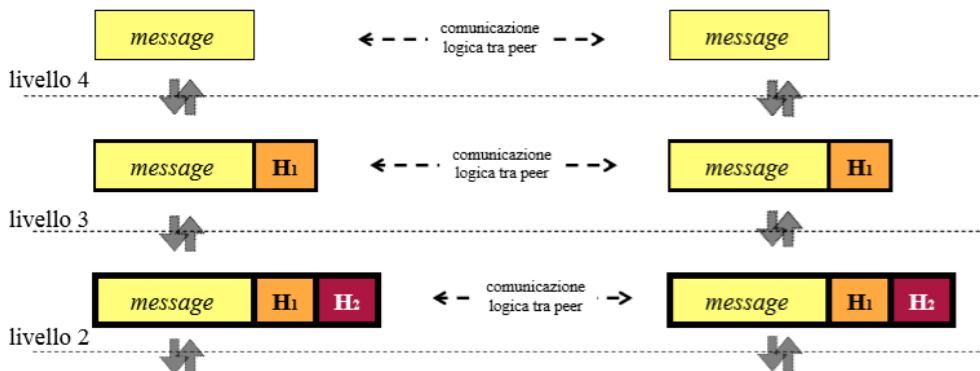
Alcune problematiche associate alla comunicazione.

- *Logica*: necessità di avere regole per lo scambio dell'informazione; presenza di varie modalità di trasferimento (simplex, half duplex, ...).
- *Fisica*: indirizzamento; controllo degli errori; affidabilità; multiplazione; controllo di flusso.

Approccio a livelli.

È un approccio *divide et impera*, perché l'informazione passa da una *catena di montaggio* in cui essa viene trasformata in modo da poter essere spedita. Dall'altra parte ci sarà una catena di montaggio inversa che restituisce l'informazione. Tradizionalmente questa catena di montaggio viene rappresentata in verticale, come una *serie di livelli*, in cui:

- Ogni livello interagisce solo con i due adiacenti (comunicazione fisica).
- Ogni livello colloquia con il suo omologo (peer) di un'altra macchina (comunicazione logica).



Perché il modello a strati?

Il modello a strati risolve le problematiche associate alla comunicazione fisica definendo, per ogni livello, i *servizi che deve offrire*, le *funzioni che deve svolgere* e le *primitive che deve mettere a disposizione*. Tuttavia, non viene definito il modo in cui implementare i suddetti servizi o funzioni. Il modello a strati rientra nella concezione di divide et impera (catena di montaggio).

Architettura di rete.

Le problematiche associate alla comunicazione logica vengono risolte attraverso la definizione di *protocolli*, ovvero insiemi di regole che sovraintendono al colloquio tra entità dello stesso livello. Lo *stack protocollore* è l'insieme dei protocolli di ciascun livello. L'*architettura di rete* è l'insieme dei livelli e dei rispettivi protocolli.

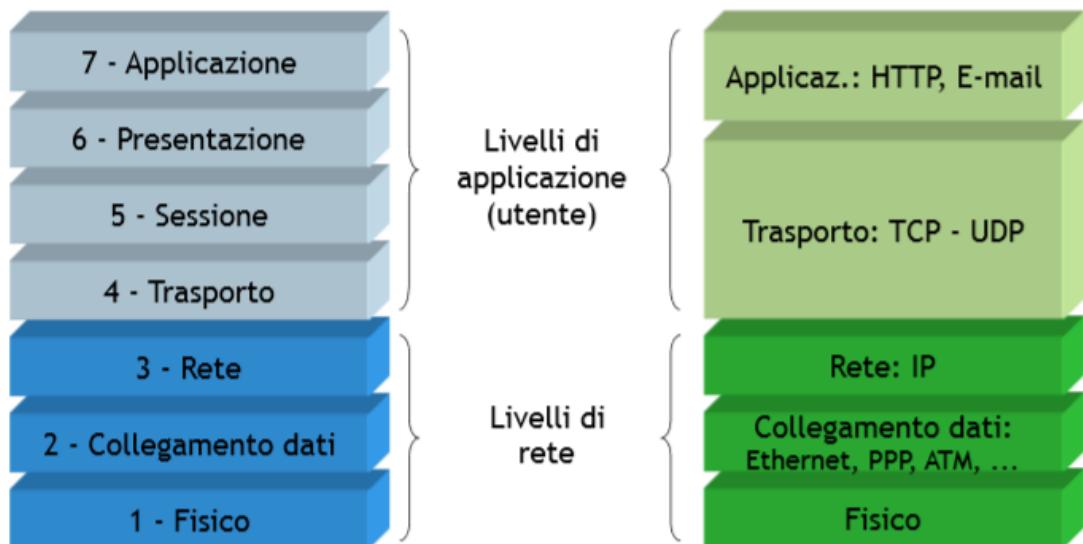
Modello ISO-OSI.

È stato il primo passo nella definizione di un'architettura di rete completa e aperta (non proprietaria). Essendo un modello che *definisce funzionalità raggruppate in livelli*, ma *non definisce in modo formale protocolli e servizi* da usare nei vari livelli, non si può considerare un'architettura di rete vera e propria.

Rimane quindi un *modello teorico utilizzato come modello di riferimento*, in quanto è stato sviluppato troppo tardi (alla pubblicazione di ISO-OSI, Internet era già una realtà).

Principi del modello ISO-OSI.

- Ci deve essere *un livello per ogni grado di astrazione*.
- Ogni livello deve eseguire *funzioni ben definite*.
- Le interfacce tra i livelli devono essere definite in modo da *minimizzare l'informazione scambiata*.
- Il *numero di livelli* deve essere *sufficientemente grande* (in modo che le stesse funzioni non siano separate in più livelli) e *sufficientemente piccolo* (in modo che l'architettura non abbia funzionalità ridondanti).



Modello ISO-OSI (teorico) vs modello TCP-IP (reale)

IL LIVELLO APPLICAZIONE

❖ Principi delle applicazioni di rete

Architetture delle applicazioni di rete.

- *Client/server*: il client comunica con il server e può avere indirizzi IP dinamici; il server è sempre attivo e ha un indirizzo IP fisso.
- *Peer-to-peer*: non c'è un server sempre attivo; coppie arbitrarie di host comunicano direttamente tra di loro; è facilmente scalabile; è difficile da gestire.
- *Ibrida*: utilizzata ad esempio da Skype (un server centralizzato ricerca l'indirizzo della parte remota, mentre la connessione client-client è diretta).

Processi e socket.

Un *processo* è un programma in esecuzione su di un host, e può essere *client* (se dà inizio alla comunicazione) oppure *server* (se è in attesa di essere contattato). Una *socket*, invece, è una porta da cui il processo invia o riceve messaggi. Processi su host differenti comunicano tra di loro attraverso lo *scambio di messaggi*.

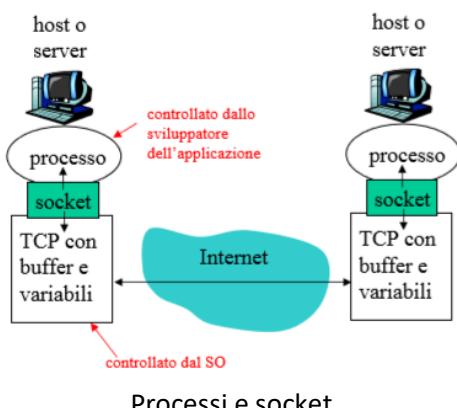
Processi di indirizzamento.

Per identificare l'host su cui è in esecuzione un processo non è sufficiente conoscerne l'indirizzo IP, in quanto sullo stesso *host* possono essere in esecuzione *molte processi*. L'*identificatore* deve quindi comprendere sia l'*indirizzo IP* che il *numero di porta* associato al processo in esecuzione su un host (ad esempio 80 per i server HTTP e 25 per i server mail).

Requisiti del servizio di trasporto.

Un'applicazione può richiedere i seguenti servizi di trasporto:

- *Tolleranza alla perdita di dati*.
- *Temporizzazione*, ovvero la richiesta di piccoli ritardi per essere realistica.
- *Throughput*, ovvero la richiesta di ampiezza di banda (minima o variabile).
- *Sicurezza*, ovvero garantire la cifratura e l'integrità dei dati.



Applicazione	Tolleranza alla perdita di dati	Throughput	Sensibilità al tempo
Trasferimento file	No	Variabile	No
Posta elettronica	No	Variabile	No
Documenti Web	No	Variabile	No
Audio/video in tempo reale	Si	Audio: da 5 Kbps a 1 Mbps Video: da 10 Kbps a 5 Mbps	Si, centinaia di ms
Audio/video memorizzati	Si	Come sopra	Si, pochi secondi
Giochi interattivi	Si	Fino a pochi Kbps	Si, centinaia di ms
Messaggistica istantanea	No	Variabile	Si e no

Processi e socket

Requisiti di trasporto di alcune applicazioni comuni

❖ Web e HTTP

Protocollo a livello di applicazione.

Un protocollo a livello di applicazione definisce: i *tipi* di messaggi scambiati, la *sintassi* di tali tipi di messaggio, il *significato delle informazioni* nei campi dei messaggi, le *regole* per determinare quando e come un processo deve *rispondere* ad un messaggio.

Servizi dei protocolli di trasporto Internet.

Nella rete vi sono due principali protocolli di trasporto:

- *TCP*, che prevede un servizio orientato alla connessione e il trasporto affidabile dei dati.
- *UDP*, che è un protocollo leggero e senza fronzoli dotato di un modello di servizio minimalista.

Applicazione	Protocollo a livello applicazione	Protocollo di trasporto sottostante
Posta elettronica	SMTP [RFC 2821]	TCP
Accesso a terminali remoti	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
Trasferimento file	FTP [RFC 959]	TCP
Multimedia in streaming	HTTP (es. YouTube) RTP [RFC 1889]	TCP o UDP
Telefonia Internet	SIP, RTP, proprietario (es. Skype)	Tipicamente UDP

Web.

Una *pagina web* è formata da vari *oggetti* (insieme di file). La maggior parte delle pagine web è formata da un *codice HTML principale* e diversi oggetti referenziati da esso. Tutte le pagine web sono identificate in modo univoco da un *URL*, che contiene il *nome di un host* seguito da “/” e seguito dal *nome del percorso* che identifica la pagina specifica. Il web è un servizio che rispetta l’architettura client/server, in quanto il server web è sempre pronto a ricevere richieste.

Panoramica su HTTP.

HTTP (HyperText Transfer Protocol) è un *protocollo a livello di applicazione del web* di tipo richiesta-risposta. Ciò significa che, ad ogni richiesta del client, il server deve sempre rispondere. In genere il server risponde con gli oggetti richiesti, ma può anche rispondere con una notifica di errore. Inoltre, siccome il server invia messaggi al client senza memorizzare nulla sullo stato del suddetto, HTTP è definibile protocollo senza stato (*stateless*). Infine, HTTP utilizza il TCP come protocollo di trasporto, e si comporta in questo modo:

- Il client inizializza la connessione TCP con il server creando una socket.
- Il server accetta la connessione TCP dal client.
- Client e server si scambiano i messaggi HTTP.
- Il client (o il server) chiude la connessione TCP.

Connessioni HTTP non persistenti.

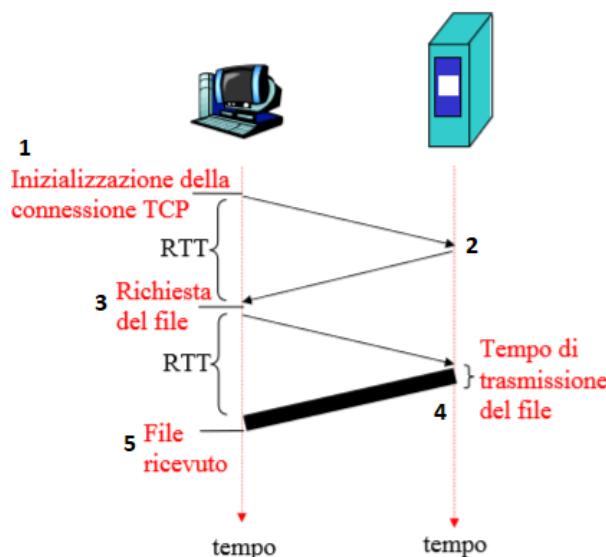
Ogni connessione TCP viene chiusa dopo l’invio di ogni oggetto, in pratica *ciascuna connessione* può trasportare *un solo messaggio* di richiesta e un solo messaggio di risposta. I *passi di un trasferimento dati* in una connessione non persistente sono i seguenti:

- 1) Il *client inizializza una connessione* TCP con il server sulla porta 80.
- 2) Il *server accetta la connessione* e avvisa il client.
- 3) Il *client*, tramite la sua porta socket, *invia* al server *un messaggio di richiesta* che include il percorso del file desiderato.

- 4) Il *server* riceve il messaggio di richiesta, recupera l'oggetto e lo incapsula in un *messaggio di risposta* che viene inviato al client tramite la sua socket.
- 5) Il *client* riceve il *messaggio di risposta* contenente l'oggetto desiderato e il *server chiude la connessione TCP*.

Schema del tempo di risposta.

RTT (Round Trip Time) indica il *tempo* che un pacchetto impiega per viaggiare da *client a server e tornare indietro*. Tale tempo è comprensivo dei ritardi di propagazione. In particolare, per le connessioni HTTP non persistenti, è necessario: un RTT per inizializzare la connessione TCP; un RTT per la richiesta del file; un tempo di trasmissione del file dal server al client. Possiamo quindi concludere che il *tempo di risposta totale* è $2RTT + \text{tempo di trasmissione}$.



Connessioni persistenti.

La connessione persistente non impone la chiusura della connessione per ogni oggetto trasferito. Infatti, una volta instaurata la *connessione*, il server la lascia *aperta anche dopo l'invio di una risposta*, e quindi essa può essere usata dal client per richiedere altri oggetti uno dietro l'altro.

Formato generale dei messaggi HTTP.

Il messaggio di richiesta (o risposta) HTTP è scritto in testo ASCII, in modo che sia leggibile dall'utente. Ogni messaggio, inoltre, può essere costituito da un *numero indefinito di righe*. La prima è detta *riga di richiesta* (o *riga di stato* se il messaggio è di risposta), mentre le successive si chiamano *righe di intestazione*, alla fine delle quali parte il *corpo* vero e proprio *del messaggio*.



Messaggio di richiesta HTTP.

La *riga di richiesta* presenta 3 campi: il campo metodo, il campo URL e il campo versione HTTP. Tra le *righe di intestazione*, invece, ci sono:

- La *riga host*, che specifica l'host sul quale è contenuto l'oggetto.
- La *riga user agent*, che specifica il tipo di browser dalla quale è stata fatta la richiesta (utile in quanto molti server hanno versioni differenti di uno stesso oggetto ottimizzate per i vari browser).
- La *riga accept language*, che indica la lingua in cui l'utente vorrebbe ricevere l'oggetto.

```
→ GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language:fr
```

Tipi di metodi.

- *GET*: include le informazioni dell'input nel campo URL della riga di richiesta.
- *POST*: include le informazioni dell'input nel corpo dell'entità.
- *HEAD*: chiede al server di escludere l'oggetto richiesto dalla risposta.
- *PUT*: include il file nel corpo dell'entità e lo invia al percorso specificato nel campo URL.
- *DELETE*: cancella il file specificato nel campo URL.

Messaggio di risposta HTTP.

La *riga di stato* presenta 3 campi: la versione di protocollo, il codice di stato ed un relativo messaggio di stato.

Tra le *righe di intestazione*, invece, ci sono:

- *Connection close*, che indica che il server, dopo la trasmissione dati, ha intenzione di chiudere la connessione.
- La *riga date*, che indica la data di creazione del messaggio.
- La *riga last modified*, che indica la data di ultima modifica del messaggio.
- La *riga content length*, che indica la lunghezza del contenuto del messaggio.
- La *riga content type*, che indica come deve essere interpretato il messaggio (testo, HTML, immagine).

Il *corpo del messaggio* conterrà i dati che il client aveva richiesto al server tramite il messaggio di richiesta.

```
→ HTTP/1.1 200 OK
Connection close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 ...
Content-Length: 6821
Content-Type: text/html

→ dati dati dati dati ...
```

Codici di stato della risposta HTTP.

Tra i più comuni codici di stato e relative espressioni troviamo:

- *200 OK*, quando la richiesta ha avuto successo e in risposta si invia l'informazione.
- *301 MOVED PERMANENTLY*, quando l'oggetto richiesto è stato trasferito in una nuova posizione.
- *400 BAD REQUEST*, quando la richiesta non è stata compresa dal server.
- *404 NOT FOUND*, quando il documento richiesto non è presente sul server.
- *505 HTTP Version Not Supported*, quando il server non ha la versione del protocollo HTTP richiesta.

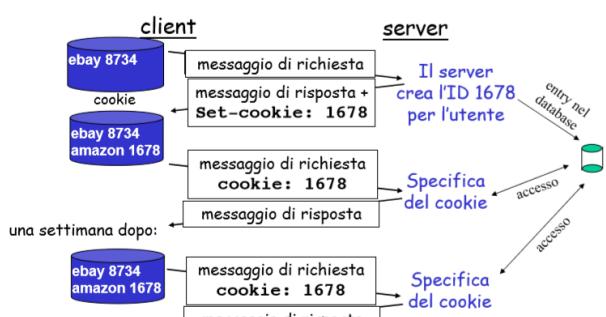
Cookie.

I cookie sono una forma di *interazione utente-server* usata da molti dei più importanti siti web. Quando si visita per la prima volta un sito che fa uso di cookie, tale sito crea un identificativo unico (ID) e una entry nel suo database, di modo che la prossima volta che si visiterà lo stesso sito si otterrà la specifica del proprio cookie. I cookie possono contenere: autorizzazioni; informazioni sulle carte per acquisti; raccomandazioni; stato della sessione dell'utente. Quindi, siccome i cookie permettono ai siti di imparare tutte queste cose sugli utenti, si crea il problema della *privacy*.

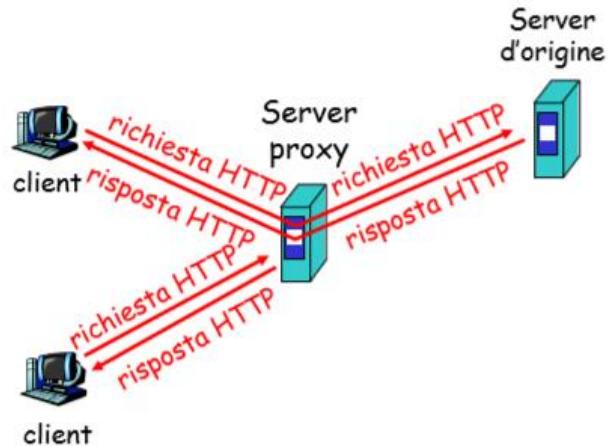
Cache web.

L'obiettivo della cache web (o *server proxy*) è quello di soddisfare la richiesta del client senza coinvolgere il server d'origine. Infatti, il browser trasmette tutte le richieste HTTP alla cache e poi: se l'oggetto è nella cache, la cache fornisce l'oggetto; altrimenti, la cache richiede l'oggetto al server d'origine e poi lo inoltra al client. La cache *opera quindi sia come client che come server*, e tipicamente viene installata da un ISP. Tra i vantaggi del caching web troviamo:

- Riduzione dei tempi di risposta alle richieste dei client.
- Riduzione del traffico sul collegamento di accesso a Internet.
- Efficacia dei provider nel fornire i dati.



Esempio di utilizzo dei cookie



Esempio di utilizzo del server proxy

GET condizionale.

L'*obiettivo* di questo metodo è quello di *non inviare un oggetto se la cache ne ha una copia aggiornata*. La cache, infatti, specifica la data della sua copia dell'oggetto nella richiesta HTTP (in particolare nella riga *if modified since*). A questo punto il server ha due possibilità: se l'oggetto è stato modificato, risponde con quest'ultimo; altrimenti, risponde semplicemente con un codice *304 Not Modified* e sarà la cache a inviare la copia dell'oggetto (ancora valida) al client.

❖ FTP

Protocollo FTP.

Il protocollo FTP (File Transfer Protocol) è utile per il *trasferimento di file a/da un host remoto*. Anche FTP segue il *modello client/server*, in quanto il client inizia il trasferimento a/da un host remoto (server). Il server FTP si trova sulla *porta 21* e *mantiene lo stato* (directory corrente, autenticazione precedente).

Connessione di controllo e connessione dati.

- Il client contatta il server FTP alla *porta 21*.
- Il client ottiene dal server l'autorizzazione sulla *connessione di controllo* (out of band).

- Il client modifica i file in remoto inviando i comandi sulla connessione di controllo.
- Il server riceve i comandi e apre una *connessione dati* con il client sulla *porta 20*.
- Dopo la modifica dei file, il server chiude la connessione dati e ne apre eventualmente una seconda per la modifica di altri file.

❖ Posta elettronica

Componenti principali.

- *Agente utente* (o mail reader): compone, modifica e legge i messaggi di posta elettronica (esempi: Outlook, Thunderbird).
- *Server di posta*: ha una *casella di posta* che contiene i messaggi in arrivo per l'utente e una *coda di messaggi* da trasmettere.
- *SMTP* (Simple Mail Transfer Protocol): protocollo usato tra i server di posta per inviare messaggi (il client è il server di posta trasmittente, mentre il server è il server di posta ricevente).

Interazione SMTP.

Il protocollo SMTP usa *TCP* (sulla *porta 25*) per trasferire in modo affidabile i messaggi di posta elettronica dal client al server. Il trasferimento è diretto, cioè il server trasmittente invia al server ricevente. SMTP usa connessioni persistenti e richiede che il messaggio (intestazione e corpo) sia nel formato ASCII a 7 bit. Come in HTTP l'interazione è di tipo *comando/risposta* (ASCII/codici di stato). A differenza di HTTP, dove ciascun oggetto è incapsulato nel suo messaggio di risposta, in SMTP più oggetti vengono trasmessi in un unico messaggio. Le *fasi del trasferimento* di un messaggio sono: handshaking (saluto); trasferimento vero e proprio del messaggio; chiusura (con CRLF/CRLF).

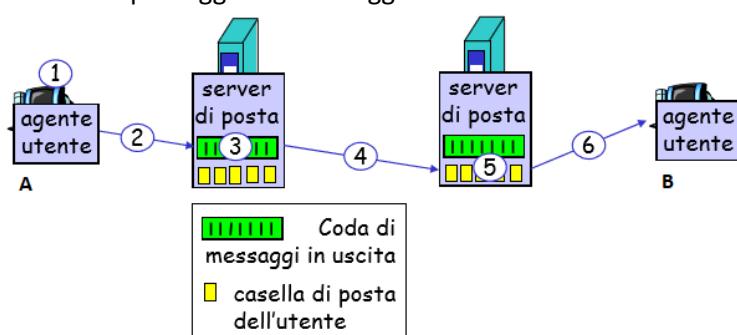
```

S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <rob@hamburger.edu>
S: 250 rob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection

```

Invio di un messaggio.

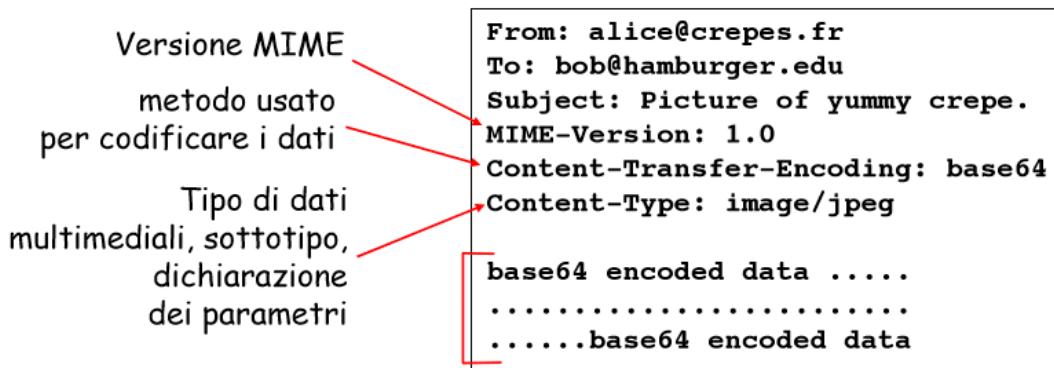
- 1) A usa il suo agente utente per comporre il messaggio da inviare a B.
- 2) L'agente utente di A invia un messaggio al server di posta di A che lo pone nella coda dei messaggi.
- 3) Il client SMTP apre una connessione TCP con il server di posta di B.
- 4) Il client SMTP invia il messaggio di A sulla connessione TCP.
- 5) Il server di posta di B pone il messaggio nella casella di posta di B.
- 6) B invoca il suo agente utente per leggere il messaggio.



Formato del messaggio.

Un messaggio di posta elettronica è formato da:

- *Righe di intestazione* (da non confondere con i comandi SMTP):
 - *From/To*, che rappresentano da chi viene inviato e a chi mandare il messaggio.
 - *Subject*, che rappresenta l'oggetto del messaggio.
 - *MIME*, ovvero estensioni di messaggi di posta multimediali (vedi immagine).
- *Riga vuota*.
- *Corpo del messaggio* scritto in caratteri ASCII.



Protocolli di accesso alla posta.

SMTP consegna e memorizza la posta sul server del destinatario. Per avere accesso alla posta e ottenere i messaggi dal server, invece, esistono due protocolli:

- *POP* (Post Office Protocol), che permette solo autorizzazione tra agente e server e download.
- *IMAP* (Internet Mail Access Protocol), che ha più funzioni come la manipolazione dei messaggi memorizzati sul server.

POP vs IMAP.

POP è un protocollo *senza stato* e usa la modalità “scarica e cancella”, quindi l’utente non può rileggere le mail se cambia client. IMAP invece *conserva lo stato* dell’utente e usa la modalità “scarica e mantieni”, quindi consente all’utente di organizzare i messaggi (che si trovano tutti sul server) in cartelle.

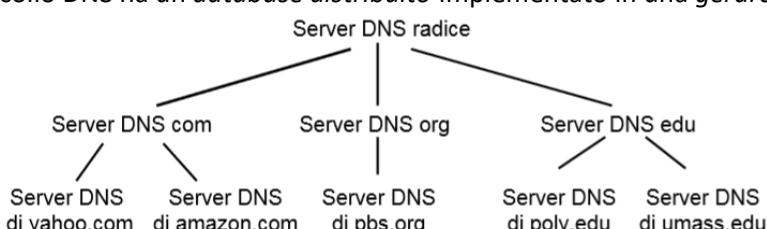
❖ DNS

Servizi DNS.

DNS (Domain Name System) è un *protocollo a livello di applicazione* che consente agli host, ai router e ai server DNS di comunicare per *risolvere i nomi*, ovvero tradurre gli indirizzi IP in host-name del tipo “www.sito.com”. Un host può avere più nomi (*host aliasing*) e un nome canonico può avere un insieme di indirizzi IP (*server web replicati*).

Database distribuiti e gerarchici.

Un *database centralizzato* su un singolo server DNS non sarebbe scalabile e avrebbe i seguenti svantaggi: singolo punto di guasto; volume di traffico elevato; distanza del database stesso; manutenzione difficile. Per questi motivi, il protocollo DNS ha un *database distribuito* implementato in una *gerarchia di server DNS*.



Server DNS radice.

Nel mondo esistono 13 server DNS radice, che vengono contattati da un server DNS locale che non riesce a tradurre il nome. Il server DNS radice, a questo punto, contatta un server DNS autorizzato per ottenere la mappatura che restituirà al server DNS locale.

Server TLD e server di competenza.

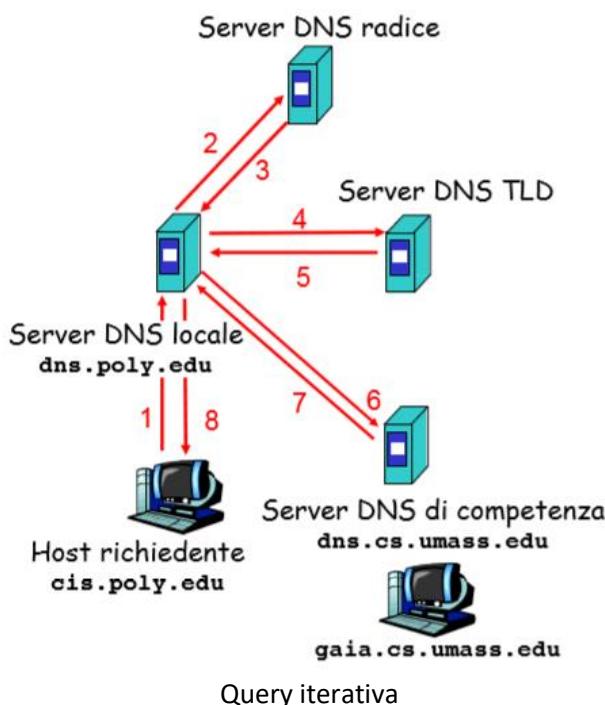
- *Server TLD (Top Level Domain)*: si occupano dei domini *com, org, net, edu, ecc.* e di tutti i domini locali di alto livello (come *uk, fr, it, ecc.*).
- *Server di competenza (authoritative server)*: possono essere mantenuti dall'organizzazione o dal service provider, infatti, ogni organizzazione dotata di host Internet pubblicamente accessibili (quali i server web e i server di posta) deve fornire i record DNS di pubblico dominio che mappano i nomi di tali host in indirizzi IP.

Server DNS locale.

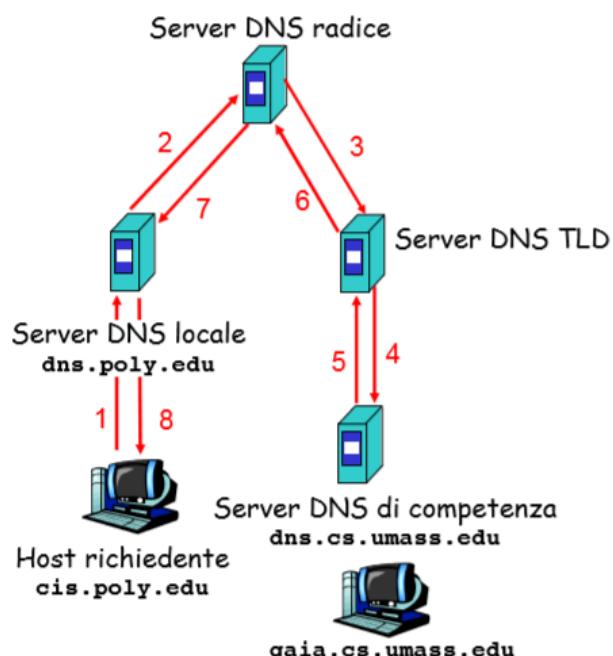
Il server DNS locale non appartiene strettamente alla gerarchia dei server. Ciascun ISP ha un server DNS locale, detto anche “*default name server*”. Quando un host effettua una *richiesta DNS*, la query viene inviata al suo server DNS locale, il quale opera da proxy e inoltra la query in una gerarchia di server DNS.

Query.

- *Iterativa*: il server contattato risponde con il nome del server da contattare; “io non conosco questo nome, ma puoi chiederlo a questo server”.
- *Ricorsiva*: l'host affida il compito di tradurre il nome al server contattato, il quale chiederà la stessa cosa ad un altro server e così via.



Query iterativa



Query ricorsiva

Caching e aggiornamento dei record.

Una volta che un server DNS impara la *mappatura*, la mette *nella cache*. Le informazioni nella cache vengono invalidate dopo un certo periodo di tempo. Tipicamente un server DNS locale memorizza nella cache gli indirizzi IP dei server TLD, quindi i server DNS radice non vengono visitati spesso.

IL LIVELLO DI TRASPORTO

❖ Generalità

Livello di trasporto.

La funzionalità del livello di trasporto è quella di creare un *canale di trasporto end-to-end* ideale e privo di errori tra due utenti. Per compiere questo obiettivo, il livello di trasporto offre dei *servizi* al livello superiore e svolge una serie di *funzioni*.

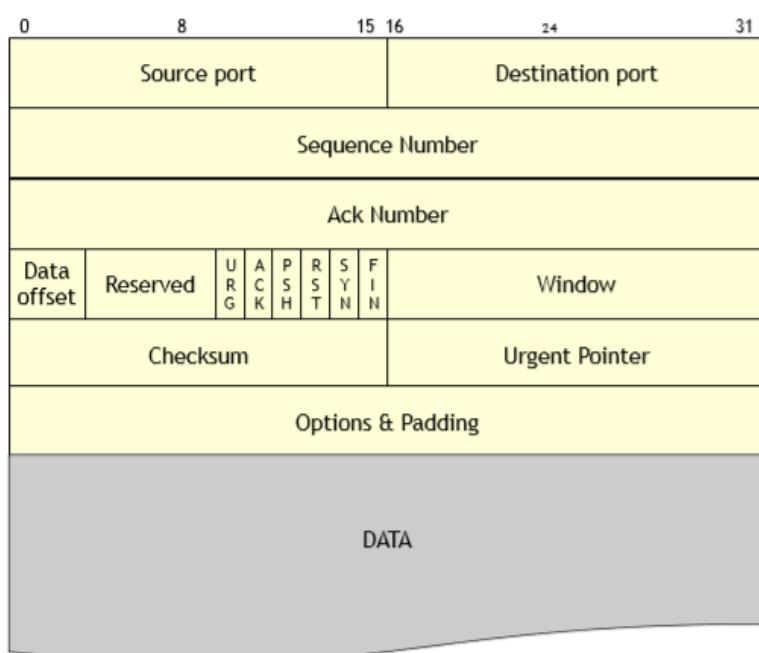
Servizi offerti al livello applicativo.

- *Connection-oriented affidabile* (TCP): per il trasferimento dei dati viene attivata una connessione e ogni segmento inviato viene “riscontrato” in modo individuale.
- *Connection-less non affidabile* (UDP): non viene attivata nessuna connessione e non si attende alcun feedback dalla destinazione sulla corretta ricezione (se un segmento viene perso non ci sono tentativi per recuperarlo).

Funzioni svolte dal livello di trasporto.

- *Indirizzamento a livello applicativo* (multiplexing/demultiplexing): il multiplexing si occupa di raccogliere dati provenienti da applicazioni diverse e trasmetterli attraverso un unico strato di rete; il demultiplexing, invece, prende i dati dallo stesso strato di rete e li smista verso le diverse applicazioni.
- *Instaurazione, gestione e rilascio della connessione*: la connessione è lo scambio di informazioni necessarie per concordare l’attivazione di un canale di comunicazione.
- *Recupero degli errori*: si definiscono politiche e azioni da svolgere in caso di errore o perdita di segmenti.
- *Consegna ordinata dei segmenti*.
- *Controllo di flusso*: si cerca di limitare l’immissione di dati in rete a seconda della sua capacità.
- *Controllo della congestione*: si definiscono azioni da intraprendere in caso di congestione di rete.

❖ Formato dei segmenti TCP



Significato dei campi.

Ogni segmento TCP (Transfer Control Protocol) ha 20 byte nella sua intestazione, così suddivisi:

- *Source port* e *Destination port* [16 bit], indirizzi della porta sorgente e della porta destinazione.
- *Sequence Number* [32 bit], numero di sequenza del primo byte del payload (per re-assemblare i dati).
- *Ack Number* [32 bit], numero di sequenza del prossimo byte che si intende ricevere (indica i dati che sono stati ricevuti e confermati e ha validità se il segmento è un ACK).
- *Offset* [4 bit], lunghezza dell'header TCP (in multipli di 32 bit).
- *Reserved* [6 bit], riservato per usi futuri.
- *Window* [16 bit], numero di byte che possono essere accettati in una sola volta dalla destinazione.
- *Checksum* [16 bit], usato per il controllo degli errori nell'intestazione del segmento e nei dati.
- *Urgent pointer* [16 bit], usato se si inviano comandi che danno inizio ad eventi asincroni urgenti.
- *Flag* [6 * 1 bit]:
 - *URG*, vale 1 se vi sono dati urgenti (in questo caso il campo *Urgent pointer* ha senso);
 - *ACK*, vale 1 se il segmento è un ACK (in questo caso l'*Ack Number* contiene un numero valido);
 - *PSH*, vale 1 quando il trasmettitore intende usare il comando di *PUSH*;
 - *RST*, resetta la connessione senza un *tear down* esplicito;
 - *SYN*, usato durante il *setup* per comunicare i numeri di sequenza iniziale;
 - *FIN*, usato per la chiusura esplicita di una connessione.

Gli ultimi due campi in un segmento TCP sono:

- *Options & Padding* [lunghezza variabile], contiene campi opzionali (si riempie a multipli di 32 bit).
- *Data* [lunghezza variabile], i dati provenienti dall'applicazione.

❖ Indirizzamento

Porta.

Gli indirizzi IP vengono utilizzati per l'instradamento dei pacchetti all'interno della rete e identificano univocamente la sorgente e la destinazione. Ma quando il pacchetto giunge alla destinazione e viene passato al livello di trasporto (a cui si appoggiano *molte applicazioni*) come è possibile distinguere un'applicazione dall'altra? Si introduce il concetto di *porta*, che non è altro che un codice che identifica l'applicazione.

Socket.

La socket è una *tupla* che identifica la connessione (e il conseguente flusso) tra due applicazioni. La socket permette di *differenziare le connessioni* e serve al TCP per il multiplexing e il demultiplexing dei dati. La socket è formata da {IP sorgente: numero di porta sorgente, IP destinazione: numero di porta destinazione}.

Indirizzamento TCP.

Le porte sorgente e destinazione non sono necessariamente uguali, infatti, il numero di porta può essere:

- *Statico* (porta nota), che è associato ad applicazioni largamente utilizzate (posta elettronica, web, FTP).
- *Dinamico* (porta effimera), che è assegnato dal sistema operativo quando si apre la connessione.

❖ Gestione delle connessioni

Instaurazione della connessione.

Siccome il TCP è un protocollo connection-oriented, prima di iniziare a trasferire i dati ci deve essere una connessione tra i due host, che viene instaurata secondo la procedura di "three-way handshake":

1. L'host A *richiede la connessione* inviando un segmento di SYN, dove specifica il numero di porta dell'applicazione a cui intende accedere e l'Initial Sequence Number (ISNA).
2. L'host B *riceve la richiesta e risponde* con un segmento di SYN, dove specifica ISNB e riscontro di ISNA.

3. L'host A *invia un segmento di ACK* all'host B per informarlo che ha riscontrato il suo segmento di SYN. I segmenti di SYN e ACK non sono altro che segmenti TCP vuoti (hanno solo l'header) in cui i bit SYN e ACK sono posti a: 1 e 0 se il segmento è SYN; 0 e 1 se il segmento è ACK; 1 e 1 se il segmento è SYN ACK.

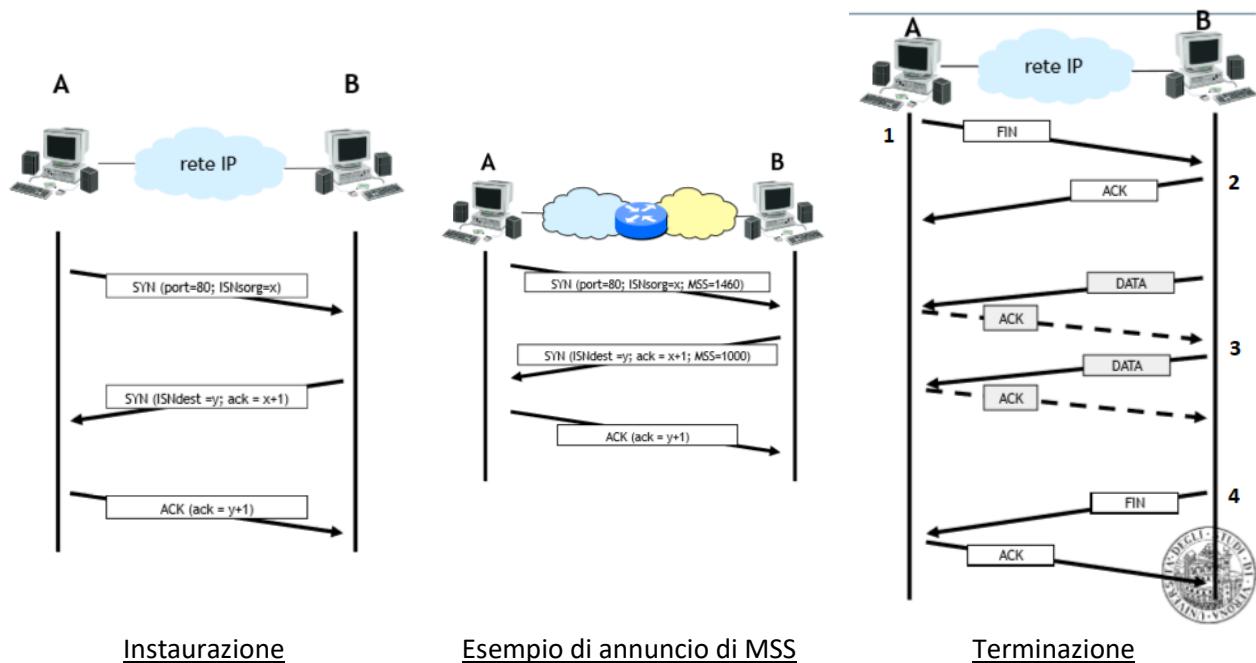
Maximum Segment Size (MSS).

L'instaurazione della connessione serve per scambiarsi i dati relativi alla comunicazione. Tra questi vi può essere anche la *MSS*, la quale consente ad un host di specificare la *dimensione massima* (espressa in numero di byte) del campo dati dei segmenti che è disponibile a ricevere. Nell'esempio, A annuncia a B di poter accettare solo segmenti la cui dimensione massima del campo dati è pari a 1460 byte, mentre B annuncia ad A che la sua MSS è di 1000 byte; A e B dunque trasmetteranno segmenti con un campo dati lungo al massimo 1000 byte.

Terminazione della connessione.

Poiché la connessione è *bidirezionale* (entrambe le stazioni possono trasferire dati), anche la terminazione deve avvenire in entrambe le direzioni:

1. L'host A non ha più dati da trasmettere e decide di chiudere la connessione, perciò *invia un segmento di FIN* (ovvero un segmento TCP con il bit FIN posto a 1 e il campo dati vuoto).
2. L'host B riceve il segmento di FIN e *invia un segmento di ACK*, indicando all'applicazione che la comunicazione è stata chiusa nella direzione entrante (half close).
3. L'host B può ancora continuare a trasferire dati (gli ACK di A non sono considerati come traffico).
4. L'host B chiude completamente la connessione (half close).



❖ Gestione degli errori e delle perdite

Affidabilità.

Il TCP è un protocollo connection-oriented affidabile, ovvero garantisce la *corretta e ordinata consegna* dei segmenti. Il TCP dunque deve essere in grado di *gestire le situazioni di errore* (con l'uso del campo checksum) e la *perdita di segmenti* (ed eventualmente di riscontri). La tecnica "*positive acknowledgement with retransmission*" prevede che, in caso di perdita di un segmento, il relativo riscontro non sarà inviato e quindi il TCP dovrà preoccuparsi di ritrasmettere il segmento perso. Nella versione base della suddetta tecnica (detta "*stop and wait*"), non viene trasmesso un nuovo segmento fino a quando l'ultimo non viene riscontrato.

Gestione del timeout.

Per sapere quanto tempo bisogna aspettare prima di poter considerare un segmento perso e ritrasmetterlo, si usa il tempo di timeout detto *RTO* (Retransmission Time Out), il quale indica l'istante di tempo entro il quale la sorgente si aspetta di ricevere un riscontro e, nel caso in cui il riscontro non arrivi, la sorgente procede alla ritrasmissione. Il *RTO* *non può essere un valore statico* predefinito, in quanto il tempo di percorrenza sperimentato dai segmenti è variabile, perché dipende: dalla distanza tra sorgente e destinazione; dalle condizioni della rete; dalla disponibilità della destinazione. Dunque, il *RTO* *dovrebbe essere calcolato dinamicamente* di volta in volta durante la fase di instaurazione della connessione e durante la trasmissione dei dati.

Stima del Round Trip Time.

Il calcolo del RTO si basa sulla misura del *RTT* (Round Trip Time), cioè dell'intervallo di tempo tra l'invio di un segmento e la ricezione del relativo riscontro. Poiché anche *RTT* *può variare molto* in base alle condizioni della rete, il valore di RTT utilizzato per il calcolo di RTO è in realtà una stima del valore medio di RTT sperimentato dai diversi segmenti (detto *SRTT*, Smoothed RTT). Nella formula per la stima del SRTT, un parametro α regolabile rende il peso della misura del RTT istantaneo più o meno incisivo: se α tende a 1, il SRTT stimato non viene influenzato dai segmenti che sperimentano RTT molto diversi; se α tende a 0, il SRTT stimato dipende fortemente dalla misura puntuale dei singoli RTT istantanei.

Stima del Retransmission Time Out.

La sorgente attende fino a $2 * SRTT$ prima di considerare il segmento perso e ritrasmetterlo. In caso di ritrasmissione, il RTO per quel segmento viene ricalcolato in quanto se è scaduto il RTO probabilmente c'è congestione, quindi meglio aumentare il RTO per quel segmento a $2 * RTO$.

❖ Controllo di flusso

Tecniche.

Il *controllo di flusso* è un'azione preventiva finalizzata a limitare l'immissione di dati in rete a seconda della capacità end-to-end di quest'ultima. Ma come fa un host a determinare la capacità e lo stato della rete? Un esempio che abbiamo già visto è la tecnica di "*positive acknowledgement with retransmission*", ovvero quando un host invia un nuovo segmento solo se l'ultimo è stato riscontrato ("*stop and wait*"). Questa tecnica, quindi, è già un controllo di flusso, in quanto il tasso di immissione di nuovi segmenti è determinato dalle condizioni della rete, poiché avviene in base alla ricezione dei riscontri. Tuttavia, tale tecnica risulta poco efficiente, in quanto viene sprecata banda nell'attesa dei singoli riscontri.

Finestra scorrevole (sliding window).

Per incrementare l'efficienza, è possibile trasmettere più segmenti consecutivamente senza attendere ogni singolo riscontro. L'intervallo dei segmenti trasmessi è detto *finestra di trasmissione*. Alla ricezione dei riscontri dei segmenti iniziali della sequenza, tale finestra scorre a destra permettendo la trasmissione di nuovi segmenti. Se la dimensione della finestra è troppo piccola, sottoutilizzo la banda; mentre se la dimensione della finestra è troppo grande, i riscontri potrebbero arrivare prima della conclusione della trasmissione facendo venir meno il controllo di flusso. Quindi, con il metodo a finestra scorrevole, l'immissione di nuovi segmenti deve dipendere sia dalle *condizioni della rete* che da quelle *del ricevente*.

Finestra scorrevole nel TCP.

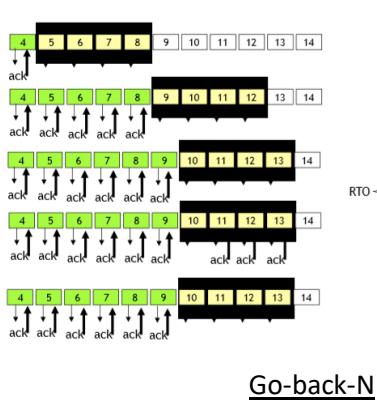
I dati provenienti dall'applicazione *vengono ordinati* in una sequenza di byte numerati progressivamente a partire dall'ISN. Poi, in base alla MSS, tali byte *vengono suddivisi in segmenti*. Il meccanismo a finestra scorrevole viene quindi applicato alla sequenza di segmenti così determinata, con la *dimensione della finestra fissata in base al campo Window* dell'header TCP (espresso dai due host durante la fase di connessione e

durante lo scambio dei dati). Scegliere la finestra basandosi su tale valore significa fare *controllo di flusso* considerando solo le condizioni della destinazione, ma non le condizioni di rete.

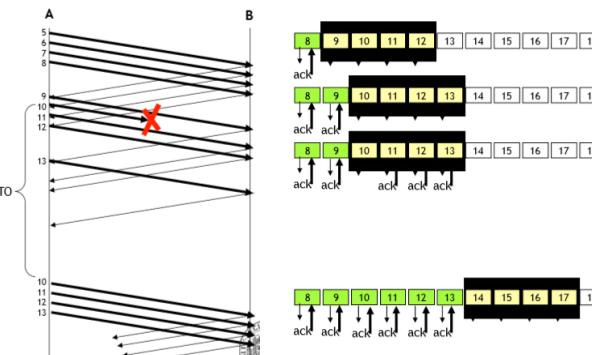
Perdita di segmenti.

In caso di perdita di un segmento, dopo lo scadere del RTO, il TCP provvede a ritrasmetterlo. Quando il segmento appartiene ad una finestra di trasmissione ci sono due possibili soluzioni:

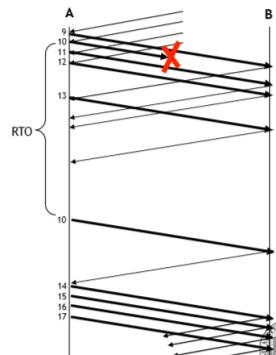
- *Go-back-N*, in cui si fa tornare indietro la finestra ritrasmettendo quindi tutti i segmenti. Tale soluzione:
 - È semplice da implementare, ma è inefficiente in caso di perdite singole.
 - Si utilizzava nelle prime implementazioni di TCP ed è stata pensata per un ambiente in cui ci sono perdite multiple contemporanee.
- *Selective Repeat*, in cui si fa ritrasmettere solo il segmento perso. Tale soluzione:
 - Ha una maggiore complessità implementativa, ma è efficiente in caso di perdite singole.
 - Non sovraccarica la rete con ritrasmissioni già riscontrate, ma siccome esistono diverse modalità di ritrasmissione sia la sorgente che la destinazione devono implementare la stessa modalità.
 - Si utilizza nelle attuali implementazioni di TCP.



Go-back-N



Selective Repeat



❖ Controllo di congestione

Tasso di immissione di nuovi segmenti.

Il controllo della congestione è una serie di azioni da intraprendere come reazione alla congestione di rete. L'unico mezzo che permette al TCP di conoscere lo stato della rete è rappresentato dalla *perdita dei segmenti* e dal relativo scadere del timeout di ritrasmissione. Siccome la perdita dei segmenti può essere *causata dalla congestione della rete* (a causa dei buffer limitati degli apparati di rete) è opportuno che la sorgente riduca il tasso di immissione dei nuovi segmenti. Questa reazione si differenzia dal controllo di flusso, che invece definisce tecniche per la prevenzione delle situazioni di sovraccarico della rete.

Congestion window.

In caso di congestione, il *controllo di flusso a finestra* protegge implicitamente anche la rete, infatti:

- Quando arrivano meno riscontri, il tasso di immissione diminuisce automaticamente.
- L'attesa dello scadere del RTO lascia un intervallo di tempo in cui non vengono immessi nuovi segmenti. Tuttavia, queste misure potrebbero non essere sufficienti, in quanto rimane da *determinare la dimensione della finestra ottimale*. La soluzione per il controllo della congestione consiste quindi nel variare dinamicamente la dimensione della finestra, permettendole di *adattarsi alle situazioni di sovraccarico e di reagire alla congestione*. Tale finestra scorrevole e variabile viene denominata “*congestion window*” (CWND).

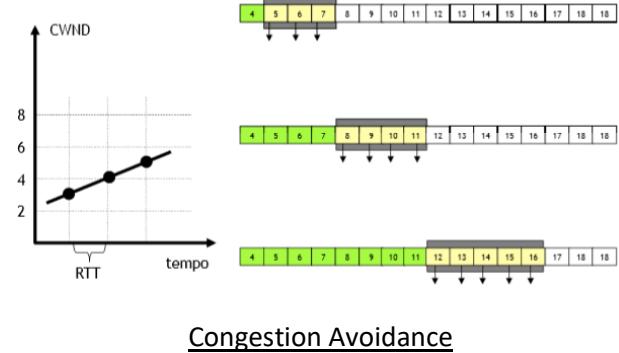
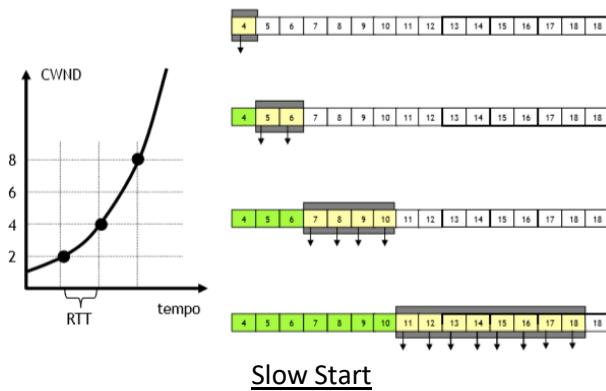
Algoritmi per il controllo della congestione.

Esistono diversi algoritmi che regolano la dimensione della finestra (CWND) al variare delle condizioni di rete. I due algoritmi di base sono *Slow Start* e *Congestion Avoidance*, ma esiste anche un altro algoritmo chiamato *Fast Retransmit – Fast Recovery* che è stato definito per aumentare l'efficienza del TCP.

Slow Start e Congestion Avoidance.

Sono due algoritmi che regolano la dimensione della finestra (l'utilizzo di uno esclude l'utilizzo dell'altro):

- *Slow Start*, per ciascun riscontro ricevuto la CWND può aumentare di un segmento (quando si riceve un riscontro si trasmettono due nuovi segmenti), quindi l'evoluzione della CWND ha un *andamento esponenziale* (1, 2, 4, 8, 16, ...).
- *Congestion Avoidance*, ad ogni RTT in cui si ricevono un numero di riscontri pari alla CWND, questa aumenta di un segmento, quindi l'evoluzione della CWND ha un *andamento lineare*.



Fast Retransmit – Fast Recovery.

Esistono principalmente due motivi che causano la perdita dei segmenti: gli errori di trasmissione, che influiscono su un singolo segmento; e la congestione, che influisce su più segmenti consecutivi. Quindi, quando viene perso anche un solo segmento, si ha una reazione "eccessiva" da parte della sorgente, che è quella di attendere lo scadere del timeout e chiudere la CWND. La soluzione a questo problema è rappresentata dall'algoritmo *Fast Retransmit – Fast Recovery*, che è specificatamente *progettato per gestire le perdite singole*. Il funzionamento consiste nel *ritrasmettere subito il segmento* considerato perso (*Fast Retransmit*), in modo da *non chiudere eccessivamente la CWND* (*Fast Recovery*).

ACK duplicati.

L'algoritmo *Fast Retransmit – Fast Recovery* si basa sugli ACK duplicati, in quanto se un *segmento* arriva fuori sequenza, la destinazione invia un *ACK* indicando il *numero di sequenza* del segmento *che non è ancora arrivato*. Infatti, nei segmenti ACK il campo *Ack Number* contiene il numero di sequenza del segmento che ci si aspetta di ricevere e questo indica anche che tutti i segmenti precedenti sono stati ricevuti correttamente.

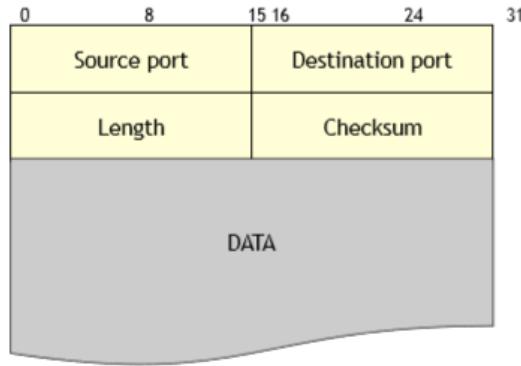
❖ Il protocollo UDP

User Datagram Protocol.

UDP è un protocollo di trasporto connection-less non affidabile che *svolge solo funzioni di indirizzamento* delle applicazioni (tramite le porte). Quindi non gestisce: connessioni, controllo di flusso, recupero degli errori (solo rilevamento), controllo della congestione, riordino dei segmenti; infatti *demandava ai livelli superiori* i compiti di: verifica della perdita dei messaggi, consegna ordinata, controllo di flusso, ecc. UDP è utilizzato principalmente per il supporto di *transazioni semplici tra applicativi* (strettamente dipendenti dal tempo) e per le *applicazioni multimediali* (più elastiche riguardo alla perdita dei dati).

Formato dei segmenti UDP.

- *Source Port* e *Destination Port* [16 bit], identificano i processi sorgente e destinazione dei dati.
- *Length* [16 bit], lunghezza totale (espressa in byte) del segmento comprensivo dell'header UDP.
- *Checksum* [16 bit], campo di controllo per sapere se il segmento contiene errori nel campo dati.



❖ Riassunto

Il TCP è un protocollo di trasporto *connection-oriented affidabile* che svolge le seguenti funzioni:

- *Indirizzamento a livello applicativo* (multiplexing/demultiplexing), ovvero utilizza il concetto di *porta* per indirizzare le diverse applicazioni.
- *Instaurazione, gestione e rilascio delle connessioni*, ovvero utilizza il concetto di *socket* per identificare una connessione che si preoccupa di gestire scambiando le *informazioni* necessarie per concordare l'attivazione di un canale di comunicazione (ISN, MSS, Window, ecc).
- *Recupero degli errori*, ovvero gestisce il recupero dei segmenti errati o persi per riuscire ad effettuare la *consegna ordinata* dei suddetti segmenti all'applicazione (si utilizzano due tempi detti *RTT* e *RTO* che vengono calcolati dinamicamente).
- *Controllo di flusso*, ovvero implementa una *finestra scorrevole* (con dimensione variabile) per controllare il *tasso di immissione* dei segmenti in rete.
- *Controllo della congestione*, ovvero reagisce alla congestione (cioè allo scadere del RTO) diminuendo l'ampiezza della finestra di trasmissione seguendo determinati *algoritmi* (SS, CA, FR/FR).

IL LIVELLO DI RETE

❖ Introduzione

Visione generale.

Il livello di rete si occupa del *trasporto dei segmenti* dall'host sorgente all'host destinazione. La sorgente incapsula i segmenti in *datagrammi*, mentre la destinazione riceve i datagrammi dal livello di trasporto. Il livello di rete è presente in ogni end-host e negli apparati intermedi, detti *router*, i quali esaminano i campi dell'header presenti nei datagrammi IP.

Funzioni chiave.

Le funzioni chiave del livello di rete sono: il *routing*, che determina il percorso che i pacchetti devono seguire dalla sorgente alla destinazione (attraverso l'esecuzione di algoritmi di routing); e il *forwarding*, cioè il trasferimento dei datagrammi da una porta di input alla porta di output appropriata. Queste funzioni richiedono come componente fondamentale l'indirizzamento (*addressing*).

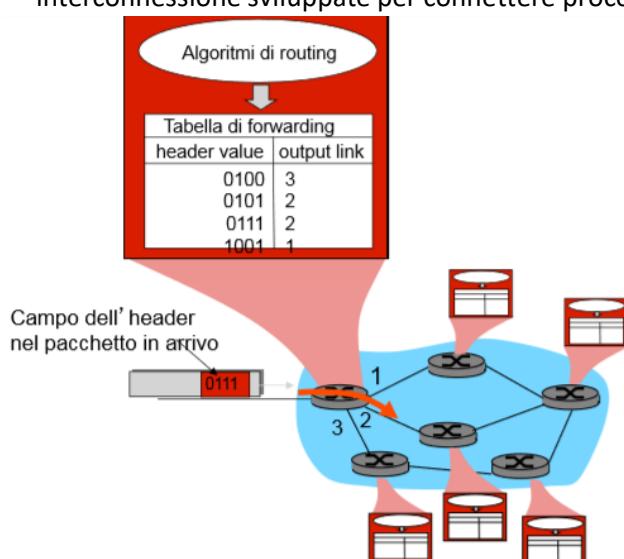
Porte di input.

Le porte di input eseguono il cosiddetto *switching decentralizzato*, ovvero, data la destinazione, determinano la porta di output utilizzando le tabelle di forwarding. Lo *scopo* di questa funzione è quello di completare l'elaborazione dei datagrammi a velocità di linea. Inoltre, le porte di input devono *gestire le code* nel caso in cui i datagrammi arrivino con velocità superiore alla velocità di switching (*commutazione*) dei pacchetti.

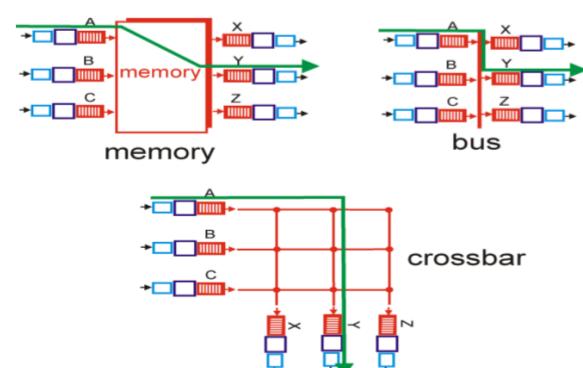
Matrici di commutazione.

Esistono *tre tipologie* di matrici di commutazione:

1. *Commutazione basata su memorie*, che appartiene alla prima generazione di router i quali non erano altro che computer tradizionali in cui il pacchetto veniva copiato nella memoria di sistema (la velocità di commutazione era limitata dalla velocità della memoria).
2. *Commutazione via bus*, dove la comunicazione tra porte di input e output avviene usando un bus condiviso (la velocità di commutazione è data dalla velocità del bus).
3. *Commutazione via rete interconnessa*, che risolve i limiti dell'architettura a bus e si basa su reti di interconnessione sviluppate per connettere processori in architetture multiprocessore.



Interazione tra routing e forwarding



Tipologie di matrice di commutazione

❖ Frammentazione IP

MTU.

A seconda della tecnologia hardware, i diversi tratti di rete possono trasportare datagrammi con una *lunghezza massima predefinita*. Tale limite è noto come Maximum Transmission Unit (*MTU*) e l'hardware di rete non è in grado di accettare o gestire datagrammi più grandi della MTU. Siccome Internet è composto da un insieme eterogeneo di segmenti di rete, la restrizione sulla MTU può creare *problemi*. Infatti, un router può connettere reti con MTU diverse, e quindi può succedere che un'interfaccia riceva un datagramma troppo grande per poter essere spedito sull'interfaccia successiva.

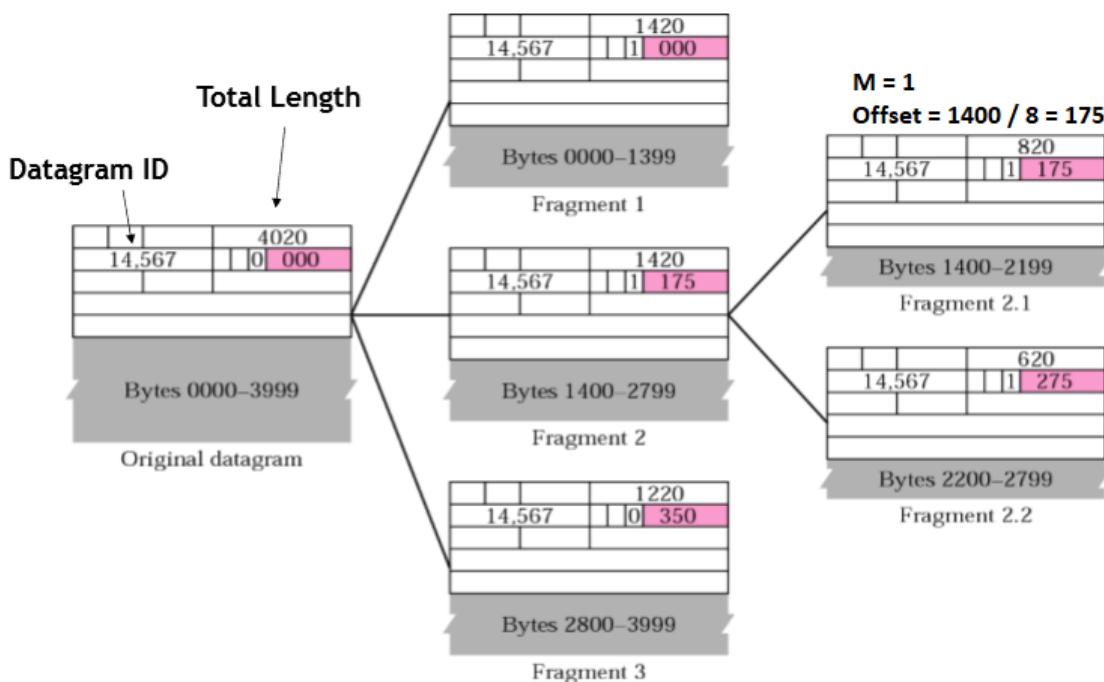
Frammentazione del datagramma.

Quando la dimensione di un datagramma è superiore alla MTU della rete verso cui deve essere inviato, il router divide il datagramma in pezzi più piccoli chiamati *frammenti* e invia ciascun frammento in modo indipendente. I frammenti si distinguono dai datagrammi completi grazie ad un bit nel campo *FLAGS*. Inoltre, vengono utilizzati anche altri campi dell'header per trasportare informazioni utili a riassemblare i frammenti a destinazione per ottenere il datagramma originale. Ad esempio, il campo *FRAGMENT OFFSET* specifica in che punto il frammento va riposizionato. Per frammentare un datagramma, il router:

1. Usa i valori di MTU e di dimensione dell'header per calcolare la *dimensione massima dei dati* che possono essere inviati in ciascun frammento e il *numero di frammenti* necessari.
2. Crea gli *header dei frammenti* usando i campi dell'header originale, cioè copia gli indirizzi IP sorgente e destinazione e gli altri campi dell'header del datagramma negli header dei frammenti.
3. Suddivide i dati del datagramma originale tra i vari frammenti.
4. Trasmette i frammenti.

Campi FLAGS e FRAGMENT OFFSET.

- Se un frammento ha il *flag M = 0*, vuol dire che non ci sono ulteriori frammenti (il frammento è l'ultimo); per dire se il pacchetto originale è stato frammentato serve anche l'*offset*.
- Se un frammento ha il *flag M = 1*, vuol dire che esiste almeno un altro frammento; se sappiamo anche l'*offset* possiamo dire se il frammento è il primo (*offset = 0*) o uno intermedio (*offset ≠ 0*).
- Se un pacchetto ha *offset = 100*, per trovare il numero del *primo byte* basta moltiplicare l'*offset* per 8, ottenendo *800*.



Riassemblaggio di un datagramma dai frammenti.

Il riassemblaggio di un datagramma dai frammenti viene fatto dalla *destinazione*. Ciò comporta due vantaggi: *riduce la quantità di dati* da memorizzare nei router, in quanto per l'operazione di forwarding un router non ha bisogno di sapere se il datagramma è intero o è un frammento; e *permette di far cambiare percorso* ai datagrammi in maniera dinamica, in quanto se un router intermedio facesse il riassemblaggio tutti i frammenti dovrebbero passare necessariamente da quel router. Il protocollo IP è quindi libero di far percorrere ai diversi frammenti il percorso più opportuno in quel momento.

Perdita dei frammenti.

Siccome si inizia a riassemblare un datagramma solo quando tutti i frammenti sono presenti, la destinazione dovrà salvare in un *buffer* tutti i frammenti in quanto essi potrebbero avere ritardi diversi. Tuttavia, i frammenti non possono essere memorizzati per sempre, infatti il protocollo IP specifica un *tempo massimo di memorizzazione dei frammenti*. Quando arriva il primo frammento di un datagramma, la destinazione fa partire un *timer*: se tutti i frammenti di un datagramma arrivano prima che il timer scada, la destinazione cancella il timer e riassembra il datagramma; viceversa, i frammenti arrivati vengono scartati. L'utilizzo di questo timer implica una politica “*tutto o niente*” (o tutti i frammenti arrivano e il datagramma viene ricostruito, o il datagramma incompleto viene scartato). Infine, la sorgente non ha informazioni riguardo alla frammentazione, in quanto non ci sono meccanismi per far sì che la destinazione comunichi quali frammenti sono arrivati. Quindi, se la sorgente ritrasmette il datagramma, il percorso seguito potrebbe essere diverso dal precedente e perciò non ci sono garanzie che il datagramma ritrasmesso venga frammentato nello stesso modo del precedente.

Esempio di frammentazione.



Gli host H1 e H2 sono connessi ad una rete con MTU pari a 1500 byte. I router R1 e R2 interconnettono ciascuno due reti con valori di MTU pari a 1500 e 1000 byte. *Se non ci fosse la frammentazione*, quando l'host H1 vuole inviare un datagramma di 1500 byte all'host H2, il router R1 non riuscirà ad inoltrarlo sulla rete Net 2 in quanto su tale rete non si possono inviare o ricevere datagrammi di dimensione maggiore a 1000 byte. *Con la frammentazione*, invece, quando l'host H1 vuole inviare un datagramma di 1500 byte all'host H2, il router R1 lo dividerà in due frammenti che verranno spediti al router R2. A questo punto, il compito di *riassembolare* i frammenti non spetta a R2 (che invia i frammenti come datagrammi a sé stanti), bensì alla destinazione (che memorizza i frammenti e li riassembra per ottenere il datagramma originale).

L'INDIRIZZAMENTO NEL LIVELLO DI RETE

❖ Introduzione

Indirizzi per Internet.

L'indirizzamento è un componente fondamentale di Internet. Tutti gli host devono utilizzare uno *schema di indirizzamento comune*, per permettere a più applicativi di comunicare senza preoccuparsi del tipo di hardware di rete utilizzato. Inoltre, ciascun *indirizzo* deve essere *unico*. Gli indirizzi MAC del livello Data Link non possono essere utilizzati, in quanto Internet contiene diverse tecnologie di rete ciascuna con il proprio formato di indirizzo MAC. Perciò si utilizzano gli indirizzi IP, che vengono assegnati da un protocollo software e non sono "hard-coded" nella tecnologia utilizzata.

Schema di indirizzamento IP e notazione.

A ciascun host viene assegnato un numero unico di *32 bit* noto come indirizzo IP o indirizzo Internet dell'host. Quando un host vuole inviare un pacchetto in Internet, la sorgente deve specificare il proprio indirizzo IP e l'indirizzo IP della destinazione. Per semplificare la gestione degli indirizzi da parte degli utenti, invece di indicare il valore binario dei 32 bit, si utilizza la *notazione decimale puntata*, la quale prevede di: dividere i 32 bit in *4 sezioni da 8 bit* ciascuna; esprimere ciascuna sezione nel corrispondente valore decimale; dividere con un punto le diverse sezioni. Per ogni sezione, il valore più piccolo è 0 mentre il valore più grande è 255, perciò il range degli indirizzi va da *0.0.0.0* a *255.255.255.255*.

Gerarchia degli indirizzi IP.

Gli indirizzi IP sono divisi concettualmente in due parti: un *prefisso* (Net ID), che *identifica la rete* fisica a cui l'host è connesso; e un *suffisso* (Host ID), che *identifica un host* specifico all'interno di una rete. Lo schema degli indirizzi IP garantisce che a *ciascun host* venga assegnato un *indirizzo unico* e che l'*assegnazione dei prefissi* venga *coordinata globalmente* (i suffissi invece vengono assegnati localmente).

❖ Indirizzamento Classful

Tradeoff dei bit.

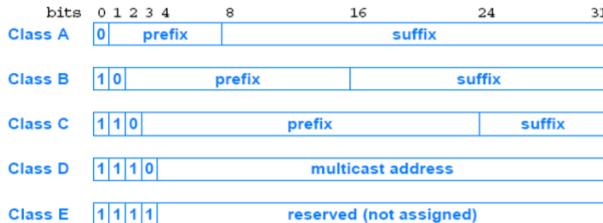
Per determinare *quanti bit usare per prefisso e suffisso*, bisogna ricordare che: il prefisso necessita di un numero di bit sufficientemente grande per identificare tutte le possibili reti fisiche in Internet; mentre il suffisso necessita di un numero di bit sufficientemente grande per poter specificare tutti i possibili host connessi ad una certa rete. Quindi *non esiste una scelta semplice* per l'allocazione dei bit, in quanto: utilizzando tanti bit per il prefisso si possono specificare molte reti, ma ciascuna rete avrà dimensione limitata; mentre utilizzando tanti bit per il suffisso si possono avere molti host su una data rete, ma il numero totale di reti sarà limitato.

Classi di indirizzi IP – soluzione originale.

Siccome Internet contiene poche reti grandi e molte reti piccole, i progettisti hanno scelto uno schema che permettesse la convivenza di combinazioni di reti grandi e piccole. L'indirizzamento IP originale, chiamato *classful*, divideva lo spazio di indirizzamento in 3 classi primarie, ciascuna con una diversa dimensione del prefisso/suffisso. I primi 4 bit di un indirizzo IP determinavano la classe di indirizzamento di appartenenza e quindi specificavano anche come il resto dell'indirizzo doveva essere diviso tra prefisso e suffisso.

Divisione dello spazio di indirizzamento.

Lo schema classful divideva lo spazio di indirizzamento in *porzioni di dimensione diverse*. I progettisti scelsero questa soluzione per poter includere *diversi scenari*. Metà degli indirizzi IP disponibili appartengono alla classe A, ma il numero di reti di classe A era limitato a 128 in quanto lo scopo era quello di permettere ai principali ISP di creare ciascuno una grande rete che connettesse milioni di host. In maniera analoga, la classe C era stata creata per permettere ad una piccola organizzazione di connettere alcuni calcolatori ad una LAN.



Classi di indirizzi IP

Address Class	Bits In Prefix	Maximum Number of Networks	Bits In Suffix	Maximum Number Of Hosts Per Network
A	7	128	24	16777216
B	14	16384	16	65536
C	21	2097152	8	256

Spazio di indirizzamento

Autorità per gli indirizzi.

L'autorità che assegna gli indirizzi e gestisce le relative dispute si chiama Internet Corporation for Assigned Names and Numbers (*ICANN*). ICANN non assegna direttamente i prefissi, ma autorizza altri *organi ufficiali* a farlo. Essi quindi assegnano i blocchi di indirizzi agli *ISP*, che forniscono a loro volta gli indirizzi ai relativi *utenti*.

❖ Indirizzamento Classless

Meccanismi.

Con l'espansione di Internet lo *schema classful* originale si è rivelato *limitante*, in quanto tutti chiedevano indirizzi di classe A o B, in modo da poter avere un numero sufficiente di indirizzi per eventuali espansioni. Per risolvere tale limitazione, sono stati proposti due meccanismi correlati: *subnet* e *indirizzamento classless*.

Indirizzamento classless – motivazioni.

Nello schema classless, invece di avere un insieme ristretto di *lunghezze per prefissi e suffissi*, si rende la scelta di tali lunghezze *arbitraria*. Si consideri un ISP che distribuisce prefissi e si assuma che un cliente chieda un prefisso per una rete che ha 55 host: in caso di indirizzamento *classful*, si dovrebbe assegnare una classe C (da 254 indirizzi host) anche se sarebbero sufficienti 6 bit per rappresentare tutti valori degli host (190 indirizzi su 254 andrebbero sprecati); in caso di indirizzamento *classless*: l'ISP può *assegnare liberamente la dimensione del prefisso* (ad esempio 26 bit per il prefisso e 6 bit per il suffisso).

Maschere.

Le decisioni di routing si prendono analizzando il prefisso, perché indica la rete di appartenenza. Con l'indirizzamento classful, i primi bit indicavano la classe e quindi la lunghezza del prefisso. Nel caso di indirizzamento classless, invece, serve aggiungere un'informazione, ovvero la *suddivisione tra prefisso e suffisso*. Infatti, invece di aggiungere la dimensione del prefisso esplicitamente, si preferisce usare questa tecnica, nota come *maschera di subnet*. Una maschera non è altro che un valore a 32 bit in cui sono posti a 1 tutti i bit fino a raggiungere la lunghezza del prefisso.

Confronto di indirizzi IP.

Il *router* mantiene in memoria una serie di *reti di destinazione* (prefissi di rete) e le *corrispondenti maschere*. Quando arriva un pacchetto con indirizzo IP generico, il router: confronta l'indirizzo di destinazione con i prefissi in memoria e fa il forward del pacchetto in base alla destinazione. Il confronto non viene fatto su tutti e 32 i bit ma, per ogni destinazione (prefisso), si considera una maschera e viene fatto l'*and logico* bit a bit *tra maschera e indirizzo IP* del pacchetto, poi si confronta il risultato con il prefisso in memoria e, se sono uguali, allora è stata determinata la destinazione del pacchetto.

Si considerino i prefissi e le relative maschere memorizzate in un router:

NetA 10000000 00001010 00000000 00000000 → 128.10.0.0

MaskA 11111111 11111111 00000000 00000000 → 255.255.0.0

NetB 01000000 00001010 00000010 00000000 → 64.10.2.0

MaskB 11111111 11111111 11111111 00000000 → 255.255.255.0

Il router analizza un pacchetto con il seguente indirizzo IP

IP addr 10000000 00001010 00000010 00000011 → 128.10.2.3

Il router applica le diverse maschere al pacchetto e confronta il risultato con i prefissi

IP & MaskA 10000000 00001010 00000000 00000000 → 128.10.0.0

IP & MaskB 10000000 00001010 00000010 00000000 → 128.10.2.0

Essendo $(IP \& MaskA) = NetA$, allora il pacchetto ha come destinazione la rete NetA



Notazione CIDR.

L'utilizzo di maschere per specificare la dimensione del prefisso viene fatta per questioni di efficienza, in quanto le operazioni di and logico bit a bit sono molto veloci in hardware. Tuttavia, per *facilitare la gestione* da parte degli utenti, si utilizza una *notazione più semplice e diretta* in cui viene specificata la dimensione del prefisso. In particolare, la notazione *CIDR* (Classless Inter-Domain Routing) prevede la seguente forma: $ddd.ddd.ddd.ddd/m$, dove ddd è un valore decimale e m è il numero di bit del prefisso. Dopo aver ricevuto il prefisso CIDR da un ISP, un cliente può assegnare liberamente gli indirizzi ai propri utenti.

Svantaggi dell'indirizzamento classless.

- Maggiore informazione da memorizzare nei router e conseguenti operazioni da svolgere per il processing di un pacchetto.
- Poiché la divisione tra prefisso e suffisso è arbitraria, usando la notazione decimale puntata non è sempre facile riuscire a leggere gli indirizzi.

❖ Indirizzi IP speciali

Introduzione.

Il protocollo IP definisce una serie di *indirizzi IP speciali riservati*, ovvero indirizzi che non possono essere assegnati agli host. Tra questi indirizzi troviamo quelli di: rete, directed broadcast, limited broadcast, questo host, loopback.

Prefix	Suffix	Type Of Address	Purpose
all-0s	all-0s	this computer	used during bootstrap
network	all-0s	network	identifies a network
network	all-1s	directed broadcast	broadcast on specified net
all-1s	all-1s	limited broadcast	broadcast on local net
127/8	any	loopback	testing

Indirizzo di rete.

Avere un indirizzo che denota il solo prefisso assegnato ad una rete può essere utile. L'indirizzo IP riservato che ha l'*host address posto a zero* viene utilizzato per identificare la rete (ad esempio l'indirizzo 128.211.0.16/28 identifica una rete, in quanto i bit oltre il 28-esimo sono posti a zero). Un indirizzo di rete, quindi, non può mai comparire come indirizzo di destinazione di un pacchetto.

Directed broadcast.

Questo indirizzo serve per semplificare il broadcasting (invio a tutti). Per ciascuna rete, infatti, viene definito un indirizzo di broadcast diretto. Un pacchetto con indirizzo di broadcast diretto viaggia su Internet finché non raggiunge la rete specificata, dove viene poi consegnato a tutti gli host. L'indirizzo di broadcast diretto è costruito ponendo tutti i *bit del suffisso a 1*.

Limited broadcast.

Il broadcast limitato si riferisce al broadcast verso gli host connessi direttamente alla rete a cui è connesso l'host che invia il pacchetto. L'indirizzo è formato ponendo *tutti i 32 bit a 1* (255.255.255.255), ed è utilizzato durante lo startup del sistema, cioè quando l'host non conosce ancora il suo indirizzo di rete. Il pacchetto con indirizzo di broadcast limitato raggiungerà tutti gli host appartenenti alla stessa rete locale della sorgente.

Questo host.

Prima di ricevere o inviare pacchetti su Internet, un host deve conoscere il proprio indirizzo IP. Lo stack TCP/IP contiene dei protocolli che possono essere utilizzati per ottenere un indirizzo IP automaticamente all'accensione dell'host, ma tali protocolli utilizzano comunque il protocollo IP per comunicare. Tuttavia, un host non può indicare un indirizzo IP sorgente corretto se non lo possiede ancora. A tale scopo, l'indirizzo con *tutti i 32 bit a 0* viene riservato per indicare "questo host".

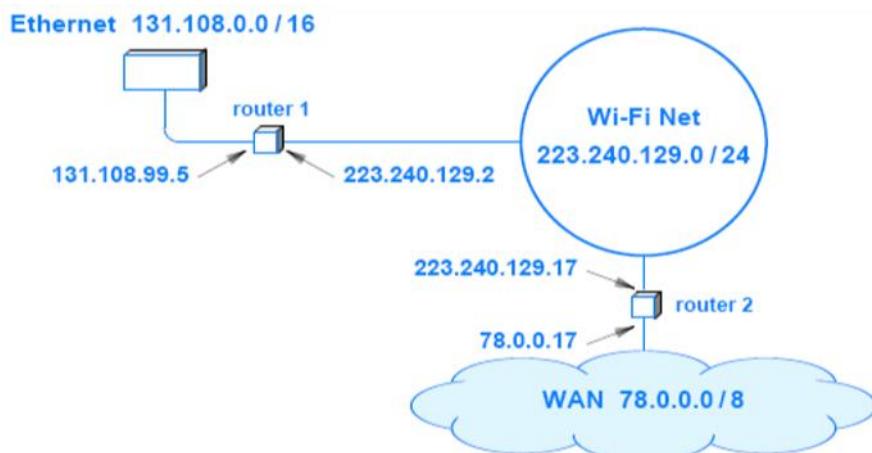
Loopback.

L'indirizzo di loopback è usato per testare applicazioni che comunicano attraverso una rete. Invece di eseguire ciascuna applicazione su host separati, si può farle eseguire entrambe su un solo host e farle comunicare tramite gli indirizzi di loopback. Così, quando un'applicazione invia dati ad un'altra, i dati viaggiano lungo lo stack protocollare fino al livello IP e il livello IP rigira il pacchetto, passando di nuovo attraverso lo stack, all'altra applicazione. In realtà, nessun pacchetto viene di fatto trasmesso, in quanto è lo strato software che gestisce il protocollo IP a rigirare i pacchetti da un'applicazione ad un'altra. Il protocollo IP riserva il prefisso 127.0.0.0/8 per il loopback, mentre il suffisso è irrilevante. Di solito si utilizza il primo disponibile, ovvero 127.0.0.1. L'indirizzo di loopback non appare mai nei pacchetti che viaggiano sulla rete.

❖ Principio di indirizzamento

Indirizzi del router.

A ciascun *router* vengono assegnati *2 o più indirizzi IP*, uno per ogni rete a cui è connesso. Per comprenderne il motivo, occorre ricordare che un router connette diverse reti ciascuna con un prefisso unico, e quindi un solo indirizzo IP non è sufficiente per ogni singolo router. Lo schema degli indirizzi IP può essere riassunto con un *principio*: *un indirizzo IP non identifica un computer specifico bensì la connessione tra un computer e la rete*.



PROTOCOLLI DI SUPPORTO

❖ Protocollo ICMP (error reporting)

Internet Control Message Protocol.

Al protocollo IP è stato associato un *protocollo complementare* chiamato *ICMP*. Questo protocollo è usato principalmente per inviare *messaggi di errore* alla sorgente in caso di problemi. IP e ICMP dipendono l'uno dall'altro. Infatti, IP dipende da ICMP per segnalare eventuali errori e ICMP utilizza IP per trasportare i messaggi di errore.

Messaggi ICMP.

ICMP contiene due tipi di *messaggi*:

- Messaggi per la *segnalazione di errori* (come Time Exceeded e Destination Unreachable).
- Messaggi per *ottenere informazioni* (come Echo Request e Echo Reply).

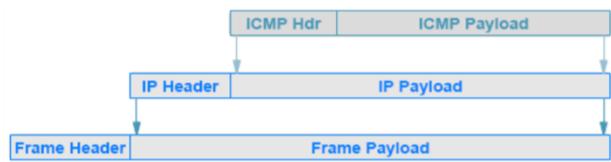
In particolare, Echo Request e Echo Reply sono usati dall'applicazione *ping* per testare la connessione. Quando un host riceve un messaggio di Echo Request, il software ICMP sull'host invia un messaggio di Echo Reply che trasporta gli stessi dati del messaggio di richiesta.

Trasporto e formato dei messaggi.

ICMP usa IP per trasportare i propri messaggi. Quando un router ha un messaggio ICMP da inviare, crea un *datagramma IP* e mette nel *payload* il *messaggio ICMP*. Il datagramma verrà poi inviato come al solito. Inoltre, i messaggi ICMP *non hanno una priorità* particolare, infatti sono inviati come ogni altro datagramma, ma con una sola eccezione: se un messaggio ICMP di errore causa un errore, non viene inviato nessun messaggio di errore (per il *rischio di effetto valanga*).

Number	Type	Purpose
0	Echo Reply	Used by the ping program
3	Dest. Unreachable	Datagram could not be delivered
5	Redirect	Host must change a route
8	Echo	Used by the ping program
11	Time Exceeded	TTL expired or fragments timed out
12	Parameter Problem	IP header is incorrect
30	Traceroute	Used by the traceroute program

Messaggi ICMP



Formato dei messaggi ICMP

❖ Protocollo DHCP (bootstrapping)

Software di protocollo.

Chi gestisce un *router* deve specificare dei *valori iniziali*, tra cui l'*indirizzo di ciascuna interfaccia di rete*, quale *software di protocollo* utilizzare e i valori iniziali delle *tabelle di forwarding*. Tale configurazione viene salvata e caricata dal router durante lo startup. La configurazione degli *host* di solito segue un processo noto come *bootstrapping*. Il protocollo che era stato ideato per permettere ad un host di ottenere una serie di parametri con una singola richiesta si chiamava *BOOTP* (Bootstrap Protocol). Attualmente, invece, è il protocollo *DHCP* ad essere usato per fornire la maggior parte delle informazioni di configurazione necessarie.

Dynamic Host Configuration Protocol.

Sono stati creati diversi meccanismi per permettere ad un host di ottenere i diversi parametri, ma ciascuno di essi veniva usato indipendentemente. DHCP racchiude molti di tali meccanismi e permette ad un host di connettersi ad una rete e ottenere l'indirizzo IP e altre informazioni automaticamente (*plug-and-play networking*).

Richiesta di un indirizzo IP.

Quando un *host* si accende, invia in *broadcast* una richiesta (*DHCP discover*). Il *server DHCP*, invia una risposta (*DHCP offer*) per dire che sta offrendo un indirizzo a quel client. È possibile configurare il server DHCP per fornire *due tipi di indirizzi*: quelli statici, cioè quelli assegnati permanentemente (come nel caso di *BOOTP*); oppure quelli *dinamici*, cioè quelli scelti da un insieme allocato appositamente. Tipicamente, gli indirizzi statici sono assegnati ai server, mentre gli indirizzi dinamici sono assegnati ad host generici.

Rilascio ed estensione.

Gli indirizzi dinamici vengono assegnati solo per un *periodo di tempo limitato* (*lease*), per permettere al *server DHCP* di tornare in possesso degli indirizzi. Quando il periodo di *lease scade*, il server considera l'indirizzo come disponibile per un'eventuale nuova assegnazione e un *host* può *liberare l'indirizzo o rinegoziare con il server DHCP* l'estensione del periodo. Di solito, il server DHCP approva le richieste di estensione, permettendo all'host di continuare a lavorare senza interruzioni. In ogni caso, un server DHCP può essere configurato per negare l'estensione per ragioni tecniche o amministrative. Se il server nega l'estensione, l'host deve smettere di usare l'indirizzo.

Ottimizzazioni del protocollo DHCP.

DHCP è stato progettato in modo che la *perdita* o la *duplicazione* di pacchetti DHCP non provochi una *configurazione errata*. Quindi: se l'host non riceve risposta, ritrasmette la richiesta; e se arriva una risposta duplicata, l'host ignora la replica. Inoltre, esiste il *caching dell'indirizzo del server*, ovvero una volta che l'host ha trovato il server DHCP se lo memorizza per utilizzi futuri. Infine, il server DHCP *limita la sincronizzazione delle richieste*, cioè usa delle tecniche per prevenire la ricezione di richieste in contemporanea.

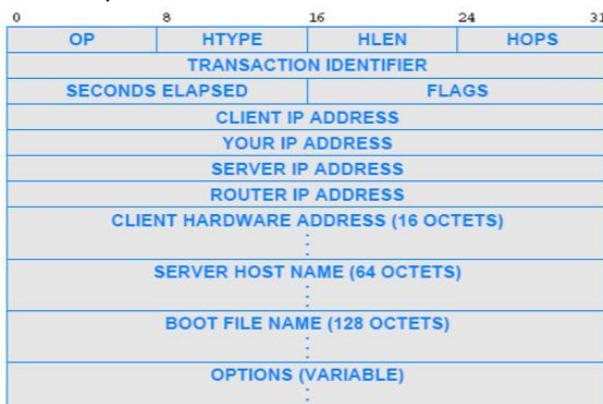
Formato dei messaggi DHCP.

DHCP adotta una versione leggermente modificata del formato dei messaggi *BOOTP*:

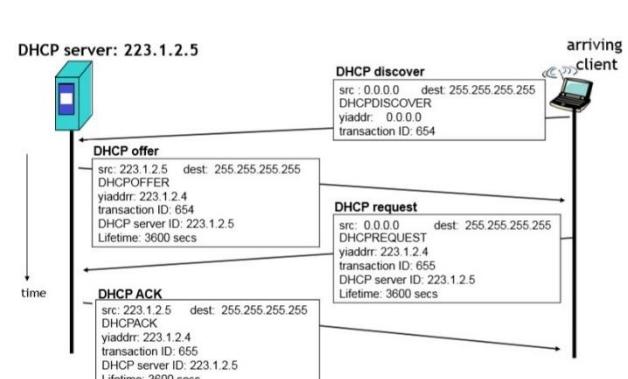
- *OP* [8 bit], indica il tipo dell'operazione contenuta nel messaggio (Discover, Offer, Request, ACK).
- *HTYPE* e *HLEN* [2 * 8 bit], indicano il tipo di hardware della rete e la lunghezza dell'indirizzo hardware.
- *HOPS* [8 bit], indica a quanti server rigirare la richiesta.
- *TRANSACTION ID* [32 bit], è il valore usato dall'host per capire se la risposta è riferita ad una sua richiesta.
- *SECONDS ELAPSED* [16 bit], indica quanti secondi sono passati dall'avvio dell'host.
- *FLAGS* [16 bit], indica se l'host può ricevere messaggi broadcast o risposte dirette.

I campi finali sono usati per trasportare informazioni verso l'host:

- Se l'host non conosce il proprio indirizzo IP, il server glielo fornisce usando il campo *YOUR IP ADDRESS*.
- Il server usa i campi *SERVER IP ADDRESS* e *SERVER HOST NAME* per fornire all'host la propria posizione.
- Il campo *ROUTER IP ADDRESS* contiene l'indirizzo IP del router di default.



Formato dei messaggi DHCP



Richiesta di un indirizzo IP

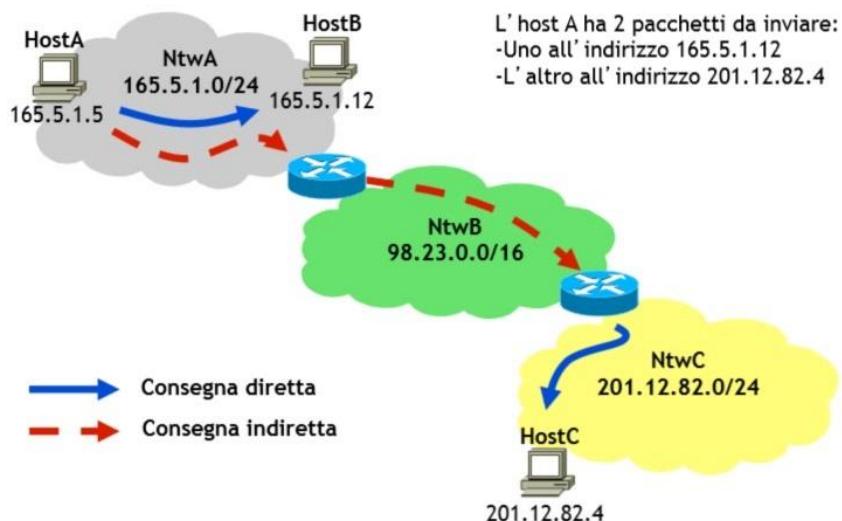
ALGORITMI DI ROUTING

❖ Introduzione

Consegna diretta/indiretta.

Quando un host vuole consegnare un messaggio ad un altro host, bisogna considerare due casi:

- Se gli host appartengono alla stessa *rete* la consegna è *diretta*, ovvero il primo host ottiene l'indirizzo IP dal server DNS, controlla se l'indirizzo appartiene alla stessa rete (tramite il prefisso) e in tal caso invia i dati al secondo host.
- Se gli host appartengono a *reti diverse* la consegna è *indiretta*, ovvero il primo host passa il messaggio al router il quale si farà carico di inviarlo verso il secondo host (i passi intermedi per raggiungerlo vengono gestiti dagli algoritmi di routing).



Routing.

Poiché un router è connesso a diverse reti, si pone il problema di come fare a sapere a chi consegnare un pacchetto avente come destinazione un host esterno alla rete (consegna indiretta). La soluzione è l'utilizzo del *routing*, cioè il processo di scoperta del cammino che va da una sorgente ad ogni destinazione nella rete. Il routing gestisce le *problematiche* relative a: quale cammino prendere; esistenza di cammini minimi (più corti e/o più veloci); guasti sul cammino individuato.

Tabelle di routing.

I protocolli di routing si occupano di gestire le *tabelle di routing* presenti nei router. Tali tabelle indicano, per ogni destinazione, qual è la porta di output sulla quale inviare il pacchetto. Per effettuare decisioni locali corrette, ciascun router deve conoscere qualcosa sullo *stato globale della rete* che, siccome è intrinsecamente grande e dinamica, rende di difficile reperibilità tali informazioni.

Gradi di libertà nelle decisioni.

- Routing *centralizzato* (semplice, ma propenso a guasti e congestioni) vs *distribuito*
- Scambio di informazioni *globale* (costoso) vs *locale*.
- Scambio di informazioni *statico* (bordi della rete) vs *dinamico* (core della rete).
- Scambio di informazioni *stocastico* (no oscillazioni, aumento pacchetti fuori sequenza) vs deterministico.
- Cammini *singoli* (primari) vs *multipli* (alternativi).
- *State-dependent* (il calcolo dipende dallo stato della rete) vs *state-independent*.

Algoritmi di routing.

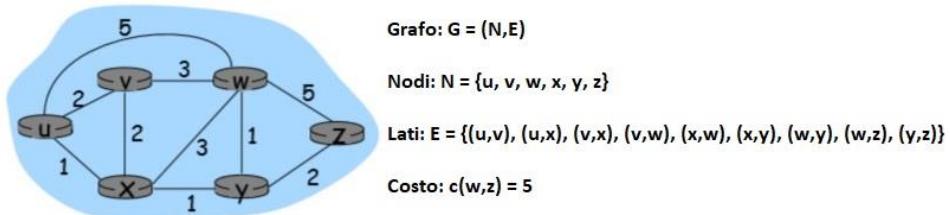
Un algoritmo di routing è un algoritmo che trova il *cammino a costo minimo*. Per assicurare che tutti i router mantengano le informazioni su come raggiungere ogni possibile destinazione, bisogna utilizzare un *protocollo di propagazione dei cammini*. Tale protocollo si occupa di scambiare le informazioni con gli altri router, cioè quando un router viene a sapere di cambiamenti nei cammini, aggiorna la propria tabella di routing. Poiché i router scambiano informazioni periodicamente, la tabella di routing locale viene aggiornata continuamente.

Requisiti degli algoritmi.

- *Minimizzare le tabelle di routing*: per velocizzare il look-up (cioè data la destinazione, trovare il next hop);
- *Minimizzare i dati da scambiare*.
- *Minimizzare il numero e la frequenza dei messaggi di controllo*.
- *Robustezza*: per evitare buchi neri, routing loop e oscillazioni.
- Utilizzo del *cammino ottimo*.

Grafi.

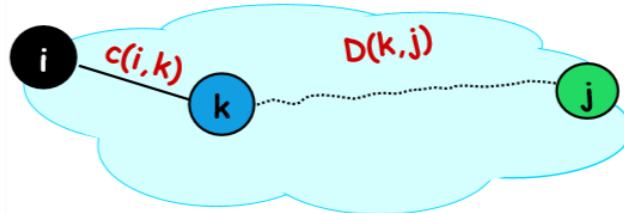
Per rappresentare in maniera astratta come funziona un algoritmo di routing, si usano i grafi. Ogni *nodo* di un grafo è un *router* e ogni *lato* rappresenta un *collegamento* tra due reti che ha un determinato *costo*.



❖ Algoritmi Distance Vector (DV)

Approccio.

Si basano sul *criterio di consistenza*, ovvero che una porzione di cammino minimo è anche il cammino minimo tra i nodi che delimitano tale porzione. Ad esempio, se $c(i,k)$ è il costo da i a k (collegamento diretto) e $D(k,j)$ è il costo del cammino minimo tra j e k , allora $D(i,j)$ (costo del cammino minimo tra i e j) è pari a $c(i,k) + D(k,j)$.



Inizializzazione.

Gli algoritmi DV mantengono: l'insieme dei valori $D(i, *)$, cioè il vettore delle distanze del nodo i ; e il valore del *next-hop*, che viene preso dalla tabella di forwarding per ogni destinazione.

Inizializzazione delle distanze:	Inizializzazione dei next-hop:
<ul style="list-style-type: none"> • $D(i,i) = 0$. • $D(i,k) = c(i,k)$ se k è un vicino diretto. • $D(i,j) = \text{INFINITO}$ per tutti gli altri nodi. 	<ul style="list-style-type: none"> • $\text{next-hop}(i) = i$. • $\text{next-hop}(k) = k$ se k è un vicino diretto. • $\text{next-hop}(j) = \text{UNKNOWN}$ altrimenti.

Scambio di messaggi.

Ad ogni iterazione, ciascun nodo invia il proprio vettore delle distanze $D(i, *)$ ai vicini diretti e, di conseguenza, riceve i vettori delle distanze dai propri vicini diretti. Per ogni vicino diretto k , se $c(i,k) + D(k,j) < D(i,j)$, allora: $D(i,j) = c(i,k) + D(k,j)$ e $\text{next-hop}(j) = k$.

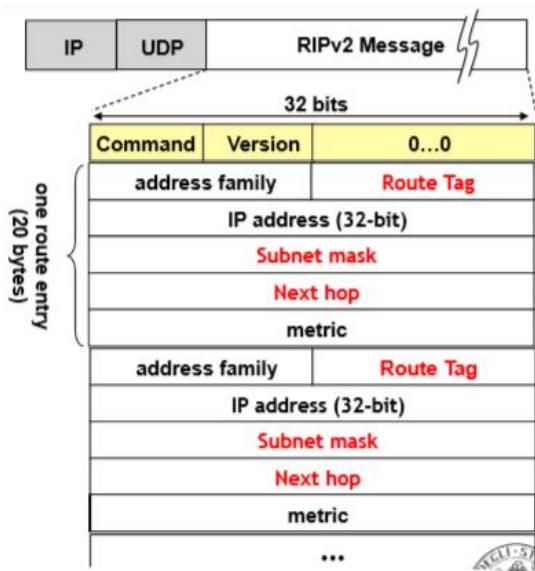
Iterazioni.

L'idea alla base degli algoritmi DV è che, periodicamente, *ogni nodo invia ai propri vicini il vettore delle distanze*. Per sapere quando inviarlo, ciascun nodo: *aspetta* che cambi il costo di un collegamento locale o di ricevere un messaggio da un vicino (approccio *asincrono*), *ricalcola* la stima delle distanze, se il DV è cambiato lo *notifica* ai vicini (approccio *distribuito*).

Routing Information Protocol (RIP).

I *messaggi* usati dal protocollo di routing RIPv2 possono contenere fino a 25 *entry* per le tabelle di routing, ognuna con il seguente *formato*:

- *Route Tag*, viene usato per trasportare informazioni di altri protocolli di routing.
- *IP address*, indica l'indirizzo della rete a cui si riferisce l'*entry*.
- *Subnet Mask*, indica la maschera di sottorete della rete identificata dall'indirizzo IP.
- *Next hop*, identifica un indirizzo di next-hop migliore rispetto a quello pubblicizzato dal router (se non esiste, il campo viene impostato a zero).



Riassunto.

- Ciascun router ha una *visione limitata* della topologia della rete.
- Data una destinazione è possibile individuare il *miglior next-hop*.
- Il cammino end-to-end è il risultato della *composizione di tutte le scelte di next-hop*.
- *Non richiede metriche uniformi* tra tutti i router.
- In caso di guasti, *gli errori si propagano su tutta la rete*.

❖ Algoritmi Link State (LS)

Confronto con DV.

Negli algoritmi DV, ciascun nodo ha *visibilità locale* e perciò le informazioni sul routing sono desunte dalle informazioni ottenute dai *vicini diretti* (la struttura del grafo che rappresenta la rete non viene specificata). Gli algoritmi LS, viceversa, cercano di ottenere una *visione globale*.

Approccio (Dijkstra).

L'approccio degli algoritmi LS è *iterativo*, ma si prende come riferimento la destinazione e i suoi predecessori. Vale quindi una versione alternativa del criterio di consistenza: $D(i,j) = D(i,k) + c(k,j)$. Ciascun nodo colleziona prima tutti i link state $c(*,*)$ e successivamente applica l'algoritmo di Dijkstra al grafo ottenuto.

Informazioni.

Dopo ciascuna iterazione, l'algoritmo trova una nuova destinazione j e il cammino minimo verso tale destinazione. Quindi, dopo m iterazioni, l'algoritmo ha esplorato i cammini fino a m hop dal nodo i . L'algoritmo di Dijkstra al nodo i mantiene *due insiemi*: l'insieme N che contiene i *nodi per cui è stato trovato il cammino minimo* e l'insieme M che contiene *tutti gli altri nodi*. Per tutti i nodi k , vengono mantenuti *due valori*: $D(i,k)$ che è il valore aggiornato della *distanza da i a k* e $p(k)$ che è il *nodo predecessore di k* lungo il cammino minimo da i .

Inizializzazione.

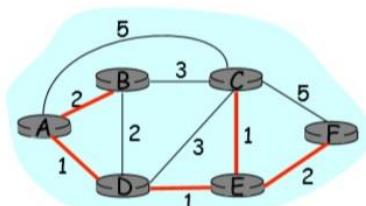
All'inizio l'insieme N ha solo il nodo i e l'insieme M ha tutti gli altri nodi. Al termine dell'esecuzione dell'algoritmo, l'insieme N conterrà tutti i nodi e l'insieme M sarà vuoto.

Inizializzazione degli insiemi:	Inizializzazione dei valori:
<ul style="list-style-type: none"> $N = \{i\}$ e $\text{next-hop}(i) = i$. $M = \{j \mid j \text{ diverso da } i\}$. 	<ul style="list-style-type: none"> $D(i,i) = 0$ e $p(i) = i$. $D(i,k) = \infty$ e $p(k) = \text{UNKNOWN}$ se k è un vicino di i. $D(i,k) = \text{INFINITO}$ e $p(k) = \text{UNKNOWN}$ altrimenti

Iterazioni.

- Un nodo j viene spostato dall'insieme M all'insieme N secondo il seguente criterio: j ha distanza minima da i tra tutti i nodi in M (in caso di distanze uguali, la scelta è casuale).
- $\text{Next-hop}(j) =$
 - Il vicino di i sul cammino minimo tra i e j .
 - $\text{next-hop}(p(j))$, se $p(j)$ è diverso da i .
 - j , se $p(j)$ è uguale a i .
- Se $D(i,k) < D(i,j) + c(j,k)$, allora la distanza di tutti i vicini k del nodo j nell'insieme M ($D(i,k)$) viene impostata a $D(i,j) + c(j,k)$ e $p(k)$ viene impostato a j .

Step	insieme N	$D(B),p(B)$	$D(C),p(C)$	$D(D),p(D)$	$D(E),p(E)$	$D(F),p(F)$
0	A	2,A	5,A	1,A	infinity	infinity
1	AD	2,A	4,D		2,D	infinity
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
5	ADEBCF					



Riassunto.

- Le *informazioni* sulla topologia sono *inviate su tutta la rete*.
- Il *miglior cammino* viene *calcolato da ciascun router localmente e determina il next-hop*.
- Funziona solo se la *metrica è condivisa e uniforme*.
- In caso di guasti, ciascun nodo calcola solamente la propria tabella e quindi *gli errori non si propagano*.

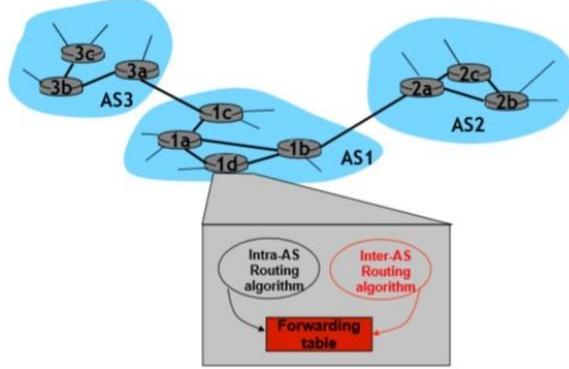
❖ Considerazioni finali

Routing gerarchico.

Il routing che abbiamo visto finora fornisce una *visione idealizzata*: tutti i router sono identici e la rete è piatta. In pratica, la situazione reale è ben diversa perché, con 200 milioni di possibili destinazioni (*scalabilità*), sarebbe impossibile memorizzare tutte le destinazioni nelle tabelle di routing e gli scambi di tabelle così grandi saturerebbero i collegamenti. Inoltre, in Internet (rete delle reti) ciascun amministratore di rete ha il controllo del routing nella propria rete (*autonomia amministrativa*).

AS interconnessi.

Una soluzione è rappresentata dall'aggregazione dei router in regioni, dette *autonomous systems* (AS). I router nello stesso AS usano lo stesso protocollo di routing (detto *protocollo intra-AS*), mentre i router in AS differenti possono usare differenti protocolli intra-AS. Il router di bordo (*gateway*) è un collegamento tra un router di un AS con un router di un altro AS. In questa configurazione le *tabelle di forwarding* sono configurate sia dagli algoritmi di routing intra-AS (per le destinazioni interne) che da quelli *inter-AS* (per le destinazioni esterne).



Compiti dei protocolli inter-AS.

Si supponga che un router in AS1 riceva un datagramma con destinazione esterna ad AS1. Il router dovrebbe rigirare il pacchetto ad un gateway, ma a quale? Per poter decidere, la rete AS1 dovrà: conoscere quali *destinazioni* sono *raggiungibili* attraverso AS2 e quali attraverso AS3 e *propagare* questa *informazione di raggiungibilità* a tutti i router in AS1 (*routing inter-AS*). Ad esempio, si supponga che AS1 sappia (attraverso un protocollo inter-AS) che una certa rete è raggiungibile attraverso AS3 (gateway 1c), ma non attraverso AS2. I protocolli inter-AS propagheranno l'*informazione di raggiungibilità* a tutti i router interni, e quindi il *router 1d*, ad esempio, determinerà attraverso i protocolli intra-AS che il next hop per raggiungere 1c è il router 1a e imposterà una nuova entry nella sua tabella di routing. Si supponga ora che AS1 sappia (attraverso un protocollo inter-AS) che una certa rete è raggiungibile attraverso AS3 (gateway 1c) e attraverso AS2 (gateway 1b). Per configurare la sua tabella di routing, il router 1d dovrebbe determinare verso quale gateway inoltrare il pacchetto. Ma siccome può *scegliere tra più AS*, in questo caso userà la tecnica “hot potato routing”, cioè invierà il pacchetto al gateway più vicino.

Protocolli di routing intra-AS e inter-AS.

Un esempio di protocollo di routing *intra-AS* (IGP, Interior Gateway Protocols) è il *RIP*. Lo standard de facto per i protocolli di routing *inter-AS*, invece, è il *BGP* (Border Gateway Protocol) il quale permette alle reti di far conoscere la propria esistenza al resto di Internet e fornisce agli AS i mezzi per: *ottenere le informazioni di raggiungibilità* delle diverse reti vicine; *propagare le informazioni di raggiungibilità* a tutti i router interni degli AS; *determinare i migliori cammini* verso le reti basandosi su informazioni di raggiungibilità e policy. Inoltre, si ricorda che è necessario *differenziare* tra routing intra-AS e inter-AS per motivi di:

- *Scalabilità*, in quanto il routing gerarchico riduce la dimensione delle tabelle e il traffico di update.
- *Policy*, perché si ha un singolo dominio amministrativo e policy uniformi (nel caso di intra-AS) e perché l'amministratore di una rete può controllare come il traffico viene instradato (nel caso di inter-AS).
- *Prestazioni*, in quanto il routing può essere focalizzato sulle prestazioni (nel caso di intra-AS) oppure le policy possono essere più importanti delle prestazioni (nel caso di inter-AS).

SOLUZIONI PER LA CARENZA DI INDIRIZZI IP

❖ Introduzione

Le ragioni del cambiamento.

Il protocollo IP usa 32 bit per l'indirizzo. Quando è stato definito il protocollo IP, lo spazio di indirizzamento sembrava sufficiente, ma la *crescita di Internet* è stata *esponenziale*. Quindi sarebbe necessario un *cambio di protocollo* per *aumentare lo spazio di indirizzamento* e *introdurre funzionalità* per applicazioni che non erano state previste durante la progettazione della versione precedente.

La difficoltà a cambiare.

Si è iniziato a lavorare su una nuova versione di IP fin dal 1993. Ma siccome non vi erano emergenze, IP non è stato cambiato. Inoltre, bisogna pensare all'importanza di IP e al costo del cambiamento infatti, secondo il *modello a classidra*, sarebbe necessario cambiare non solo gli end-host ma anche i router. Tuttavia, relativamente al problema dello spazio di indirizzamento, esiste una soluzione temporanea.

❖ NAT

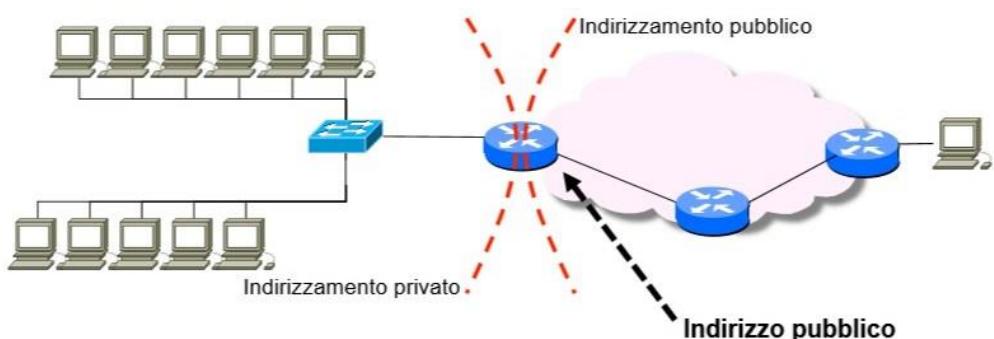
Indirizzi privati.

Sono stati definiti alcuni range di indirizzi all'interno dello spazio di indirizzamento IP da utilizzare solamente in ambito privato (*private addresses* o *non-routable addresses*). Ogni volta che un router pubblico riceve un pacchetto destinato ad un indirizzo IP privato, viene segnalato un errore. Gli indirizzi privati si impiegano solitamente in *reti con pochi punti di connessione* ad Internet.

Prefisso	Indirizzo iniziale	Indirizzo finale
10.0.0.0/8	10.0.0.0	10.255.255.255
172.16.0.0/12	172.16.0.0	172.31.255.255
192.168.0.0/16	192.168.0.0	192.168.255.255
169.254.0.0/16	169.254.0.0	169.254.255.255

Network Address Translation.

Per permettere di *ricevere i pacchetti* all'interno della rete privata e quindi risolvere i *problemi di instradamento* tra una rete ad indirizzamento privato ed una rete ad indirizzamento pubblico, è necessario introdurre un'ulteriore *funzionalità sul bordo tra privato e pubblico*. Questa funzionalità è chiamata *NAT* e consiste nell'assegnare un *indirizzo pubblico* al *router di confine* tra privato e pubblico sull'interfaccia verso la rete esterna.

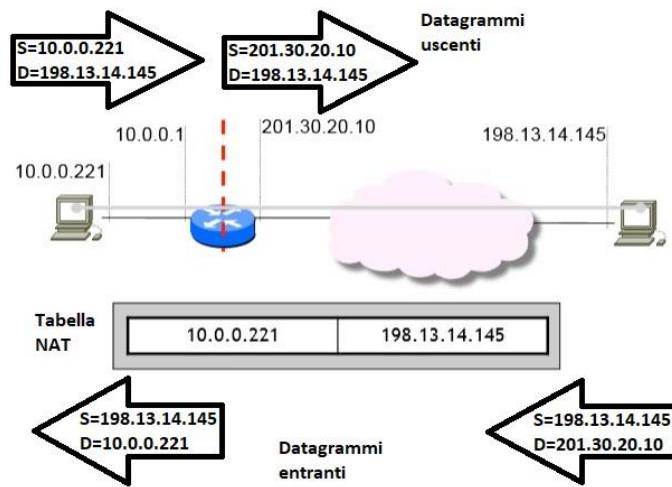


Datagrammi uscenti/entranti.

Il router NAT: traduce l'indirizzo IP dei *datagrammi uscenti* sostituendo l'indirizzo sorgente con il proprio indirizzo pubblico; e traduce l'indirizzo IP dei *datagrammi entranti* sostituendo l'indirizzo destinazione con l'indirizzo privato dell'host corretto.

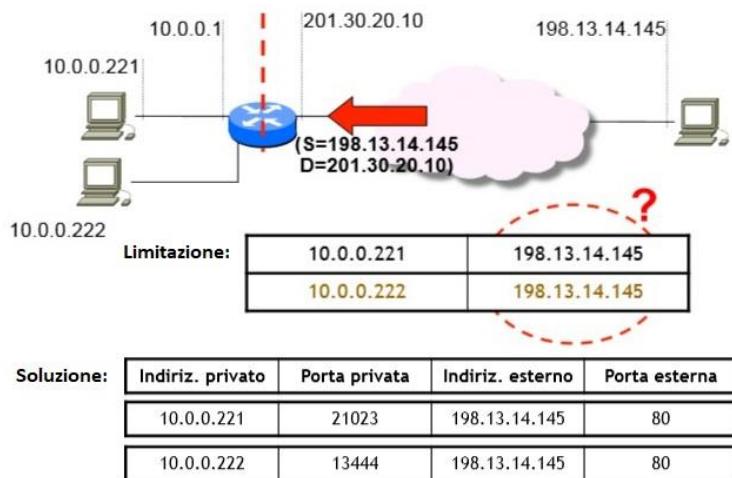
Tabella NAT.

Per tradurre l'indirizzo IP dei datagrammi entranti, il router NAT deve mantenere al suo interno una *tavella di mapping* tra indirizzo privato sorgente ed indirizzo pubblico destinazione. L'aggiornamento della tabella NAT si può basare su: *configurazione manuale*, cioè quando il gestore della rete configura in modo statico i record della tabella; o *datagrammi uscenti*, cioè quando i record vengono creati in modo dinamico ogni volta che un pacchetto verso una data destinazione attraversa il NAT. Nell'ultimo caso i record vengono cancellati con un meccanismo di *timeout* e si garantisce una *maggior sicurezza* non permettendo l'attivazione di una comunicazione dall'esterno.



Port-mapped NAT (NAPT).

Il NAT basato unicamente sull'indirizzo non permette a differenti host privati di connettersi contemporaneamente allo stesso host pubblico. La *soluzione* a questa *limitazione* è quella di far agire il *router NAT* da *gateway di livello 4*, ossia affidargli la *traduzione* sia dell'*indirizzo IP* che della *porta* (TCP/UDP).



Considerazioni finali.

Il NAT è compatibile con l'attuale architettura, quindi *solo il router di bordo deve essere aggiornato*. Il NAT permette di *limitare il numero di indirizzi pubblici* necessari, risolvendo il problema dello spazio di indirizzamento. Il NAT *non permette l'instaurazione di connessioni dall'esterno* che da un lato aumenta la sicurezza ma dall'altro limita le applicazioni.

❖ IPv6

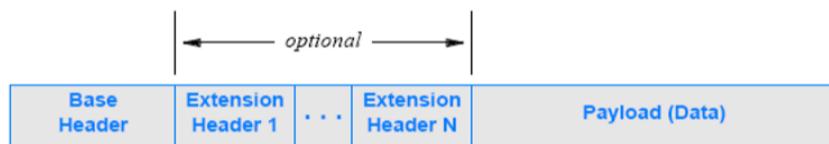
Caratteristiche.

IPv6 mantiene molte delle caratteristiche di IPv4 infatti, come IPv4, anche IPv6 è *connection-less* e l'header del datagramma IPv6 contiene anch'esso un *numero massimo di hop* che il datagramma può fare prima di essere scartato. Tuttavia, molti dettagli sono cambiati, come:

- La *dimensione degli indirizzi*, che raggiunge 128 bit in modo da rendere lo spazio di indirizzamento sufficiente a contenere eventuali crescite future.
- Il *formato dell'header*, che è completamente differente rispetto a quello di IPv4.
- L'introduzione del concetto di *Extension Header*, cioè il raggruppamento di informazioni in header separati (un datagramma IPv6 perciò avrà un header di base, zero o più Extension Header, un payload).
- Il *supporto del traffico real-time*, cioè l'introduzione di un meccanismo che permette di creare un cammino tra una sorgente e una destinazione e di associare i datagrammi a tale cammino (utilizzato da applicazioni audio e video o da applicazioni che richiedono una maggiore qualità del servizio).

Formato dei datagrammi IPv6.

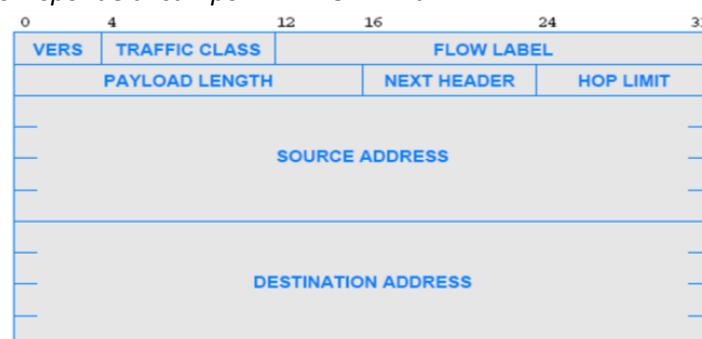
IPv6 è un *protocollo estensibile*. Ciò rende IPv6 *più flessibile* di IPv4 e permette di *aggiungere funzionalità* se necessario. Infatti, IPv6 permette alla sorgente di aggiungere informazioni addizionali al datagramma tramite gli Extension Header. Ciascun datagramma IPv6 inizia con un *header di base*, seguito da zero o più *Extension Header* (che possono essere più grandi dell'header di base), seguiti dal *payload*.



Formato dell'header di base.

Sebbene l'header IPv6 (40 byte) sia grande il doppio dell'header IPv4 (20 byte), contiene meno campi:

- VERS [4 bit], indica la *versione* (nel caso di IPv6 vale 6).
- TRAFFIC CLASS [8 bit], specifica la *classe di traffico* in base al tipo di traffico (differentiated services).
- FLOW LABEL [20 bit], usato per associare un *datagramma con un cammino specifico*.
- PAYLOAD LENGTH [16 bit], specifica la *dimensione del payload* (cioè i dati trasportati dopo l'header).
- NEXT HEADER [8 bit], specifica il *tipo di informazione che segue l'header* corrente (se ci sono extension header, ne indica il tipo; se non ci sono extension header, indica il tipo di dati trasportato nel payload).
- HOP LIMIT [8 bit], *corrisponde al campo TIME TO LIVE di IPv4*.



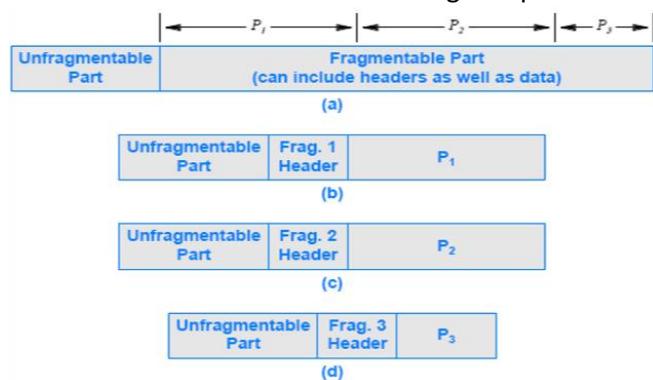
Extension Header.

Lo standard specifica un *valore unico* per ciascun Extension Header possibile, per garantire la non ambiguità nell'interpretazione del campo NEXT HEADER. Un nodo elabora gli header sequenzialmente, usando il campo NEXT HEADER per capire cosa segue. Al di là dell'header di base (che ha dimensione fissa), gli Extension Header possono avere *dimensione variabile*, ma devono contenere informazioni riguardo la propria dimensione (campo HEADER LEN).



Frammentazione.

La frammentazione di IPv6 è simile quella di IPv4, ma ci sono comunque delle differenze. *Come in IPv4*: il prefisso in ciascun datagramma viene copiato in ciascun frammento, e la dimensione del payload viene modificata in base alla Maximum Transmission Unit (MTU) della rete da attraversare. *Diversamente da IPv4* non esistono campi predeterminati nell'header di base per la frammentazione, ma bisogna aggiungere un Extension Header con le informazioni sulla frammentazione. Quindi la presenza stessa di un Extension Header di tipo "frammentazione" in un datagramma, indica che si tratta di un frammento. Inoltre, va ricordato che l'header di base e gli Extension Header che controllano il routing non possono essere frammentati.



Motivazioni per la definizione di header multipli.

- Motivazioni *economiche*: si risparmia spazio e si fa usare ad un datagramma solo un sottoinsieme limitato delle funzionalità disponibili.
- Motivazioni *di estensibilità*: è possibile aggiungere un ampio insieme di nuove funzionalità senza imporre che tutti gli header abbiano un campo predeterminato (basta aggiungere un nuovo Extension Header).

Indirizzamento.

Come con CIDR, la divisione tra prefisso e suffisso è arbitraria. IPv6 introduce il concetto di *gerarchia multilivello*, infatti, sebbene l'assegnazione degli indirizzi non è fissa, si può assumere che: il livello più alto corrisponde agli ISP e i livelli successivi corrispondono ad organizzazioni, aziende, siti specifici, e così via.

Indirizzi speciali.

- *Unicast* (singolo host): un datagramma inviato a tale indirizzo viene instradato sul cammino minimo.
- *Multicast* (insieme di host): viene consegnata una copia del datagramma a ciascun membro dell'insieme.
- *Anycast* (insieme di host con lo stesso prefisso): il datagramma viene consegnato ad un host qualsiasi.

Notazione esadecimale.

L'indirizzo IPv6 è composto da 128 bit. Per ridurre il numero di caratteri da scrivere per indicare un indirizzo, è stata introdotta una *notazione esadecimale*, in cui ciascun gruppo di 16 bit è scritto in esadecimale e viene separato da un altro gruppo usando ":". Le sequenze di zeri si possono comprimere (ad esempio FF0C:0:0:0:0:0:B1 diventa FF0C : : B1). Il *mapping* degli indirizzi IPv4 esistenti in indirizzi IPv6 viene fatto ponendo a zero i primi 96 bit.

IL LIVELLO DATA LINK

❖ Introduzione

Obiettivi e problematiche.

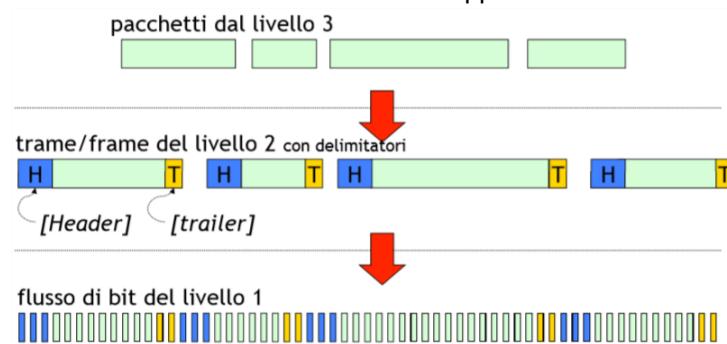
L'obiettivo principale del livello 2 è quello di fornire al livello di rete di due macchine adiacenti, cioè fisicamente connesse, un *canale di comunicazione* (cavo coassiale, doppino telefonico) il più possibile *affidabile*, per garantire che i bit siano ricevuti nello stesso ordine in cui vengono inviati. Per compiere questo obiettivo, come tutti i livelli OSI, il livello 2 offre dei *servizi* al livello superiore (livello di rete) e svolge una serie di *funzioni*. Inoltre, siccome un canale fisico non è del tutto ideale, possono insorgere *problematiche* quali: *errori di trasmissione* tra sorgente e destinazione; necessità di dover *gestire la velocità di trasmissione* dei dati; *ritardo di propagazione* non nullo.

Tipologie di servizi.

- *Connection-less senza acknowledge* (maggior parte delle LAN): non viene attivata nessuna connessione e si inviano le trame senza attendere alcun feedback dalla destinazione, ovvero se una trama viene persa non ci sono tentativi per recuperarla in quanto tale compito è lasciato ai livelli superiori.
- *Connection-less con acknowledge*: non viene attivata nessuna connessione, ma ogni trama inviata viene "riscontrata" in modo individuale.
- *Connection-oriented con acknowledge*: viene attivata una connessione che, al termine del trasferimento, viene abbattuta e ogni trama inviata viene "riscontrata" in modo individuale.

Funzioni – framing.

Il livello 2 riceve dal livello superiore dei pacchetti. Considerando che la lunghezza dei pacchetti e delle corrispondenti trame (frame) è variabile, che i sistemi non sono sincronizzati tra loro e che il livello 1 tratta solo bit, nasce il problema della *delimitazione delle trame*. La funzionalità di *framing* è dunque quella di *rendere distinguibile una trama dall'altra* attraverso l'uso di opportuni codici all'inizio e alla fine della trama.



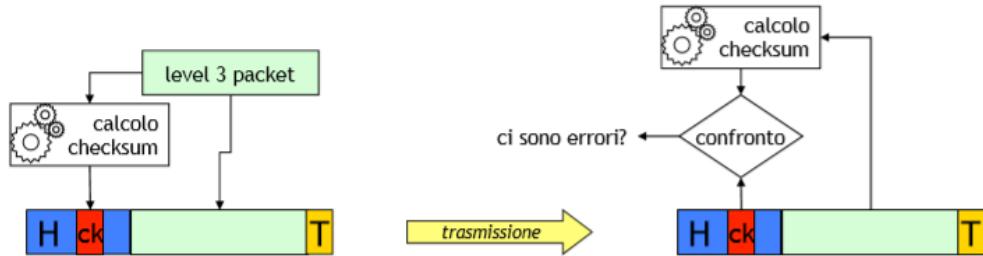
Modalità di framing.

Esistono diverse tecniche per implementare il framing. Si potrebbe *inserire intervalli temporali fra trame consecutive*, ma si avrebbe il problema che, per natura intrinseca, le reti di telecomunicazione non danno garanzie sul rispetto delle caratteristiche temporali delle informazioni trasmesse e quindi gli intervalli inseriti potrebbero essere espansi o ridotti generando problemi di ricezione. In alternativa si può *marcare inizio e fine di ogni trama* attraverso:

- *Character count*, cioè un campo nell'header che indica il numero di caratteri contenuti nella trama.
- *Bit stuffing*, cioè far iniziare e terminare ogni trama con uno speciale pattern di bit (01111110) chiamato byte di flag (se la trama contiene al suo interno il byte di flag, la sorgente aggiungerà uno "0" dopo aver scritto cinque "1" consecutivi e la destinazione toglierà tale "0" dopo aver letto i cinque "1" consecutivi).

Funzioni – rilevazione degli errori.

Il livello fisico offre un canale di trasmissione *non privo di errori*, infatti ci possono essere: errori sul singolo bit; replicazioni di bit; perdite di bit. Per la rilevazione di tali errori, il livello 2 inserisce nell'header di ogni trama un campo denominato *checksum*, che è il risultato di un calcolo fatto utilizzando i bit della trama. Una volta ricevuta la trama, la destinazione ripete questo calcolo e confronta il risultato con il checksum. Se i due valori coincidono, allora la destinazione può dedurre che la trama ricevuta è corretta.



Funzioni – controllo di flusso.

Siccome la sorgente trasmette le trame ad una velocità superiore a quella che la destinazione utilizza per accettare l'informazione, ci può essere congestione nel nodo destinazione. La soluzione a questo problema è quella di implementare il *controllo di flusso*, cioè gestire la *velocità di trasmissione* della sorgente. Per fare questo ci si basa su feedback inviati dalla destinazione che indicano la quantità di informazione che essa è ancora in grado di gestire e, eventualmente, di bloccare la trasmissione fino a un comando successivo.

❖ Il sotto-livello MAC

Introduzione di un nuovo sotto-livello.

Il livello 2 ha a che fare direttamente con il livello 1, cioè con il mezzo fisico. Tale mezzo fisico può essere dedicato (*reti punto-punto*) o condiviso (*reti broadcast*). Se il *mezzo fisico* è *condiviso*, nascono una serie di problematiche relative all'accesso a tale mezzo, come la selezione dell'host che ha il diritto di trasmettere sul mezzo condiviso e la competizione per la risorsa trasmissiva. Per gestire queste problematiche viene introdotto un *sotto-livello* al livello 2, chiamato *MAC* (Medium Access Control). Infine, quindi, il livello 2 avrà una *parte di collegamento dati* che gestisce le funzioni viste precedentemente, e una *parte MAC* che gestisce le politiche di accesso al mezzo fisico condiviso.

Tecniche di allocazione del canale.

Sono una *serie di regole* definite per poter utilizzare un mezzo condiviso, cioè un canale che può essere usato da più sorgenti. Infatti, se due sorgenti riescono a parlare contemporaneamente vi sarà una *collisione* e l'informazione andrà persa. Le tecniche di allocazione del canale possono essere statiche o dinamiche.

Allocazione statica.

Si ha quando alle diverse sorgenti viene data una *porzione* del mezzo trasmissivo. Il partizionamento può avvenire in base:

- *Al tempo* (TDM, Time Division Multiplexing), ogni sorgente ha a disposizione il mezzo per un determinato periodo.
- *Alla frequenza* (FDM, Frequency Division Multiplexing), ogni sorgente ha a disposizione una determinata frequenza.

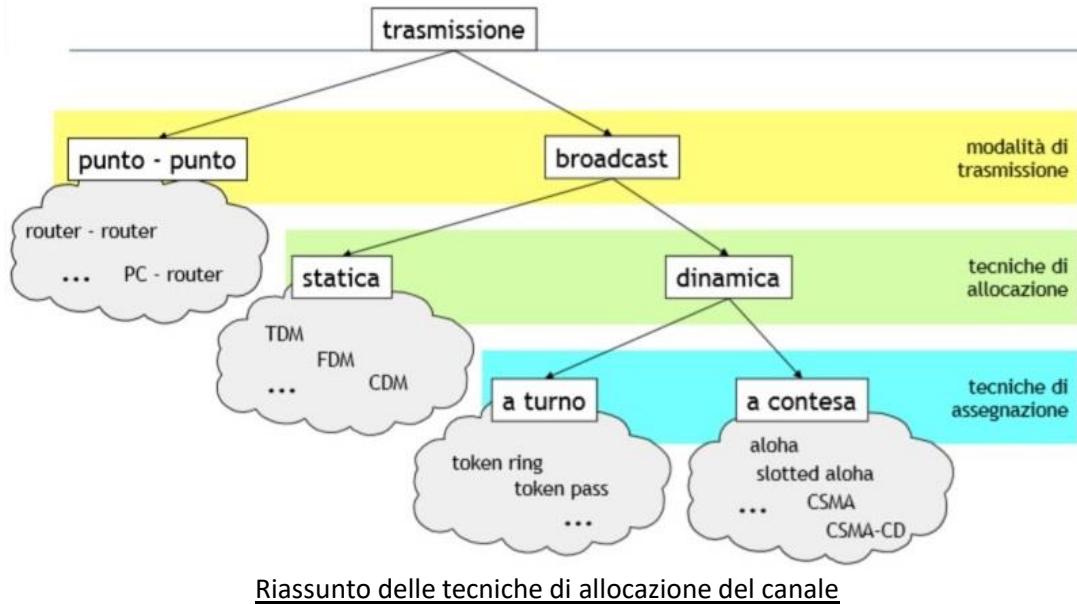
Queste tecniche sono di *semplice implementazione* e garantiscono una buona efficienza in situazioni con pochi utenti e con molto carico costante nel tempo. Tuttavia, in situazioni con molti utenti e con traffico discontinuo, generano una *scarsa efficienza* di utilizzo delle risorse trasmissive, in quanto le *risorse dedicate agli utenti "momentaneamente silenziosi"* vengono sprecate.

Allocazione dinamica.

Si ha quando il canale viene assegnato di volta in volta a chi ne fa *richiesta*. L'assegnazione può avvenire:

- *A turno*, quando viene distribuito il “permesso” di trasmettere (con durata decisa dalla sorgente).
- *A contesa*, quando ciascuna sorgente prova a trasmettere indipendentemente dalle altre.

Nel primo caso si presuppone la presenza di meccanismi per l'assegnazione del permesso di trasmettere (si crea overhead di gestione). Nel secondo caso, invece, non sono previsti meccanismi particolari. I *protocolli* che gestiscono la *trasmissione a contesa* sono generalmente i *più utilizzati*.



❖ Algoritmi di allocazione dinamica a contesa

Ipotesi.

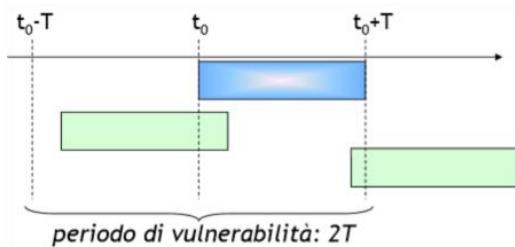
- *Single channel assumption*: c’è un unico canale per tutte le comunicazioni.
- *Station model*: ci sono N stazioni indipendenti che trasmettono trame di lunghezza fissa pari a T.
- *Collision assumption*: due trame contemporaneamente presenti sul canale generano collisione.
- *Tempo*
 - *continuo*: la trasmissione della trama può iniziare in qualunque istante.
 - *discreto (slotted)*: la trasmissione della trama può iniziare solo in istanti discreti.
- *Carrier sense*: le stazioni possono ascoltare il canale (anche prima di iniziare la trasmissione di una trama).

Algoritmo Pure ALOHA.

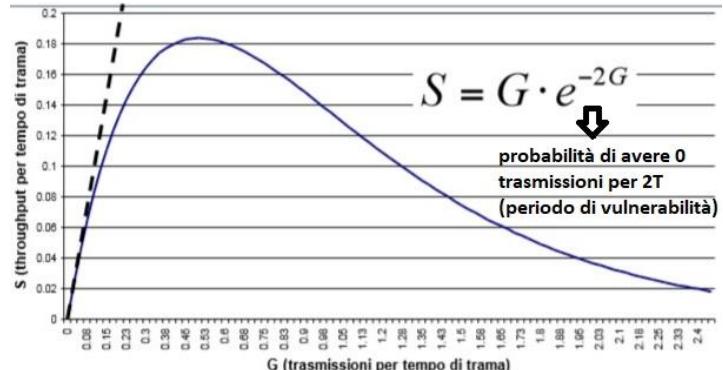
È stato definito nel 1970 all'università delle Hawaii. Il suo funzionamento è il seguente:

- Una *sorgente* può trasmettere una *trama* ogni qualvolta vi sono *dati da inviare* (tempo continuo).
- La sorgente *rileva*, ascoltando il canale, l'eventuale *collisione*.
- Se c’è stata collisione, la sorgente *aspetta* un tempo casuale e *rtrasmette* la trama (un tempo deterministico porterebbe ad una situazione di collisione all’infinito).

L'intervallo di tempo in cui può avvenire una collisione che invalida una trasmissione è detto *periodo di vulnerabilità* ed è pari al *doppio del tempo di trama*. Nel momento in cui una sorgente inizia a trasmettere (t_0), nessun'altra sorgente deve aver iniziato la trasmissione dopo l'istante di tempo t_0+T e nessun'altra sorgente deve iniziare la trasmissione fino a t_0+T . Le *prestazioni* di Pure ALOHA garantiscono di sfruttare al massimo il 19% degli slot liberi (nel caso in cui mediamente vengano generate 0,5 trasmissioni per tempo di trama).



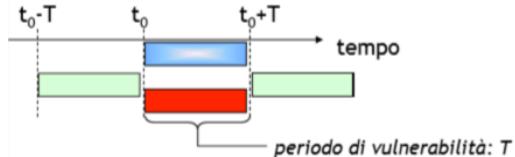
Periodo di vulnerabilità (Pure ALOHA)



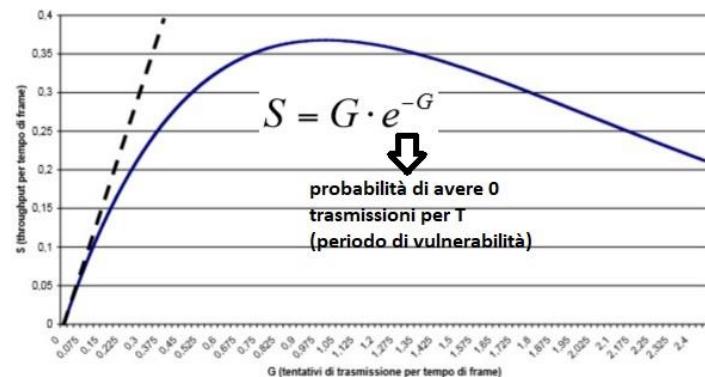
Prestazioni (Pure ALOHA)

Algoritmo Slotted ALOHA.

È stato proposto nel 1972 per duplicare la capacità di Pure ALOHA. In questa versione di ALOHA, infatti, il tempo viene suddiviso in *intervalli discreti*, permettendo così di dimezzare il periodo di vulnerabilità. Il suo funzionamento è uguale a quello di Pure ALOHA, con la differenza che la trasmissione di una trama può iniziare solo ad intervalli discreti. È quindi necessario sincronizzare le varie stazioni. Le *prestazioni* di Slotted ALOHA garantiscono di sfruttare al massimo il 37% degli slot liberi (nel caso in cui mediamente venga generata 1 trasmissione per tempo di trama).



Periodo di vulnerabilità (Slotted ALOHA)



Prestazioni (Slotted ALOHA)

Algoritmo CSMA.

L'algoritmo CSMA (Carrier Sense Multiple Access) è molto usato in ambito LAN, in quanto le stazioni, prima di iniziare a trasmettere, possono *ascoltare il canale* per verificare se c'è già una trasmissione in corso. Il funzionamento di CSMA è il seguente:

- Se il canale è *libero*, si trasmette.
- Se il canale è *occupato*, si rimanda la trasmissione ad un nuovo istante di tempo casuale (variante non-persistent) oppure si inizia a trasmettere nel momento in cui si libera il canale (variante persistent).
- Se c'è *collisione*, come in ALOHA, si attende un tempo casuale e poi si cerca di ritrasmettere.

Il *periodo di vulnerabilità* è legato al ritardo di propagazione del segnale (τ) e vale 2τ . Perciò se una stazione ha iniziato a trasmettere, ma il suo segnale non è ancora arrivato a tutte le stazioni, qualcun altro potrebbe iniziare la trasmissione. In generale, il CSMA viene usato in reti in cui il ritardo di propagazione è molto minore del tempo di trama.

CSMA – modalità p-persistent.

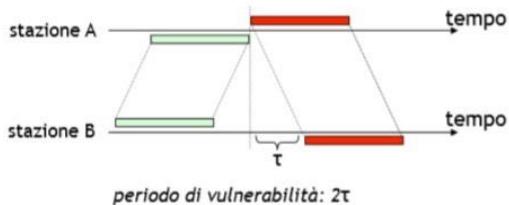
In questa modalità, il *tempo* viene suddiviso in *intervalli* di lunghezza pari al periodo di vulnerabilità (2τ). Il suo funzionamento è il seguente:

- Se il canale è *libero*, si può trasmettere con probabilità p :
 - Se si è deciso di *trasmettere*, si passa al punto 2.

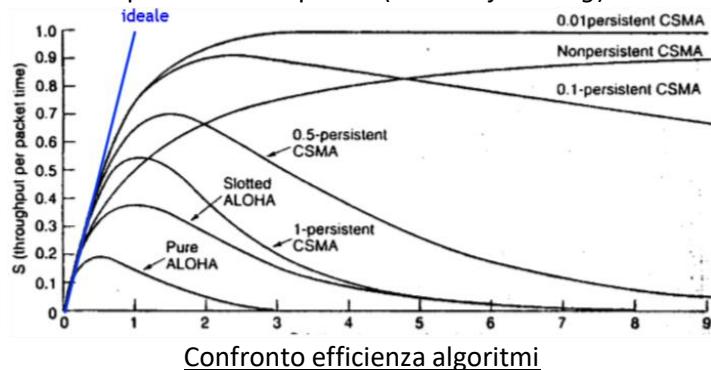
- Se si è deciso di *non trasmettere*, si attende un intervallo di tempo e si torna al punto 1.
- Se il canale è *occupato*, si attende un intervallo di tempo e si torna al punto 1.
- Se c'è *collisione*, si attende un tempo casuale e si torna al punto 1.

CSMA con Collision Detection.

Il CSMA-CD è un miglioramento del CSMA. La differenza con quest'ultimo sta nel fatto che, se la stazione che sta trasmettendo rileva una collisione, *interrompe immediatamente* la trasmissione. In questo modo, una volta rilevata la collisione, *non si spreca tempo* a trasmettere trame già corrotte. Inoltre, per far sentire a tutte le stazioni che vi è stata collisione, si trasmette una particolare sequenza (detta di *jamming*).



Periodo di vulnerabilità (CSMA)



❖ LAN estese

Local Area Networks.

La scelta di utilizzare mezzi condivisi per l'accesso al canale di trasmissione è stata fatta sia per necessità che per motivi economici. Grazie proprio a questi ultimi, tale tecnologia si è diffusa particolarmente nelle reti locali, o LAN. La rappresentazione tipica di una LAN è una *serie di stazioni* (PC) connesse ad un *segmento* di cavo (bus). L'*attenuazione del segnale* e la *disposizione spaziale delle stazioni* fanno sì che il segmento non possa essere troppo lungo. Da qui nasce il problema di come estendere le LAN. Per farlo, esistono 3 tipi di *apparati* (in ordine crescente di complessità): Repeater e Hub, Bridge, Switch.

Domini di collisione e broadcast.

Il *dominio di collisione* è la parte di rete per cui, se due stazioni trasmettono dati contemporaneamente, il segnale ricevuto dalle stazioni risulta danneggiato. Il *dominio di broadcast*, invece, è la parte di rete raggiunta da una trama avente come destinazione l'indirizzo di broadcast. Le stazioni appartenenti alla medesima rete di livello 2 condividono lo stesso dominio di broadcast, perciò gli apparati che estendono le LAN possono influire solo sul dominio di collisione.

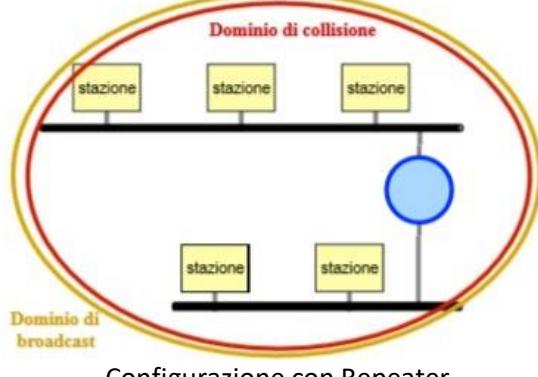
Repeater e Hub.

Sono apparati che intervengono solo a *livello fisico*. Infatti, ogni trama in arrivo da un segmento viene *replicata* in un altro segmento, amplificandone il segnale. Gli *Hub* sono dei Repeater che possono connettere *più di due segmenti*. In questo caso il segnale trasmesso da una stazione viene propagato a tutte le uscite dell'Hub e quindi le trame ricevute vengono copiate su tutte le porte. Se una configurazione di rete prevede l'utilizzo di Repeater o di Hub, il dominio di collisione coinciderà con il dominio di broadcast. Il problema legato a questo tipo di configurazioni è l'*eccessiva estensione del dominio di collisione*, in quanto è come se tutte le stazioni condividessero lo stesso mezzo fisico.

Bridge.

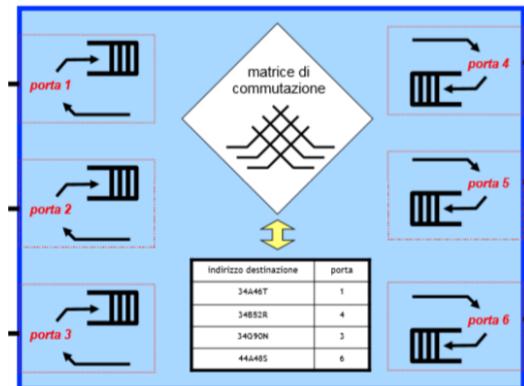
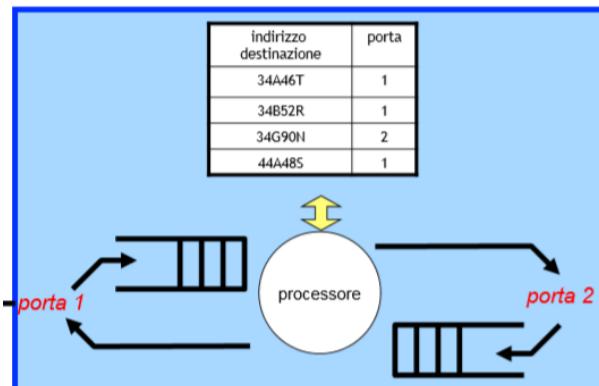
È un apparato che *collega due segmenti* di rete ed è dotato di *intelligenza*. Il Bridge, infatti, seleziona se ripetere una trama in un altro segmento in base ad una *tabella*, in cui c'è scritto quali stazioni fanno parte di

ciascun segmento. Quindi, quando riceve una trama, il Bridge legge l'indirizzo di destinazione e, in base alla propria tabella, decide se propagarla nell'altro segmento o meno. Il Bridge *spezza il dominio di collisione*.



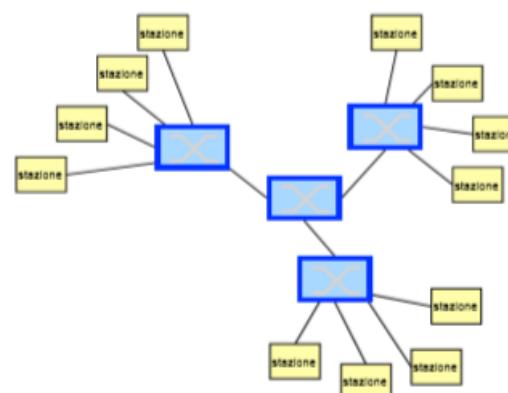
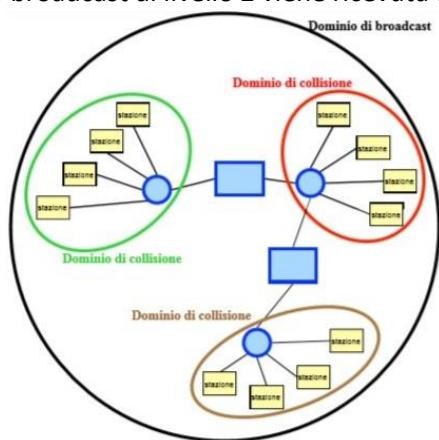
Switch.

Lo Switch è un *Bridge multi-porta* e mantiene al suo interno una *tabella* in cui sono associati indirizzi di livello 2 e relativi segmenti di rete di appartenenza. Spesso ogni porta è connessa ad un'unica stazione (invece che ad un segmento di rete). In questo modo si realizza un *accesso dedicato per ogni nodo* e si eliminano le collisioni. Dunque, aumenta la capacità e si riescono a supportare conversazioni multiple contemporanee.



Esempio di configurazione.

Il Bridge spezza il dominio di collisione, ovvero ciascun segmento di rete è conteso solo da chi è attestato sull'Hub. Gli Hub vedono il Bridge come una stazione qualsiasi che genera trame. La trama è propagata dal Bridge solo se il destinatario è attestato su un Hub diverso da quello di origine. Ogni trama indirizzata ad un indirizzo broadcast di livello 2 viene ricevuta da tutti i nodi del segmento. Si può fare lo stesso con gli Switch.



PROTOCOLLI DI LIVELLO 2

❖ Incapsulamento di IP

Introduzione.

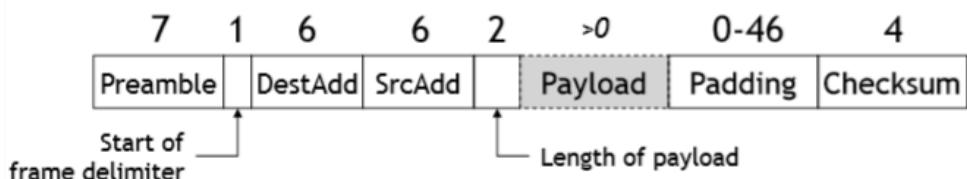
Il livello 2 svolge una serie di *funzionalità* (framing, rilevazione errori, controllo di flusso) che consentono il trasferimento hop-by-hop. Inoltre, in caso di mezzo condiviso, è necessaria la presenza di un sotto-livello di accesso al mezzo. Le suddette funzionalità sono implementate dai *protocolli di livello 2*, che definiscono il *formato dei messaggi* e regolano la comunicazione tra entità. Ogni hop (di livello 3) può avere un protocollo di livello 2 differente dall'hop successivo. Infatti, esistono diverse modalit  di incapsulamento dei pacchetti IP. Le soluzioni maggiormente utilizzate prevedono l'uso dei protocolli Ethernet e PPP.

Ethernet (standard IEEE 802.3).

È un protocollo utilizzato nell'ambito delle *reti locali* (LAN). È una *tecnologia economica* di facile installazione e manutenzione, che si interfaccia direttamente con il *livello fisico*. Il *formato della trama* prevede:

- *Preamble* [7 Byte], che indica la sequenza di Byte utilizzata per sincronizzare il ricevitore.
- *Start of frame* [1 Byte], che indica il flag di inizio della trama.
- *Addresses* [2 * 6 Byte], che indicano gli indirizzi destinazione e sorgente della trama.
- *Length* [2 Byte], che indica la lunghezza in Byte della trama (0-1500) o il protocollo trasportato (>1500).
- *Payload*, che indica l'informazione trasmessa.
- *Checksum* [4 Byte], che indica il codice utile per la rilevazione degli errori.

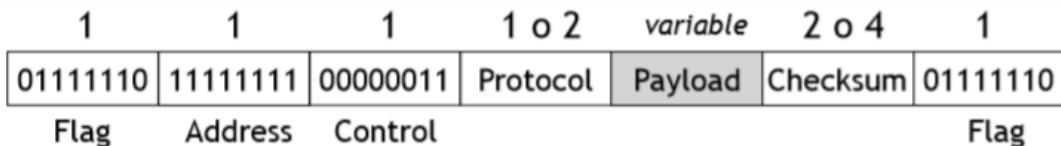
Ethernet si è evoluto in Fast Ethernet (100 Mbps) e successivamente in Gigabit Ethernet (1 o 10 Gbps). Ciò ha permesso di velocizzare le moltissime LAN Ethernet già presenti sostenendo costi non molto elevati, in quanto è stata necessaria la sola sostituzione degli apparati di rete (Hub, Switch).



PPP.

Il Point-to-Point Protocol è un protocollo di livello 2 utilizzato per i *collegamenti punto-punto* tra router e per i collegamenti punto-punto tra un PC e un router. In quest'ultimo caso, PPP viene utilizzato per l'accesso a Internet tramite modem (56 Kbps) o ADSL (banda del doppino telefonico). Il *formato della trama* prevede:

- *Flag* [2 * 1 Byte], che identifica inizio e fine della trama.
- *Address* [1 Byte], che viene utilizzato in configurazione "tutti gli host".
- *Control* [1 Byte], che assume sempre un valore predefinito perché PPP non fornisce un servizio affidabile.
- *Protocol* [1 o 2 Byte], che identifica il tipo di livello delle trame.
- *Payload*, che indica l'informazione trasmessa.
- *Checksum* [2 o 4 Byte], che serve per l'identificazione degli errori.



❖ Risoluzione degli indirizzi

Indirizzi hardware.

Ogni scheda di rete ha un indirizzo univoco associato chiamato indirizzo hardware, o MAC. Quindi un host o un router ha *un indirizzo MAC per ogni scheda di rete*. Inoltre, l'indirizzo MAC viene creato e assegnato al momento della produzione della scheda di rete e quindi, diversamente dall'indirizzo IP che viene assegnato in base alla rete, *non cambia mai*. Anche se ci sono già gli indirizzi IP, c'è pure bisogno degli indirizzi MAC per garantire la consegna diretta/indiretta.

Consegna diretta/indiretta.

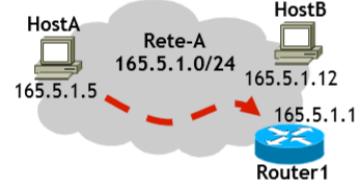
La *consegna diretta* si ha quando sorgente e destinazione appartengono alla *stessa rete*, ovvero quando hanno lo stesso prefisso. In questo caso la sorgente può inviare direttamente il pacchetto alla destinazione, e quindi la presenza l'indirizzo MAC sembrerebbe inutile. La *consegna indiretta*, invece, si ha quando sorgente e destinazione appartengono a *reti diverse*, ovvero quando hanno prefisso diverso. In questo caso, la sorgente manda il messaggio al router di default che si farà carico della consegna tramite il routing.

Ancuni campi del pacchetto
 - IP sorg: 165.5.1.5
 - IP dest: 165.5.1.12
 - MAC sorg: MAC-A
 - MAC dest: MAC-B



Consegna diretta

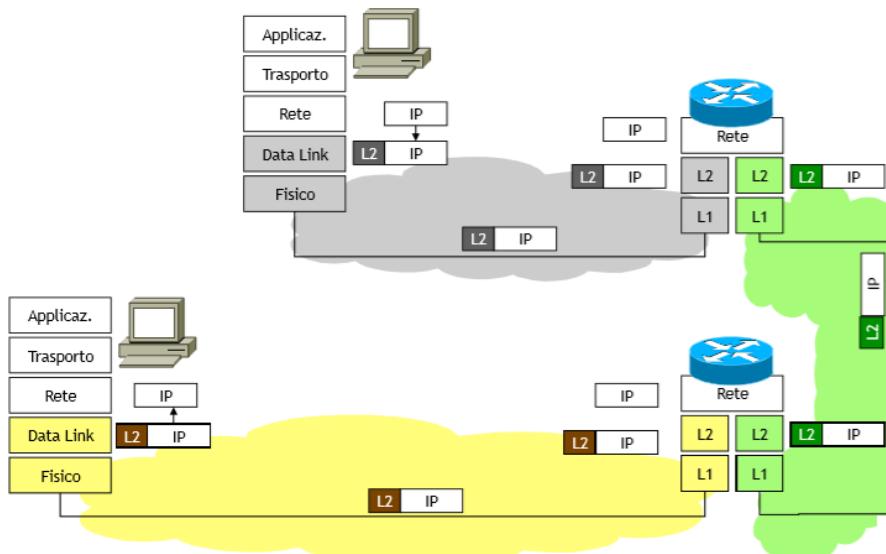
Ancuni campi del pacchetto
 - IP sorg: 165.5.1.5
 - IP dest: 201.12.82.4
 - MAC sorg: MAC-A
 - MAC dest: MAC-Router1



Consegna indiretta

Livello di indirizzamento MAC.

Se consideriamo il caso della consegna indiretta, si pone il problema di come riuscire a consegnare il pacchetto al router. Infatti, non possiamo usare il suo indirizzo IP come destinazione. La soluzione sarà quindi quella di usare un *ulteriore livello di indirizzamento*: gli indirizzi MAC. In questo modo, quando il router riceve una trama che ha come indirizzo MAC di destinazione il proprio indirizzo MAC, *estrae il payload* e lo passa al livello di rete. A questo punto, *controlla l'indirizzo IP di destinazione* e, se non coincide con il proprio indirizzo IP, *inoltra il pacchetto al next hop* (tramite le tabelle di routing). Per fare questo, incapsula il pacchetto in una *nuova trama* in cui l'*indirizzo MAC di destinazione* sarà quello del *next hop*.



Address Resolution Protocol.

Dato un *indirizzo IP*, per riuscire a risalire al suo *indirizzo MAC*, è necessario un protocollo apposito chiamato ARP. Un problema simile era quello di risalire all'*indirizzo IP* dato un *indirizzo logico* (del tipo www.google.it) e, in quel caso, il protocollo utilizzato si chiamava DNS.

Procedura di risoluzione.

Un host può risolvere l'indirizzo di un altro host solo se entrambi sono connessi alla medesima rete fisica. Nel caso di una *consegna diretta* si deve *risolvere l'indirizzo IP di destinazione*, mentre nel caso di una *consegna indiretta* si deve *risolvere l'indirizzo IP del router* (che l'host ha ricevuto dal DHCP). Si supponga che B debba risolvere l'indirizzo IP di C (vedi immagine):

1. B invia una richiesta in broadcast per dire che ha bisogno dell'indirizzo MAC dell'host con indirizzo IP *ipC*.
2. Il broadcast (limitato) viaggia solo sulla rete locale e quindi la richiesta ARP raggiunge tutti gli host.
3. Quando l'host C riceve la richiesta, risponde direttamente all'host B dicendo di essere l'host con indirizzo IP *ipC* e con indirizzo MAC *macC*.



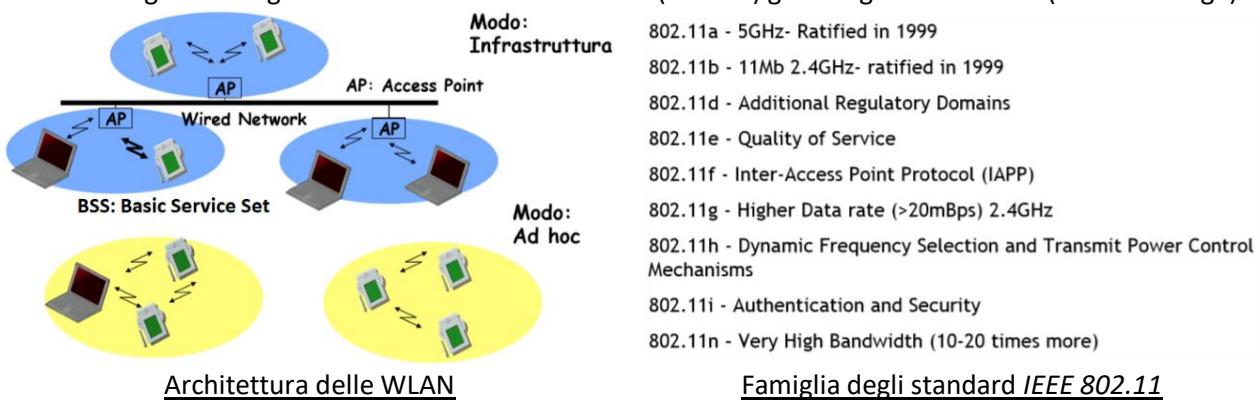
Trasporto dei messaggi e caching.

Inviare una richiesta ARP per ciascun datagramma è inefficiente, in quanto *tre trame* attraverserebbero la rete *per ciascun datagramma* (richiesta ARP, risposta ARP, dati). Per ridurre il traffico di rete, il software ARP estrae e salva le informazioni delle risposte ARP in una *tabella* in modo da poterle utilizzare anche in futuro. Tuttavia, il software non mantiene tali informazioni per sempre. Infatti, ARP gestisce la tabella come una *cache*: un'associazione (tra indirizzo IP e MAC) viene aggiornata quando si riceve una risposta; se la tabella ha raggiunto la sua dimensione massima e arriva una nuova informazione, si procede alla rimozione delle informazioni più vecchie; se un'informazione non è stata aggiornata per molto tempo, viene rimossa. A questo punto, prima di inviare una richiesta ARP, si controlla se l'informazione esiste già nella cache: *se sì*, ARP utilizza l'associazione degli indirizzi senza dover inviare una nuova richiesta; *altrimenti*, ARP invia una nuova richiesta per aggiornare la cache e usa l'informazione ottenuta per l'invio della trama.

❖ Wireless LAN (standard IEEE 802.11)

Architettura di riferimento.

Una WLAN è formata da tanti terminali con capacità di accesso al mezzo wireless (detti *stazioni*) i quali, se usano le stesse frequenze, vengono riuniti in un *BSS*. Nel caso di un'*infrastruttura* con reti cablate e wireless, i vari BSS vengono collegati alla *rete di interconnessione* (cablata) grazie agli *Access Point* (simili ai Bridge).



Livello MAC.

È possibile utilizzare CSMA anche per le WLAN. Tuttavia, siccome la risorsa di trasmissione è molto preziosa e le collisioni sprecano banda aumentando il ritardo, bisogna modificare il CSMA con lo scopo di *ridurre la*

possibilità di collisione. Il concetto che si utilizza è detto *DCF* (Distributed Coordination Function), cioè si utilizza un algoritmo per la *risoluzione delle contese* per fornire accesso a tutti i tipi di traffico.

Time slot.

Il tempo è suddiviso in intervalli chiamati *slot*. Uno slot rappresenta l'unità di tempo del sistema e la sua durata dipende dall'implementazione del livello fisico. Usando il time slot, le *stazioni* sono *sincronizzate*.

Inter-Frame Space (IFS).

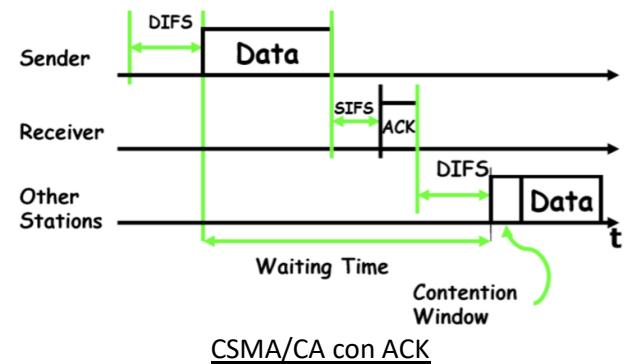
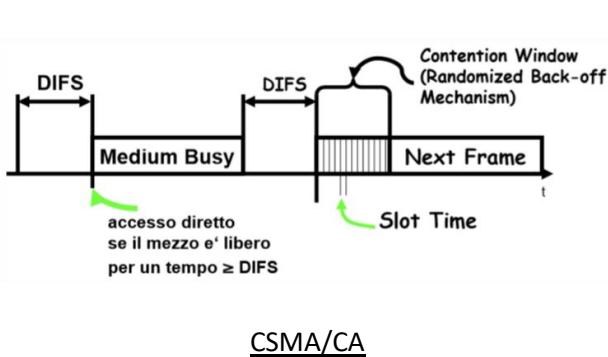
È l'intervallo di tempo che passa tra la trasmissione delle trame, ed è usato per stabilire dei livelli di *priorità* nell'accesso al canale. La sua durata dipende dall'implementazione fisica. Ci sono 2 tipi di IFS: *SIFS* e *DIFS*. Il primo è usato per separare la trasmissione di trame appartenenti allo stesso dialogo e corrisponde alla priorità più alta. Il secondo, invece, corrisponde ad *un SIFS più 2 time slot* e ha priorità più bassa. Viene usato dalle trame per l'invio asincrono con ritardo minimo nel caso di contesa del canale.

DCF con CSMA/CA.

Una stazione con dei dati da trasmettere ascolta il canale e si comporta secondo il seguente algoritmo:

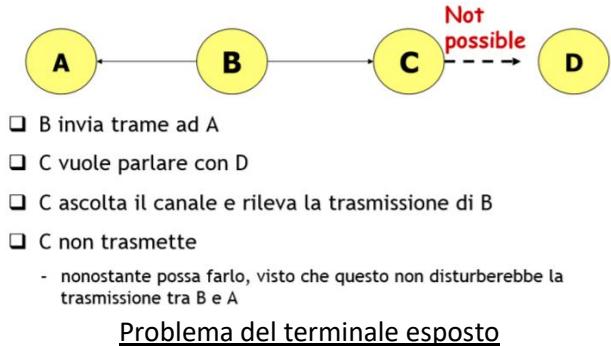
- Se il canale è *libero*, continua ad ascoltarlo per un tempo pari a *DIFS* per capire se rimane libero (in tal caso la stazione può *trasmettere* immediatamente).
- Se il canale è *occupato* (sia perché lo era fin dall'inizio, sia perché lo è divenuto durante il periodo di *attesa*), la stazione continua a monitorarlo fino a quando la trasmissione corrente è finita.
- Quando la trasmissione corrente è finita, la stazione aspetta un altro *DIFS* e, se il canale è di nuovo *occupato*, si torna al punto 2.
- Se il mezzo rimane *libero* durante il periodo di *attesa*, la stazione estrae un numero casuale di slot detto *contatore di backoff* (preso all'interno di una Contention Window) e, fintantoché il canale rimane libero, la stazione decrementa il contatore man mano che il tempo passa (se il contatore arriva a zero, la stazione può *trasmettere*).
- Se il canale torna ad essere *occupato* prima che il contatore arrivi a zero, il contatore viene congelato e si torna al punto 2 (tuttavia al punto 4 non verrà estratto un nuovo valore del contatore, ma si userà il valore congelato).

In definitiva: la prima stazione a cui scade il backoff inizia la trasmissione, mentre le altre stazioni bloccano il loro backoff (lo faranno ripartire nel successivo periodo di contesa). Inoltre, in caso di *collisione*, si raddoppia il valore massimo della Contention Window. Quindi, nel caso di ritrasmissioni multiple, la lunghezza del tempo di backoff viene aumentata esponenzialmente. Infine, esiste anche l'algoritmo *CSMA/CA con ACK*, in cui la stazione ricevente manda un ACK dopo la ricezione di una trama.



Terminali nascosti e esposti – problema.

Il segnale generato dalle stazioni (o dall'access point) è percepibile solo fino ad una certa distanza, che dipende dalla potenza di emissione del segnale. Quando il segnale è troppo debole, non è possibile ricostruirlo. Ci sono delle *particolari disposizioni spaziali* per cui il segnale emesso da una stazione può essere percepito solo da un sottoinsieme di altre stazioni e quindi nascono i problemi di *terminali nascosti e esposti*.



DCF con RTS/CTS.

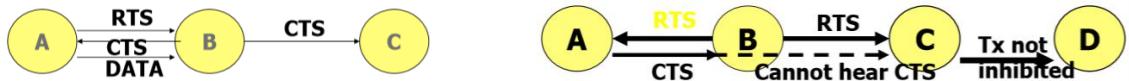
Per risolvere il problema del terminale nascosto:

- La sorgente invia una trama **RTS** (Request To Send) se ha trovato il canale libero per un intervallo DIFS.
- Il ricevente risponde con una trama **CTS** (Clear To Send) dopo un intervallo SIFS.
- Quando la sorgente riceve la trama CTS (dopo un intervallo SIFS), può iniziare a *trasmettere* i dati.

In definitiva, le trame RTS e CTS servono a *riservare il canale* per la trasmissione dei dati, per fare in modo che le eventuali collisioni possano avvenire solo tra i messaggi di controllo.

Terminali nascosti e esposti – soluzione.

Usando RTS/CTS, si permette alla stazione A di includere la *lunghezza dei dati da trasmettere* nella trama RTS inviata per riservare il canale. Tale informazione verrà poi inclusa dalla stazione B nella trama CTS, che verrà trasmessa in broadcast. Quindi anche la stazione C riceverà la durata della trasmissione e perciò calcolerà per quanto tempo inibire le sue trasmissioni.



- A invia una trama RTS a B
 B invia una trama CTS (in broracast)
 A e C ricevono la trama CTS
 C blocca la sua trasmissione
 A invia i dati a B con successo

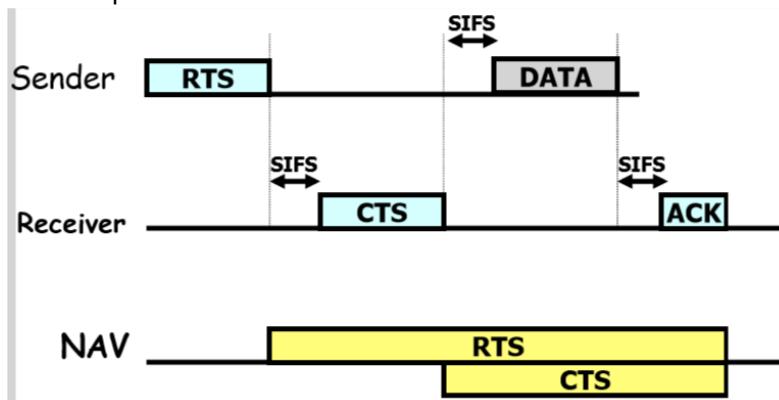
Soluzione al problema del terminale nascosto

- B invia una trama RTS ad A (percepita anche da C)
 A invia una trama CTS a B
 C non riceve la trama CTS di A (fuori range)
 C assume che non sia raggiungibile
 C non inibisce la trasmissione a D

Soluzione al problema del terminale esposto

Network Allocation Vector (NAV).

Quando una stazione vuole trasmettere, invia prima di tutto l'*informazione sulla durata della trasmissione* nella trama RTS. Questa informazione raggiungerà poi tutte le altre stazioni attraverso la trama CTS. Così facendo la stazione ha comunicato per quanto tempo il canale deve rimanere riservato per la sua trasmissione. Tutte le altre stazioni, invece, avranno salvato la suddetta informazione nel proprio *NAV*, in modo tale da sapere che non potranno usare il canale finché è riservato.



DOMANDE SULLA TEORIA

❖ Livello data link

Si spieghi che cosa si intende per framing.

Il livello 2 riceve dal livello superiore dei pacchetti. Considerando che la lunghezza dei pacchetti e delle corrispondenti trame (frame) è variabile, che i sistemi non sono sincronizzati tra loro e che il livello 1 tratta solo bit, nasce il problema della delimitazione delle trame. La funzionalità di *framing* è dunque quella di rendere distinguibile una trama dall'altra attraverso l'uso di opportuni codici all'inizio e alla fine della trama.

Si descrivano le diverse modalità di framing.

Esistono diverse modalità di framing, tra cui quella di *inserire intervalli temporali fra trame consecutive* o quella di *marcare inizio e fine di ogni trama* attraverso:

- *Character count*, cioè un campo nell'header che indica il numero di caratteri contenuti nella trama.
- *Bit stuffing*, cioè far iniziare e terminare ogni trama con uno speciale pattern di bit chiamato byte di flag.

Descrivere l'algoritmo CSMA 0-persistent.

Il CSMA (Carrier Sense Multiple Access) è un *algoritmo di allocazione dinamica del canale a contesa*, ovvero assegna l'utilizzo del canale condiviso di volta in volta alle stazioni che ne fanno richiesta. Inoltre, siccome si tratta di una assegnazione a contesa, ciascuna stazione proverà a trasmettere indipendentemente dalle altre. Il CSMA è molto usato in *ambito LAN* e il suo *funzionamento* prevede che una stazione, prima di trasmettere, ascolti il canale per verificare che non ci sia già una trasmissione in corso. A questo punto: se il canale è libero, trasmette; altrimenti, rimanda il tentativo di trasmissione ad un istante di tempo scelto in maniera casuale. Quest'ultima operazione è propria solamente della variante del CSMA detta 0-persistent (*o non-persistent*), infatti nella variante *persistent*, la stazione inizia a trasmettere nel momento in cui si libera il canale.

Cosa si intende per periodo di vulnerabilità negli algoritmi CSMA?

Negli algoritmi CSMA, il periodo di vulnerabilità è legato al *ritardo di propagazione del segnale* (τ) e vale 2τ . Perciò se una stazione ha iniziato a trasmettere, ma il suo segnale non è ancora arrivato a tutte le stazioni, qualcun altro potrebbe iniziare la trasmissione e generare una collisione. In generale, il ritardo di propagazione deve essere molto minore del tempo di trama, per non rischiare di *non percepire le collisioni*.

Si descriva l'algoritmo CSMA p-persistent.

Nell'algoritmo CSMA p-persistent, il tempo viene suddiviso in intervalli di lunghezza pari al periodo di vulnerabilità (2τ). Il suo funzionamento è il seguente:

1. Se il canale è libero, si può decidere di trasmettere (passa al punto 2) o di attendere (ritorna al punto 1).
2. Se il canale è occupato, bisogna attendere (ritorna al punto 1).
3. Se c'è collisione, bisogna attendere un tempo casuale (ritorna al punto 1).

Si descriva l'algoritmo CSMA nella sua variante Collision Detection (CSMA-CD), indicando il motivo che ha portato all'introduzione di tale variante.

Il CSMA-CD è un miglioramento del CSMA. La differenza con quest'ultimo sta nel fatto che, se la stazione che sta trasmettendo *rileva una collisione*, *interrompe* immediatamente *la trasmissione*. Il motivo che ha portato all'introduzione di tale variante è il fatto che non si vuole *sprecare tempo a trasmettere trame già corrotte*: una volta rilevata la collisione. Inoltre, per far sentire a tutte le stazioni che vi è stata collisione, si trasmette una particolare sequenza (detta di jamming).

Si dia la definizione di dominio di collisione e dominio di broadcast.

Il *dominio di collisione* è la parte di rete per cui, se due stazioni trasmettono dati contemporaneamente, il segnale ricevuto dalle stazioni risulta danneggiato. Il *dominio di broadcast*, invece, è la parte di rete raggiunta da una trama avente come destinazione l'indirizzo di broadcast. Le stazioni appartenenti alla medesima rete di livello 2 condividono lo stesso dominio di broadcast, perciò gli apparati che estendono le LAN possono influire solo sul dominio di collisione.

Come cambia un sistema di comunicazione se si introduce un Bridge?

Il Bridge è un apparato usato per estendere le LAN, il quale *collega due segmenti* di rete ed è dotato di intelligenza. Infatti, il Bridge seleziona se ripetere una trama in un altro segmento in base ad una *tabella*, in cui c'è scritto quali stazioni fanno parte di ciascun segmento. Quindi, quando riceve una trama, il Bridge legge l'indirizzo di destinazione e, in base alla propria tabella, decide se propagarla nell'altro segmento o meno. In questo modo il Bridge riesce a spezzare il *dominio di collisione*, cioè la parte di rete per cui, se due stazioni trasmettono dati contemporaneamente, il segnale ricevuto dalle stazioni risulta danneggiato. Così facendo, due stazioni facenti parte dello stesso segmento non disturbano anche le comunicazioni nell'altro segmento.

Si illustri la variante del CSMA chiamata CSMA/CA (Collision Avoidance) utilizzata nelle wireless LAN.

Si ascolta il canale per un *DIFS* e, se risulta libero, si trasmette immediatamente. In caso contrario, si aspetta che il canale ritorni libero. A quel punto di aspetterà un *DIFS* più un contatore casuale di *backoff*. Così facendo la prima stazione a cui scade il backoff inizierà la trasmissione, mentre le altre stazioni bloccheranno il loro backoff e lo faranno ripartire nel successivo periodo di contesa.

Si spieghi il problema dei terminali nascosti/esposti nelle reti wireless e si discuta una possibile soluzione.
Il problema del *terminale nascosto* si ha quando due stazioni (che non si vedono) parlano entrambe con un'altra stazione (che vede tutte e due). A questo punto, se le due stazioni trasmettono in contemporanea alla stazione centrale, quest'ultima percepisce collisione, ma per le due sorgenti la trama sarà andata a buon fine. Il problema del *terminale esposto*, invece, si ha quando una stazione vuole parlare con un'altra ma non può farlo perché sente la comunicazione tra altre due stazioni, anche se potrebbe farlo visto che non la disturberebbe. La *soluzione* consiste nell'usare le trame RTS (Request To Send) e CTS (Clear To Send) per riservare il canale. In questo modo, il problema del terminale nascosto viene risolto perché solo una delle due stazioni riceverà un CTS dalla stazione centrale, e quindi l'altra bloccherà la sua trasmissione. Anche il problema del terminale esposto viene risolto perché la stazione che vuole parlare con un'altra non si bloccherà, in quanto non riceverà il CTS che riguarda la comunicazione tra le altre due stazioni.

Per consentire il risparmio di energia nelle WLAN, le stazioni utilizzano il cosiddetto “Network Allocation Vector” (NAV): si spieghi che cos’è e come viene utilizzato.

Per ridurre la possibilità di collisione si utilizza un concetto chiamato DCF. Questo concetto prevede l'utilizzo dell'algoritmo CSMA/CA e delle trame RTS/CTS. Quindi, quando una stazione vuole trasmettere, invia prima di tutto l'*informazione sulla durata della trasmissione* nella trama RTS. Questa informazione raggiungerà poi tutte le altre stazioni attraverso la trama CTS. Così facendo la stazione ha comunicato per quanto tempo il canale deve rimanere riservato per la sua trasmissione. Tutte le altre stazioni, invece, avranno salvato la suddetta informazione nel proprio NAV, in modo tale da sapere che non potranno usare il canale finché è riservato e risparmiare energia, in quanto la risorsa di trasmissione wireless è molto preziosa.

❖ **Livello di rete**

Si spieghi, attraverso un esempio, come viene utilizzato il campo “Fragment Offset” dell'header IP.

Il campo FRAGMENT OFFSET dell'header IP specifica l'*offset del frammento rispetto al datagramma originale* (il valore del campo deve essere moltiplicato per 8 per ottenere il vero offset). Tale campo, quindi, specifica

in che punto il router deve posizionare il frammento. Inoltre, anche per dire se un pacchetto è stato frammentato serve sapere l'offset, infatti possiamo subito dire se il frammento è il primo ($offset = 0$) o uno intermedio ($offset \neq 0$).

Si spieghi che cosa contiene e come viene utilizzato il campo “Identification” dell’header IP.

Il campo IDENTIFICATION è un *numero sequenziale* assegnato al pacchetto che viene utilizzato per ricomporre un datagramma nel caso in cui esso venga frammentato. Il riassemblaggio di un datagramma dai frammenti viene fatto dalla destinazione per ridurre la quantità di dati da memorizzare nei router e per permettere di far cambiare percorso ai datagrammi in maniera dinamica. Infatti, se un router intermedio facesse il riassemblaggio, tutti i frammenti dovrebbero passare necessariamente da quel router. Il protocollo IP è quindi libero di far percorrere ai diversi frammenti il percorso più opportuno in quel momento.

Si spieghi come viene utilizzato e perché è stato introdotto il campo “Time To Live” dell’header IP.

Il campo TIME TO LIVE è un *numero inizializzato dalla sorgente* che viene *decrementato da ciascun router attraversato* dal datagramma. Se il TTL raggiunge il valore 0, il datagramma viene scartato e viene inviato un *messaggio di errore* alla sorgente attraverso il protocollo *ICMP*. Quest’ultimo infatti definisce diversi messaggi per la segnalazione di errori, tra cui appunto il messaggio di Time Exceeded il quale indica che il TTL è scaduto oppure che un frammento non è più valido.

Spiegare l’utilizzo degli indirizzi IP speciali.

- *Indirizzo di rete* (suffisso a zero): denota solo il prefisso assegnato ad una rete e quindi non può mai comparire come indirizzo di destinazione di un pacchetto.
- *Directed broadcast* (suffisso a uno): un pacchetto con questo indirizzo viaggia su Internet finché non raggiunge la rete specificata, dove viene poi consegnato a tutti gli host.
- *Limited broadcast* (255.255.255.255): un pacchetto con questo indirizzo raggiungerà tutti gli host appartenenti alla stessa rete locale della sorgente.
- *Questo host* (0.0.0.0): usato nello startup di un host, cioè quando non ha ancora un proprio indirizzo IP.
- *Loopback* (127.0.0.1): usato per testare applicazioni di rete (simula una comunicazione tra host separati).

Spiegare la differenza tra consegna diretta e indiretta.

Quando un host vuole consegnare un messaggio ad un altro host, bisogna considerare due casi:

- Se gli host appartengono alla stessa rete la consegna è *diretta*, ovvero il primo host ottiene l’indirizzo IP dal server DNS, controlla se l’indirizzo appartiene alla stessa rete (tramite il prefisso) e in tal caso invia i dati al secondo host.
- Se gli host appartengono a reti diverse la consegna è *indiretta*, ovvero il primo host passa il messaggio al router di default, il quale si farà carico di inviarlo verso il secondo host (i passi intermedi per raggiungerlo vengono gestiti dagli algoritmi di routing).

Si supponga che una rete utilizzi RIP come algoritmo di routing. In caso di guasto di un link, si possono verificare situazioni di routing loop?

Gli *algoritmi di routing* si occupano di trovare il cammino a costo minimo. Inoltre, i router mantengono al loro interno le informazioni su come raggiungere ogni possibile destinazione. Per scambiare queste informazioni e tenere costantemente aggiornata la *tavella di routing*, bisogna utilizzare i protocolli di propagazione dei cammini, dei quali ne esistono di due tipi: *Distance Vector* e *Link State*. Se si usano i primi: ciascun router ha una visione limitata della topologia della rete; il cammino end-to-end è il risultato della composizione di tutte le scelte di next hop; non si richiedono metriche uniformi tra i router; gli errori si propagano su tutta la rete in caso di guasti. Se si usano i secondi: ciascun router cerca di ottenere una visione globale della rete; le informazioni sulla topologia sono inviate su tutta la rete; il miglior cammino viene calcolato da ciascun router localmente; si richiedono metriche condivise e uniformi; gli errori non si

propagano in caso di guasti. Quindi, siccome il *RIP* è un protocollo di tipo DV, è possibile che si verifichino dei *routing loop* se un link si guasta.

Quando viene aggiornata la tabella di routing?

Nel caso degli algoritmi DV ogni nodo invia *periodicamente* ai propri vicini il *vettore delle distanze*. Per sapere quando inviarlo, ciascun nodo: *aspetta* che cambi il costo di un collegamento locale o di ricevere un messaggio da un vicino, *ricalcola* la stima delle distanze e, se il vettore è cambiato, lo *notifica* ai vicini.

Dare una definizione di indirizzo privato e indirizzo pubblico, spiegando come vengono utilizzate le due diverse tipologie di indirizzo.

Sono stati definiti alcuni range di indirizzi all'interno dello spazio di indirizzamento IP da utilizzare solamente in ambito privato. Gli *indirizzi privati* si impiegano solitamente in reti con pochi punti di connessione ad Internet. Gli *indirizzi pubblici*, invece, devono necessariamente essere assegnati all'interfaccia del router che si affaccia su una rete a indirizzamento pubblico.

Si spieghi che cos'è il Network Address Translation (NAT), specificando per quale motivo tale funzionalità è stata introdotta.

Per permettere di ricevere i pacchetti all'interno della rete privata e quindi risolvere i problemi di instradamento tra una rete ad indirizzamento privato ed una rete ad indirizzamento pubblico, è necessario introdurre un'ulteriore funzionalità sul bordo tra privato e pubblico. Questa funzionalità è chiamata *NAT* e funziona in questo modo: il router traduce l'indirizzo IP dei *datagrammi uscenti* sostituendo l'indirizzo sorgente con il proprio indirizzo pubblico e traduce l'indirizzo IP dei *datagrammi entranti* sostituendo l'indirizzo destinazione con l'indirizzo privato dell'host corretto. Per tradurre l'indirizzo IP dei datagrammi entranti, il router NAT deve mantenere al suo interno una *tabella di mapping* tra indirizzo privato sorgente ed indirizzo pubblico destinazione. Inoltre, tale tabella mantiene anche informazioni sulle porte di livello 4 utilizzate per permettere a differenti host privati di connettersi contemporaneamente allo stesso host pubblico. Il NAT risolve il problema dello *spazio di indirizzamento* aggiornando solo i router di bordo.

❖ Riassunto

Si spieghi cosa si intende per consegna diretta/indiretta e come questa viene effettuata.

La *consegna diretta* si ha quando gli host appartengono alla stessa rete. A *livello di rete*: la sorgente ottiene l'indirizzo IP della destinazione dal server DNS, controlla se l'indirizzo appartiene alla stessa rete (tramite il prefisso) e in tal caso invia i dati. A *livello data link*: la sorgente può inviare direttamente il pacchetto alla destinazione, e quindi la presenza l'indirizzo MAC sembrerebbe inutile. La *consegna indiretta*, invece, si ha quando gli host appartengono a reti diverse. A *livello di rete*: la sorgente passa il messaggio al router di default, il quale si farà carico di inviarlo verso la destinazione (i passi intermedi per raggiungerlo vengono gestiti dagli algoritmi di routing). A *livello data link*: il router riceve una trama, estrae il payload e lo passa al livello di rete. A questo punto, controlla l'indirizzo IP di destinazione e inoltra il pacchetto al next hop (tramite le tabelle di routing). Per fare questo, incapsula il pacchetto in una nuova trama in cui l'indirizzo MAC di destinazione sarà quello del next hop. Il protocollo *ARP* risolve gli indirizzi IP di destinazione (consegna diretta) e gli indirizzi IP dei router (consegna indiretta).

❖ Livello di trasporto

Si spieghi il concetto di porta del livello di trasporto.

La *socket* è una tupla che identifica una connessione tra due applicazioni, ed è formata da {IP sorgente: numero di porta sorgente, IP destinazione: numero di porta destinazione}. Gli *indirizzi IP* vengono utilizzati per l'instradamento dei pacchetti all'interno della rete. La *porta*, invece, è un codice che identifica

un'applicazione ed è utile quando il pacchetto giunge a destinazione e viene passato al livello di trasporto, in quanto a esso si appoggiano molte applicazioni e bisogna quindi distinguerle una dall'altra.

Si spieghi che cosa sono le “porte note” e il motivo per cui sono state introdotte.

Le porte sorgente e destinazione non sono necessariamente uguali, infatti, il numero di porta può essere statico o dinamico. Nel primo caso si parla di *porte note*, cioè quell'insieme di porte che è associato ad applicazioni largamente utilizzate (posta elettronica, web, FTP). Nel secondo caso, invece, si parla di *porte effimere*, cioè quelle porte assegnate dal sistema operativo quando si apre la connessione. Le porte note assumono valori da 0 a 1023 permettendo di rendere facilmente riconoscibile un servizio su un server. Le porte effimere, invece, assumono valori superiori a 1024 permettendo ad un singolo host di aprire più connessioni contemporaneamente.

Come viene instaurata una connessione TCP?

Tra le funzioni svolte dal livello di trasporto c'è la cosiddetta *gestione delle connessioni*. Il protocollo TCP instaura una connessione (*connection-oriented*) per scambiare alcune informazioni necessarie a concordare l'attivazione di un canale di comunicazione *affidabile*. La procedura di instaurazione della connessione è detta “*three-way handshake*” e funziona in questo modo:

1. L'host che richiede la connessione invia un segmento SYN (mettendo a 1 il corrispondente flag nell'header TCP) dove specifica il numero di porta dell'applicazione a cui intende accedere, l'*Initial Sequence Number* e la *Maximum Segment Size* (cioè la dimensione massima di dati che può ricevere).
2. La destinazione riceve la richiesta e risponde inviando un segmento SYN-ACK (per dire che accetta la richiesta di connessione) dove specifica il proprio ISN e la propria MSS.
3. L'host che ha richiesto la connessione invia un segmento ACK per concludere la procedura.

A questo punto la connessione è instaurata e i due host possono scambiarsi segmenti con un campo dati pari alla MSS minima.

Come viene abbattuta una connessione TCP?

La connessione TCP è *bidirezionale*, in quanto entrambe le stazioni possono trasferire dati. Per cui anche la terminazione della connessione deve avvenire in entrambe le direzioni, in questo modo:

1. Quando un host non ha più dati da trasmettere e vuole chiudere la connessione, invia un segmento FIN.
2. La destinazione riceve il segmento FIN e invia un segmento ACK, indicando all'applicazione che la comunicazione è stata chiusa nella sola direzione entrante (*half close*).
3. A questo punto la destinazione può ancora continuare a trasferire dati.
4. Quando anche la destinazione vuole chiudere la connessione, invia un segmento FIN (*half close*).

Come viene aggiornato il valore medio del RTT (SRTT) durante una connessione?

Con *RTT* (Round Trip Time) si intende l'intervallo di tempo tra l'invio di un segmento e la ricezione del relativo riscontro. Poiché RTT può variare molto in base alle condizioni della rete, bisogna stimarne un valore medio (detto *SRTT*, Smoothed RTT) guardando al RTT dei diversi segmenti. Il SRTT così stimato può venire influenzato o meno dai segmenti che sperimentano RTT molto diversi.

Si descriva come il TCP calcola il Retransmission Time Out (RTO).

Per sapere quanto tempo bisogna aspettare prima di poter considerare un segmento perso e ritrasmetterlo, si usa il RTO, il quale indica l'istante di tempo entro il quale la sorgente si aspetta di ricevere un riscontro e, nel caso in cui il riscontro non arrivi, la sorgente procede alla ritrasmissione. Il RTO deve essere *calcolato dinamicamente* di volta in volta durante la fase di instaurazione della connessione e durante la trasmissione dei dati. Di norma la sorgente attende fino a $2 * SRTT$ prima di considerare il segmento perso e ritrasmetterlo. In caso di ritrasmissione, il RTO per quel segmento viene ricalcolato in quanto se è scaduto il RTO probabilmente c'è congestione, quindi meglio aumentare il RTO per quel segmento a $2 * RTO$.

PROCEDIMENTI PER RISOLVERE GLI ESERCIZI

❖ Esercizi sul livello di trasporto (TCP)

Evoluzione della CWND – variabili considerate.

- *Congestion Window (CWND)*: è la dimensione della finestra di trasmissione (espressa in byte o in numero di segmenti).
- *Receive Window (RCVWND)*: è la dimensione della finestra di ricezione (espressa in byte o in numero di segmenti) annunciata dalla destinazione, ovvero il limite massimo che la CWND può assumere.
- *Slow Start Threshold (SSTHRESH)*: è una dimensione della finestra (espressa in byte o in numero di segmenti) raggiunta la quale, invece di seguire l'algoritmo di Slow Start, si segue l'algoritmo di Congestion Avoidance.
- *Round Trip Time (RTT)*: è il tempo trascorso tra l'invio di un segmento e la ricezione del riscontro (in condizioni di stabilità della rete e del carico RTT rimane pressoché costante).
- *Retransmission Time Out (RTO)*: è il tempo che la sorgente aspetta prima di ritrasmettere un segmento non riscontrato.

Evoluzione della CWND – algoritmo.

L'algoritmo che regola la dimensione della CWND è il seguente:

- All'inizio della trasmissione si pone $CWND = 1$ segmento (ovvero un numero di byte pari a MSS) e $SSTHRESH = RCVWND$ oppure $RCVWND / 2$ (dipende dalle implementazioni).
- La CWND evolve secondo l'algoritmo di *Slow Start* fino al raggiungimento della SSTHRESH.
- Raggiunta la soglia SSTHRESH, la dimensione di CWND è regolata dall'algoritmo di *Congestion Avoidance*.
- La finestra cresce fino al raggiungimento di RCVWND.

```
IF (CWND_old < SSTHRESH)
    CWND_new = min((CWND_old + numero_ACK); SSTHRESH; RCVWND)
ELSE
    CWND_new = min((CWND_old + (numero_ACK / CWND_old)); RCVWND)
```

Evoluzione della CWND – errori o perdite.

Nei periodi di “rete fuori uso” (intervalli aperti), tutti i segmenti in transito vengono persi, interrompendo così la trasmissione di nuovi segmenti (la finestra non si sposta). A questo punto:

- Si attende lo scadere del timeout RTO (per determinarlo vedere il testo dell'esercizio).
- Allo scadere di RTO, si pone $SSTHRESH = \max((CWND / 2); 2)$ e $CWND_new = 1$.
- Si riprende a ritrasmettere con la tecnica di *Go-back-N*.
- La CWND evolve secondo l'algoritmo di *Slow Start* fino al raggiungimento della SSTHRESH, ecc.

In caso di errore o perdita consecutiva:

- Al primo errore o perdita, quando il timeout scade e si ritrasmette il primo segmento non riscontrato, il timeout per quel segmento viene raddoppiato (exponential back off), cioè $RTO_new = 2 * RTO_old$.
- La CWND rimane a 1, mentre la SSTHRESH si pone a 2 segmenti.

Modalità di svolgimento – ipotesi generali.

Negli esercizi si ragiona sempre per *numero di segmenti*, quindi bisogna fare la *conversione* da byte a numero di segmenti (usando la MSS). Inoltre, si considerano gli RTT come *intervalli chiusi* e quindi:

- I segmenti vengono inviati contemporaneamente all'inizio di ciascun intervallo e con tempo di trasmissione trascurabile.

- Tali segmenti vengono ricevuti e riscontrati simultaneamente dal ricevitore.
- I riscontri (ACK) arrivano simultaneamente al termine dell'intervallo RTT.

Obiettivi.

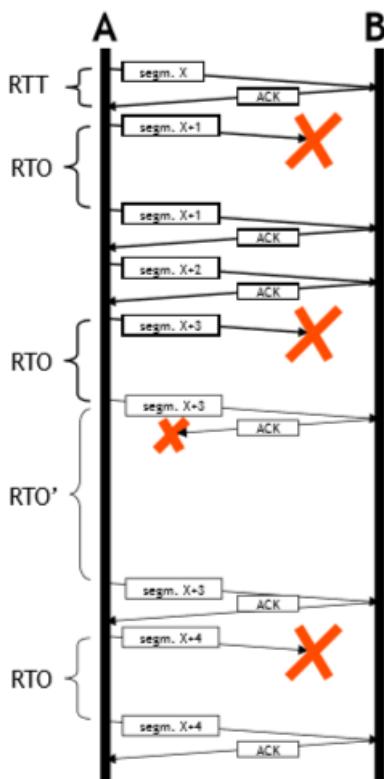
L'obiettivo degli esercizi è quello di *disegnare* l'evoluzione della trasmissione fino a quando non sono stati ricevuti tutti i riscontri degli ultimi segmenti inviati, determinando ad ogni RTT: *CWND*, *SSTHRESH*, *segmenti inviati* (che non sempre coincidono con la dimensione della *CWND*).

Notazione.

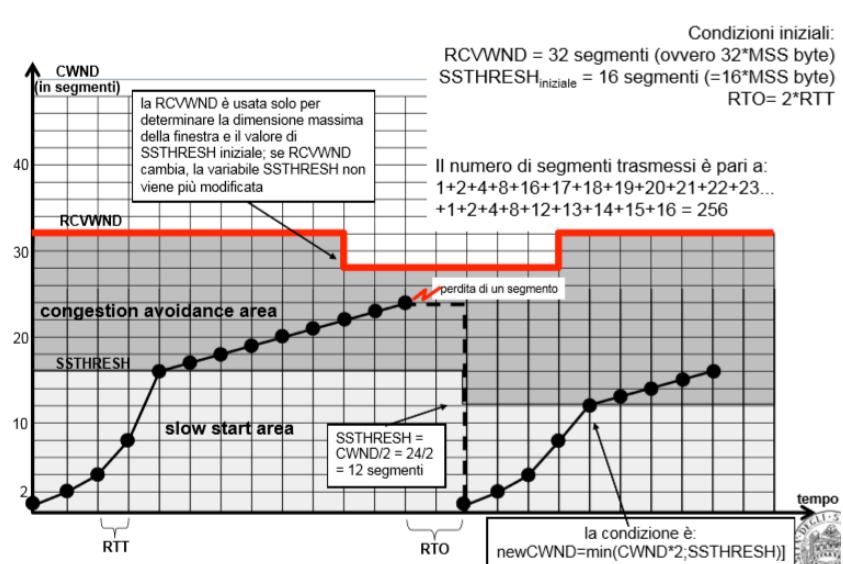
Il grafico dell'evoluzione della trasmissione ha come unità di misura:

- Il Round Trip Time (RTT) sull'asse x.
- La *CWND* (espressa in numero di segmenti) sull'asse y.

I punti sul grafico rappresentano dunque il *valore della CWND* per ogni RTT (ricordarsi che, per chiarezza, si deve esplicitare il valore assunto dalla *CWND* a fine trasmissione). I numeri dei segmenti effettivamente inviati si tengono da parte e, in caso di perdita, la notazione prevede di barrare tale numero. *SSTHRESH* e *RCVWND* sono delle *linee* che si devono distinguere dal resto del grafico e tra loro.



Perdite consecutive



Esempio di svolgimento di un esercizio

❖ Esercizi sul livello di rete (indirizzamento IP)

Notazione decimale puntata – regole.

- Non si antepongono zeri nella notazione decimale puntata (es. 111.56.045.78).
- Non si possono avere più di 4 sezioni nella notazione decimale puntata (es. 221.34.7.8.20).
- Nella notazione decimale puntata, ciascun numero deve essere minore o uguale a 255 (es. 75.45.301.14).
- Non è permesso mescolare notazione decimale puntata e binaria (es. 11100010.23.14.67).

Indirizzo di rete (blocco CIDR).

L'indirizzo di una rete si trova mantenendo fissi i bit del prefisso e ponendo a zero i rimanenti (suffisso). Per trovare l'indirizzo di rete dell'host con indirizzo $167.199.170.82/27$, faremo:

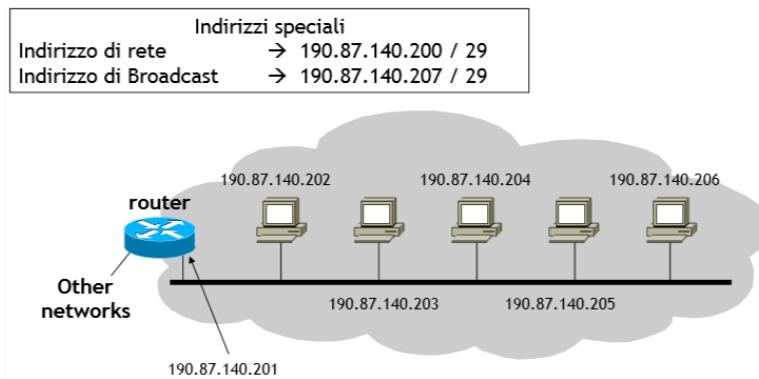
Traduzione in binario	$10100111\ 11000111\ 10101010\ 01010010$
Ultimi 5 bit a zero	$10100111\ 11000111\ 10101010\ \underline{01000000}$
Traduzione in decimale	$167.199.170.64 / 27$

Dimensione del blocco.

Per trovare il numero di indirizzi (inclusi quelli riservati) di un blocco, dato l'indirizzo di un host, bisogna considerare la lunghezza del prefisso (p). La dimensione cercata sarà: 2^{32-p} indirizzi. Ad esempio, se uno degli indirizzi è $140.120.84.24/20$, il blocco avrà $2^{32-20} = 2^{12} = 4096$ indirizzi.

Configurazione di rete.

Per disegnare una configurazione di rete bisogna ricordare che ci sono due *indirizzi speciali* che non possono essere usati per gli host: l'*indirizzo di rete* (bit del suffisso tutti a 0) e l'*indirizzo di broadcast* limitato (bit del suffisso tutti a 1). Ad esempio, se il blocco CIDR è $190.87.140.200/29$ avremo 8 indirizzi così suddivisi:

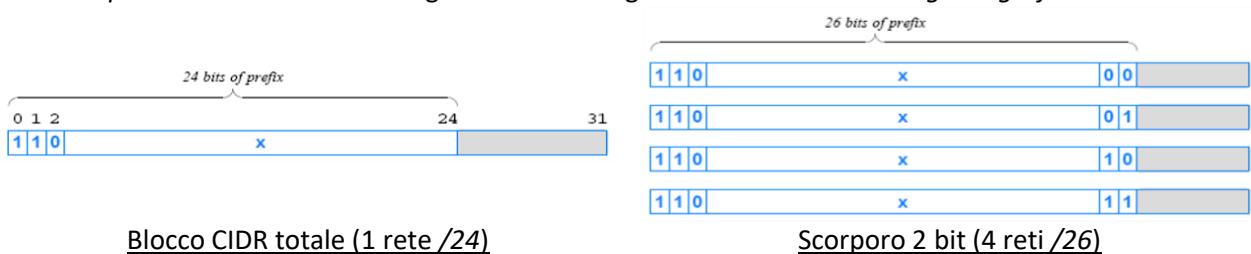


Obiettivi.

- Assegnare gli *indirizzi di rete alle LAN richieste*, partizionando il blocco CIDR in modo che ciascuna LAN possa contenere il numero di stazioni richiesto.
- Specificare per ogni LAN l'indirizzo di rete (e quello di broadcast).
- Se ci sono router, assegnare i primi indirizzi disponibili di ogni LAN all'interfaccia del loro router.
- Rispettare i vincoli nella consegna (se viene dato l'indirizzo di un host di una LAN invece del blocco CIDR).

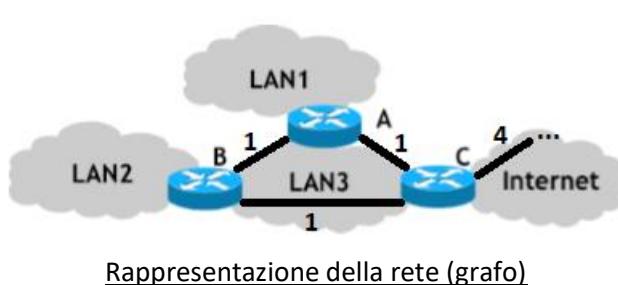
Modalità di svolgimento.

1. Per ogni LAN, si ricava il *numero di indirizzi necessari* prendendo la potenza di 2 immediatamente superiore al numero di host.
2. Si calcola la *dimensione del blocco* prendendo la potenza di 2 immediatamente superiore alla somma degli indirizzi necessari ad ogni LAN.
3. Si calcola la *lunghezza del prefisso* come $2^{32 - \text{blocco}}$.
4. Si ottiene il *blocco CIDR totale* mettendo a 0 i bit del suffisso.
5. Si scorporano i bit in base alle esigenze della configurazione di rete e si disegna il grafico a torta.



Grafi e tabelle di routing.

Dalla figura della rappresentazione della rete si ricava un *grafo*, in cui ogni nodo è un router e ogni lato rappresenta un collegamento tra due router avente un determinato costo (di norma pari a 1). Per scrivere la *tavella di routing* di un determinato router si deve indicare, per ogni destinazione, il *costo* del collegamento e il *next hop*, ovvero da quale router bisogna passare per raggiungere la destinazione. In caso di *consegna diretta*, non si passa da alcun router e il costo è sempre pari a 1.



A	Costo	Next hop
LAN1	1	Diretta
LAN2	2	B
LAN3	1	Diretta
Internet	5	C
B	1	B
C	1	C

Tabella di routing del router A

❖ Esercizi sul livello data link

Caratteristiche del sistema.

Un certo numero di stazioni comunicano a livello data link tramite un determinato *protocollo*. Il sistema è formato da uno o più segmenti ognuno con la propria *velocità* v e il proprio *ritardo di propagazione* τ . Le stazioni, inoltre, generano delle trame di *lunghezza* L in corrispondenza di determinati istanti temporali detti *istanti di inizio trasmissione*.

Tempo di trama.

Il *tempo di trama* T di ogni segmento si ricava facendo il rapporto tra la lunghezza delle trame e la velocità del segmento considerato ($T = L/v$). Prima di eseguire tale operazione, però, bisogna *convertire la lunghezza* delle trame da Byte a bit moltiplicandola per 8 (perché 1 Byte = 8 bit). Una volta ottenuta questa misura si può determinare graficamente in quali istanti avvengono le trasmissioni delle diverse trame.

Collisioni.

In caso di *collisione* (sovraposizione di due trame), le stazioni devono aspettare un tempo casuale pari a Z millisecondi prima di cercare di ritrasmettere la trama. Per esigenze di comparabilità tra gli esercizi, il numero Z viene deciso secondo il seguente metodo: $Z = Sc * N + T$, dove Sc è la somma delle cifre che compongono l'istante di inizio trasmissione e N è il numero di collisioni subite dalla trama. Ad esempio, consideriamo che una trama di lunghezza pari a 12 ms e con istante di inizio trasmissione pari a 418 ms subisca la prima collisione. A questo punto si avrà $Z = (4+1+8) * 1 + 12 = 25$, perciò la trama verrà ritrasmessa dopo la fine della sua trasmissione corrotta, e quindi all'istante 455 ms .

Algoritmo ALOHA.

1. Una stazione *inizia a trasmettere* ogni qualvolta ha una trama da inviare.
2. La stazione *ascolta il canale* e rileva l'eventuale collisione.
3. Se c'è stata *collisione*, la stazione *aspetta* un tempo casuale (vedi Collisioni) e *ritrasmette* la trama.

Con l'algoritmo ALOHA il *ritardo di propagazione* è sempre *pari a 0*, perciò le stazioni percepiscono immediatamente se un'altra stazione ha iniziato a trasmettere.

Algoritmo CSMA persistent.

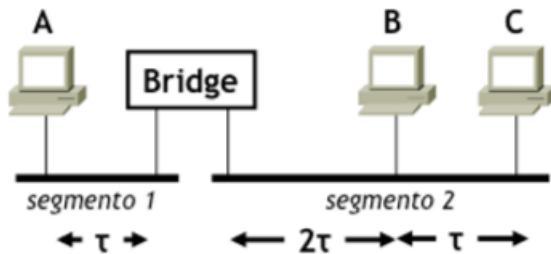
1. La stazione *ascolta il canale* per verificare che non ci sia già una trasmissione in corso.
2. Se il canale è *libero*, la stazione *inizia a trasmettere*.
3. Se il canale è *occupato*, la stazione *rimanda la trasmissione* fino al momento in cui si libera il canale.

4. Se c'è *collisione*, la stazione *aspetta* un tempo casuale (vedi Collisioni) e poi si ritorna al punto 1.

Con l'algoritmo CSMA il *ritardo di propagazione* è *pari a* τ , perciò se una stazione ha iniziato a trasmettere, ma il suo segnale non è ancora arrivato a tutte le altre stazioni, una di quelle potrebbe iniziare una nuova trasmissione generando una collisione.

Bridge.

Il Bridge è un particolare tipo di stazione che *memorizza* le trame che gli arrivano da ogni segmento di rete a cui è collegato. Quando il Bridge riceve completamente una trama, *controlla* se essa è destinata all'altro segmento e in tal caso la *rtrasmette* (questo comportamento è valido in entrambi i sensi, in modo indipendente l'uno dall'altro). Le trame restano nella memoria del Bridge fino a quando la trasmissione sull'altro segmento non è andata a buon fine.



Sistema formato da due segmenti uniti per mezzo di un Bridge

Buffer.

Ogni stazione possiede un buffer (*coda*) di tipo *FIFO*. Le trame si accumulano nel buffer quando una stazione genera una *nuova trama* e, al contempo, deve anche *risolvere una collisione*. A quel punto bisognerà prima aspettare il nuovo istante di trasmissione (dovuto alla collisione), e poi cominciare a trasmettere le trame in *serie*. Nel caso del Bridge, inoltre, quando una trama viene memorizzata può essere subito rtrasmessa e perciò non bisogna aspettare che finisca la coda.

Periodo di vulnerabilità.

L'intervallo di tempo in cui può avvenire una collisione che invalida una trasmissione è detto *periodo di vulnerabilità*. Nel caso dell'algoritmo ALOHA esso vale *due volte il tempo di trama*, quindi nessun'altra sorgente deve aver iniziato/iniziare la trasmissione nel periodo di tempo $[t_0-t, t_0+t]$ dove t_0 è l'istante di inizio trasmissione. Nel caso dell'algoritmo CSMA, invece, il periodo di vulnerabilità vale *due volte il ritardo di propagazione*, infatti anche se una stazione ha già iniziato a trasmettere qualche altra stazione potrebbe iniziare una nuova trasmissione in quanto non avrà ancora ricevuto il segnale della prima.