

Reti di calcolatori

Sistemi interconnessi

- "comodo" → pericolos
- spreci di risorse

Un sistema di comunicazione è composto da:

- hardware fisico
- software

PAG 35

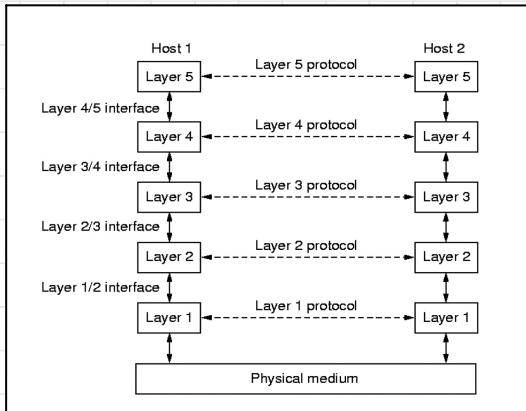
Un protocollo definisce il formato e l'ordine dei messaggi scambiati tra due o più entità in comunicazione, così come le azioni intraprese in fase di trasmissione e/o di ricezione di un messaggio o di un altro evento.

HTTP non tiene conto delle sfiducie della comunicazione e degli utenti
esse soffrirebbe a questa mancanza attraverso l'utilizzo di cookie.

TCP ha un handshake a tre vie: la connessione avviene scambiando 3 messaggi. TCP garantisce che il primo host arrivato non si prenda tutta la banda disponibile.
L'ideale è garantire una suddivisione equa della banda in modo che sia egualmente distribuita senza specie (proprietà fairness).

TCP fa tutto ciò attraverso il "controllo congestione".

HTTP 3.0 trova un modo per garantire che non reiga by-passando il controllo congestione



Ogni livello comunica col livello immediatamente precedente o successivo e fornisce dei servizi sfruttando i servizi offerti dal livello inferiore.

Le comunicazioni tra host differenti avvengono in "orizzontale", gestiti dai protocolli di comunicazioni orizzontali.

Ci sono comunicazioni anche intere, che avvengono tramite i protocolli verticali. Questo tipo di comunicazione interna avviene sempre correttamente ma quella (orizzontale) tra host differenti ha molte probabilità di fallire.

La comunicazione orizzontale avviene facendo ricorrere al mezzo già a livello fisico, esso trasferisce i bit al livello fisico dell'altro host che si occuperà di farlo risalire a livello applicativo.

Se l'hardware tra i livelli avviene tramite un interazione ed un payload che verrà incorporato e spedito al livello sottostante.

Il livello sottostante riceve il messaggio e mette la propria interazione che sarà necessaria al funzionamento del protocollo.

La comunicazione avviene tramite una trasmissione "fisica" nelle seguenti modalità:

- simplex
- half-duplex
- full-duplex

SIMPLEX => Un nodo trasmette l'informazione mentre l'altro riceve. In questo caso A non può ricevere e B non può trasmettere. La comunicazione risulta essere **unidirezionale**



HALF DUPLEX => Entrambi i nodi possono trasmettere e ricevere ma se A trasmette allora B riceve e quando B trasmette, allora A riceve



ex. Walkie Talkie

• **FULL-DUPLEX** \Rightarrow Entrambi i nodi sono in grado di trasmettere e ricevere contemporaneamente



ex. chiamata Telefonica

3 messaggi a basso livello vengono trasmesse in modo frammentato, infatti essi vengono suddivisi in frammenti per garantire che non siano di lunghezza arbitraria

Internet mette a disposizione delle applicazioni due

PAG 113

protocolli di trasporto : TCP e UDP

Se un'applicazione utilizza **TCP** come protocollo di trasporto ha i seguenti servizi :

- servizio orientato alla connessione: i dispositivi che vogliono comunicare instaurano prima una connessione mediante una procedura di **handshake** con la quale si instaura una connessione TCP di tipo **full-duplex** tra i due dispositivi

- trasferimento dati affidabile: i dispositivi hanno la certezza che i pacchetti trasmessi arrivino in maniera completa e corretta, grazie ad un meccanismo che prende la **numerazione e l'ordinamento**

dei pacchetti. Si necessita di un **feedback** di ricezione del pacchetto, ovvero un segnale di **ACK** per comunicare che il pacchetto è stato ricevuto correttamente.

Essendo trasparente al mittente, questa volta necessita di un meccanismo di **timeout** che econometra il tempo necessario al destinatario per ricevere il messaggio, in modo da spedirlo nuovamente se non ricevuto correttamente. Il destinatario inoltre deve identificare il messaggio in modo da riuscire a capire che il messaggio era già stato ricevuto, così da scaricarlo.

TCP include anche un meccanismo di **controllo congestione** che si occupa di regolare l'arrivo dei pacchetti quando la rete è trafficata. Questi meccanismi che garantiscono affidabilità introducono ritardo. La connessione TCP è **full duplex** e sposta l'architettura peer-to-peer, ovvero gli host comunicano con altre porte e non c'è master/slave.

Invece **UDP** è un protocollo di trasporto **non affidabile alla connessione**, per tanto non ci è nessuna procedura di handshaking tra i dispositivi. Inoltre esso è **affidabile** in quanto non anticipa

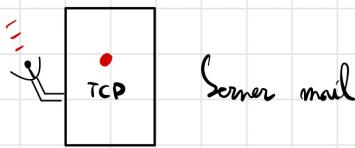
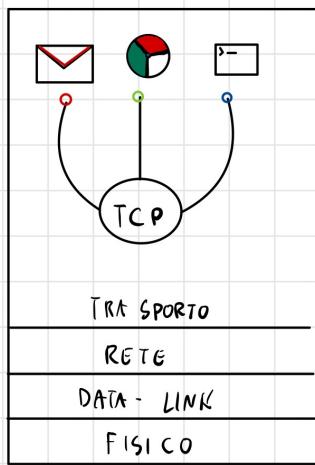
che il trasferimento del pacchetto arrivi a destinazione in modo corretto e non colpisca i pacchetti spediti.

UDP non include il meccanismo di controllo confezione, per tanto spedisce pacchetti ad una velocità molto più alta.

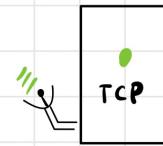
Questi sono protocolli veloci che permettono il trasferimento dati in modo veloce seppur inaffidabile. Questo viene utilizzato quando si ha bisogno di trasferire in tempi reali.

Multiplicazione \Rightarrow differenziare le diverse applicazioni

Il livello applicazione non è omogeneo ma si compone di diverse applicazioni, in esecuzione sulla macchina (es. chrome, porta, terminale)



Server mail



Server WEB



Server SSH

Si vuole connettere il servizio di posta sul proprio computer al server mail (anche esso programma applicativo). I segmenti inviati, dal un certo punto, verranno ricevuti dal livello 4 del server delle mail, essi li innalzerà al livello 5.

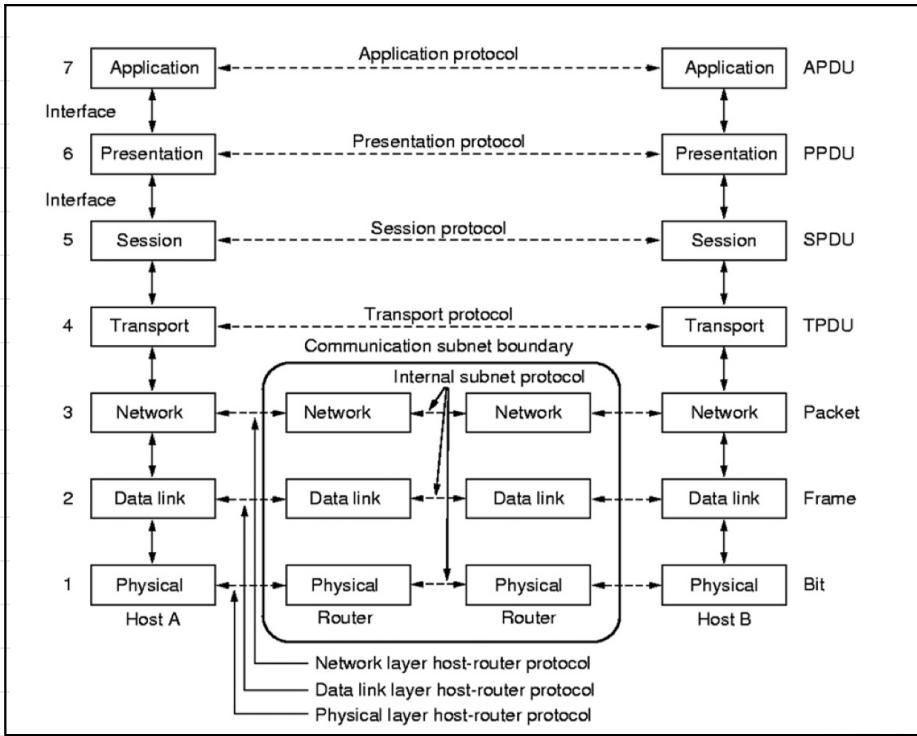
Quando il server mail restituisce una risposta deve sapere quale tra i tre processi in corso ha richiesto la connessione.

Si identifica il portale attraverso il **numero di porta**.

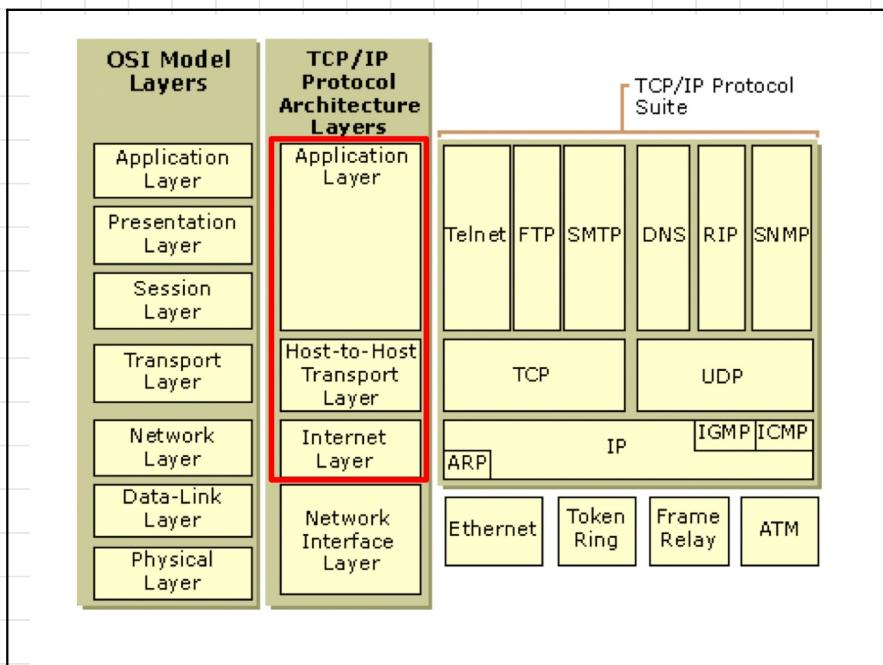
Per Tanto una connessione TCP è identificata univocamente da

- indirizzo IP dei dispositivi
 - numeri di porta dei processi
- } La coppia IP - PORTA di un dispositivo
è detto **Socket**

Modello 7 strati (pila ISO/OSI)



Standard de-facto (effettivamente realizzato)

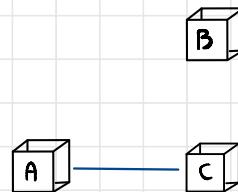


=> Esso non è suddiviso a livelli

Tipologie di trasmissione

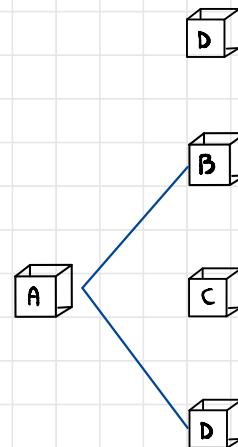
Trasmissione point-to-point

La trasmissione point-to-point è una modalità di comunicazione diretta tra due dispositivi, dove i dati vengono trasmessi da un dispositivo sorgente a un dispositivo di destinazione specifico. Questa modalità di comunicazione può avvenire in modalità simplex, half-duplex o full-duplex.



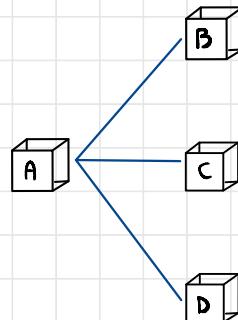
Trasmissione multicast

La trasmissione multicast prevede la trasmissione di dati a un gruppo di dispositivi contemporaneamente. In una trasmissione multicast, un dispositivo trasmittente invia un solo flusso di dati che viene ricevuto da tutti i dispositivi appartenenti al gruppo multicast. Questo è utile per inviare informazioni a un gran numero di dispositivi contemporaneamente, come nella diffusione di streaming video o audio su Internet.



Trasmissione broadcast

La trasmissione broadcast si riferisce a una modalità di trasmissione in cui i dati vengono inviati a tutti i dispositivi della rete contemporaneamente. In una trasmissione broadcast, tutti i dispositivi ricevono i dati trasmessi, e ogni dispositivo elabora solo i dati che sono destinati a esso. Dato che tutti i dispositivi possono parlare si necessita di regole di comunicazione.



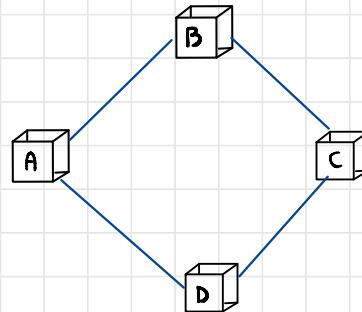
Trasmissione a bus condiviso

La trasmissione a bus condiviso è un metodo in cui tutti i dispositivi della rete condividono lo stesso canale di comunicazione, noto come "bus". I dati vengono trasmessi da un dispositivo alla volta, e ogni dispositivo della rete riceve tutti i dati trasmessi, ma solo il destinatario effettivo elabora i dati ricevuti.



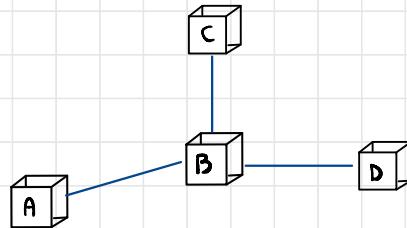
Trasmissione ad anello

La trasmissione ad anello, invece, coinvolge un anello di dispositivi, in cui ogni dispositivo è collegato direttamente a due dispositivi vicini. I dati vengono trasmessi in un unico senso intorno all'anello, e ogni dispositivo riceve i dati trasmessi e li inoltra al dispositivo successivo fino a quando i dati non raggiungono il destinatario effettivo.



Trasmissione a stella

La trasmissione a stella prevede una topologia di rete in cui ogni dispositivo è collegato direttamente a un dispositivo centrale, noto come hub o switch. I dati vengono trasmessi dal dispositivo sorgente all'hub o switch centrale, che inoltra i dati al dispositivo destinatario.



Commutazione a livello di circuito

La comunicazione a livello di circuito è una tecnica di instradamento dei dati nelle reti di computer in cui viene stabilito un circuito dedicato per l'intera durata della comunicazione. In questa tecnica, viene stabilita una connessione tra il mittente e il destinatario prima che i dati vengano trasmessi. Durante la connessione, vengono riservate risorse di rete dedicate (come la larghezza di banda) per la comunicazione, che vengono utilizzate esclusivamente per il circuito stabilito.

Nella comunicazione a livello di circuito, i dati vengono trasmessi attraverso il circuito dedicato in modo sequenziale, seguendo il percorso stabilito durante la connessione. La connessione viene mantenuta per tutta la durata della comunicazione, anche se non vi sono dati da trasmettere.

Quando la comunicazione viene terminata, il circuito viene rilasciato e le risorse di rete precedentemente riservate vengono liberate.

La comunicazione a livello di circuito viene spesso utilizzata in applicazioni che richiedono una comunicazione affidabile e senza interruzioni, come le telefonate. In una chiamata telefonica, ad esempio, viene stabilito un circuito dedicato tra il mittente e il destinatario prima che la conversazione inizi. Durante la conversazione, il circuito dedicato viene utilizzato esclusivamente per la trasmissione della voce, senza interferenze da altri dati o connessioni di rete.

Tuttavia, la comunicazione a livello di circuito può comportare uno spreco di risorse di rete se il circuito viene stabilito ma non viene effettivamente utilizzato per la trasmissione di dati. Inoltre, se il percorso dedicato risulta congestionato o si verifica un guasto durante la comunicazione, la connessione può interrompersi e la comunicazione può essere interrotta.

Commutazione a livello di pacchetto

La commutazione a livello di pacchetto è una tecnica di instradamento dei dati nelle reti di computer in cui i dati vengono divisi in pacchetti prima di essere trasmessi. In questa tecnica, i pacchetti vengono instradati attraverso la rete in base alla destinazione di ogni pacchetto, e possono seguire percorsi diversi lungo la rete.

In una rete a commutazione a livello di pacchetto, ogni pacchetto viene elaborato singolarmente e in modo indipendente dagli altri pacchetti trasmessi nella rete. Ciò significa che ogni pacchetto può essere instradato in modo indipendente in base alla destinazione specifica del pacchetto, senza dover aspettare il completamento della trasmissione di altri pacchetti nella rete.

Uno dei principali vantaggi della commutazione a livello di pacchetto è la flessibilità nella gestione della rete. Poiché ogni pacchetto viene instradato singolarmente, la rete può utilizzare più percorsi e risorse di rete in modo efficiente. Inoltre, se un percorso diventa congestionato o si verifica un guasto nella rete, i pacchetti possono essere instradati su percorsi alternativi per evitare ritardi o perdite di dati.

Tuttavia, poiché i pacchetti sono elaborati individualmente, la commutazione a livello di pacchetto può richiedere maggiori risorse di elaborazione rispetto alla commutazione a livello di circuito (dove viene stabilito un circuito dedicato per l'intera durata della comunicazione). Inoltre, la commutazione a livello di pacchetto può comportare ritardi maggiori nella consegna dei pacchetti, poiché ogni pacchetto deve essere instradato in base alla destinazione specifica.

Commutazione a circuito virtuale

La comunicazione a circuito virtuale è una tecnica di instradamento dei dati nelle reti di computer che combina le caratteristiche della comunicazione a livello di circuito e della commutazione a livello di pacchetto. In questa tecnica, viene stabilito un percorso virtuale tra il mittente e il destinatario prima che i dati vengano trasmessi. Durante la creazione del percorso virtuale, vengono riservate risorse di rete dedicate per la comunicazione, come la larghezza di banda.

Viene inviato un pacchetto esploratore che esplora il cammino commutatori (router e switch) una sola volta e dopo di che si sfrutta questo flusso che viene identificato specificando solo l'host di destinazione.

Una volta che il percorso virtuale è stato creato, i dati vengono divisi in pacchetti e trasmessi lungo il percorso virtuale, seguendo le informazioni di instradamento incluse nei pacchetti.

Durante la trasmissione dei pacchetti, le risorse di rete riservate per il percorso virtuale vengono utilizzate esclusivamente per tale percorso.

La comunicazione a circuito virtuale combina i vantaggi della comunicazione a livello di circuito e della commutazione a livello di pacchetto. Come la comunicazione a livello di circuito, la comunicazione a circuito virtuale garantisce una connessione affidabile e senza interruzioni tra il mittente e il destinatario, utilizzando risorse di rete dedicate. Tuttavia, come la commutazione a livello di pacchetto, la comunicazione a circuito virtuale consente di utilizzare le risorse di rete in modo efficiente, poiché i pacchetti possono essere instradati attraverso la rete in base alla destinazione specifica di ogni pacchetto.

La comunicazione a circuito virtuale viene spesso utilizzata in applicazioni che richiedono una comunicazione affidabile e con una bassa latenza, come le videochiamate o la trasmissione di dati in tempo reale. Inoltre, la comunicazione a circuito virtuale consente di ridurre lo spreco di risorse di rete rispetto alla comunicazione a livello di circuito, poiché le risorse di rete vengono riservate solo per la durata del percorso virtuale, e non per l'intera durata della comunicazione.

Per la progettazione di reti informatiche vengono utilizzati i seguenti modelli architetturali :

Client-Server (HTTP, IMAP, FTP)

Questo modello architettonico prevede l'esistenza di due entità :

- Client : un host non necessariamente attivo dotato di un indirizzo **IP dinamico** e che non comunica direttamente con tutti gli altri host client. Si organizzano comunicazioni tra client passando per i server
- Server : host sempre attivi con un indirizzo **IP fisso** locati in dei data-center per garantire una connessione e un hardware ottimali

La comunicazione viene iniziata sempre dal client, esso invia una richiesta al server attraverso la rete e attende la risposta, che viene inviata dal server. In questo modello, il server ha il compito di elaborare le richieste dei client e di fornire loro le informazioni richieste. Il modello client-server è spesso utilizzato in contesti in cui il controllo centrale è importante, come ad esempio nella gestione di database o nella distribuzione di contenuti web. Il server è sempre attivo e controlla le richieste dei client, fornendo risposte appropriate.

Nel modello client-server, il server dispone di un'interfaccia di rete attraverso la quale ascolta le richieste dei client. Questa interfaccia è definita come "ascolto attivo" o "listening socket" ed è una delle entità fondamentali del server. Quando il server riceve una richiesta dal client, crea una "socket di connessione" attraverso la quale il server e il client possono comunicare tra loro.

Peer-to-Peer (P2P)

Il modello architettonico peer-to-peer (P2P) prevede la presenza di dispositivi che agiscono sia come client che come server. In questo modello, ogni dispositivo è in grado di richiedere informazioni o servizi ad altri dispositivi nella rete e di fornire informazioni o servizi ad altri dispositivi. Non c'è un server centrale che gestisce tutte le richieste, ma ogni dispositivo è in grado di comunicare con gli altri dispositivi della rete. In questo modo, ogni dispositivo può contribuire alla rete, inviando e ricevendo informazioni e servizi.

Il modello architettonico client-server prevede la presenza di un server centrale che fornisce informazioni e servizi ai client, mentre il modello architettonico peer-to-peer prevede la condivisione di informazioni e servizi tra i dispositivi nella rete, senza la presenza di un server centrale.

Due host comunicano attraverso un'interfaccia di comunicazione fornita dal SO. Un client comunica col server tramite messaggi e il loro scambio permette la **sincronizzazione** tra processi.

Il client invia sempre la comunicazione ed il server riceve rimanendo sempre in ascolto.

Socket

3 processi inviano / ricevono messaggi alti/ dalle loro socket.
Una socket è una coppia univoca IP, numero di porta che consente la comunicazione tra processi su sistemi operativi diversi.

Sono un modo per entrare/uscire dal SO.

es. IP address : 128.113.245.12
port number : 80

HTTP server : 80
mail server : 50

Il server ha porte e IP fatti e finiti mentre il client ha IP variabile e porta assegnata dal SO al momento della comunicazione.

Socket \Rightarrow è un'interfaccia software per la trasmissione e la ricezione di dati attraverso una rete. Essa c'è il punto in cui il codice applicativo di un processo accede alla comunicazione. Come un processo sull'una altra macchina tramite un porto.

Un socket permette di instaurare le varie comunicazioni all'interno di una rete, sfruttando la pila TCP/IP.

Un socket è composto da:

- indirizzo IP: codice univoco della macchina interessata (grandezza 32 BIT)
- numero di porta: numero usato dal singolo processo per richiedere il protocollo (grandezza 16 BIT)

I messaggi scambiati devono sottostare al protocollo, ed ogni protocollo è documentato dalla RFC.

I protocolli di trasporto godono delle seguenti proprietà:

- Trasferimento dati affidabile: i dati arrivano in modo corretto e completo
- Throughput: quantità di pacchetti spediti nell'unità di tempo (diverso dalla banda che rappresenta invece la capienza massima)
- Timing: avere la certezza che i dati arrivino entro un determinato lasso di tempo
- Sicurezza: fornire riservatezza impedendo che host terzi possano accedere ai messaggi scambiati

Classifichiamo nel seguente modo le porte

- 1) Well-known ports: sono le porte numerate da 0 a 1023. Sono riservate per l'uso da parte di protocolli noti e standardizzati.
 - Porta 20 e 21: utilizzate dal protocollo **FTP** (File Transfer Protocol) per il trasferimento di file tra server e client.
 - Porta 22: utilizzata dal protocollo **SSH** (Secure Shell) per consentire la connessione remota sicura a un server.
 - Porta 23: utilizzata dal protocollo **Telnet** per la connessione remota a un server.
 - Porta 25: utilizzata dal protocollo **SMTP** (Simple Mail Transfer Protocol) per l'invio di e-mail.
 - Porta 53: utilizzata dal protocollo **DNS** (Domain Name System) per la risoluzione dei nomi di dominio.
 - Porta 80: utilizzata dal protocollo **HTTP** (Hypertext Transfer Protocol) per la comunicazione tra client e server web.
 - Porta 110: utilizzata dal protocollo **POP3** (Post Office Protocol 3) per la ricezione di e-mail.
 - Porta 143: utilizzata dal protocollo **IMAP** (Internet Message Access Protocol) per la ricezione di e-mail.
 - Porta 443: utilizzata dal protocollo **HTTPS** (Hypertext Transfer Protocol Secure) per la comunicazione sicura tra client e server web.
- 2) Registered ports: sono le porte numerate da 1024 a 49151. Sono assegnate a protocolli di rete specifici, ma possono essere utilizzate anche da applicazioni utente.
 - Porta 1433: utilizzata dal protocollo SQL Server per la comunicazione con il database.
 - Porta 3306: utilizzata dal protocollo MySQL per la comunicazione con il database.
 - Porta 8080: utilizzata dal protocollo HTTP alternativo per la comunicazione web.
- 3) Dynamic ports: sono le porte numerate da 49152 a 65535. Sono utilizzate da applicazioni che richiedono l'uso di porte effimere, che vengono assegnate dinamicamente dal sistema operativo durante la creazione di una connessione di rete.

Telnet → porta 23 TCP

Telnet è un protocollo di livello applicativo che utilizza il modello client-server per consentire agli utenti di accedere ai servizi di un dispositivo remoto, solitamente per la configurazione di applicativi. Un utente (client) si connette a un server Telnet sulla porta 23 e, una volta stabilita la connessione, può interagire con il dispositivo remoto. Viene utilizzato per stabilire una connessione remota con un dispositivo di rete, in particolare, Telnet consente agli utenti di accedere e controllare un dispositivo remoto come se fossero fisicamente presenti di fronte ad esso. Telnet utilizza un formato di messaggio basato su testo, in cui i comandi e le risposte sono trasmessi come stringhe di caratteri ASCII.

Tuttavia, Telnet ha alcuni problemi di sicurezza, in quanto le credenziali di accesso (username e password) e tutti i dati trasmessi sono inviati in chiaro, senza crittografia. Per questo motivo, Telnet è stato sostituito dal protocollo SSH (Secure Shell), che fornisce un livello di sicurezza superiore attraverso la crittografia dei dati trasmessi.

FTP (File Transfer Protocol) —> porta 20 e 21

FTP è un protocollo di rete di livello applicativo utilizzato per trasferire file da un host a un altro su una rete IP che utilizza il modello client-server. Un server FTP viene eseguito su un host remoto e gli utenti possono connettersi al server utilizzando un client FTP. Il client e il server FTP comunicano tra loro utilizzando una serie di comandi FTP e risposte, che sono trasmessi tramite una connessione TCP (Transmission Control Protocol) sulla porta 21.

Durante la connessione FTP, il client FTP si connette al server FTP sulla porta 21 e invia le credenziali di accesso (username e password). Una volta autenticato, il client può inviare comandi FTP al server per eseguire operazioni di trasferimento file, come ad esempio l'elenco dei file, la creazione di directory, la cancellazione di file e il trasferimento di file da un host a un altro.

Il protocollo FTP prevede anche l'uso di una seconda connessione TCP per il trasferimento dei dati, che viene aperta dal server sulla porta 20. Tuttavia, a causa di alcune problematiche legate alla sicurezza e alla configurazione dei firewall, FTP può essere configurato per utilizzare una singola connessione TCP per il trasferimento dei dati (modo passivo). FTP è stato uno dei primi protocolli di trasferimento file sviluppati per la rete, ma presenta alcune vulnerabilità di sicurezza, in quanto tutte le informazioni, comprese le credenziali di accesso, vengono trasmesse in chiaro. Gli host devono autenticarsi prima di iniziare la connessione.

HTTP (Hypertext Transfer Protocol) —> porta 80

HTTP è un protocollo di livello applicativo che viene utilizzato per la trasmissione di informazioni su internet, in particolare per la richiesta di pagine web. HTTP è basato sul modello client-server, in cui un client richiede delle informazioni a un server e il server risponde fornendo i dati richiesti. La connessione HTTP è basata su una connessione TCP stabilita sulla porta 80 (o sulla porta 443 se si utilizza HTTPS, la versione crittata di HTTP). Quando un client richiede una risorsa ad un server, invia una richiesta HTTP che contiene un metodo (ad esempio GET, POST, PUT, DELETE) ed un URL che identifica la risorsa richiesta. Il server risponde inviando una risposta HTTP che contiene uno stato (ad esempio 200 OK, 404 Not Found, 500 Internal Server Error) ed i dati richiesti.

Ad ogni richiesta la sessione viene chiusa, quindi viene generata una nuova sessione per ogni richiesta. Se in una pagina sono presenti degli oggetti viene generata una sessione per ogni oggetto (con HTTP 1.0). HTTP 1.1 invece genera una sessione per ogni pagina, quindi chiude la sessione solo dopo aver caricato tutti gli oggetti che conteneva, in un'unica sessione.

HTTP 1.0 è un perfetto esempio di comunicazione stateless perché non tengono conto dello stato della connessione e dello storico degli host, infatti ogni richiesta HTTP è totalmente indipendente dalle precedenti e dalle successive. Supponiamo che si voglia contattare una pagina web che richieda di impostare la lingua per migliorare la visualizzazione del contenuto. Se non si tiene conto dello stato dell'utente ogni volta si va a richiedere la lingua per ogni sessione generata. HTTP 1.1 sopperisce a questa mancanza tramite l'utilizzo dei cookie.

Cookie

I cookie nel protocollo HTTP sono dei piccoli file di testo che vengono memorizzati sul dispositivo del client (ad esempio il browser) dal server quando si effettua una richiesta HTTP. Questi file di testo contengono informazioni sulle preferenze dell'utente e sullo stato della sessione tra il client e il server. In questo modo, il server può utilizzare le informazioni contenute nel cookie per mantenere lo stato della sessione tra le varie richieste HTTP inviate dal client.

I cookie possono contenere informazioni come le preferenze dell'utente, le informazioni sul login, le informazioni sul carrello degli acquisti, ecc. Inoltre, i cookie possono essere utilizzati anche per tracciare l'attività dell'utente sul sito web, ad esempio per fornire pubblicità mirate.

HTTP Request e HTTP Response

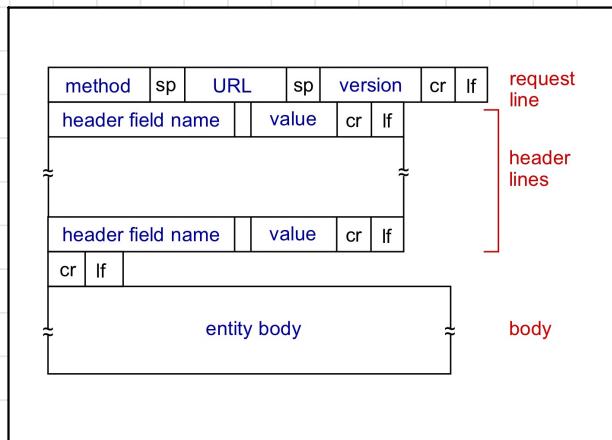
Una richiesta HTTP è costituita da una riga di richiesta (request line) seguita da una serie di header HTTP (HTTP headers) e, optionalmente, dal body della richiesta (request body). La riga di richiesta specifica il metodo HTTP utilizzato per la richiesta (ad esempio GET, POST, PUT, DELETE), l'URL della risorsa richiesta e la versione di HTTP utilizzata. Gli header HTTP contengono informazioni aggiuntive sulla richiesta, come ad esempio il tipo di contenuto richiesto (Content-Type), l'indirizzo IP del client (X-Forwarded-For), il tipo di compressione supportato (Accept-Encoding), ecc. Il body della richiesta, presente solo in alcuni metodi HTTP come POST o PUT, contiene i dati trasmessi dal client al server, come ad esempio un modulo di registrazione o un file da caricare.

Una risposta HTTP è costituita da una riga di stato (status line) seguita da una serie di header HTTP (HTTP headers), optionalmente, dal body della risposta (response body). La riga di stato specifica lo stato della richiesta, rappresentato da un codice numerico a tre cifre, come ad esempio 200 OK per una richiesta eseguita con successo, 404 Not Found per una risorsa non trovata, ecc. Gli header HTTP contengono informazioni aggiuntive sulla risposta, come ad esempio il tipo di contenuto restituito (Content-Type), la data e l'ora di generazione della risposta (Date), la dimensione del contenuto (Content-Length), ecc. Il body della risposta contiene i dati restituiti dal server al client, come ad esempio il contenuto di una pagina web o un file richiesto.

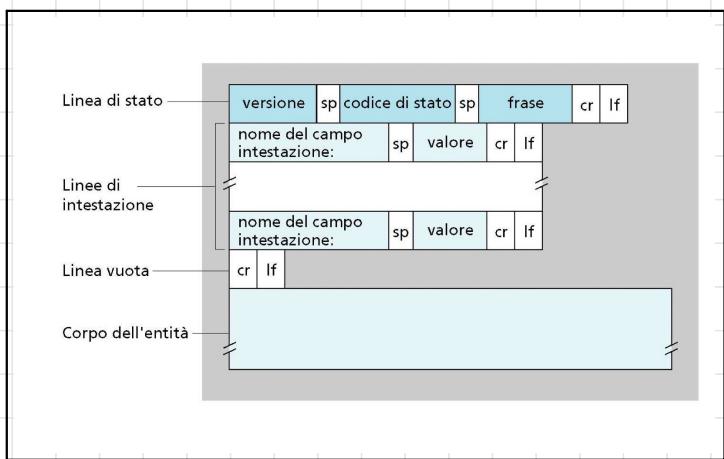
In sintesi, una richiesta HTTP è un messaggio inviato da un client a un server per richiedere una risorsa, mentre una risposta HTTP è un messaggio inviato dal server al client in risposta ad una richiesta, contenente la risorsa richiesta o un messaggio di errore. Entrambe le richieste e le risposte sono costituite da una riga di stato (request line o status line), una serie di header HTTP (HTTP headers) e optionalmente dal body della richiesta o della risposta (request body o response body).

HTTP - Request

La struttura di una richiesta HTTP



La struttura di una richiesta HTTP



I metodi presenti nel metodo della richiesta HTTP sono i seguenti:

Method	Description
GET	Request to read a Web page
HEAD	Request to read a Web page's header
PUT	Request to store a Web page
POST	Append to a named resource (e.g., a Web page)
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Reserved for future use
OPTIONS	Query certain options

put=Richiesta di memorizzazione di una pagina Web

get=Richiesta di lettura di una pagina web

GET => mette nell' URL i campi richiesti al web server

POST => ha lo stesso scopo del metodo GET ma non scrive nell' URL i campi (es. usato per richiedere i login)

le rigiste HTTP fornire i seguenti codici di riferimento al clientante (browser)

Information Codes

100 Continue

Success Codes

200 OK

203 Non-Authoritative Information

204 No Content

Redirection Codes

305 Use Proxy

Client Error Codes

400 Bad Request

403 Forbidden

404 Not Found

405 Method Not Allowed

Server Error Codes

500 Internal Server Error

505 HTTP Version not supported

HTTP era inizialmente un modo per trasferire file da un server ad un client. Una sua evoluzione consiste nel rendere dinamico l'url che genera un file HTML compatibile contenente la pagina web richiesta. La pagina HTML viene generata in C. Nascono dei linguaggi destinati alla creazione di pagine web (PHP) e si differenzia la programmazione lato client e lato server.

Proxy server

Host differenti richiedono la stessa pagina web in differenti istanti di tempo. La prima richiesta passa come se il proxy non esistesse ma attiva un meccanismo di web-caching che mantiene le pagine caricate di recente . I client che richiedono le pagine trovano eventi di cache hit ed il proxy gliela fornisce senza passare dal server principale (fornisce un netto miglioramento delle prestazioni)

Un proxy server è interposto tra client e server.

SMTP(Simple Mail Transer Protocol) → porta 25

Un protocollo di push basato sul modello client-server che consente l'invio di Mail da una macchina all'altra, in modo affidabile, grazie all'uso di TCP. Il protocollo deve conoscere la macchina di destinazione e devo fornire anche l'account mail della macchina da raggiungere. In SMTP un client (solitamente un programma di posta elettronica) si connette ad un server SMTP per inviare un messaggio di posta elettronica. La comunicazione con SMTP avviene attraverso una connessione client-server, dove il client SMTP invia un messaggio email al server SMTP, che a sua volta consegna il messaggio al destinatario. Il processo di invio di un messaggio email tramite SMTP avviene in questo modo.:

1. L'utente della macchina A scrive e compone un messaggio di posta elettronica utilizzando un client di posta elettronica, ad esempio Outlook o Gmail.
2. Il client di posta elettronica sulla macchina A si connette al server di posta elettronica della macchina A utilizzando il protocollo SMTP e gli invia il messaggio.
3. Il server di posta elettronica della macchina A riceve il messaggio di posta elettronica e lo memorizza nella coda di posta in uscita.
4. Il server di posta elettronica della macchina A contatta il server di posta elettronica della macchina B utilizzando il protocollo SMTP, e invia il messaggio di posta elettronica.
5. Il server di posta elettronica della macchina B riceve il messaggio di posta elettronica e lo memorizza nella coda di posta in arrivo.
6. Il destinatario della macchina B controlla la propria casella di posta elettronica e legge il messaggio di posta elettronica.

SMTP è un protocollo affidabile ma non cifrato, il che significa che il testo del messaggio email può essere intercettato durante il trasferimento.

È importante notare che SMTP si occupa solo dell'invio di email e non della ricezione. Per la ricezione di email, viene invece utilizzato un protocollo differente, come ad esempio POP3 o IMAP.

POP3 (Post Office Protocol version 3) e IMAP (Internet Message Access Protocol)

SMTP, essendo un protocollo di push, trasferisce il messaggio dallo user agent del mittente al suo mail server e successivamente fra i mail server di due host, non può essere usato per prelevare messaggi dal mail server.

POP3 è un protocollo di tipo "pull", ovvero il client si connette al server per scaricare le email e le scarica completamente, eliminandole dal server. In altre parole, il client si comporta come una sorta di "postino" che va a ritirare la posta dal server e la consegna al destinatario. Questo comporta che le email scaricate dal client non siano più disponibili sul server e non possono essere accedute da altri dispositivi o client.

IMAP è un protocollo di posta elettronica basato sul modello client-server, che consente ai client di posta elettronica di selezionare e visualizzare i messaggi di posta elettronica presenti sul server, inviando richieste specifiche per recuperare informazioni come le intestazioni dei messaggi o il testo completo dei messaggi. In questo modo, il client di posta elettronica può visualizzare, gestire e modificare i messaggi di posta elettronica senza doverli scaricare e archiviare localmente sul dispositivo client.

In sintesi, POP3 è un protocollo di posta elettronica che scarica le email dal server sul client, eliminandole dal server, mentre IMAP è un protocollo di posta elettronica che permette di visualizzare e gestire le email direttamente sul server, mantenendole sempre disponibili e sincronizzate su diversi dispositivi e client. La scelta del protocollo dipende dalle esigenze dell'utente, ad esempio se si vuole accedere alle email da più dispositivi o se si vuole mantenere uno storico delle email sul server.

DOMINIO

Dominio si riferisce a un gruppo di computer e dispositivi connessi in una rete e gestiti come un'entità unica. Un dominio di rete rappresenta un gruppo di computer che condividono un insieme comune di risorse e servizi.

L'indirizzo di posta della macchina diventa "nome utente"-“dominio” così da non dover specificare l'identificativo della macchina ma un semplice dominio che rappresenta un insieme di macchine , tutte equivalenti tra di loro , alle quali l'utente si connette per ottenere servizi. Avere molte macchine mi consente di ovviare al problema del guasto di una macchina, infatti a fronte di guasti non perdo comunque mail grazie alla sincronizzazione con gli altri server.

Esistono le Black-List , ovvero delle mail-list nella quale vengono indirizzate tutte le mail che provengono da mittenti non affidabili . Ogni server può utilizzare la propria lista nera ed è compito dell'amministratore di sistema occuparsi che le proprie macchine finiscano in qualche lista nera.

DNS (domain name system)

-> in rete locale

Un server DNS è un server con una lista di indirizzi IP associati ai nomi, che viene sempre aggiornata. Quando un client richiede l' IP di un determinato indirizzo al DNS , esso lo restituisce correttamente. Un DNS consente quindi di ottenere l'IP a partire da un determinato nome, in pratica associa nomi ad indirizzi IP.

-> su rete internet

Quando un utente digita un nome di dominio nel proprio browser web, il browser invia una richiesta a un server DNS, che cerca quindi l'indirizzo IP associato a quel nome di dominio e lo restituisce al browser, consentendo al browser di connettersi al server corretto su Internet.

- I **DNS Root Server** sono un insieme di server DNS che costituiscono il punto di partenza della gerarchia del DNS. Essi contengono informazioni sui server DNS autoritativi per ciascun Top Level Domain (TLD) e forniscono risposte alle query DNS inviate dai client.
- I **Top Level Domain (TLD)** sono i domini di primo livello nella gerarchia del DNS, come .com, .org, .net, .it, .fr, ecc. Essi rappresentano il livello più alto della gerarchia del DNS e sono gestiti da organizzazioni chiamate registri.
- I **DNS Autoritativi** sono i server DNS che contengono le informazioni ufficiali sui nomi di dominio e gli indirizzi IP corrispondenti. Essi sono responsabili della risposta alle query DNS relative ai nomi di dominio per cui sono autorizzati. Ospitano gli indirizzi IP forniti dalle aziende.
- I **DNS Locali**, noti anche come resolver DNS, sono i server DNS utilizzati dai client per inviare le query DNS. Essi memorizzano temporaneamente le informazioni DNS ottenute dalle query precedenti, in modo da ridurre i tempi di risposta per le query successive. I DNS Locali possono essere gestiti dal provider di servizi Internet (ISP) dell'utente o da un server DNS configurato localmente.

Si pensa ad un sistema di interrogazione di un database distribuito : l'informazione reale è conosciuta dal root server DNS del dominio a cui si vuole fare riferimento.

es. chi è l'indirizzo IP di www.alfiospoto.unict.it è conosciuto dal server root DNS del dominio unict

Un indirizzo viene letto da destra verso sinistra

- > it (mi trovo in italia)
- >unict (server DNS)

La struttura è una gerarchia ad albero dove sono presenti anche dei rami trasversali per evitare traffico risalendo sempre in radice (non è necessario passare per i root-server), quindi la struttura diventa un grafo.

Server DNS

Si realizza un DNS primario e dei DNS secondari che hanno una copia (periodicamente aggiornata) del DNS primario, usati per ricevere le richieste dei client e moderare il traffico sul DNS primario. Le richieste , dette Query DNS possono essere :

- **autoritative** : arrivano dal DNS primario
- **non autoritative** : le richieste costituite dal meccanismo di caching dei DNS

I provider modificano l'indirizzo ip del client continuamente in modo da non consentire all'utente medio di mettere su un server.

Esistono i DNS dinamici : tramite login si ha la possibilità di aggiornare la coppia client-IP.

Un'alternativa consiste nel far comunicare il firmware con un server web locato in un determinato luogo e con indirizzo IP fisso.

Un record DNS è strutturato nel seguente modo

RR format: (name, value, type, ttl)	
type=A	type=CNAME
<ul style="list-style-type: none">• name is hostname• value is IP address	<ul style="list-style-type: none">• name is alias name for some "canonical" (the real) name• www.ibm.com is really servereast.backup2.ibm.com• value is canonical name
type=NS	type=MX
<ul style="list-style-type: none">• name is domain (e.g., foo.com)• value is hostname of authoritative name server for this domain	<ul style="list-style-type: none">• value is name of mailserver associated with name

AAAA = indirizzo IPV6

SNMP (simple network management protocol)

Pensato per poter gestire la rete . Ha un agent con i permessi di amministratore in grado di interrogare il Sistema Operativo . Questo agent organizza i dati ottenuti in un database organizzato ad albero , il MIB. SNMP deve essere esplicitamente abitato sul dispositivo , abbiamo tre versioni di SNMP .

Livello di trasporto

Un protocollo di trasporto sfrutta il modello end-to-end. Il termine "end-to-end" si riferisce al fatto che il protocollo di trasporto opera direttamente tra gli host (cioè le estremità) che vogliono comunicare tra loro, senza che nessun altro dispositivo di rete intervenga nella gestione della comunicazione. Il livello di trasporto svolge un ruolo importante nella gestione delle connessioni di rete. Grazie ai suoi meccanismi di controllo del flusso e di controllo della congestione, TCP garantisce una trasmissione affidabile e ordinata dei dati su una rete IP. Tuttavia, in alcune situazioni, come quelle in cui è necessaria una bassa latenza, UDP può essere preferibile per la sua maggiore leggerezza e velocità.

UDP (User Datagram Protocol) —> RFC 768 (request for comment)

UDP è un protocollo di trasporto non orientato alla connessione, per tanto non vi è nessuna procedura di handshake tra gli host comunicanti. Inoltre esso è inaffidabile: invia i dati sotto forma di pacchetti noti come datagrammi, che possono essere trasmessi sulla rete senza alcuna garanzia di consegna o di sequenza. Ciò significa che non vi è alcuna garanzia che il destinatario riceva tutti i dati inviati, né che li riceva nell'ordine corretto. Questo protocollo non include nessun meccanismo di controllo congestione, per tanto spedisce pacchetti ad una velocità molto più alta di TCP.

UDP è utilizzato principalmente in applicazioni che richiedono una trasmissione veloce dei dati o applicazioni che richiedono una bassa latenza di rete e che riescono ad ammettere inaffidabilità nella trasmissione di dati (**loss-tollerant**).

Un pacchetto UDP, anche noto come datagramma UDP, è costituito da un'intestazione (header) e dai dati da trasmettere.

L'intestazione UDP è composta da 4 campi di 2 byte ciascuno, per un totale di 8 byte. I campi sono i seguenti:

- Porta di origine: indica la porta di origine del mittente del pacchetto UDP.
- Porta di destinazione: indica la porta di destinazione del destinatario del pacchetto UDP.
- Lunghezza: indica la lunghezza totale del datagramma UDP, espressa in byte, incluso l'intestazione.
- Checksum: è un valore di controllo di 2 byte calcolato sui dati e sull'intestazione del pacchetto, utilizzato per verificare l'integrità del datagramma.

Checksum

I controlli sulla correttezza di ciò che arriva vengono fatti a livello molto più basso. Si sommano i valori dei campi di intestazione e contenuto dei dati. Non è raro che una macchina ignori il checksum.

Il checksum non assicura che il risultato sia giusto ma serve solo a vedere in che condizioni è sicuramente sbagliato. Tra richiesta e risposta non cambia nulla, vengono solo invertiti i numeri di porta tra sorgente e destinazione.

Per la realizzazione di un canale affidabile è essenziale numerare e mantenere ordinato l'invio dei pacchetti. Si ha a disposizione soltanto un canale inaffidabile che in buona percentuale viene reso affidabile grazie a degli accorgimenti. La situazione viene complicata dal fatto che gli host comunicanti non si vedono.

Come si ottiene un canale affidabile a partire da qualcosa di inaffidabile?

Sfruttiamo gli automi a stati finiti e lo stato viene definito da alcune "variabili di stato".

RDT(Reliable Data Transfer)

RDT è un protocollo che viene utilizzato per garantire la consegna affidabile dei dati tra due entità di una rete di calcolatori, anche in presenza di errori di trasmissione.

Il protocollo RDT utilizza diverse tecniche per garantire la consegna affidabile dei dati, come **l'acknowledgement (ACK)** e **il retransmission timeout (RTO)**. In particolare, il mittente invia i dati al destinatario e attende un ACK di conferma. Se l'ACK non arriva entro un certo intervallo di tempo (RTO), il mittente reinvia i dati. In questo modo, il protocollo RDT garantisce che i dati vengano consegnati in modo affidabile, anche in presenza di errori di trasmissione.

RDT 1.0

Trasferimento dati su un canale perfettamente affidabile (banale). Il Sender crea un pacchetto e lo spedisce al canale di comunicazione mentre il Receiver prende un pacchetto dal canale ed estrae i dati. Con un canale così affidabile non è necessario che le due macchine comunichino tra di loro in quanto nulla può andare storto.

RDT 2.0

Si verifica spesso, data l'inaffidabilità, che si perdono i pacchetti e le macchine lo rilevano attraverso un meccanismo di **checksum**. Quando viene inviato un pacchetto si sfrutta il meccanismo **stop-and-wait**. La macchina ricevente notifica di aver ricevuto correttamente il pacchetto attraverso un messaggio di ACK(dati ricevuti correttamente), se viene ricevuto con errore si chiede la ritrasmissione con un NAK(dati corrotti). Si necessita di aggiungere uno stato di stop and wait alla macchina Sender. Il Sender spedisce un pacchetto alla volta : se non riceve una notifica di ACK/NAK per quel pacchetto, non proseguirà con l'invio del pacchetto successivo. Se gli errori sono presenti anche sul canale di ritorno bisogna aggiungere un checksum anche sul ACK/NAK e ritrasmettere l'ultimo pacchetto a seguito della ricezione di un ACK/NAK corrotto. Tuttavia questo introduce il problema di pacchetti duplicati, in quanto non è detto che il pacchetto sia stato realmente perduto. L'unica cosa che si può fare è rispedire l'ultimo pacchetto distinguendo pacchetti pari e dispari .

RDT 2.1

La soluzione consiste nell'etichettare i messaggi inviati con un **numero di sequenza (0/1)** e al Sender sarà sufficiente valutare questo numero per capire se il pacchetto rappresenti una ritrasmissione o meno. Sender e Receiver introducono quindi di un nuovo stato che consentono di valutare il numero di sequenza del pacchetto (basta un solo bit per valutare lo stato 0/1).

RDT 2.2

Ci si rende conto che i **NAK non sono necessari**. Possono essere eliminati grazie all'invio sempre di segmenti di ACK con l'identificativo dell'ultimo pacchetto ricevuto correttamente. Se il Sender riceve due ACK con lo stesso identificativo (ACK duplicati) allora sa che il Receiver non ha ricevuto correttamente il pacchetto successivo e glielo ritrasmette. Il funzionamento del receiver viene quindi semplificato.

RDT 3.0

Se i pacchetti inviati vengono persi ?

Si necessita di un meccanismo di temporizzazione (**meccanismo di timeout**) che risincronizza le operazioni.

RTO (RetransmissionTimeOut) è utilizzato dal protocollo RDT per garantire che i dati inviati dal mittente siano ricevuti dal destinatario in modo affidabile. In RDT, il mittente invia un segmento di dati al destinatario e attende una conferma di ricezione (ACK) dal destinatario. Se il mittente non riceve l'ACK entro un determinato intervallo di tempo RTO, assume che il segmento sia andato perso e lo ritrasmette. Nelle reali implementazioni si lancia un'eccezione se il pacchetto viene perso un certo numero di volte per evitare di mandare in loop il sistema.

Si può causare un ritardo nell'invio dei segmenti di ACK e l'unica circostanza che fa fallire il protocollo è data dal tempo di viaggio dei pacchetti che **non è costante** : il sender riceve una risposta di ACK relativa ad un pacchetto vecchio che viene scambiata per il pacchetto attuale , quindi nessuna delle due macchine capisce che c'è stato un errore nella trasmissione.

Si necessita quindi di aumentare i bit usati per l'identificazione del pacchetto (solo un bit per 0/1) per evitare di confondere la risposta ad un pacchetto recente con quella di un pacchetto con lo stesso identificativo ma inviato precedentemente.

Bisogna modificare il timeout : non posso introdurre in meccanismo di temporizzazione costante perché non tutti i pacchetti hanno la stessa strada da fare, alcuni hanno percorsi molto più brevi di altri.

Un valore troppo basso di RTO può causare una ritrasmissione eccessiva dei segmenti, riducendo le prestazioni della rete. D'altra parte, un valore troppo alto può causare un ritardo nella ritrasmissione dei segmenti persi, aumentando il tempo di consegna dei dati.

Con "throughput" intendiamo la quantità di lavoro svolta nell'unità di tempo. Utilizzando lo stop and wait il tempo totale che impiega il datagramma per viaggiare nella rete, arrivare al destinatario ed essere elaborato sarà dato dalla seguente formula :

$$t = T_{\text{frame}} + \text{RTT} + T_{\text{elaboration}} + T_{\text{ack}}$$

- **T_frame** : tempo necessario per trasmettere un frame di dati.
- **RTT** (Round Trip Time) : tempo che impiega un pacchetto per andare dal mittente al destinatario e tornare indietro.
- **T_elaboration** : tempo che impiega il ricevente per elaborare il frame ricevuto, che può includere la decodifica del frame, la verifica degli errori e l'eventuale invio di un messaggio di errore al mittente.
- **T_ack** : il tempo necessario per trasmettere un pacchetto di ACK dal destinatario al mittente per indicare che il frame è stato ricevuto correttamente.

In una comunicazione "stop and wait", il mittente trasmette un singolo frame e poi attende la conferma del destinatario prima di trasmettere il successivo. Pertanto, il tempo totale di trasmissione è la somma di tutti i tempi sopra elencati

RDT 3.0 risulta affidabile ma inefficiente in termini di prestazioni. Il problema risiede nella strategia stop and wait. Il mittente non potrà mai sfruttare tutta la banda disponibile dato che prima di spedire un pacchetto, deve attendere l'ACK del pacchetto precedente, inoltre non va bene in quanto introduce un grosso dispendio di tempo per le distanze lunghe.

Pipelining

Si introduce un meccanismo di Pipelining che prevede che al posto della fase di wait successiva all'invio del primo pacchetto si inviano altri pacchetti in quella **finestra di spedizione** che tiene un certo numero di pacchetti fino al primo ACK. Si ha un sistema che mi permette di minimizzare il numero di tempi morti, ovvero i momenti in cui il sender rimane in attesa di una risposta senza inviare pacchetti, infatti se la finestra di spedizione è di 3 pacchetti, allora avrò migliorato le prestazioni in quanto 3 pacchetti accodati occuperanno il canale e non attendo che si sia ricevuto il pacchetto per spedire l'altro (avrò triplicato il Troughput). Si può implementare il pipelining attraverso gli approcci Go-Back-N e Ripetizione Selettiva.

GBN (Go Back N)

RDT che sfrutta solo stop and wait va bene solo per le distanze brevi. Go back N prevede una finestra di trasmissione ma anche di ricezione. In caso di perdita di un pacchetto si ritorna indietro dal punto in cui si era interrotta la sincronizzazione , di cui si tiene conto tramite il segmento di **ACK cumulativo** . Indica che fino al pacchetto n, sono stati ricevuti correttamente tutti i pacchetti precedenti, il re-invio dei pacchetti riguarda quindi tutti quelli da n+1 a seguire. Il problema è che si reinvia anche tutti i pacchetti che seguono il pacchetto perso che erano stati ricevuti correttamente. Si basa sulla suposizione che i pacchetti successivi al primo rovintato, probabilmente siano rovinati anch'essi.

Ripetizione selettiva

Quando la finestra è grande, nella pipeline si possono trovare numerosi pacchetti. Se un pacchetto viene ricevuto in maniera errata o perso, verranno ritrasmessi anche tutti quelli che seguono, anche se per la maggior parte di questi è stata ricevuta correttamente. Se le perdite sono sporadiche si reinvia solo i pacchetti su cui vi è un sospetto di errore. Infatti il destinatario invia un riscontro per i pacchetti ricevuti fuori sequenza.

TCP / IP v4 (transmission control protocol)

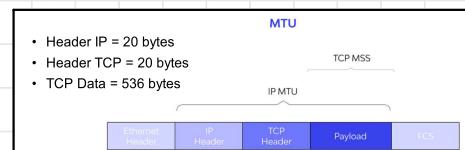
TCP è un protocollo a livello di trasporto che offre alle applicazioni un servizio di comunicazione affidabile e orientato alla connessione che va in esecuzione solo su sistemi periferici, i router intermedi sono ignari della connessione TCP.

TCP offre una connessione di tipo **full-duplex** in quanto i dati possono fluire da mittente a destinatario e viceversa. Questo protocollo implementa il modello **end-to-end**, ossia la comunicazione ha luogo esclusivamente fra due host (multicast non ottenibile tramite TCP). TCP frutta l'architettura **peer-to-peer** in quanto gli host comunicanti sono alla pari (non c'è un dispositivo master e uno slave).

L' MSS (Maximum segment size) definisce la massima quantità di dati inseriti in un segmento. Questo valore sommato all'intestazione TCP (20 byte) deve essere minore dell'unità trasmissiva massima (MTU) ovvero la grandezza massima di un frame che può essere inviato in un collegamento.

Struttura di un segmento TCP

Un pacchetto TCP (Transmission Control Protocol) è composto da un payload (al massimo MSS) e da un header :



- Porta di origine (2 byte): il numero di porta del mittente.
- Porta di destinazione (2 byte): il numero di porta del destinatario.
- Numero di sequenza (4 byte): il numero di sequenza del primo byte di dati contenuti nel pacchetto.
- Numero di conferma (4 byte): il numero di sequenza del prossimo byte di dati attesi dal mittente.
- Lunghezza dell'header (4 bit): la lunghezza dell'header TCP in parole da 4 byte.
- Bit di controllo (6 bit): contiene vari flag di controllo, tra cui il flag **SYN**, **ACK**, **FIN** e altri.
- Window size (2 byte): il numero di byte di dati che il mittente può accettare dal destinatario.
- Checksum (2 byte): un valore di controllo che viene utilizzato per rilevare eventuali errori nel pacchetto.
- Puntatore di urgente (2 byte): se il flag di urgente è impostato, il puntatore di urgente viene utilizzato per indicare il byte di dati successivo che richiede un'attenzione immediata.
- Opzioni (variabile): opzioni aggiuntive, come ad esempio il massimo numero di segmenti di dati che possono essere inviati prima di ricevere una conferma.

Il payload TCP contiene i dati veri e propri che vengono trasmessi dal mittente al destinatario. La dimensione del payload è variabile e dipende dalla quantità di dati da trasmettere.

TCP indica il numero del primo byte del segmento, per esempio se un host deve inviare segmenti di 1000 byte ciascuno , allora il primo ha un numero di sequenza 0, il secondo 1000, il terzo 2000 ecc..

Timeout TCP

TCP usa un timer per far fronte alla possibile perdita di pacchetti. Definiamo RTT il tempo che intercorre tra l'invio di un segmento e la ricezione del suo ACK di verifica. Per calcolare il RTT in TCP, si utilizza spesso la media ponderata **EWMA** (Exponentially Weighted Moving Average). Questa tecnica calcola una media mobile dei tempi di risposta precedenti, assegnando maggior peso ai tempi di risposta più recenti. L'equazione per il calcolo del RTT EWMA è la seguente:

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

- RTT è il valore attuale del RTT calcolato;
- alpha è un fattore di peso ($\alpha = 0.125$) che indica quanto peso assegnare al SampleRTT più recente rispetto ai valori precedenti;
- SampleRTT è il tempo di risposta campionato per il pacchetto più recente.

La DevRTT viene utilizzata per calcolare una stima della variazione del RTT nel tempo, cosa di cui non teneva conto la EWMA. Questa informazione può essere utilizzata da applicazioni di rete per adattare il loro comportamento alle attuali condizioni di rete. La formula per calcolare la deviazione media ponderata EWMA del RTT (DevRTT) è la seguente:

$$\text{DevRTT} = (1 - \text{beta}) * \text{DevRTT_precedente} + \text{beta} * |\text{RTT} - \text{Estimated_RTT}|$$

- DevRTT è la deviazione media ponderata EWMA del RTT calcolata in un determinato momento
- DevRTT_precedente è il valore di DevRTT calcolato al momento precedente
- beta (beta = 0.25) è un parametro di smoothing che determina il peso relativo dei valori passati e presenti di RTT nella media mobile.
- RTT è il tempo impiegato per inviare un pacchetto di dati da un computer a un altro e ricevere una risposta (Round-Trip Time)
- Estimated_RTT è il valore di RTT medio calcolato al momento precedente

I tempo massimo di attesa per una risposta, RTO, viene calcolato come la somma dell'Estimated RTT (RTT stimato) e di un valore di DevRTT moltiplicato per un fattore di 4.

$$\text{RTO} = \text{Estimated RTT} + 4 * \text{DevRTT}$$

Questo valore di DevRTT moltiplicato per 4 rappresenta un'ampiezza di banda di sicurezza per la finestra di trasmissione. In pratica, il RTO viene impostato in modo che la probabilità di perdere un pacchetto sia inferiore al 1%.

ACK Cumulativi

significa che il destinatario di un flusso di dati TCP conferma la ricezione di tutti i dati fino a un certo punto in una sola volta, invece di confermare la ricezione di ciascun pacchetto singolarmente. L'uso di ACK cumulativi in TCP consente al protocollo di gestire in modo efficiente la ritrasmissione dei dati mancanti, evitando di inviare pacchetti duplicati e migliorando l'affidabilità della trasmissione dei dati.

Fast Retransmitt

Al fine di diminuire i tempi di attesa nella ritrasmissione, TCP metta a disposizione un ulteriore servizio. Se il mittente riceve tre ACK duplicati per un segmento, allora egli considera il segmento che lo segue perduto e quindi lo ritrasmette, ancor prima che il timer per il segmento smarrito, scada. Ad esempio potrebbe accadere che il pacchetto n venga perso ma il destinatario riceva i pacchetti n+1, n+2 ed n+3 prima che scada il timer. Come conseguenza, conserva i pacchetti nel buffer ed invia un ACK duplicato per il pacchetto atteso, n appunto. Quando il mittente TCP riceve i tre ACK duplicati capisce che il pacchetto n si è perso e quindi lo ritrasmette.

Controllo di flusso

TCP offre un servizio di controllo di flusso affinché il mittente non saturi il buffer del ricevente.

NB : non bisogna confondere controllo di flusso con controllo di congestione. La differenza fondamentale è che il controllo del flusso si concentra sul flusso di dati tra due dispositivi di comunicazione, mentre il controllo della congestione si concentra sulla gestione del traffico in una rete di comunicazione più ampia.

Il receiver comunica al sender quale è la capienza di byte che può ricevere e il sender invia nel RcvBuffer dei pacchetti della capienza consentita. Mi serve un meccanismo che verifica se accumulare pacchetti(bufferizzare) ed inviare o se spedire pacchetti poco alla volta.

Algoritmo di Nagle

L'algoritmo di Nagle è un algoritmo utilizzato nei protocolli di trasmissione a livello di rete come il TCP ed è stato progettato per migliorare l'efficienza della rete riducendo il numero di pacchetti trasmessi.
Esso funziona nel seguente modo:

1. Il mittente deve accumulare dati sufficienti prima di inviare un pacchetto. In altre parole, un pacchetto viene inviato solo quando il mittente ha una quantità di dati significativa da trasmettere, invece di inviare un pacchetto ogni volta che ha un byte di dati da inviare.
2. Il mittente invierà un pacchetto immediatamente se il pacchetto precedente è stato confermato dal destinatario.
3. Se il pacchetto precedente non è stato ancora confermato, il mittente attenderà prima di inviare il nuovo pacchetto. Questa attesa viene chiamata "ritardo di Nagle" e dura circa 200 millisecondi.

L'algoritmo di Nagle aiuta a ridurre il traffico sulla rete, poiché riduce il numero di pacchetti inviati, ma può anche aumentare la latenza, in quanto i pacchetti vengono inviati con un certo ritardo.

Si necessita delle parti del protocollo TCP che servono per aprire e chiudere la connessione.

TCP offre un servizio orientato alla connessione : i dispositivi che vogliono comunicare instaurano prima una connessione mediante una procedura di handshake a tre vie con la quale si instaura una connessione TCP tra gli host che devono scambiarsi dei segmenti preliminari per stabilire come avverrà la trasmissione dei dati.

Si aveva inizialmente un handshake a 2 vie ma si ha un problema in quanto la risposta al mittente è standard e non viene data in funzione di quanto ricevuto. Si ha un handshake a tre vie per garantire che il messaggio di risposta sia conseguenza del messaggio di richiesta e poi si invia un messaggio al mittente che risponderà anch'esso in funzione di quanto richiesto.

Nella connessione TCP si ha un numero di sequenza e un ACK . Il numero di sequenza dei pacchetti parte da un numero pseudo-randomico per garantire che la risposta sia in funzione della richiesta : di solito si usano le cifre meno significative della traduzione binaria dell'ora corrente.

Concetto di sfida

Host 1 vuole aprire una connessione con Host 2. E Host 2 risponde. Dato che Host 1 non vede l'Host 2 e Host 2 non vede l'Host 1, ma semplicemente possono scambiarsi il messaggio; chi c'è dall'altra parte? Ci sono messaggi vecchi provenienti da Host 2 che si sono persi? Che spuntano improvvisamente senza motivo? Oppure c'è un reale tentativo di aprire la connessione? Dato che non si vedono, la sfida è "io ti mando un numero, se tu ora sei presente, sei online, allora mi devi rispondere con un valore legato al numero che ti sto mandando". Poi la sfida viene invertita. L'Host 2 sfida l'Host 1 a fare la stessa operazione. Qui non è un problema di sicurezza, è un problema solo per garantire che vecchi messaggi che circolano nella rete per qualunque motivo, che non ci interessa, non vadano ad aprire connessioni, quindi a occupare risorse in maniera inutile. Host 2 apre la connessione solamente se la sua sfida, quella che manda verso Host 1, ha successo.

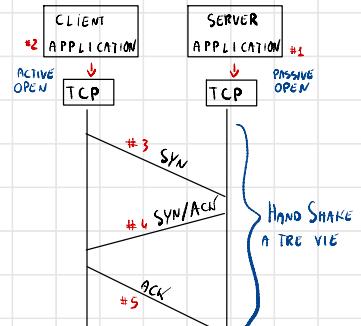
Apertura connessione TCP

L'apertura di una connessione TCP prevede che il dispositivo client invii un pacchetto di richiesta di connessione al dispositivo server. Questo pacchetto viene chiamato "SYN" (synchronize) e **contiene informazioni sulla porta di origine e di destinazione, oltre a un numero di sequenza iniziale**. Il dispositivo server, se disponibile e pronto a stabilire una connessione, risponderà con un pacchetto "SYN-ACK" (synchronize-acknowledge) che **conferma la disponibilità del server e invia un numero di sequenza iniziale**. Il dispositivo client quindi risponderà con un pacchetto "ACK" (acknowledge) che **conferma la ricezione del pacchetto SYN-ACK e completa la connessione**. Il bit SYN viene **posto a 1 per i primi due pacchetti inviati**. Il mittente invia il primo pacchetto ed avvia un timer . Non avendo campioni su cui basarmi imposto un timer molto grande per essere sicuro di coprire il tempo di risposta.

Questo approccio ha un problema di sicurezza : vengono bloccate delle risorse che verranno istanziate per tutta la durata del timer. Se si inviano delle richieste continue al server esso si blocca in quanto si satura il buffer di richieste.

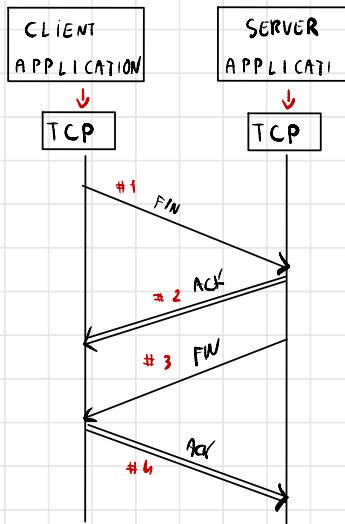
D-DOS (Distributed delay of service)

Il Delay of Service DOS è un attacco informatico che mira a rendere un servizio online inaccessibile ai suoi utenti legittimi, aumentando i tempi di risposta del servizio fino a renderlo inutilizzabile. Un approccio "man in the middle" prevede che si inserisca una macchina in mezzo alle due comunicanti che riceva le richieste senza inoltrarle direttamente al server.



Chiusura di una connessione TCP

La chiusura in TCP è unidirezionale. Se una macchina chiude la propria connessione non può inviare più nulla all'altra ma deve continuare a rispondere ai messaggi con degli ACK, per tanto bisogna che entrambi gli host chiudano la loro connessione TCP. La chiusura di una connessione TCP prevede che uno dei dispositivi invii un pacchetto "FIN" (finish) per indicare che la connessione deve essere chiusa. L'altro dispositivo risponderà con un pacchetto "ACK" per confermare la ricezione del pacchetto FIN. Una volta ricevuto il pacchetto "FIN" di conferma, entrambi i dispositivi completeranno la chiusura della connessione e rilasceranno le risorse associate ad essa. Se un segmento di ACK si perde, l'altra macchina non vede alcuna risposta e chiude il canale di comunicazione.



Congestione

la congestione si verifica quando il numero di pacchetti trasmessi attraverso la rete è superiore alla capacità di trasmissione della rete stessa. In un canale i pacchetti viaggiano sempre alla velocità massima. La congestione è creata dai router e non dal canale. Non si può saturare un canale e non è il canale a causare la congestione. Quando la congestione in rete risulta molto alta, è possibile che si verifichino degli overflow nei router causando perdita di pacchetti. La congestione spesso è dovuta alla presenza in rete di copie di copie di pacchetti ritrasmessi perché il collegamento ha generato del ritardo sui router o perché andati persi. Sostanzialmente si arriva ad un situazione in cui, la maggior parte dei pacchetti in rete sono copie che non hanno alcuna utilità ed intasano la rete.

La finestra di congestione TCP si riferisce alla quantità di dati che un mittente può inviare prima di ricevere un acknowledgement (ACK) dal destinatario.

Fase AIMD

La fase AIMD (Additive Increase Multiplicative Decrease) è un algoritmo di controllo della congestione utilizzato dai protocolli di trasporto come TCP (Transmission Control Protocol) per regolare la velocità di invio dei dati in una rete. Durante la fase AIMD, il protocollo TCP aumenta gradualmente la sua finestra di congestione in modo additivo, ovvero incrementando la quantità di dati inviati di un certo valore a ogni round trip time (RTT), fino a quando non si verifica una perdita di pacchetti o un timeout. A questo punto, la finestra di congestione viene ridotta drasticamente in modo moltiplicativo per alleviare la congestione nella rete. Questo processo viene ripetuto continuamente durante la trasmissione dei dati.

Slow start TCP

Lo Slow Start, d'altra parte, è utilizzato all'avvio della connessione TCP o quando la connessione subisce una riinizializzazione. In questa fase, il protocollo aumenta esponenzialmente il tasso di trasmissione di dati finché non viene rilevata una perdita di pacchetti, a quel punto il protocollo passa alla fase AIMD per regolare il flusso di dati.

La fase AIMD e lo Slow Start sono due algoritmi di controllo della congestione utilizzati in TCP per regolare il flusso di dati sulla rete. La fase AIMD regola il flusso di dati quando la rete è congestionata, mentre lo Slow Start viene utilizzato all'inizio della connessione o in caso di riinizializzazione.

Politica TCP Tahoe (Slow start , AIMD, fast retrasmitt)

Tahoe prevede che ogni qual volta si verifichi un evento perdita (o evento di congestione) di qualsiasi tipo, la finestra di congestione venga dimezzata e questo nuovo valore viene memorizzato nella soglia SSTresh. Fatto questo la trasmissione dei dati ricomincia impostando il valore iniziale della finestra di congestione corrente pari ad 1MSS. Si riinizializza la trasmissione con Slow Start, ovvero la crescita della finestra di congestione avviene esponenzialmente fino a raggiungere il valore di soglia prima determinato. Oltre questo valore la crescita avviene linearmente nel tempo secondo la modalità AIMD, fino a quando non si verifica nuovamente un evento perdita e si riparte dalla fase Slow Start.

È importante sottolineare come la riduzione della finestra di congestione ad 1 MSS comporti una repentina riduzione della velocità di trasmissione dei dati nella connessione TCP. Questo effetto, da un lato, de-congestiona la rete, ma, dall'altro, limita temporaneamente (ma fortemente) la velocità di trasmissione/ricezione dei dati. Pertanto, la crescita esponenziale fino al livello di soglia consente alla connessione TCP di recuperare prontamente (ma parzialmente) una parte della banda ormai persa in seguito all'evento di congestione. Una volta raggiunto il livello di soglia, la crescita avviene lentamente per sondare il livello di banda effettivamente disponibile in rete e cercare di raggiungere nuovamente il livello di congestione il più lentamente possibile.

Politica TCP Reno (Slow start , AIMD, fast retrasmitt, fast recovery)

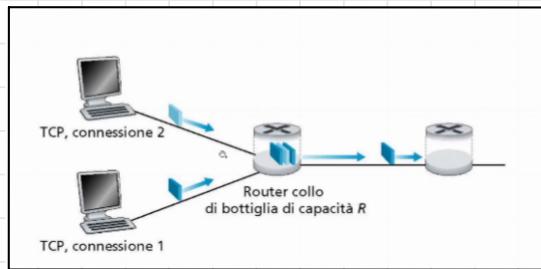
TCP Reno reagisce in maniera diversa ai due eventi che segnalano la possibile perdita di un pacchetto dati. L'evento timeout scaduto viene considerato un indicatore "forte" di congestione, e pertanto, TCP Reno, così come TCP Tahoe, fa ripartire la fase di Slow Start da un valore della finestra di congestione pari ad 1 MSS, dimezza la SSTresh e continua in SlowStart. L'evento tre ACK duplicati, invece, viene considerato un indicatore "debole" di congestione, perché sintomatico del fatto che comunque la rete ha consegnato ulteriori pacchetti, dopo un pacchetto probabilmente perso. Pertanto, in TCP Reno questa circostanza produce la ritrasmissione veloce (Fast Retransmit) del segmento che contiene i dati già trasmessi per i quali non è ancora arrivato il riscontro, senza attendere lo scadere del relativo timeout, si riparte con la fase AIMD (incremento lineare) senza riportare a 1MSS la finestra di congestione (Fast Recovery).

Le due politiche iniziano l'invio dei pacchetti sfruttando lo slow start (aumento esponenziale) e proseguono con la fase AIMD (aumento lineare) se si rileva una perdita tramite triplo ACK duplicato. Se si utilizzasse prima AIMD (incremento lineare del numero di pacchetti trasmessi) e dopo Slow Start (incremento esponenziale) la Fairness non sarebbe garantita (la curva divergerebbe).

Proprietà Fairness

Molteplici flussi di dati possono competere per la larghezza di banda disponibile, e senza una regolamentazione adeguata, un flusso di dati più aggressivo potrebbe saturare la rete impedendo ad altri flussi di ricevere una quota equa di larghezza di banda. Il concetto di **fairness** si riferisce alla capacità di gestire equamente le richieste di larghezza di banda da parte di più flussi di dati, in modo che nessun flusso abbia un vantaggio ingiusto rispetto agli altri. Ciò significa che TCP deve essere in grado di allocare equamente la banda disponibile tra tutti i flussi di dati che condividono una connessione di rete.

Fairness

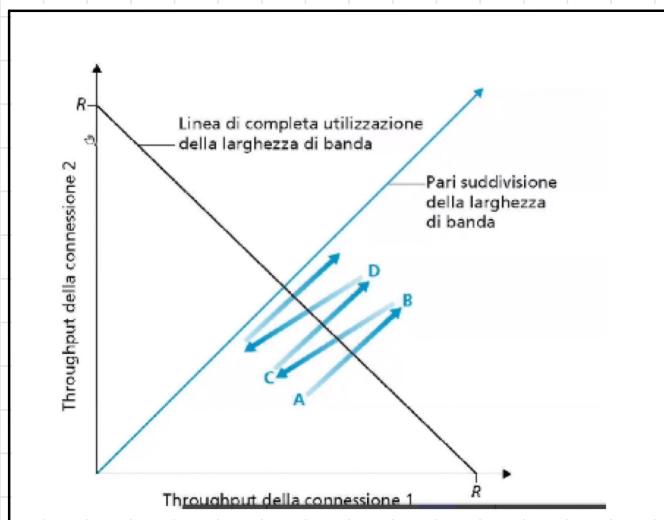


Abbiamo due connessioni TCP, un router centrale che fa da collo di bottiglia e ha una certa capacità R .

L'obiettivo finale è che la banda tra queste due connessioni venga equamente divisa in questo link.

Le due connessioni non sanno che l'altra esiste. Il router non può fare niente: il suo unico compito è di inoltrare i pacchetti fuori dal suo buffer con la sua politica FIFO e scartare i pacchetti che arrivano quando il buffer è pieno.

Dimostriamo che la proprietà della fairness esiste con una “dimostrazione grafica”:



Andiamo a diagrammare il throughput delle due connessioni. Abbiamo un punto con delle coordinate (x,y) che rappresenta il throughput delle due connessioni.

Dato che R rappresenta la massima larghezza di banda disponibile in uscita, individuiamo altri due punti: $R1 = (R,0)$ e $R2 = (0,R)$, che indicano rispettivamente che la connessione 1 o la connessione 2 hanno preso tutta la banda disponibile. Infine, il punto di coordinate $(R/2, R/2)$ indica l'equa suddivisione della banda (al centro del grafico).

Tutta la bisettrice infatti è fatta da punti che hanno le coordinate (x,y) uguali, cioè i throughput sono uguali.

La fairness vuol dire che il throughput sta sulla bisettrice, e l'ideale è stare fermi sul punto a $R/2$.

Il segmento disegnato tra i punti $(0,R)$ e $(R,0)$ ha tutti i punti di coordinate (x,y) tali che $x+y = R$, perché tracciando una retta che passa per i punti $R1$ e $R2$ avrà coordinate:

$$y = -x + R \Rightarrow y + x = R$$

Quindi tutti i punti su questo segmento rispetteranno l'equazione $y = -x + R$. -1 è quindi il coefficiente angolare di questa retta (che estende il segmento) e una retta con coefficiente -1 è perpendicolare alla bisettrice (che ha coefficiente 1 e equazione $y = x$ ovviamente, senza termine noto perché passa per l'origine).

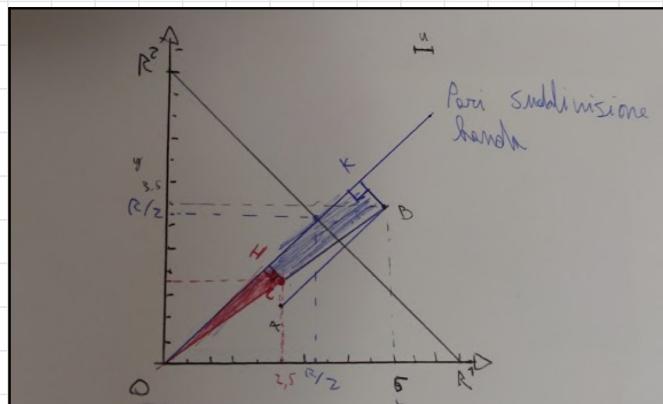
Quindi è importante dire che il segmento tra i punti $R1$ e $R2$ è un segmento e non una curva o un'iperbole.

Tutti i punti che sono nel triangolo rettangolo di vertici $OR1R2$ sono punti tali che la somma delle coordinate è minore o uguale a R . Quindi il segmento indica la Linea di completa utilizzazione possibile della banda come dice il grafico. Va da sè dire che all'origine le connessioni sono ferme.

Dato che la somma è al di sotto di R il collegamento non è sfruttato al massimo.

Possiamo avere un throughput maggiore di R quindi? Teoricamente no, ma... invece sì. Come?

Consideriamo il fatto che ci sono dei buffer di ingresso che possono accumulare pacchetti, allora può capitare che ci sia un istante in cui il throughput è maggiore di R , semplicemente per il fatto che i pacchetti si stanno accumulando.



Quindi può capitare di arrivare nella zona oltre il segmento $R1R2$. Però arrivare in questa zona del grafico, e quindi di avere un punto in questa zona oltre il max throughput, vuol dire che i pacchetti si stanno accumulando e quindi dato che si accumulano le code si riempiono e prima o poi si arriva alla perdita di qualche pacchetto per timeout o triplo ACK duplicato.

Livello di rete

Il livello di rete riguarda tutti i dispositivi intermedi presenti nella comunicazione **end-to-end**. Finora abbiamo parlato di livello applicativo e livello di trasporto, che erano incentrati esclusivamente nella macchina mittente e nella macchina destinazione, il livello di rete è il primo che si occupa invece della comunicazione verso tutti i dispositivi intermedi presenti nella rete.

Forwarding e Routing

Il forwarding è il processo di invio dei pacchetti di dati da un nodo della rete a un altro. In pratica, il forwarding viene effettuato dai dispositivi di rete come switch e router, che sono in grado di analizzare l'intestazione dei pacchetti e inoltrarli al nodo successivo nella catena di nodi intermedi. Il forwarding è un'operazione di tipo locale, in quanto viene eseguito solo tra i dispositivi adiacenti e non richiede una conoscenza completa della topologia della rete.

Il routing, al contrario, è il processo di determinazione del percorso migliore per i pacchetti attraverso la rete. Questo processo coinvolge l'analisi della topologia della rete, la valutazione di diverse metriche (ad esempio, la larghezza di banda disponibile, la congestione della rete e la qualità del servizio richiesta) e l'utilizzo di algoritmi di routing per determinare il percorso ottimale per i pacchetti. Il routing è un'operazione di tipo globale, in quanto richiede una conoscenza completa della topologia della rete e delle capacità di instradamento dei singoli dispositivi.

Data plane (Forwarding)

Il data plane viene eseguito dai dispositivi di rete come switch e router, che analizzano l'intestazione dei pacchetti e le informazioni contenute nella tabella di routing, ed inoltre i pacchetti attraverso la rete. Quando un pacchetto arriva in un dispositivo di rete, **il data plane utilizza le informazioni nella tabella di routing per determinare il percorso migliore per inoltrare il pacchetto al nodo successivo nella catena di nodi intermedi**. Il data plane è quindi responsabile dell'elaborazione e dell'inoltro dei pacchetti di dati attraverso la rete utilizzando il percorso più efficiente possibile.

Control plane (Routing)

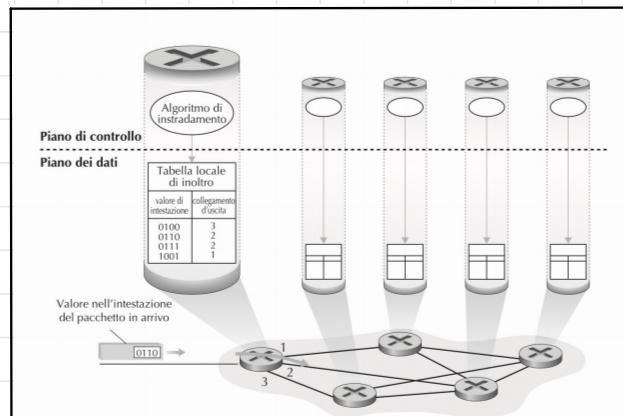
Il control plane, invece, utilizza le regole di funzionamento generale della rete per determinare le configurazioni dei dispositivi di rete. Queste regole consentono al control plane di vedere la rete in senso globale, e di determinare il modo migliore per instradare i pacchetti attraverso la rete sulla base delle risorse disponibili.

Il control plane utilizza **algoritmi di routing** per analizzare la topologia della rete, le metriche di rete e le informazioni sullo stato della rete per determinare il percorso ottimale per i pacchetti attraverso la rete. In base a queste informazioni, il control plane **determina le configurazioni dei dispositivi e le regole di funzionamento generale della rete**, che vengono semplificate e inserite nella tabella di forwarding utilizzata dal data plane per inoltrare i pacchetti attraverso la rete. Il control plane utilizza informazioni come la disponibilità delle risorse (ad esempio, preferendo una scheda cablata rispetto a una wireless) per determinare l'uscita migliore per i pacchetti.

Control plane : approccio distribuito tradizionale

In questo approccio, l'algoritmo di routing è implementato in ogni router, che quindi svolge sia la funzione di inoltro che quella di instradamento internamente (non vi è una netta separazione tra piano dati e di controllo). In pratica, ogni dispositivo di rete (router, switch, etc.) ha il compito di prendere autonomamente le decisioni di instradamento del traffico sulla base dell'informazione di routing che ha a disposizione.

L'approccio a controllo distribuito prevede l'uso di protocolli di routing che consentono ai dispositivi di rete di scambiarsi informazioni sulle rotte disponibili. In questo modo, **ogni dispositivo di rete può determinare autonomamente ed in modo decentralizzato il percorso migliore per instradare il traffico verso la destinazione desiderata.**

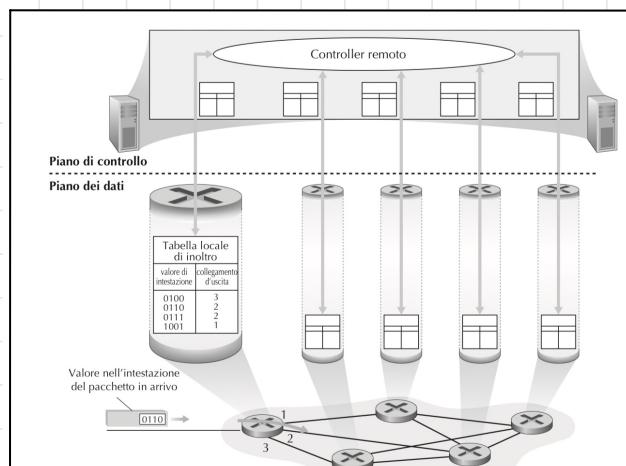


Control plane : approccio SDN(software defined network)

Questo approccio prevede una separazione tra il controllo della rete (control plane) e il trasferimento dei dati (data plane). SDN adotta un approccio a controllo centralizzato per il control plane, che consente di gestire la rete in modo più flessibile e dinamico rispetto ai tradizionali approcci a controllo distribuito.

Nell'approccio SDN, il control plane è centralizzato in un'unità software chiamata **controller**. Il controller gestisce la topologia della rete, raccogliendo informazioni dai dispositivi di rete (switch, router, etc.) attraverso un protocollo di comunicazione e decidendo come instradare il traffico sulla base delle politiche definite dall'amministratore.

Il data plane, invece, è costituito dai dispositivi di rete (switch, router, etc.) che instradano il traffico sulla base delle istruzioni ricevute dal controller. I dispositivi di rete non devono avere funzionalità avanzate di routing, poiché la maggior parte delle decisioni di routing sono prese dal controller.



Servizio datagram e a circuito virtuale

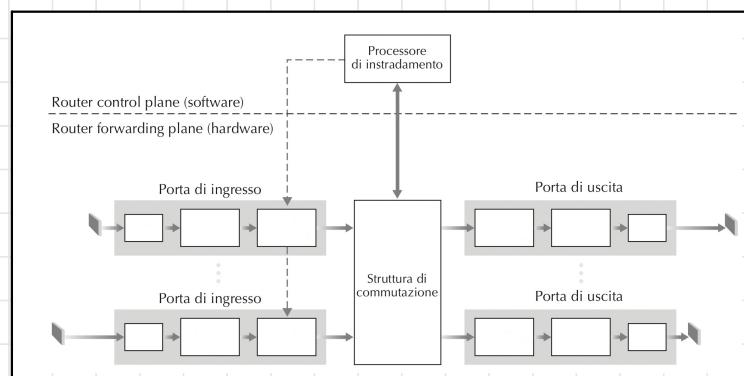
Il servizio datagram e il servizio a circuito virtuale rappresentano due approcci differenti per la gestione della comunicazione dati all'interno di una rete.

Il servizio datagram, utilizzato a livello di rete di Internet, mette a disposizione un servizio noto come **best-effort**, ossia "col massimo impegno possibile". Con questo servizio, non c'è garanzia che i pacchetti vengano ricevuti nell'ordine in cui sono stati inviati, così come non è garantita la loro eventuale consegna. Non c'è garanzia sul ritardo end-to-end, così come non c'è garanzia su una larghezza di banda minima. In questo caso, i pacchetti dati vengono inviati attraverso la rete **senza un percorso prestabilito**, e ogni nodo di rete prende le proprie decisioni di routing in base alle informazioni contenute nella tabella di instradamento. Tuttavia, questo approccio è facile da implementare e funziona bene se la probabilità di insuccesso nella trasmissione è bassa, cosa che viene garantita dall'hardware.

Il servizio a circuito virtuale è un approccio che prevede l'individuazione esatta delle macchine da attraversare per la comunicazione dati, e la riservazione di una porzione di banda per la durata della comunicazione stessa. In questo caso, un percorso prestabilito viene creato prima dell'inizio della comunicazione, e la banda viene divisa tra i vari utenti che utilizzano il circuito virtuale. Ciò **garantisce un livello di affidabilità e prestazioni elevate**, oltre alla possibilità di controllare la congestione della rete. Tuttavia, questo approccio richiede una maggiore complessità operativa e di gestione della rete.

Router

Un router è un dispositivo di rete che consente di instradare i pacchetti di dati tra diverse reti, utilizzando una tabella di routing per determinare il percorso migliore. Esso è fondamentale per il funzionamento di internet e delle reti aziendali, e può offrire anche funzionalità di sicurezza.



• porte di ingresso

Le porte di ingresso di un router rappresentano una parte fondamentale del dispositivo, in quanto consentono di ricevere i pacchetti di dati provenienti dalle diverse reti e instradarli verso la destinazione desiderata.

Nelle porte di input di un router, si ha un componente che lavora a livello data-link e che si occupa di ricevere i pacchetti di dati provenienti dalle diverse interfacce di rete. Successivamente, i dati ricevuti vengono bufferizzati per consentire una gestione efficiente del traffico, in quanto non è detto che i pacchetti vengano inoltrati con la stessa velocità con cui vengono ricevuti. Per instradare i pacchetti di dati verso la destinazione desiderata, i router utilizzano delle tabelle di routing, che suddividono i gruppi di pacchetti in base alla destinazione e assegnano loro le uscite predefinite.

Per assegnare un'uscita a un pacchetto in ingresso, si cerca una voce della tabella di inoltro con il prefisso di indirizzo più lungo che matcha all'indirizzo di destinazione.

Le porte di ingresso di un router quindi sono fondamentali per il funzionamento del dispositivo, poiché consentono di ricevere i pacchetti di dati dalle diverse interfacce di rete, bufferizzarli e instradarli verso la destinazione desiderata utilizzando le tabelle di routing.

Memorie associative CAM e TCAM

Le memorie associative CAM e TCAM (Content-Addressable Memory e Ternary Content-Addressable Memory) sono importanti componenti utilizzati nei router per la ricerca di informazioni all'interno delle tabelle di routing, in quanto consentono di effettuare ricerche molto rapide all'interno delle tabelle di routing, poiché sono in grado di confrontare l'intero indirizzo IP di un pacchetto di dati con gli indirizzi presenti nella tabella di routing in modo simultaneo.

Le CAM sono utilizzate per la ricerca di informazioni esatte all'interno della tabella di routing, mentre le TCAM consentono di effettuare ricerche parziali utilizzando maschere di bit, permettendo di individuare i percorsi di routing per più destinazioni contemporaneamente. Le memorie associative rendono possibile l'elaborazione di un alto volume di pacchetti di dati in tempo costante.

• struttura di commutazione

Dopo l'arrivo dei pacchetti nella parte di input del router, segue un meccanismo che sposta i pacchetti nella coda di uscita corretta, precedentemente identificata. Questo meccanismo di commutazione può utilizzare diverse architetture, tra cui la memoria condivisa, il bus condiviso e la rete di interconnessione.

Nella architettura della **memoria condivisa**, i pacchetti di dati che arrivano in ingresso al router vengono salvati in memoria e successivamente letti dalla memoria per essere instradati verso la porta di uscita corretta. Tuttavia, questa architettura può essere molto lenta in quanto richiede molti accessi alla memoria, ma è facile da implementare.

Nella architettura del **bus condiviso**, i pacchetti di dati vengono bufferizzati nelle code di ingresso e uscita, eliminando la necessità di accedere alla memoria. In questo modo, il router può gestire un traffico di dati più elevato rispetto alla memoria condivisa. Tuttavia, questa architettura richiede un bus più robusto per garantire la corretta gestione del traffico.

Nella architettura della **rete di interconnessione**, i pacchetti di dati possono essere instradati verso diverse porte di uscita selezionate per gestire l'inoltro dei pacchetti. In questo modo, si possono selezionare diverse porte di ingresso e uscita per gestire l'inoltro dei pacchetti. Tuttavia, questa architettura può essere più complicata da implementare e richiedere costi maggiori. Inoltre, non è possibile instradare pacchetti di diverse porte di input sulla stessa uscita.

Organizzando molti più piani di commutazione parallelamente, è possibile organizzare le connessioni su più porte in ingresso e instradarle con una velocità molto più elevata, ma a fronte di un maggiore costo. La scelta dell'architettura dipende dalle specifiche esigenze di prestazioni, costi e scalabilità del router.

- **Porte di uscita** : memorizzano i pacchetti che provengono dalla struttura di commutazione e li trasmettono sul collegamento in uscita, operando le funzionalità necessarie del livello di collegamento e fisico.
- **Processore di instradamento** (routing processor) : segue le funzioni del piano di controllo. Nei router tradizionali, esegue i protocolli di instradamento, gestisce le tabelle di inoltro e le informazioni sui collegamenti attivi, ed elabora la tabella di inoltro per il router. Nei router SDN, il processore di instradamento è responsabile della comunicazione con il controller remoto, in modo da ricevere le occorrenze della tabella di inoltro e installarle alle porte di ingresso.

Se la memoria del router è eccessiva rispetto al traffico di dati gestito, i pacchetti di dati molto probabilmente non andranno persi, ma potrebbero subire un **aumento dei tempi di attesa e creare congestione**. Inoltre, aumentando la capacità della memoria del router, i pacchetti di dati vengono bufferizzati per un periodo di tempo più lungo.

La formula che spiega quanto bufferizzare in relazione alla capacità di rete è :

$$\text{Buffering} = \frac{\text{RTT} \cdot C}{\sqrt{N}} \rightarrow \text{bit/s}$$

- **RTT** rappresenta il ritardo di andata e ritorno (Round Trip Time), ovvero il tempo necessario per un pacchetto di dati per viaggiare dalla sorgente alla destinazione e ritornare indietro.
- **C** rappresenta la capacità del link di rete, ovvero la quantità massima di dati che il link può trasferire in un'unità di tempo (ad esempio, in bit al secondo).
- **N** rappresenta il numero di flussi di dati presenti nella rete.

La formula indica che il buffering necessario per gestire correttamente i flussi di dati dipende dal prodotto tra il RTT e la capacità del link di rete, diviso per la radice quadrata del numero di flussi presenti nella rete. In altre parole, maggiore è il RTT e la capacità del link, maggiore sarà il buffering necessario per gestire i flussi di dati.

Tuttavia, è importante notare che troppo buffering può causare congestione di rete e aumentare i ritardi, in particolare per le applicazioni in tempo reale. Pertanto, la scelta del valore di buffering dipende dalle specifiche esigenze della rete e del tipo di applicazioni che viaggiano sulla rete stessa.

Smaltimento dei buffer

Dato che la porta di uscita può trasmettere un solo pacchetto in un intervallo prestabilito (tempo di trasmissione del pacchetto), i pacchetti in arrivo dovranno mettersi in coda per la trasmissione sul collegamento in uscita. In assenza di sufficiente memoria per inserire nel buffer il nuovo pacchetto in ingresso, occorrerà stabilire se scartarlo o se rimuoverne uno o più, fra quelli già in coda, per far posto al nuovo arrivato.

Si utilizzano degli algoritmi per prendere la scelta sul pacchetto da scartare :

FCFS (First-Come, First-Served) è un altro algoritmo utilizzato per la gestione dei pacchetti in una rete. Questo algoritmo gestisce i pacchetti in ordine di arrivo, ovvero il primo pacchetto che arriva viene trasmesso per primo. L'algoritmo FCFS è molto semplice ed efficiente, poiché non richiede alcun tipo di ordinamento dei pacchetti. Tuttavia, questa politica di gestione dei pacchetti può portare a problemi di congestione di rete in situazioni di traffico intenso.

RED (Random Early Detection) prevede la gestione delle code di I/O sui router con un modello FIFO. Quando il buffer di una coda raggiunge il 75% della sua capienza, alcuni pacchetti vengono eliminati in modo "casuale" per prevenire la saturazione del buffer e i pacchetti vengono quindi trasmessi nello stesso ordine con cui sono arrivati in coda. In questo modo, si evita che TCP saturi i buffer e rallenti l'invio dei pacchetti, utilizzando strategie di controllo di congestione messe in atto da TCP Tahoe o Reno. La politica RED è particolarmente utile per evitare la congestione di rete in situazioni di traffico intenso.

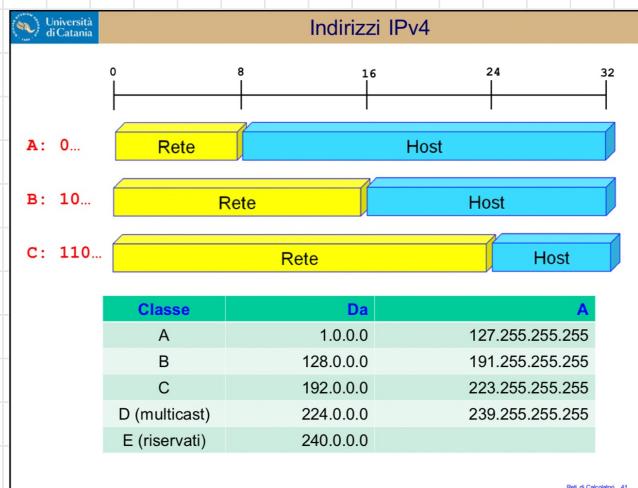
RR (Round Robin), invece, scandisce le code di bufferizzazione e le gestisce tutte in parallelo, inviando un pacchetto per ciascuna coda. Questa politica permette di gestire in modo più efficiente il traffico di dati e prevenire la congestione di rete, garantendo una distribuzione equa del traffico su tutte le code di bufferizzazione. Tuttavia, la politica RR richiede una maggiore complessità nell'implementazione rispetto alla politica RED.

IPv4 (internet protocol versione 4)

Il pacchetto a livello di rete è noto come datagramma. IPv4 è un protocollo di rete utilizzato per l'instradamento dei pacchetti di dati su Internet. Gli indirizzi IPv4 sono composti da 32 bit, divisi in quattro ottetti da 8 bit ciascuno. Questi indirizzi sono suddivisi in classi di indirizzi, che sono gruppi di indirizzi utilizzati nelle tabelle di routing.

• Classi di indirizzi IP

Le classi di indirizzi IPv4 sono A, B, C, D ed E. Ogni classe di indirizzi ha una diversa lunghezza della parte di rete e della parte di host. La parte esplicita dell'indirizzo identifica la rete, mentre la parte con gli asterischi identifica l'host. Nella classe A, il primo ottetto identifica la rete, mentre gli ultimi tre ottetti identificano l'host. Nella classe B, i primi due ottetti identificano la rete, mentre gli ultimi due ottetti identificano l'host. Nella classe C, i primi tre ottetti identificano la rete, mentre l'ultimo ottetto identifica l'host. Le classi di indirizzi D e E, utilizzate per scopi speciali come la multicast e la sperimentazione.



Il formato dei datagrammi IPv4 sono formati da un **header di 20byte** a meno di eventuali campi opzionali . La memoria associativa TCAM sfrutta la lunghezza dell'header per cercare match nelle tabelle di inoltro, informazione riportata nel campo “lunghezza dell'intestazione”.

L'intestazione di un pacchetto IPv4 contiene diverse informazioni che vengono utilizzate dai nodi intermedi per instradare il pacchetto attraverso la rete. L'intestazione contiene i seguenti campi:

- **Versione:** indica la versione del protocollo IPv4 utilizzata ("4").
- **Lunghezza dell'intestazione:** indica la lunghezza dell'intestazione in parole da 32 bit. Questo campo può variare a seconda dei campi opzionali presenti nell'intestazione ma solitamente è di 20 byte a meno di campi opzionali.
- **Tipo di servizio:** fornisce informazioni sul tipo di servizio richiesto per il pacchetto, ad esempio la priorità e la qualità del servizio (QoS) richiesta.
- **Lunghezza totale:** indica la lunghezza totale del pacchetto (intestazione e payload) in byte.
- **Identificativo, flag e offset di frammentazione:** questi campi vengono utilizzati per gestire i pacchetti che sono troppo grandi per essere trasmessi in un'unica volta attraverso la rete e devono essere frammentati in pacchetti più piccoli.
- **Tempo di vita (TTL):** indica il numero massimo di nodi intermedi attraverso i quali il pacchetto può essere instradato prima di essere scartato.
- **Protocollo:** indica il protocollo di livello superiore utilizzato dal pacchetto (ad esempio TCP o UDP).
- **Checksum dell'intestazione:** viene utilizzato per verificare l'integrità dell'intestazione.
- **Indirizzo IP di origine e destinazione:** indicano gli indirizzi IP del mittente e del destinatario del pacchetto.

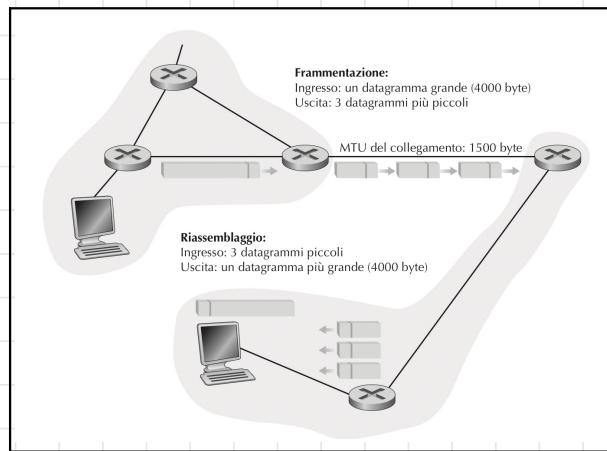
Il carico utile (payload) di un pacchetto IPv4 contiene i dati effettivi che vengono trasmessi dal mittente al destinatario. In generale, i pacchetti IPv4 vengono trasmessi attraverso la rete in modo indipendente gli uni dagli altri e possono seguire percorsi diversi. La loro elaborazione e instradamento sono gestiti dai nodi intermedi della rete, che utilizzano le informazioni contenute nell'intestazione per determinare il percorso migliore per il pacchetto.

Frammentazione

Nella gestione dei pacchetti di dati su Internet, spesso è necessario suddividere un pacchetto grande in unità separate, adattando la loro dimensione all'MTU (Maximum Transmission Unit) del collegamento. Questo processo è noto come frammentazione del pacchetto.

Per effettuare la frammentazione del pacchetto, si suddivide il pacchetto in unità separate, ma aventi lo stesso ID, in modo da identificare che appartengono allo stesso pacchetto originale. Inoltre, si utilizza un campo **offset** per identificare di quanto è sfasata rispetto all'origine la posizione del pacchetto. Tuttavia, se si perde un frammento di un pacchetto, è necessario reinviare l'intero pacchetto. Ogni frammento ha anche un flag **more fragments**, che indica se ci sono altri frammenti dopo o se è l'ultimo frammento del pacchetto. Questo sistema fornisce flessibilità, in quanto un frammento può essere a sua volta frammentato. Alcuni pacchetti, tuttavia, non necessitano di essere frammentati, e l'informazione relativa è contenuta in un campo **don't fragment**.

In IPv4 qualsiasi router di transito può frammentare un pacchetto, ciò non avviene in IPv6.



Indirizzamento

Generalmente un host ha un solo collegamento con la rete e quando l'host vuole inviare un datagramma, lo fa su tale collegamento. Il confine tra host e collegamento fisico viene detto interfaccia. Invece, dato che il compito di un router è ricevere datagrammi da un collegamento e inoltrarli su un altro, questo deve necessariamente essere connesso ad almeno due collegamenti e anche il confine tra un router e i suoi collegamenti è chiamato interfaccia. Il router presenta più interfacce, una su ciascuno dei suoi collegamenti. Dato che host e router sono in grado di inviare e ricevere datagrammi, IP richiede che tutte le interfacce abbiano un proprio indirizzo IP. Pertanto, l'indirizzo IP è tecnicamente associato a un'interfaccia, anziché all'host o al router che la contiene.

Gli indirizzi IP sono lunghi 32 bit (4 byte) e quindi ci sono in totale 2^{32} indirizzi IP, cioè circa 4 miliardi. L'indirizzo 193.32.216.9 in notazione binaria diventa :

11000001 00100000 11011000 00001001

Ogni interfaccia di host e router di Internet ha un indirizzo IP globalmente univoco (eccetto quelle gestite da NAT)

Indirizzi Gerarchici (geografici)

Gli indirizzi IPv4 sono **composti da 32 bit** e sono assegnati in modo gerarchico, fornendo informazioni sulla posizione geografica del dispositivo. Le cifre dell'indirizzo seguono una gerarchia, che identifica la rete e l'host. Questo tipo di indirizzamento è utile per instradare i pacchetti di dati su Internet, in quanto fornisce informazioni sulla strada da seguire per arrivare alla destinazione.

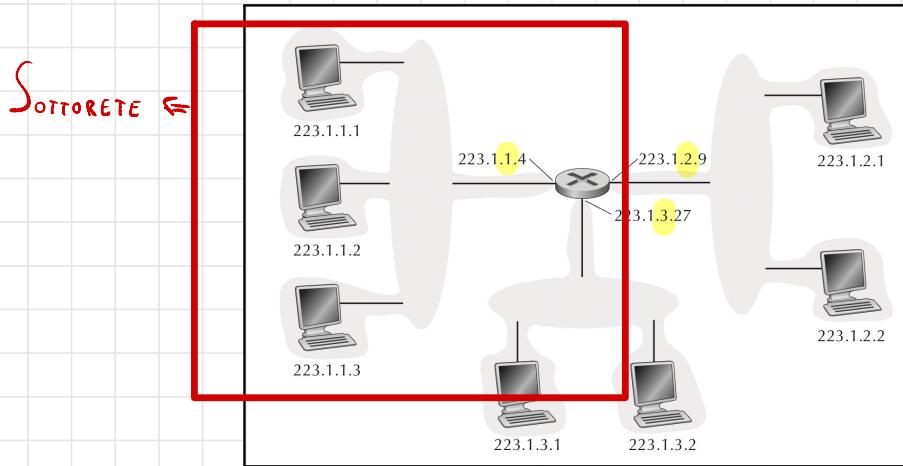
Indirizzi Flat

Sono indirizzi che non forniscono informazioni sulla posizione della destinazione. Questo tipo di indirizzamento è utile in contesti in cui non è necessario conoscere la posizione geografica del dispositivo, come ad esempio in reti locali (MAC).

Classful addressing e CIDR

La strategia di assegnazione degli indirizzi Internet è detta classless interdomain routing **CIDR**. Prima dell'adozione di CIDR, le parti di rete di un indirizzo IP dovevano essere lunghe 8, 16 o 24 bit. Tale schema di indirizzamento era noto come **classful addressing** e le sottoreti con indirizzi di sottorete da 8, 16 e 24 bit erano note rispettivamente come reti di classe A, B e C. Il requisito che la parte di sottorete di un indirizzo IP fosse lungo esattamente 1, 2 o 3 byte si rivelò problematico. Una sottorete di classe C (/24) poteva ospitare solo fino a $2^8 - 2 = 254$ host (due dei 256 indirizzi sono riservati per usi speciali): troppo pochi per molte organizzazioni.

Se specifico un numero come 25 verranno utilizzati i primi 25 bit dell'indirizzo IP per indicare la subnet mask, mentre i rimanenti 7 bit saranno usati per indicare gli host disponibili (da 0 a 127). Questo schema è un metodo per suddividere gli indirizzi IP allo scopo di allocarli con il minimo spreco possibile, infatti si possono specificare numeri di bit variabili e non interi blocchi da 8 bit. Si ottiene quindi un'ottimizzazione dello spazio di indirizzamento.



Per IP, la rete che interconnecta le tre interfacce di host e l'interfaccia di un router forma una sottorete. IP assegna a questa sottorete l'indirizzo 223.1.1.0/24, dove la notazione /24, detta anche maschera di sottorete (subnet mask), indica che i 24 bit più a sinistra dell'indirizzo definiscono l'indirizzo della sottorete. Di conseguenza, la sottorete **223.1.1.0/24** consiste di tre interfacce di host (223.1.1.1, 223.1.1.2, 223.1.1.3) e di un'interfaccia di router (223.1.1.4). Ogni altro host connesso alla sottorete 223.1.1.0/24 deve avere un indirizzo della forma 223.1.1.xxx.

Subnetting

Una subnet mask rappresenta un'organizzazione fisica delle classi di indirizzi all'interno della stessa LAN, che consente di suddividere la rete in sotto-reti. Un indirizzo IPv4 è un campo a 32 bit, che può essere suddiviso in due parti: la parte di rete e la parte di host. La subnet mask identifica quanti bit, di un indirizzo IP, vengono utilizzati per **identificare la rete** e sono detti bit fissi, in quanto sono uguali per tutti gli host all'interno della stessa rete. Tutti gli altri bit sono detti variabili e servono ad identificare l'host.

La scelta dell'utilizzo di una subnet mask dipende dalle specifiche esigenze della rete e dei dispositivi connessi. La suddivisione della rete in sotto-reti consente di ottimizzare il routing dei pacchetti di dati su Internet, migliorando l'efficienza e la sicurezza della rete.

La subnet mask viene indicata dopo l'indirizzo IP attraverso /N, dove N è il numero di bit che identificano la rete di appartenenza :

- 192.168.1.7 / 24 - solo i primi 24 bit identificano la rete di appartenenza
 - i rimanenti 8 indicano l'host

infatti la traduzione binaria della maschera è la seguente :

01100000 . 01011010 . 00000001 . 00000111	IP
11111111 . 11111111 . 11111111 . 00000000	subnet mask host (da 0 a 2^8)

Indirizzi speciali

Alcuni indirizzi speciali non possono essere utilizzati per configurare host all'interno di una rete :

L'indirizzo **0.0.0.0** viene utilizzato come **indirizzo di default** o **indirizzo di "tutti gli indirizzi"** in alcune situazioni. Ad esempio, un dispositivo potrebbe utilizzare l'indirizzo 0.0.0.0 come indirizzo IP sorgente quando invia un pacchetto su una rete se non ha ancora un indirizzo IP assegnato.

L'indirizzo **255.255.255.255**, noto anche come **indirizzo di broadcast limitato**, viene utilizzato per inviare pacchetti di broadcast a tutti i dispositivi nella stessa rete locale. Quando un dispositivo invia un pacchetto a questo indirizzo, il pacchetto viene trasmesso a tutti i dispositivi nella rete locale.

In alcuni casi particolari, come quando sono presenti solo due router agli estremi di un collegamento senza alcuna macchina intermedia, è possibile utilizzare una subnet mask con **prefisso /31**. In questo caso, non è necessario considerare gli indirizzi di broadcast e l'intera rete, poiché tutti i bit sono utilizzati per identificare gli host.

Configurazione delle reti

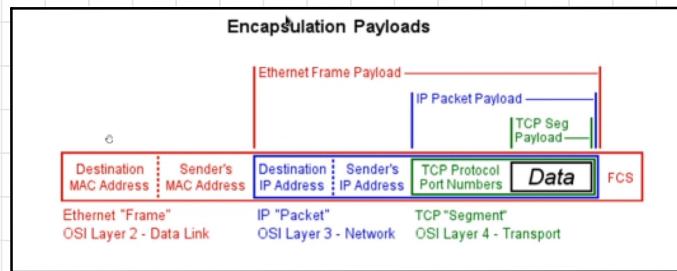
Se viene cambiato l'IP ad una macchina questa non farà più parte della rete. Se vengono numerate diversamente un numero di macchine esse possono comunicare tra loro ma non con la rete. Si possono mettere due indirizzi IP alla stessa scheda di rete con lo scopo di far comunicare delle macchine isolate con la rete .

NB un indirizzo IP è associato ad una sola scheda di rete ma ad una scheda di rete posso associare più indirizzi IP.

Incapsulamento

La comunicazione tra macchine diverse avviene tramite l'incapsulamento. Il processo di encapsulamento inizia dal livello applicativo, dove il **messaggio di livello applicativo** viene suddiviso in **segmenti di livello di trasporto**, ad esempio utilizzando il protocollo TCP o UDP. Il segmento di livello di trasporto contiene le informazioni necessarie per instradare il pacchetto attraverso la rete, come ad esempio i numeri di porta di origine e di destinazione. Successivamente, il segmento di livello di trasporto viene incapsulato in un **datagramma a livello di rete**, che contiene l'indirizzo IP di origine e di destinazione, necessario per instradare il pacchetto attraverso la rete. Il datagramma a livello di rete viene quindi incapsulato in un **frame a livello di collegamento dati**, ad esempio utilizzando il protocollo Ethernet. Il frame a livello di collegamento dati contiene gli indirizzi MAC di origine e di destinazione, necessari per instradare il pacchetto attraverso la rete locale. Infine, il frame a livello di collegamento dati viene tradotto in **bit a livello fisico**, che vengono trasmessi attraverso il mezzo di trasmissione fisico, come ad esempio un cavo Ethernet.

Il processo di decapsulamento avviene in modo inverso quando il pacchetto raggiunge la destinazione finale. Il livello fisico riceve i bit e li passa al livello di collegamento dati, che rimuove l'intestazione del frame a livello di collegamento dati. Successivamente, il livello di rete rimuove l'intestazione del datagramma a livello di rete e infine il livello applicativo riceve il segmento di livello di trasporto.



Ethernet

Ethernet è un protocollo di livello DLL (livello 2 con riferimento alla pila ISO/OSI) che si occupa di trasmissione di dati su reti locali LAN cablate attraverso i cavi Ethernet. Grazie a Ethernet, i dispositivi di rete possono comunicare tra loro in modo **efficiente e affidabile**, all'interno di una rete locale (LAN). Un frame Ethernet è il pacchetto di dati che viene trasmesso attraverso la rete Ethernet.

MAC address (media access control)

Nella rete LAN assume una certa rilevanza l'indirizzo MAC dei singoli dispositivi, oltre a quello IP, in quanto il pacchetto deve comunque passare dal livello DLL per essere inviato ad un'altra macchina.

Un MAC address è un indirizzo univoco ed è costituito da **6 byte di dati** espressi in formato esadecimale.

00 : 1A : 2B : 3C : 4D : 5E

Il MAC viene assegnato a una scheda di rete dal produttore e viene utilizzato per identificare in modo univoco i dispositivi all'interno di una rete. MAC è un **indirizzo flat** poiché non ha una struttura gerarchica e non è possibile dedurre l'ubicazione geografica di un dispositivo di rete solo dal suo MAC address.

Poiché l'indirizzo MAC è un identificatore flat, viene utilizzato principalmente per identificare i dispositivi all'interno di una rete locale (LAN), dove viene utilizzato dal protocollo Ethernet per instradare i pacchetti attraverso la rete locale.

ARP (address resolution protocol)

ARP è un protocollo utilizzato per mappare un indirizzo IP a un indirizzo MAC sulla stessa subnet, quindi funziona solo nella LAN locale. Questo processo è necessario perché i pacchetti vengono trasmessi utilizzando gli indirizzi IP, ma la comunicazione effettiva avviene attraverso gli indirizzi MAC.

Quando un dispositivo su una rete locale deve inviare un pacchetto IP a un altro dispositivo sulla stessa subnet, utilizza ARP per ottenere l'indirizzo MAC del dispositivo di destinazione. Il dispositivo mittente invia una **ARP request** broadcast che arriva a tutte le macchine in rete contenente l'indirizzo IP del dispositivo di destinazione (nota che invece, a livello IP, il messaggio è diretto ad una precisa macchina). Tutti i dispositivi sulla stessa subnet ricevono la richiesta e il dispositivo di destinazione risponde con un pacchetto **ARP reply**, contenente il proprio indirizzo MAC.

La risposta ARP viene quindi memorizzata nella cache ARP del dispositivo mittente, che associa l'indirizzo IP del dispositivo di destinazione con il corrispondente indirizzo MAC. In questo modo, il dispositivo mittente può utilizzare l'indirizzo MAC per inviare il pacchetto IP al dispositivo di destinazione. La **cache ARP** viene utilizzata per memorizzare le corrispondenze tra gli indirizzi IP e MAC e viene aggiornata periodicamente o in risposta a nuove richieste ARP. Tuttavia, la cache ARP può essere soggetta ad attacchi di spoofing, in cui un attaccante invia pacchetti ARP falsificati per inserire informazioni false nella cache ARP di un dispositivo.

- **Vulnerabilità ARP**

Una delle principali vulnerabilità di ARP è che **non richiede autenticazione**, il che significa che un dispositivo a cui non è realmente destinato un frame ARP può riceverlo. Questo può portare a diverse forme di attacchi, come l'ARP spoofing, in cui un attaccante invia deliberatamente un frame ARP modificato per mappare l'indirizzo MAC del suo dispositivo all'indirizzo IP di un altro dispositivo, al fine di **dirottare il traffico IP** destinato a quel dispositivo.

Inoltre, ARP è **stateless**, ovvero non tiene traccia delle richieste e risposte degli utenti. Ciò significa che una ARP response può essere ricevuta senza l'invio di una precedente ARP request, il che può essere sfruttato per attacchi in cui un attaccante invia una ARP response fraudolenta per aggiornare la cache ARP di un host con un indirizzo MAC falso.

I controlli sulla correttezza dell'indirizzo MAC del destinatario vengono fatti ad un livello più alto secondo protocolli che richiedono autenticazione.

RARP (Reverse Address Resolution Protocol)

Questo è un protocollo utilizzato per mappare un indirizzo MAC a un indirizzo IP sulla stessa subnet. A differenza di ARP, che mappa un indirizzo IP a un indirizzo MAC, RARP utilizza il processo inverso.

RARP è stato sviluppato per risolvere il problema di avvio dei computer diskless (senza disco), che non hanno un indirizzo IP assegnato in modo permanente. In questo caso, il computer avvia il proprio sistema operativo tramite il network, ma non ha un indirizzo IP assegnato, quindi non può comunicare con altri computer sulla rete.

Per risolvere questo problema, il computer diskless invia un pacchetto RARP broadcast contenente il proprio indirizzo MAC e richiedendo l'assegnazione di un indirizzo IP valido. Il server RARP sulla rete riceve la richiesta e invia un pacchetto RARP response contenente l'indirizzo IP richiesto.

Tuttavia, RARP ha alcune vulnerabilità di sicurezza simili ad ARP, come l'assenza di autenticazione e la possibilità di attacchi di spoofing.

BOOTP (bootstrap protocol)

Il BOOTP è un protocollo di rete utilizzato per consentire a un computer di avviarsi e di ricevere un indirizzo IP, in particolare viene utilizzato durante la fase di boot del computer, quando il sistema deve acquisire un indirizzo IP valido per poter comunicare sulla rete.

Il client BOOTP invia un messaggio di richiesta di boot (BOOTREQUEST) contenente il proprio indirizzo MAC al server BOOTP, che risponde con un messaggio di offerta di boot (BOOTREPLY) contenente l'indirizzo IP da assegnare al client.

Requisiti di configurazione

Per essere configurata in una rete, una macchina ha bisogno di diverse informazioni e impostazioni, tra cui:

1. Indirizzo IP: è l'indirizzo numerico univoco assegnato alla macchina per identificarla sulla rete. L'indirizzo IP può essere assegnato manualmente o automaticamente da un server DHCP.
2. Subnet mask: è un valore numerico che permette di definire la suddivisione logica della rete in sottoreti. La subnet mask definisce quali bit dell'indirizzo IP identificano la parte di rete e quali bit identificano la parte di host.
3. Gateway predefinito: è l'indirizzo IP del router che consente alla macchina di comunicare con le reti esterne alla sua sottoretina.
4. DNS: è l'indirizzo IP del server DNS (Domain Name System) che consente alla macchina di risolvere i nomi dei domini in indirizzi IP.
5. Nome della macchina: è il nome utilizzato per identificare la macchina sulla rete, che può essere utilizzato per la risoluzione dei nomi di dominio.
6. Indirizzo MAC: è l'indirizzo univoco assegnato dalla scheda di rete del dispositivo, utilizzato a livello di collegamento dati per identificare la macchina all'interno della rete locale.
7. Protocolli di rete: la macchina deve essere configurata in modo da supportare i protocolli di rete utilizzati nella rete, come ad esempio il protocollo Ethernet.

Comunicazione su reti differenti

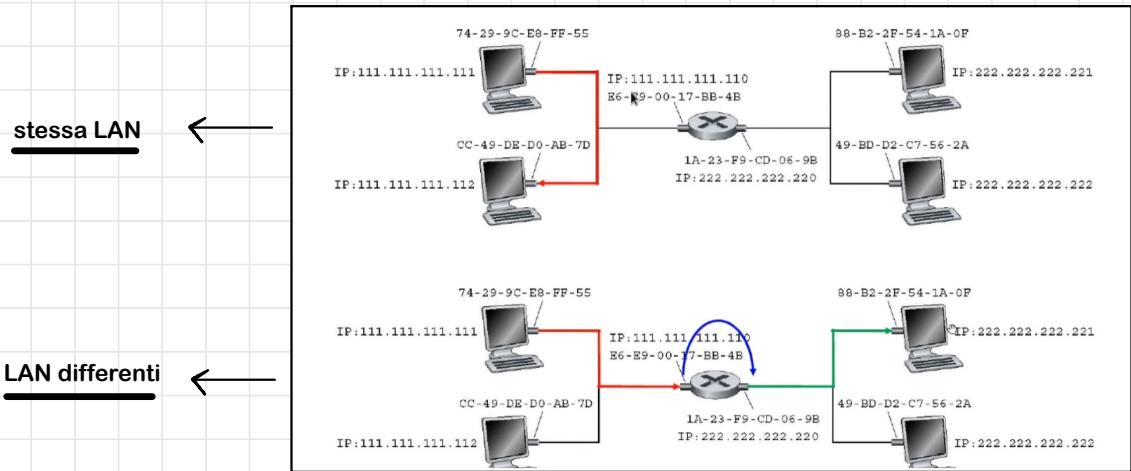
Se le macchine si trovano nella stessa rete LAN possono comunicare. Nel caso in cui le macchine si trovano in reti differenti la situazione si complica: quando un pacchetto viene trasmesso attraverso una rete, ogni dispositivo di rete che inoltra il pacchetto sostituisce l'indirizzo MAC del mittente con il proprio indirizzo MAC come sorgente del pacchetto, e l'indirizzo MAC della destinazione con l'indirizzo MAC del prossimo dispositivo di rete che inoltrerà il pacchetto. In questo modo, il pacchetto può essere correttamente instradato attraverso la rete.

Tuttavia, è importante sottolineare che l'indirizzo MAC non cambia casualmente, ma viene sostituito da una macchina intermedia solo durante il passaggio del pacchetto attraverso quella macchina. L'indirizzo MAC originale del mittente e della destinazione viene preservato all'interno del pacchetto stesso.

Se la destinazione del pacchetto si trova al di fuori della LAN, allora il pacchetto deve essere inviato al router di rete. Il router funge da gateway predefinito per la rete e viene utilizzato per instradare i pacchetti di dati tra le diverse reti. Il router analizza l'indirizzo IP di destinazione del pacchetto e lo inoltra alla rete corretta fino a quando non raggiunge la destinazione finale.

Discussione d'esame

Due LAN separate da un Router (Livello 3 quindi) di mezzo. Ha infatti sia indirizzi IP che MAC per ogni interfaccia. Supponiamo che una macchina abbia necessità di parlare con un'altra macchina, che potrebbe trovarsi sia nella sua stessa LAN che in una LAN differente, anche attraversando più router.



Le due operazioni sono nettamente differenti:

Stessa LAN :

Il mittente deve solo scoprire che il destinatario appartiene alla stessa LAN, quindi può preparare una frame Ethernet (ricordiamo che la comunicazione avviene a livello DLL, non IP, perché il datagramma IP è incapsulato nella frame Ethernet). Deve solo scoprire il MAC Address del destinatario, cosa che può fare con ARP mandando in broadcast la richiesta. Questa macchina potrà rispondere tranquillamente alla richiesta broadcast in quanto si trova sulla stessa LAN

LAN Differenti :

Se il destinatario è fuori dalla sua LAN non può usare ARP per scoprire il MAC address della destinazione, per due motivi:

- a. Il router non fa passare nulla a livello DLL, perché per quanto riguarda il router ha due livelli DLL differenti, quello della LAN rossa e quello della LAN verde. Anche se il mittente riuscisse a scoprire il MAC address che gli interessa, la frame Ethernet verrebbe comunque spacciata una volta arrivato al router, e ovviamente scartata.
- b. Non esiste solo Ethernet: Ci sono reti DLL che non hanno il MAC Address (per esempio tante macchine collegate tutte punto a punto), quindi semplicemente dire che il mittente vuole conoscere il MAC Address della destinazione è inutile, perché potrebbe anche non esistere.

Quindi se la macchina è fuori dalla LAN devo indirizzare il pacchetto al router, il quale spacciherà la frame Ethernet, leggerà il pacchetto IP e creerà un'ulteriore frame Ethernet da spedire dentro l'altra LAN (verde).

Tabella host

Ogni macchina all'interno di una rete fa da router ed è dotata di una tabella di routing che **serve ad associare un indirizzo IP di destinazione a un'interfaccia di rete** sulla stessa macchina o su un router di rete.

La tabella di routing contiene informazioni sui router, insieme alle relative interfacce di rete e ai percorsi di instradamento. Quando una macchina deve inviare un pacchetto di dati, consulta la tabella di routing per determinare il percorso migliore per la destinazione. Se la destinazione si trova sulla stessa subnet, la macchina invia il pacchetto direttamente alla destinazione. In caso contrario, cerca il router di rete predefinito(gateway predefinito) e invia il pacchetto a questo router per l'instradamento verso la destinazione.

DHCPv4 (Dynamic Host Configuration Protocol versione 4)

Con l'avvento dei portatili nasce l'esigenza di assegnare un indirizzo ad una macchina che non ne ha uno.

Il protocollo DHCPv4 è un protocollo di rete utilizzato per l'assegnazione dinamica degli indirizzi IP alle macchine sulla rete. **DHCPv4 consente alle macchine di ottenere un indirizzo IP valido in modo automatico e semplificato**, senza la necessità di assegnare manualmente un indirizzo IP a ciascuna macchina.

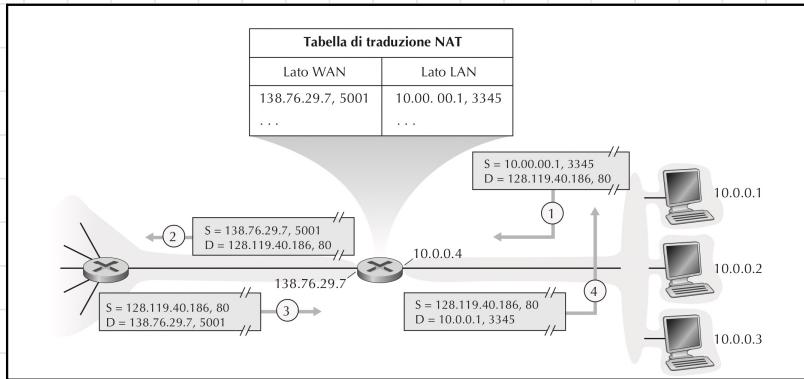
Il protocollo DHCPv4 supporta anche l'assegnazione di indirizzi IP temporanei, noti come indirizzi IP "leasing". In questo caso, l'indirizzo IP viene assegnato alla macchina solo per un periodo di tempo limitato, dopo il quale la macchina deve rinnovare l'assegnazione dell'indirizzo IP. Questo consente di gestire in modo più efficiente gli indirizzi IP disponibili sulla rete. Il protocollo DHCPv4 prevede quattro fasi principali per l'assegnazione dell'indirizzo IP:

- 1. Scoperta DHCP:** la fase di Scoperta DHCP inizia quando il client DHCP si connette alla rete e invia un messaggio di scoperta in broadcast sulla rete locale alla ricerca di un server DHCP disponibile. Il messaggio contiene informazioni come l'identificatore del client, il tipo di hardware utilizzato e le opzioni di configurazione richieste.
- 2. Offerta DHCP:** il server DHCP riceve il messaggio di scoperta e invia un messaggio di offerta al client. In questo messaggio, il server offre un indirizzo IP disponibile al client DHCP, insieme ad altre informazioni di configurazione come la subnet mask e l'indirizzo del gateway.
- 3. Richiesta DHCP:** il client DHCP riceve il messaggio di offerta dal server e invia un messaggio di richiesta per confermare l'assegnazione dell'indirizzo IP offerto dal server. In questo messaggio, il client DHCP richiede l'assegnazione dell'indirizzo IP offerto dal server.
- 4. ACK DHCP:** il server DHCP riceve il messaggio di richiesta e invia un messaggio di ACK al client DHCP per confermare l'assegnazione dell'indirizzo IP richiesto. Il messaggio ACK contiene l'indirizzo IP assegnato al client DHCP, insieme alle altre informazioni di configurazione richieste.

Il protocollo DHCP prevede queste quattro fasi principali per l'assegnazione dell'indirizzo IP, e solo dopo aver completato queste fasi il client può utilizzare l'indirizzo fornito dal server DHCP. Il server DHCP mantiene un registro dello stato delle assegnazioni degli indirizzi IP ai client, però **non prevede l'autenticazione**.

NAT (network acces translation)

Nella figura vi è una rete privata, ossia una rete i cui indirizzi hanno significato solo per i dispositivi interni ed un router abilitato al NAT. Le quattro interfacce della rete domestica hanno lo stesso indirizzo di sottorete, 10.0.0.0/24. Esistono centinaia di migliaia di reti private, molte delle quali usano un identico spazio di indirizzamento, 10.0.0.0/24, per scambiare pacchetti fra i loro dispositivi. Ovviamente, quelli inviati sull'Internet globale non possono utilizzare questi indirizzi come sorgente o destinazione. Ma se gli indirizzi privati hanno significato solo all'interno di una data rete, come viene gestito l'indirizzamento dei pacchetti relativi all'Internet globale, in cui gli indirizzi sono necessariamente **univoci**? La risposta è il NAT.



I router abilitati al NAT non appaiono come router al mondo esterno, ma si comportano come un unico dispositivo con un unico indirizzo IP. Supponiamo che un utente che si trovi nella rete domestica dietro l'host 10.0.0.1 richieda una pagina web da un server (porta 80) con indirizzo IP 128.119.40.186. L'host 10.0.0.1 assegna il numero di porta di origine (arbitrario) 3345 e invia il datagramma nella rete locale. Il router NAT riceve il datagramma, genera per esso un nuovo numero di porta di origine 5001, sostituisce l'indirizzo IP sorgente con il proprio indirizzo IP sul lato WAN 138.76.29.7 e sostituisce il numero di porta di origine iniziale 3345 con il nuovo numero 5001. Il NAT del router aggiunge inoltre una riga alla propria **tabella di traduzione NAT**.

Il web server, ignaro della manipolazione subita dal datagramma in arrivo con la richiesta HTTP, risponde con un datagramma con l'indirizzo IP del router NAT come destinazione e il cui numero di porta di destinazione è 5001. Quando questo datagramma arriva al router NAT, quest'ultimo consulta la tabella di traduzione NAT usando l'indirizzo IP di destinazione e il numero di porta di destinazione per ottenere l'appropriato l'indirizzo IP (10.0.0.1) e il corretto numero di porta di destinazione (3345) del browser nella rete domestica. Il router quindi riscrive l'indirizzo di destinazione del datagramma e il suo numero di porta di destinazione e inoltra il datagramma nella rete domestica.

NB

- Il server NAT non necessariamente è un router ma potrebbe essere una macchina sulla rete.
- Nella rete NAT il primo pacchetto scambiato deve partire necessariamente dalla rete locale.

Problemi con IPv4 e IPv6

IPv4 presenta delle problematiche che verranno risolte con l'introduzione di IPv6 :

Tabelle di routing: Dentro i router sono presenti delle tabelle di instradamento che indicano le porte di uscita in base alla destinazione che il pacchetto in ingresso deve raggiungere. Le tabelle di routing iniziano a diventare parecchio complicate a causa del vecchio standard IP che assegnava in modo disordinato le locazioni geografiche .

Indirizzi : Dato che gli indirizzi IPv4 sono gerarchici, bisogna distribuirli secondo un certo criterio, ma purtroppo i gruppi di indirizzi sono stati assegnati in maniera disordinata (a casaccio).

In IPv4, gli indirizzi sono a 32 bit, scritti in formato decimale puntato e sono divisi in classi di indirizzi. Ciò significa che solo un numero limitato di indirizzi IPv4 sono disponibili per l'uso (circa 4 miliardi).

In IPv6, gli indirizzi sono a 128 bit e sono scritti in formato esadecimale. Questo sistema di indirizzamento offre un numero molto più grande di indirizzi rispetto ad IPv4, quasi illimitato.

Sicurezza : riguarda la mancanza di meccanismi di sicurezza adeguati a proteggere i dati trasmessi sulla rete. Il livello di sicurezza di IPv4, posto dopo il livello di trasporto, tiene in chiaro tutti i dati successivi, inclusi gli indirizzi IP e le porte. Questi dati sensibili vengono trasmessi in chiaro, rendendoli vulnerabili agli attacchi di spoofing degli indirizzi IP. IPv6 invece supporta la crittografia e la sicurezza dei pacchetti attraverso l'utilizzo di IPsec, un insieme di protocolli di sicurezza che forniscono autenticazione, integrità dei dati e crittografia per i pacchetti IP.

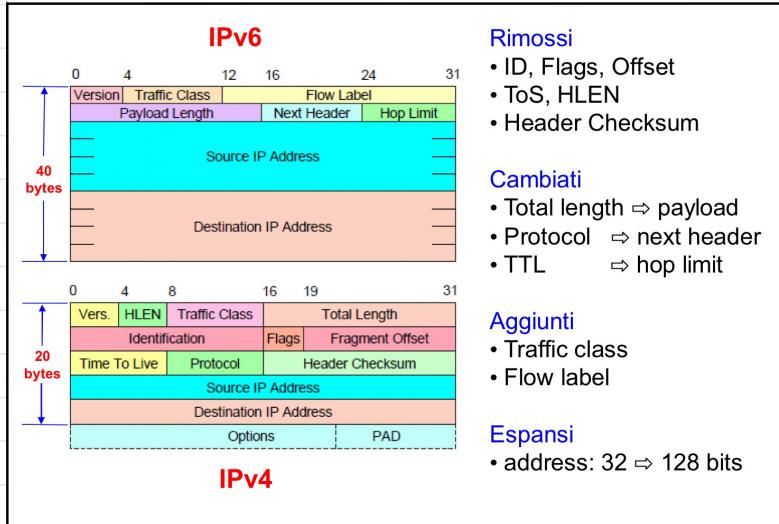
Autoconfigurazione (Plug & Play): in IPv4, la configurazione degli indirizzi IP e delle impostazioni di rete di un dispositivo deve essere effettuata manualmente o mediante l'utilizzo di protocolli come DHCP. L'autoconfigurazione plug and play è una funzionalità di IPv6 che consente ai dispositivi di configurare automaticamente la propria interfaccia di rete, senza la necessità di un amministratore di rete o di un server DHCP.

Gestione della qualità del servizio (QoS): la gestione della QoS è un problema importante nella comunicazione di rete, poiché consente di garantire determinati livelli di servizio per le applicazioni. In IPv4, la QoS è gestita mediante il campo Type of Service (ToS) nell'intestazione IP ma non viene mai sfruttato. In IPv6 il bit Flow Label consente di fornire un livello di servizio differenziato per flussi di traffico specifici sulla rete.

Indirizzamento multicast: in IPv4, l'indirizzamento multicast è limitato ai soli indirizzi di classe D, che consentono la trasmissione di dati a un gruppo di destinatari. Lo spazio di indirizzamento multicast in IPv4 è limitato e non è sufficiente per supportare un gran numero di gruppi multicast. Gli indirizzi multicast IPv6 sono un sottoinsieme degli indirizzi IPv6 e hanno una struttura specifica per supportare l'invio di pacchetti a più host contemporaneamente. Gli indirizzi multicast IPv6 iniziano con il prefisso **'ff'**, seguito da altri campi che consentono una maggiore flessibilità nella definizione di gruppi multicast

Indirizzamento host mobili: consente a un dispositivo di mantenere la stessa configurazione di rete IPv6 mentre si sposta da una rete a un'altra. Un dispositivo mobile possiede un indirizzo IPv6 permanente, noto come "indirizzo di casa", che è associato alla sua interfaccia di rete. Quando il dispositivo si sposta in una nuova rete, ottiene un nuovo indirizzo IPv6 temporaneo, noto come "indirizzo di cura", sulla nuova rete. Il dispositivo continua a utilizzare l'indirizzo di casa come il suo indirizzo permanente, ma utilizza l'indirizzo di cura come l'indirizzo sorgente per tutte le comunicazioni in uscita sulla nuova rete. In questo modo, le comunicazioni in uscita sembrano provenire dalla nuova rete, anche se il dispositivo conserva il suo indirizzo permanente

Datagramma IPv6



Rimossi

- ID, Flags, Offset
- ToS, HLEN
- Header Checksum

Cambiati

- Total length \Rightarrow payload
- Protocol \Rightarrow next header
- TTL \Rightarrow hop limit

Aggiunti

- Traffic class
- Flow label

Espansi

- address: 32 \Rightarrow 128 bits

La dimensione del datagramma IPv4 è variabile a seguito di campi opzionali che fanno passare la dimensione dell'header da 20 fino a 60 byte, questo rende complicata la ricerca attraverso le memorie associative. L'header in IPv6 ha infatti una lunghezza fissa di 40 byte, il che permette di rimuovere il campo che specifica la lunghezza dell'header.

In quanto non veniva utilizzato è stato rimosso anche il controllo dell'integrità dell'header tramite l'header checksum.

Il campo TTL non rappresentava un tempo ma un numero di nodi nel caso di IPv4, con IPv6 questo campo identifica invece un tempo reale massimo, entro il quale il pacchetto deve arrivare a destinazione o viene scartato.

In IPv6 viene meno il campo "protocol" e viene sostituito con **next header**, ovvero delle liste linkate nelle quali ogni nodo è un servizio o header opzionale che viene aggiunto al servizio già esistente. Un header opzionale contiene come primo campo un parametro "next header" che contiene il link al prossimo campo, messo all'inizio in quanto più veloce da trovare perché ogni campo ha lunghezza variabile. Un secondo campo specifica appunto la lunghezza dell'header opzionale. Sono stati tolti i vari flag, offset ecc e sono stati inseriti tutti come header opzionali.

Non è possibile riassemblare la lista linkata inserendo dei nodi in mezzo e quindi il solo router mittente può eseguire la frammentazione del pacchetto ed è stato inoltre specificato un **MTU minimo** di 1280 byte per supportare un trasferimento tramite IPv6. In IPv4 se si perde un frammento si necessita di ritrasmettere l'intero pacchetto, IPv6 quindi obbliga a non poter frammentare. Viene previsto un header anche per la sicurezza "authentication header" che codifica tutto il resto del pacchetto.

In pratica con IPv6, il campo di offset del frammento è stato rimosso dall'intestazione del pacchetto, e quindi i router intermedi non possono frammentare il pacchetto. Invece, la frammentazione deve essere gestita dai dispositivi sorgente, che devono suddividere i dati in pacchetti con una dimensione inferiore al payload massimo definito da IPv6.

Struttura indirizzo IPv6

Un indirizzo IPv6 ha una struttura di 16 byte, ovvero 128 bit, divisi in 8 gruppi di 16 bit (ovvero da 4 nibble) ciascuno, separati da due punti (:).

Un indirizzo IPv6 è scritto in notazione esadecimale, che rappresenta ogni gruppo di 16 bit con un numero esadecimale da 0 a F. Per facilitare la scrittura degli indirizzi IPv6, è possibile omettere gli zeri iniziali di ogni gruppo di 16 bit e comprimere i gruppi di zeri consecutivi in un singolo gruppo di zeri, indicato con i due punti (::). Tuttavia, questa compressione può essere utilizzata **solo una volta** in ogni indirizzo IPv6.

Un esempio di indirizzo IPv6 scritto in notazione esadecimale è:

2001 : 0db8 : 85a3 : 0000 : 0000 : 8a2e : 0370 : 7334

Questo stesso indirizzo potrebbe essere scritto con la compressione dei gruppi di zeri consecutivi:

2001 : db8 : 85a3 :: 8a2e : 370 : 7334

L'indirizzo IPv6 può anche includere una prefisso di rete, che specifica la porzione dell'indirizzo che identifica la rete, e una porzione di interfaccia, che identifica il singolo host all'interno della rete. Il prefisso di rete è solitamente specificato con una lunghezza di prefisso, indicando il numero di bit che compongono la parte di rete (solitamente i primi 2 o 3 blocchi di 4 nibble) dell'indirizzo IPv6.

Ad esempio, un prefisso di rete di 64 bit indica che i primi 64 bit dell'indirizzo IPv6 sono riservati alla rete, mentre gli ultimi 64 bit sono utilizzati per identificare l'interfaccia host all'interno di quella rete.

Subnet mask in IPv6

La subnetting in IPv6 è analoga a quella in IPv4, ma con l'utilizzo di indirizzi a 128 bit anziché a 32 bit. Ciò significa che il numero di subnet disponibili è molto più elevato rispetto ad IPv4.

Per indicare la maschera di sottorete in IPv6, si utilizza il prefisso di sottorete, che specifica quanti bit dell'indirizzo IPv6 sono riservati alla sottorete. Il prefisso di sottorete viene solitamente specificato come un multiplo di 16 bit, ovvero un blocco di 4 nibble. Ad esempio, se si utilizza un prefisso di sottorete di /64, i primi 64 bit dell'indirizzo IPv6 sono riservati alla sottorete, mentre i restanti 64 bit sono riservati all'identificatore dell'interfaccia.

Indirizzi riservati

Gli indirizzi IPv6 riservati sono indirizzi che non sono assegnati ad alcuna interfaccia di rete e sono stati riservati per scopi speciali. Questi indirizzi **non sono utilizzabili per la comunicazione su Internet** e non possono essere assegnati a nessun host.

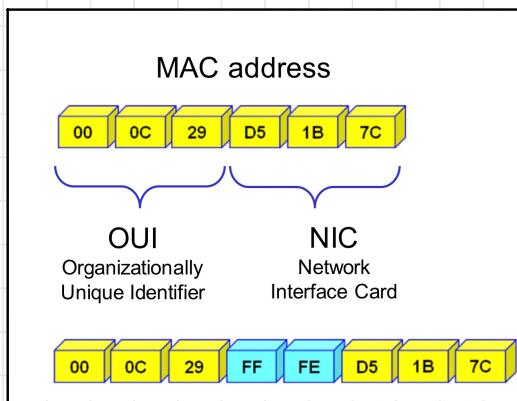
- Gli indirizzi **FEC0::/48** sono stati originariamente riservati per l'indirizzamento di sito locale, ovvero per la comunicazione tra nodi all'interno di un sito
- Gli indirizzi **FE80::/64** sono utilizzati per gli indirizzi di **local-link**, ovvero indirizzi che possono essere utilizzati solo all'interno di una rete locale e che non sono instradati su Internet. Questi indirizzi vengono utilizzati per la comunicazione tra nodi della **stessa rete locale**, ad esempio tra due computer collegati alla stessa LAN e non possono essere assegnati a dispositivi su reti differenti.

Gli indirizzi local link IPv6 sono simili agli indirizzi privati IPv4 in quanto entrambi sono utilizzati all'interno di una rete privata e non sono accessibili direttamente dall'esterno della rete.

EUI-64 (extended unique identifier-64)

Gli indirizzi IPv6 globali, ovvero quelli che è possibile utilizzare per navigare in rete, vengono autogenerati a partire dal MAC address della scheda di rete del dispositivo stesso.

L'EUI-64 è un metodo di generazione degli identificatori per le interfacce di rete in IPv6. EUI-64 usa l'indirizzo MAC della scheda di rete, composto da 6 byte (48 bit) e lo modifica inserendo un blocco **FF:FE** da 2 byte (16 bit) in mezzo tra i primi e gli ultimi tre byte del MAC, ottenendo i 64 bit che identificano l'interfaccia di rete.



Questo metodo semplifica la configurazione automatica degli indirizzi IPv6 per le interfacce di rete, garantendo anche l'unicità degli identificatori.

Comunicazione tra reti con IPv6

Gli indirizzi IPv6 sono suddivisi in quattro tipi principali:

- **Unicast**: identificano in modo univoco un'interfaccia di rete su una singola macchina. Sono utilizzati per la comunicazione diretta tra due host.
- **Broadcast**: in IPv6 non esiste più il concetto di indirizzo broadcast. Viene invece utilizzato il concetto di indirizzo multicast per la comunicazione a tutti i nodi di una rete.
- **Multicast**: identificano un gruppo di interfacce di rete che hanno una qualifica specifica e vengono utilizzati per la comunicazione di gruppo. Sono identificati da un prefisso iniziale "FF" seguito da un numero identificativo che specifica il gruppo di destinazione.
- **Anycast**: novità di IPv6, identificano un insieme di interfacce di rete, ma solo una di queste viene selezionata come destinazione del pacchetto. Viene utilizzato il concetto di anycast per **identificare il nodo più vicino in una rete** che condivide lo stesso indirizzo anycast.

Comunicazione tra reti con IPv6

Quando un dispositivo IPv6 ha bisogno di comunicare con un altro dispositivo **sulla stessa rete**, utilizza NDP (ARP) per determinare il suo indirizzo MAC. In questo modo, il dispositivo che ha iniziato la comunicazione può mappare l'indirizzo IPv6 del destinatario con il suo indirizzo MAC e inviare i pacchetti direttamente a livello di link layer.

Per quanto riguarda la comunicazione **tra LAN differenti**, viene utilizzato il protocollo DHCPv6 per assegnare gli indirizzi IPv6 ai client e per fornire altre informazioni di configurazione come il gateway predefinito e i server DNS. Non ha più senso utilizzare NAT in IPv6 poiché lo spazio degli indirizzi è molto ampio e non esiste più il problema di esaurimento degli indirizzi come in IPv4.

Firewall

Un firewall è un dispositivo (hardware) o un software che controlla il traffico di rete in entrata e in uscita verso e da una rete. Il suo obiettivo principale è quello di proteggere la rete da accessi non autorizzati e da attacchi informatici, impedendo l'accesso a risorse di rete non autorizzate.

- **Firewall software**

Il filtraggio in ingresso controlla il traffico di pacchetti che entra nella rete, mentre il filtraggio in uscita controlla il traffico di pacchetti che esce dalla rete. Entrambi i tipi di filtraggio possono essere configurati per bloccare o consentire il traffico in base a criteri di sicurezza specifici, come l'indirizzo IP di origine e destinazione, il numero di porta, il protocollo di rete, il tipo di dati contenuti nei pacchetti e così via. Il **blocco e il passaggio del traffico** sono le due operazioni principali eseguite dal firewall software. Il blocco del traffico impedisce il passaggio di pacchetti di rete non autorizzati o sospetti, mentre il passaggio del traffico consente il passaggio di pacchetti di rete autorizzati e legittimi. In questo modo, il firewall software può proteggere la rete da attacchi informatici, malware e altre minacce informatiche.

- **Firewall hardware**

Un **application gateway** è una macchina **non raggiungibile** che viene interposta tra due LAN private per consentire il passaggio dei pacchetti da una rete privata all'altra. A differenza del gatweway predefinito (che mi dice solo come trovare l'uscita da una rete) un application gateway è un firewall hardware che guarda il contenuto del pacchetto e prende decisioni sul proseguimento o scarto del pacchetto.

Il costo dei firewall hardware è molto più elevato dei firewall software, questo li rende molto meno utilizzati. Un firewall software è meno sicuro di uno hardware per il semplice motivo che si trova già sulla macchina, che può essere a sua volta compromessa.

Il concetto di porta aperta definisce una porta di ascolto dove un processo chiede di ricevere i pacchetti che il sistema operativo riceve su quella determinata porta, con porta chiusa si intende che il sistema operativo scarta a priori il pacchetto nonostante dei processi siano in ascolto su quella porta.

DMZ (zona demilitarizzata)

La zona demilitarizzata (DMZ) è un'area di rete separata che viene creata all'interno di una rete aziendale o di un'altra rete protetta. La DMZ viene utilizzata per ospitare server e applicazioni che devono essere accessibili dall'esterno della rete, come server web, server di posta elettronica e server FTP. La DMZ è separata sia dalla rete interna che da quella esterna ed è protetta solo parzialmente dal router tramite uno o più firewall, che ne controllano l'accesso da parte degli utenti esterni e proteggono la rete interna da eventuali attacchi provenienti dalla DMZ. In genere, i firewall sono configurati per consentire solo il traffico di rete autorizzato tra la DMZ e la rete interna o esterna.

Algoritmi di routing

Gli algoritmi di routing sono algoritmi utilizzati dai router per determinare il percorso ottimale per instradare i pacchetti attraverso una rete. Quando si parla di **routing distribuito** si fa riferimento ad un approccio nel quale l'algoritmo di routing è in esecuzione in ogni nodo della rete in modo distribuito ed indipendente dagli altri nodi. Ogni nodo utilizza le informazioni locali per determinare il percorso migliore per instradare i pacchetti. In un approccio distribuito, per garantire il corretto funzionamento della rete, bisogna che risulti chiaro quando l'algoritmo inizia la propria esecuzione e quando l'algoritmo si conclude.

Flooding

Flooding è un algoritmo di routing che prevede l'invio di tutti i pacchetti di dati a tutti i nodi della rete, indipendentemente dalla destinazione finale del pacchetto. In altre parole, ogni nodo inoltra il pacchetto ricevuto a tutti i nodi ad esso adiacenti, che a loro volta lo inoltrano ai propri vicini, e così via, fino a quando il pacchetto raggiunge la destinazione finale. L'algoritmo di flooding è molto semplice da implementare e **non richiede conoscenza della topologia della rete**. Tuttavia, può causare un'elevata congestione della rete, poiché ogni pacchetto viene trasmesso a ogni nodo della rete. In tali reti, l'algoritmo di flooding può garantire una maggiore resilienza alla rete, poiché ogni nodo ha la possibilità di inoltrare il pacchetto a tutti i nodi vicini, indipendentemente dalla loro posizione o dal loro stato di connessione.

Si necessita di un modo per bloccare il flooding in quanto la ritrasmissione continua di pacchetti su tutti i link provoca una congestione dovuta al fatto che molti pacchetti vengono rispediti molte volte sugli stessi link. A tal proposito sono state proposte diverse soluzioni :

- **Usare un ID per i pacchetti**

Per bloccare il flooding si può pensare di aggiungere un identificativo ad ogni pacchetto così da poter tenere conto di tutti i pacchetti di cui si fa forwarding, in modo da non ritrasmettere copie dello stesso pacchetto se è già stato inoltrato. Questo approccio risolve la congestione ma sorge il problema della dimensione della tabella che tiene conto di tutti i pacchetti inoltrati e diventa difficoltoso memorizzarla a meno che il passaggio di un pacchetto per un link non sia sporadico

- **TTL limite**

Se si possono fare assunzioni sulla rete si può invece pensare di stabilire un TTL limite pari al diametro del grafo, ovvero la distanza massima tra una coppia di nodi del grafo. Quindi si fissa il numero massimo di nodi che il pacchetto può attraversare prima di essere scartato pari al valore di diametro. Il valore limite di TTL viene utilizzato come meccanismo di sicurezza per prevenire il loop infinito dei pacchetti in una rete. Ogni volta che il pacchetto viene inoltrato a un nodo, il valore di TTL viene decrementato di uno. Se il valore di TTL raggiunge lo zero, il pacchetto viene scartato. In questo modo, il pacchetto viene inoltrato solo per un numero limitato di nodi, prevenendo il loop infinito. Il problema è che spesso non è possibile fare assunzioni sulla rete.

- **Spanning tree protocol**

L'obiettivo dello STP è quello di creare un albero di copertura minima, ovvero un sottografo dell'intera topologia di rete che contiene tutti i nodi e che non ha cicli ed elimina gli archi che portano a nodi già raggiungibili.

Si elegge un nodo radice sulla base dei confronti tra i vari MAC address dei nodi, in particolare la radice è costituita dal dispositivo che ha il **MAC address minore** e si effettua una visita in ampiezza (BFS) a partire da quel nodo.

Tuttavia, il protocollo STP riscontra problemi di applicabilità in quanto richiede la conoscenza completa della topologia di rete e può causare ritardi nella convergenza della rete in caso di cambiamenti nella topologia di rete.

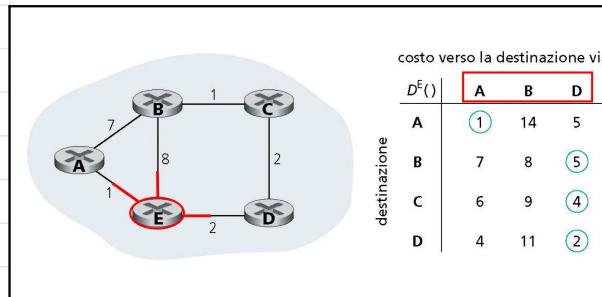
Il flooding, d'altra parte, è un algoritmo di routing **distribuito** che inoltra tutti i pacchetti di dati a tutti i nodi della rete, indipendentemente dalla destinazione finale del pacchetto. Questo algoritmo non viene utilizzato in Internet perché può causare una congestione di rete e perché non fornisce un percorso ottimale per il pacchetto.

Distance Vector (DV) :

Viene costruita una tabella dove con riferimento ad un certo nodo sorgente (nell'esempio E) si mettono in riga le destinazioni ed in colonna vengono inseriti i nodi verso i quali si obbliga il primo salto.

La tabella è rettangolare e non quadrata, in quanto si hanno nelle righe gli N nodi del grafo e nelle colonne solo gli M nodi verso i quali è possibile obbligare il primo salto, ovvero raggiungibili dalla sorgente con un solo arco.

Nella posizione (i, j) della tabella si trova **il costo minimo per andare dalla sorgente alla destinazione 'i' obbligando il primo salto verso il nodo 'j'**. Si nota che dal secondo in poi ogni salto può prendere qualunque direzione e si può tornare anche indietro. Si ricava il vettore delle distanze scegliendo i **costi minimi** in ogni riga, ovvero i percorsi di costo minore verso tutte le destinazioni.



Inizialmente non si possono fare assunzioni sulla tabella e vengono posti tutti i costi ad infinito. Ogni nodo costruisce il proprio vettore delle distanze e gli altri nodi nella rete cercano di migliorare i valori del proprio vettore utilizzando le proprie stime di cammino minimo, facendo variare la stima proposta dagli altri nodi.

Questo è un algoritmo **distribuito** basato su Bellman-Ford, e la convergenza viene raggiunta rapidamente in quanto ogni nodo lo esegue in modo distribuito ed indipendente dagli altri nodi.

Sono noti i punti di inizio e fine del protocollo distribuito : Distance Vector inizia quando si ha una variazione nella stima dei cammini minimi e termina quando non si verifica nessun cambiamento durante l'iterazione precedente, ovvero quando si arriva a convergenza (e lo si fa rapidamente) e la situazione di rete diventa stabile quando non si hanno più variazioni nelle stime.

La tabella di inoltro dice quale è il prossimo passo da seguire nel cammino, quella di routing fornisce il quadro generale della rete. L'output dell'algoritmo distance vector è la **tabella di inoltro**. Si nota che questo algoritmo non implica la conoscenza a priori del grafo, quindi ogni nodo si affida alle informazioni ricevute dai nodi ad esso adiacenti.

Questo implica un forte problema di sicurezza dovuto al fatto che un nodo malevolo potrebbe **dirottare il traffico** destinato ad altri nodi. Non c'è soluzione a questo problema e si necessita di fidarsi. La terminazione dell'algoritmo avviene anch'essa in modo distribuito. Questo è stato il primo algoritmo ad essere utilizzato su internet seppur con delle precisazioni da stabilire, come il concetto di infinito uguale a 15, peso unitario dei link ecc... .

Distance vector risponde male al peggioramento delle condizioni di rete

Un problema può sorgere quando il peso di un arco varia peggiorando le stime note ed i nodi che nella costruzione dei loro vettori tengono conto del peso di quell'arco avranno una stima fittizia, dato che tengono conto di un peso che è stato modificato.

Split horizon e Poisoned Reverse

Lo **split horizon** è una tecnica utilizzata in Distance Vector per prevenire i loop di rete dovuti al peggioramento di una stima. In pratica, lo split horizon prevede che un nodo non invii informazioni sul cammino minimo per raggiungere una destinazione a un altro nodo facente parte di tale cammino, al fine di evitare che i pacchetti vadano a ciclare indefinitamente nella rete.

Questo perché se il nodo ricevente utilizza il percorso ricevuto dal nodo mittente per migliorare la propria stima, essa terrà conto del peso fittizio dell'arco che ha peggiorato la propria stima, causando un loop di rete.

La tecnica del **poisoned reverse** viene utilizzata insieme allo split horizon per risolvere il problema delle stime fittizie causate dalla variazione del peso di un arco. Quando c'è un peggioramento del peso di un arco, tutti i nodi che utilizzavano quell'arco per il calcolo del cammino minimo pongono la propria stima ad **infinito**, in modo da evitare che il pacchetto vada a ciclare nella rete.

Considerazioni sull'approccio distribuito

Un sistema distribuito garantisce affidabilità ed alta efficienza e resistenza ai guasti. Si necessita di una sincronizzazione continua tra i dispositivi per mantenerli aggiornati e di uno scambio limitato di informazioni per garantire alti livelli di sicurezza, infatti se non si conosce l'informazione si rimanda a chi la conosce.

L'algoritmo inoltre deve arrivare velocemente a convergenza, per cui devono essere ben note le condizioni di inizio e fine di cui si tiene conto nell'esecuzione.

Link State Routing (LSR)

Link State Routing è un algoritmo che nasce per sopperire ad alcune carenze di Distance Vector, come il fatto che non tiene conto dello stato dei link ma attribuisce banalmente un peso unitario a tutti i collegamenti. Questo è un algoritmo globale (tutti i nodi sincronizzati sulle stesse informazioni) basato su Dijkstra, ipotesi applicabile in pratica dato che nessun arco ha peso negativo. L'output dell'algoritmo è l'albero dei cammini minimi, ovvero un **minimum spanning tree** che può non essere unico se esistono diversi cammini che portano alla stessa destinazione con uguale peso.

L'algoritmo tiene conto dello stato del link, la condizione di partenza dell'algoritmo è a tempo, infatti periodicamente ogni nodo informa quelli ad esso adiacenti delle condizioni dei link. Le condizioni di terminazione riguardano il raggiungimento di una situazione stabile nel grafo, ma come si può conoscere la topologia del grafo?

Ogni nodo sa solo chi sono i propri vicini e che nessun arco ha peso negativo o nullo. Ogni nodo sfrutta il flooding per inviare in broadcast a tutti i vicini la propria tabella di nodi adiacenti, contenente il peso dell'arco che permette di raggiungerli. L'approccio possibile per fermare il Flooding è tenere conto dell'ID ma la soluzione è applicabile solo se il messaggio da inviare è sporadico, ovvero se l'introduzione di altri nodi non fa cambiare le tabelle degli adiacenti molto frequentemente. La conseguenza di questo comportamento è che ogni router applica Dijkstra.

LSR è robusto agli attacchi, in quanto ogni nodo non può fornire informazioni fittizie dei link ai nodi adiacenti perché verrebbero immediatamente contrarie dall'altro nodo, in quanto si riferiscono allo stato di un link che è uguale in entrambe le direzioni.

Problemi con LSR

- Un problema di **oscillazione** riguarda il fatto che si alternano continuamente diversi percorsi minimi dati da diversi alberi dei cammini minimi. L'oscillazione crea problemi di sincronia in quanto LSR sfrutta la sincronizzazione tra le informazioni nei nodi per la propria esecuzione.
- Un altro problema riguarda la **dimensione del grafo**, infatti il flooding causa una forte congestione all'aumentare del numero di nodi. Inoltre considerare porzioni troppo grandi del grafo comporta ricoprire AS differenti, il che non interessa i link considerati.

RIP (routing information protocol)

Il Routing Information Protocol (RIP) è un protocollo di routing che fornisce l'implementazione di distance vector :

RIPv1

RIP v1 utilizza una metrica di distanza basata sul **numero di hop** (o salti) necessari per raggiungere una destinazione, infatti ogni router mantiene una **tavella di routing** che indica il numero di hop necessari per raggiungere ogni nodo nella rete. Si fissa il numero massimo di salti a 15 per prevenire ritardi nella convergenza dell'algoritmo dovuti ad eccessive distanze tra nodi. Un router sceglierà il percorso più breve in termini di numero di hop per raggiungere la destinazione.

Con RIP ogni router in una rete trasmette periodicamente (ogni 30 secondi) le proprie informazioni di routing agli altri router tramite messaggi di aggiornamento. RIP utilizza un **timer di invalidità** per determinare se un percorso di routing non è più valido, in particolare se un percorso non viene aggiornato entro il tempo di invalidità, viene considerato non valido e rimosso dalla tabella di routing.

Infine, il protocollo RIP sfrutta un **garbage-collection timer** per rimuovere i router che non sono più raggiungibili, dalla tabella di routing. Se un router non trasmette informazioni di routing per il tempo di 120 secondi scandito da questo timer, allora viene considerato irraggiungibile e rimosso dalla tabella di routing.

Nel pacchetto RIPv1 non è specificata una maschera in quanto opera utilizzando le classi di indirizzi.

Una **tavella di routing** contiene i seguenti campi:

- Indirizzo di destinazione
- Distanza dalla destinazione in hop (massimo 15)
- Next hop: router adiacente a cui inviare i pacchetti
- Timeout (ogni 30 secondi)
- Garbage-collection timer (120 secondi)

RIP V2

RIPv2 aggiunge diverse caratteristiche rispetto a RIPv1 :

RIPv2 supporta **l'indirizzamento CIDR** e il subnetting con maschere di sottorete variabili, ciò significa che RIPv2 può gestire sottoreti di dimensioni diverse, migliorando l'efficienza nell'utilizzo degli indirizzi IP.

Questa versione supporta **l'autenticazione dei messaggi**, che consente ai router di autenticare l'origine dei messaggi di aggiornamento della tabella di routing e di evitare l'inserimento di informazioni di routing fittizie nella tabella di routing. Consente ai router di specificare il prossimo hop per raggiungere una determinata destinazione. Ciò significa che i router possono scegliere il prossimo hop per un percorso di routing in base a fattori come il costo del percorso.

RIPv2 utilizza il "**split horizon**" per prevenire i loop di routing tra i router. Questo significa che un router non annuncerà una destinazione a un altro router se ha appreso quella destinazione da quel router stesso. Inoltre, RIPv2 supporta anche la tecnica del "**poisoned reverse**", che prevede che un router annuncerà una destinazione con una metrica di distanza infinita (di solito 16) a un altro router dal quale ha appreso che quella destinazione non è raggiungibile. Questo aiuta a prevenire loop di routing rimuovendo immediatamente la destinazione dalla tabella di routing del router che ha appreso l'informazione.

RIP ng (new generation)

Questa è una versione aggiornata di RIP che usa indirizzi IPv6

Internet Routing

Internet è organizzata in autonomous systems (AS).

Un Autonomous System (AS) è un insieme di reti interconnesse e gestite da un'unica entità amministrativa che utilizza un protocollo di routing comune. In altre parole, un AS è un sistema autonomo e indipendente che si occupa di instradare il traffico Internet all'interno della propria rete e verso altre reti.

Distinguiamo due tipologie di protocolli di routing :

- **Inrerior gateways protocol** : i protocolli di routing di gateway interni (IGP) sono utilizzati all'interno di una singola Autonomous System (AS) e consentono ai router di comunicare e scambiarsi informazioni sulla topologia di rete. Alcuni esempi di protocolli IGP includono RIP (Routing Information Protocol) e OSPF (Open Shortest Path First).
- **Exterior gateways protocol** : i protocolli di routing di gateway esterni (EGP) sono utilizzati per il routing tra diverse AS e consentono ai router di comunicare con altre reti e di condividere informazioni sulla raggiungibilità delle destinazioni. BGP (Border Gateway Protocol) è un esempio di protocollo EGP.

OSPF (Open shortest path first)

L'implementazione di LSR è OSPF : è un protocollo di routing a stato di link (LSR) utilizzato per instradare il traffico all'interno di una rete IP. In pratica, OSPF determina il percorso migliore tra due punti di una rete, basandosi sulla topologia di rete e sulla metrica di costo, che tiene conto di fattori come la larghezza di banda, il ritardo e il traffico.

L'implementazione utilizza i messaggi scambiati tra i router che partecipano al protocollo OSPF, al fine di costruire un **database** di stato dei collegamenti (LSDB) che rappresenta la topologia della rete. Una volta che il database di stato dei collegamenti è stato costruito, OSPF utilizza l'algoritmo Dijkstra per determinare il percorso più breve tra due punti della rete. Questo percorso viene quindi utilizzato per instradare il traffico da una sorgente a una destinazione.

OSPF è un protocollo di routing dinamico, il che significa che è in grado di adattarsi automaticamente ai cambiamenti nella topologia della rete, come l'aggiunta o la rimozione di router o sottoreti. In OSPF lo **stato del link** viene valutato tramite l'invio di pacchetti di tipo "hello" tra i router che partecipano al protocollo. Questi pacchetti contengono informazioni sullo stato dei collegamenti, come l'indirizzo IP del router, l'ID dell'area di routing e la priorità del router all'interno dell'area.

OSPF è ampiamente utilizzato nelle reti di grandi dimensioni, come quelle delle aziende o dei provider di servizi Internet (ISP). In generale, OSPF è considerato un protocollo di routing robusto e affidabile, che fornisce una buona scalabilità, flessibilità e sicurezza.

Compiti del DLL

Data link layer si interfaccia con il livello fisico, in particolare ricepisce i segnali del livello fisico ed impacchetta i bit in dei frame di livello DLL. Fornisce un **recapito affidabile** se richiesto. Gestisce gli errori (non volontari) dovuti al canale di trasmissione dovuti al fatto che il livello fisico "rovina" il messaggio. Si possono rilevare e correggere gli errori, solo che mentre la rilevazione si può fare abbastanza bene mentre per la correzione si può al massimo garantire una qualità elevata del servizio ma non sempre è possibile.

Framing dei dati

Il "framing dei dati" consiste nell'incapsulare i dati in un frame di livello data-link, aggiungendo informazioni necessarie per garantire la corretta trasmissione dei dati sulla rete. Il framing dei dati è importante perché garantisce che i dati inviati siano **codificati** e ricevuti correttamente dal destinatario.

Codifica di due livelli : un segnale ad 1 implica che ci sia una comunicazione mentre a zero significa o stato logico basso, oppure assenza di comunicazione. Come si distingue tra 0 e assenza di comunicazione ? Si introducono più livelli.

Codifica di tre livelli o cinque livelli : portano più informazioni , infatti avrei più bit per ogni posizione.

Avere più livelli permette di perdere meno informazioni durante il passaggio da uno stato logico ad un altro in quanto se si verifica un errore su cinque livelli si perderà solo 1/5 dell'informazione, mentre con un errore in un livello nel caso di tre soli livelli si perderebbe 1/3 dell'informazione.

Codifica 4b-5b

La tecnica di codifica 4B5B viene utilizzata a livello fisico di una rete per aumentare l'affidabilità della trasmissione e semplificare la sincronizzazione tra mittente e destinatario. La tabella di codifica 4B5B è conosciuta sia dal mittente che dal destinatario .

Nella codifica 4B5B, ogni gruppo di 4 bit viene mappato in un gruppo di 5 bit, utilizzando una tabella di codifica predefinita, così da poter **aumentare l'affidabilità** della trasmissione. In particolare, ogni possibile combinazione di 4 bit viene mappata in un codice di 5 bit univoco. Questo significa che ci sono un totale di 16 combinazioni di 4 bit che verranno usate per la comunicazione tra i 32 codici possibili di 5 bit. La decodifica 4B5B garantisce una corretta ricezione dei dati, evitando le sequenze 1111 che potrebbero causare problemi di sincronizzazione. Scartare queste combinazioni tuttavia induce lo spreco di un bit ogni 5.

Codifica 8B-10B

La codifica 8B10B è una tecnica di codifica utilizzata per la trasmissione di dati su reti a larga banda come HDMI. La tecnica prevede la codifica di ogni gruppo di 8 bit di dati in un simbolo a 10 bit. Ciò significa che ogni byte di dati viene convertito in un simbolo a 10 bit prima della trasmissione. La codifica 8B10B è stata sviluppata per garantire, sulle lunghe sequenze uguale numero di zeri e uno nella codifica.

La codifica 8B10B utilizza una tabella di codifica predefinita per codificare ogni gruppo di 8 bit di dati in un simbolo a 10 bit. La tabella di codifica è progettata in modo da garantire che il numero di zeri e uno nella codifica sia bilanciato, in modo da **azzerare lo scambio energetico** nella trasmissione. La codifica con uguale numero di zeri e uno nella sequenza permette un annullamento dello scambio energetico.

Quantità di informazione

Usando tre livelli logici (alto, medio, basso) nella trasmissione si possono trasmettere più informazioni grazie alle 2^3 combinazioni possibili. Aumentando il numero di simboli (forme ottenibili con i livelli) si può **aumentare la quantità di informazioni** portate dal segnale, misurata in bit. Ad esempio con 3 simboli si hanno alto-alto-alto, alto-alto-medio, ecc... Si discrimina assenza di segnale come tutti i simboli allo stato logico medio.

Non c'è una reale corrispondenza biunivoca tra i simboli a livello fisico e i simboli a livello DLL, infatti con strategie software si possono raggruppare i simboli al fine di ottenere una maggiore velocità di trasmissione in bps (bit per secondo).

Rilevazione e Correzione degli errori

La **rilevazione** degli errori consente di individuare la presenza di un errore di trasmissione in una frame, ma non di correggerlo direttamente. Invece, la **correzione** degli errori consente sia di rilevare che di correggere gli errori di trasmissione.

Entrambi i metodi sono basati sulla presenza di **ridondanza** nella comunicazione, ovvero dati che non aggiungono informazioni che non si hanno già.

La rilevazione degli errori utilizza tecniche come la checksum o il controllo di parità per aggiungere informazioni ridondanti ai dati trasmessi. Queste informazioni ridondanti permettono al destinatario di confrontare i dati ricevuti con la somma di controllo o il valore di parità e individuare eventuali errori di trasmissione.

La correzione degli errori è più complessa e richiede più risorse di elaborazione rispetto alla rilevazione degli errori.

- **Controllo di parità**

Il controllo di parità è una tecnica di rilevazione degli errori utilizzata nelle trasmissioni seriali di dati. In questa tecnica, viene aggiunto un bit di parità alla fine di ogni byte di dati trasmesso. Il bit di parità viene calcolato in modo da garantire che il numero totale di bit di valore 1 nel byte di dati, compreso il bit di parità, sia pari o dispari.

Durante la ricezione, il destinatario calcola il bit di parità del byte di dati ricevuto e lo confronta con il bit di parità trasmesso. Se i due bit di parità non corrispondono, significa che si è verificato un errore di trasmissione nel byte di dati. In questo caso, il destinatario può richiedere una ritrasmissione del byte di dati o adottare altre tecniche di correzione degli errori.

Il controllo di parità è una tecnica relativamente semplice e **facile da implementare**, ma ha alcuni limiti. In particolare, il controllo di parità può individuare solo gli errori a singolo bit. Se ci sono più di due bit errati nel byte di dati trasmesso, il controllo di parità non sarà in grado di individuare l'errore.

- **Parità a due dimensioni**

Il controllo di parità può essere utilizzato anche per controllare sequenze di bit in matrici. In questo caso, vengono utilizzati un controllo di parità in riga e uno in colonna per individuare e correggere eventuali errori nella matrice.

Durante la ricezione, il destinatario calcola sia il bit di parità della riga che quello della colonna del byte di dati ricevuto e li confronta con i bit di parità trasmessi. Se uno dei due bit di parità non corrisponde, significa che si è verificato un errore nella riga o nella colonna del byte di dati. Il destinatario può quindi individuare la posizione esatta dell'errore, che si trova all'intersezione della riga e della colonna errate.

Tuttavia la correzione degli errori si basa sulla **probabilità che l'errore sia solo uno**, pertanto si necessita di massimizzare la probabilità che ci sia un solo errore.

- **CRC (codice di ridondanza ciclica)**

$$\rightarrow R(x) = (x^r) * M(x) \bmod G(x)$$

La tecnica CRC prevede la generazione di un polinomio di grado **n**, dove **n** è il numero di bit della sequenza di dati. Si prende il polinomio **M(x)** ottenuto dalla sequenza **m** da trasmettere che presenta una ridondanza di **r** bit. Si prende il resto della divisione tra il polinomio **M(x)** ed un generatore **G(x)**, noto a mittente e destinatario, e lo si moltiplica per un polinomio di grado **r**.

Durante la trasmissione, il generatore del CRC viene trasmesso insieme ai dati. Il destinatario riceve i dati e il valore di controllo CRC e utilizza lo stesso polinomio generato dal mittente per calcolare il valore di controllo CRC sui dati ricevuti. Se il valore di controllo CRC calcolato dal destinatario corrisponde al valore trasmesso dal mittente, allora i dati sono stati trasmessi correttamente. In caso contrario, significa che si è verificato un errore di trasmissione.

Il generatore **G(x)** deve avere alcune caratteristiche, ovvero deve avere il primo e l'ultimo bit posti ad 1 e non deve essere ottenibile come prodotto di altri polinomi.

Il CRC è un controllo semplice da implementare ed efficace dal punto di vista della rilevazione.

Distanza di Hamming

Date due codeword è possibile definire una distanza formale in termini di differenze tra i singoli bit. Se le due codeword sono a distanza 2 è necessario cambiare due bit per passare dall'una all'altra.

Il canale di trasmissione fisica "rovina il messaggio", ovvero introduce errori di trasmissione nella sequenza inviata. Normalmente in un byte di dati è più probabile che sia un solo bit ad essere modificato rispetto a 2 o più bit.

Il controllo di parità rileva gli errori in numero dispari, ovvero se c'è un errore in un solo bit lo rileva ma non su due, ed è poco probabile che ci siano 2 o più errori, per tanto funziona abbastanza bene.

Keyword con 2 bit per i dati e 1 per la parità (parità pari sul bit 1)

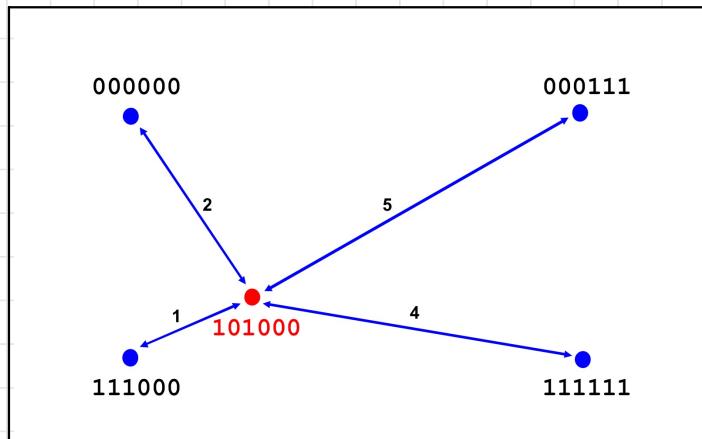
000	001
011	010
101	100
110	111

distanza del
vocabolario = 2

	0	3	5	6	$P(e=1) \gg P(e>1)$
0	0	2	2	2	
3	2	0	2	2	$\Rightarrow \text{minim} = 2$
5	2	2	0	2	
6	2	2	2	0	

Il trasmittitore invia di certo l'informazione corretta (combinazioni di sinistra, compreso bit di parità in verde), tuttavia il ricevitore potrebbe ricevere informazioni errate (combinazioni in rosso) per via di errori nel canale di trasmissione. Il destinatario conosce le combinazioni errate e le scarta in caso di ricezione.

Nella ricezione della risposta si traccia la possibile sorgente dell'errore valutando, tra quelle possibili, quale di queste ha distanza minore da quella ricevuta, in quanto il canale è probabile che abbia cambiato meno bit possibile. Il mittente stesso corregge l'errore sulla base della sequenza più vicina.



Il vocabolario con controllo di parità ha sempre distanza = 2.

La regola generale dice che :

- Per rilevare e errori è necessario un vocabolario con distanza $e + 1$.
- Per correggere e errori è necessario un vocabolario a distanza $2e + 1$

Ridondanza

La ridondanza è funzione dei dati già noti e non presenta informazioni aggiuntive rispetto a quelle già note.
Avendo m bit di dati, quanti bit r di ridondanza devo aggiungere per **rilevare** gli errori singoli? Serve aggiungere un solo bit di ridondanza, quello di parità.
Bisogna poter calcolare quanti sono gli r bit di ridondanza da introdurre per **correggere** gli errori singoli e la formula per il calcolo di r è funzione di m . Perchè possa correggere errori singoli server un dizionario a distanza $d=3$.

Consideriamo un vocabolario con m bit dati e r bit di controllo

Poniamo $m+r = n$

Le combinazioni possibili (con n bit) sono 2^n , di cui solo 2^m sono valide

Consideriamo una codeword valida: **10001**

Cambiando **un solo bit per volta**, possiamo scrivere altre n codeword, tutte non valide:

11001

Se $d=3$, allora $(n+1)2^m \leq 2^n$

Semplificando:

Combinazioni errate ottenibili cambiando uno alla volta i bit nella sequenza per ogni codeword corretta(compresa)

$$(m+r+1)2^m \leq 2^{m+r}$$

$$(m+r+1) \leq 2^r$$

$$m+1 \leq 2^r - r$$

Nota: L'espressione $m+1 \leq 2^r - r$ è valida **solo** per $d=3$.

Esempi:

$$m=8 \Rightarrow r=4 \quad (8+4+1) \leq 2^4 \quad 13 < 16$$

$$m=11 \Rightarrow r=4 \quad (11+4+1) \leq 2^4 \quad 16 = 16$$

Codici di hamming

Quando passo da un valore di n al successivo introduco un bit di parità (controllo). Tra i valori di n mancano le potenze del due e in corrispondenza di queste inserisco il bit di controllo ottenuto dal XOR tra i bit dei dati, in particolare si fa lo XOR delle sole posizioni per le quali il bit di parità figura nella espansione binaria dei bit dei dati.

Esempio:	
Dati originali	10010001100
	$\begin{matrix} & 10010001100 \\ xx1x001x0001100 \end{matrix}$
	$\begin{matrix} \uparrow & \uparrow & \uparrow & \uparrow \\ 1 & 2 & 4 & 8 \end{matrix}$
	$3 = 1+2$
	$5 = 1 + 4$
	$6 = 2+4$
	$7 = 1+2+4$
	$9 = 1 + 8$
	$10 = 2 + 8$
	$b_1 = 3 \otimes 5 \otimes 7 \otimes 9 \otimes 11 \otimes 13 \otimes 15$
	$b_2 = 3 \otimes 6 \otimes 7 \otimes 10 \otimes 11 \otimes 14 \otimes 15$
	$b_4 = 5 \otimes 6 \otimes 7 \otimes 12 \otimes 13 \otimes 14 \otimes 15$
	$b_8 = 9 \otimes 10 \otimes 11 \otimes 12 \otimes 13 \otimes 14 \otimes 15$
	$11 = 1+2 + 8$
	$12 = 4+8$
	$13 = 1 + 4+8$
	$14 = 2+4+8$
	$15 = 1+2+4+8$

Al verificarsi di un errore nella trasmissione della sequenza vengono a cambiare i valori ottenuti dagli XOR nei bit di controllo e quindi si riesce facilmente ad individuare la posizione esatta del bit in cui si è verificato l'errore.
Se la posizione modificata è una potenza del due allora si ha un errore nella trasmissione del bit di controllo e si può facilmente implementare la correzione di un bit di controllo attraverso un contatore.

Accesso al canale trasmissivo

Un'altro dei compiti del DLL è gestire l'**accesso al canale di comunicazione**. Se il canale è punto-punto bidirezionale, non serve neanche un protocollo. Nel caso di comunicazione broadcast si deve cominciare a pensare come gestire l'accesso al canale. Il DLL in questo caso si divide in due parti : LLC (Logical Link Control) e MAC (Media Access Control). Il livello LLC si occupa della gestione dei protocolli di comunicazione e della gestione degli errori, mentre il livello MAC gestisce l'accesso al mezzo trasmisivo.

Problemi :

- Collisione: si verifica quando due o più nodi tentano di trasmettere contemporaneamente nello stesso canale di comunicazione, causando un conflitto e la perdita di dati.

TDMA (time division multiple acces) : il tempo viene suddiviso in slot temporali di durata fissa e assegnati ai vari nodi per la trasmissione dei dati. Ogni nodo può occupare uno o più slot di tempo, a seconda delle necessità di comunicazione. Il sistema di gestione del canale, stabilisce gli slot temporali e li assegna ai vari nodi. Una volta stabiliti gli slot, essi rimangono assegnati ai rispettivi proprietari e non cambiano a meno che non vengano apportate modifiche al sistema di gestione del canale.

FDMA (frequency division multiple acces) : le frequenze vengono suddivise in canali e assegnati ai vari nodi per la trasmissione dei dati. Ogni nodo occupa un canale di frequenza specifico e non lo condivide con altri nodi. In altre parole, ogni nodo utilizza lo slot di frequenza assegnatogli per tutto il tempo, ma solamente il suo slot di frequenza. La suddivisione delle frequenze in canali e l'assegnazione dei canali ai nodi sono stabiliti dal sistema di gestione del canale, che cerca di minimizzare le interferenze tra i nodi.

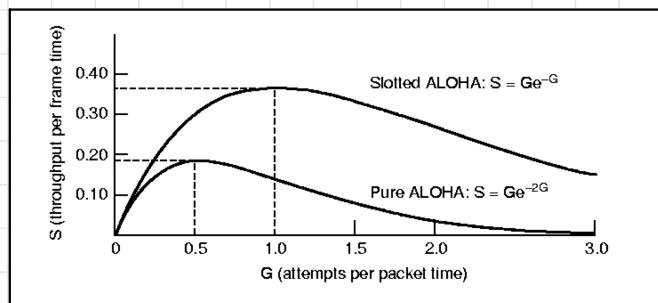
CDMA (code division multiple acces) : ogni nodo utilizza un codice unico per codificare i propri dati prima di trasmetterli sul canale, e ogni ricevente applica il codice corrispondente per estrarre i dati originali dal segnale. Per evitare interferenze tra le comunicazioni, i codici devono essere tra loro indipendenti, ovvero non correlati tra di loro.

ALOHA

La regola è che parla quando vuole, se deve comunicare trasmette, altrimenti no. Si usa un'antenna per trasmettere e una ricevente nelle stazioni periferiche, con un'antenna centrale che fungeva da ripetitore. Quando un ente vuole trasmettere la sua frame può farlo : la frame arrivava all'antenna centrale che la ripete su un determinato canale. In caso di collisione viene trasmesso il segnale rovinato, altrimenti il segnale arriva correttamente a destinazione.

La situazione ideale è data dall'assenza di perdita per qualunque richiesta di traffico che non superi la soglia di traffico massimo. All'atto pratico si hanno delle perdite anche al di sotto di quella soglia in quanto il traffico richiesto (G) supera quello massimo consentito (S) .

Il canale viene utilizzato al più per il 18.4% delle sue potenzialità quando $G = 0.5$, infatti $S = 0.184$.



Non si può evitare la collisione quindi si sfrutta il fatto che statisticamente pochi utenti hanno necessità di iniziare la trasmissione contemporaneamente e quindi si ha bassa probabilità di sovrapposizione di slot. Più si richiede al canale e più si abbassa il throughput ottenibile.

Una macchina poteva rilevare se una frame era stata rovinata prima ancora di ricevere riscontro dai livelli superiori, semplicemente ascoltando il canale di ritorno. Se sul canale di ritorno riesce a sentire la sua frame vuol dire che non è stata rovinata, se sente rumore invece il trasmettitore deve ritrasmettere il messaggio.

Quindi nonostante non fosse definito in maniera esplicita, di fatto c'è un **controllo di collisione**, proprio perché i canali di andata e ritorno erano separati.

Carrier sense

Prima di trasmettere, un dispositivo utilizza il Carrier Sense per ascoltare il canale e rilevare se c'è già una trasmissione in corso. Se rileva una portante di modulazione o un'altra forma di segnale di trasmissione, il dispositivo rimane in silenzio e attende il momento appropriato per trasmettere. Questo aiuta a ridurre le collisioni tra le trasmissioni

CSMA 1-persistent

Un dispositivo tenta di trasmettere immediatamente dopo aver rilevato la fine di una trasmissione precedente. Se più dispositivi utilizzano la stessa strategia e rilevano la fine della trasmissione contemporaneamente, potrebbe verificarsi una collisione. In tal caso, i dispositivi devono attenderne la risoluzione e ritentare in seguito.

CSMA p-persistent

Un dispositivo rileva la fine di una trasmissione precedente, genera un numero casuale e se il numero generato è inferiore o uguale a una certa probabilità p , il dispositivo inizia la trasmissione. Questa strategia offre la possibilità di ottenere un throughput quasi massimo per un dispositivo fortunato, in quanto il resto del traffico è stato tagliato via dal meccanismo di probabilità.

CSMA non-persistent

Un dispositivo aspetta un periodo di tempo casuale dopo aver rilevato la fine di una trasmissione precedente prima di tentare la trasmissione. Questo aiuta a evitare la sincronizzazione tra i dispositivi e riduce la probabilità di collisioni ripetute. Tuttavia, l'attesa casuale può causare un leggero ritardo nella trasmissione dei dati.

Aloha e CSMA non risolvono il problema delle collisioni.

CSMA/CD (carrier sense multiple access with collision detection)

Si ha un numero di slot pari al numero di stazioni (numerate) che possono parlare contemporaneamente nel mezzo.

Si invia un segnale di beacon agli utenti che vogliono comunicare ed essi comunicano la propria volontà di voler comunicare settando un apposito bit e si saprà esattamente chi è interessato a parlare e l'ordine di prenotazione.

Terminato il periodo di contesa, tutte le macchine sanno chi vuole parlare, avendo ascoltato il canale, e allora trasmetteranno in ordine. Non garantisce la totale assenza di collisione, ci può ancora essere perché qualcuno potrebbe non rispettare il protocollo.

Problemi con CSMA/CD :

- Se ci sono tante macchine il periodo di contesa diventa inutilmente lungo ed è uno schema utile se tutte le macchine presenti tendono a comunicare. In caso contrario il canale rimane inutilizzato per gran parte del tempo.
- Deve essere noto il numero di macchine presenti nel sistema e non si possono aggiungere/togliere macchine a piacimento. L'aggiunta o rimozione di slot deve essere sincronizzata con le altre macchine.

CSMA/BA

Gli utenti comunicano utilizzando segnali a due livelli: uno segnale alto per trasmettere il bit 1 e il silenzio per trasmettere il bit 0. Se due macchine dicono contemporaneamente "1", il segnale non va in collisione e il risultato è un bit 1. Se una macchina dice "0" e l'altra dice "1", prevale il bit 1 e lo 0 scompare senza collisioni.

Bit time	0 1 2 3
0 0 1 0	0 - - -
0 1 0 0	0 - - -
1 0 0 1	1 0 0 -
1 0 1 0	1 0 1 0
Result	1 0 1 0

Alla fine comunica la macchina con il MAC address più alto ed inoltre alla fine si ha già nel canale il MAC del mittente e non si spreca lo spazio per slot di contesa. Il problema è dato dal fatto che i MAC address più alti saranno avvantaggiati nella comunicazione, ma è un modo per garantire la priorità a delle macchine privilegiate.

Token bus

Si da un messaggio (token) ad una macchina e chi ha il token occupa il canale, al termine passa il token alla macchina successiva. Il token è una frame e potrebbe collidere con altre frame o anche perdersi a causa di interferenze. Se si perde il token non finisce tutto. Se dopo un certo tempo tutte notano che il token non sta girando allora tutte sono autorizzate a generare un nuovo token, nella speranza che venga generato un solo token alla volta.

Se ci sono due token si deve prevedere un sottoprotocollo per la rimozione del superfluo. Poi una macchina deve poter entrare/uscire dall'anello. Uscire è facile, ma come si fa per entrare? Un sottoprotocollo deve poter gestire questi casi. Questo approccio evita le collisioni ma è di difficile implementazione.

Ethernet RFC->802.3 (CSMA/CD)

Il protocollo CSMA/CD è stato sviluppato per gestire i canali di trasmissione condivisi, come quelli utilizzati nell'Ethernet, in cui più dispositivi competono per l'accesso al canale. La combinazione di "carrier sense" e "collision detection" consente di ottimizzare l'efficienza della trasmissione e di gestire efficacemente le collisioni quando si verificano.

Per determinare l'intervallo di attesa prima di un nuovo tentativo, Ethernet utilizza un algoritmo di **backoff esponenziale**. Se un dispositivo non riesce a trasmettere correttamente, ha la possibilità di ritentare la trasmissione fino a un certo numero di tentativi, solitamente 10. Il dispositivo seleziona casualmente un numero nell'intervallo da 0 a un valore massimo specificato, che cresce con ogni tentativo.

Se si verifica una collisione, il dispositivo selezionerà casualmente un tempo di attesa nell'intervallo da 0 a 10 secondi per il secondo tentativo. Nel caso di ulteriori collisioni, il dispositivo selezionerà casualmente un tempo di attesa nell'intervallo da 0 a 20 secondi per il terzo tentativo, e così via. Il valore massimo dell'intervallo di attesa cresce in modo esponenziale, fino a un massimo di 1023. In questo modo ogni dispositivo che ha subito una collisione aspetta un tempo casuale prima di ritentare la trasmissione. Ciò aumenta le possibilità che un dispositivo trovi il canale libero e possa trasmettere senza collisioni.

Codifica Manchester

Ci sono due livelli, alto basso, il bit 1 è identificato da un passaggio alto-basso, mentre il bit 0 da un passaggio basso-alto. Si identificano i bit con la **variazione dello stato logico** e non con il livello perché è più facile identificare il passaggio da uno stato logico all'altro più che un livello fermo. Si necessita di inviare un **preamble** per sincronizzare mittente e destinatario, ovvero una sequenza di bit terminata da **11**, dove **11** è un'onda quadra di frequenza 20 MHz, questo permette al destinatario di sincronizzarsi col mittente.

Ethernet 10Mbps

Lo standard Ethernet 802.3 sfrutta la codifica machester

Name	Cable	Max. seg.	Nodes/seg.	Advantages
10Base5	Thick coax	500 m	100	Original cable; now obsolete
10Base2	Thin coax	185 m	30	No hub needed
10Base-T	Twisted pair	100 m	1024	Cheapest system
10Base-F	Fiber optics	2000 m	1024	Best between buildings

- Il primo è il cavo coassiale grosso **10Base5**, molto rigido perché era un pezzo di rame unico non intrecciato, una guaina isolante e una calza metallica di protezione. Ovviamente poi c'è la guaina esterna di protezione totale. La lunghezza massima per un segmento unico era di 500 metri, e potevano essere collegati tra di loro con dei ripetitori / amplificatori di segnale fino a un massimo di 5 segmenti consecutivi, quindi 2.5km.
- Per quanto riguarda **10Base2** è molto più flessibile e per estendere il cavo semplicemente si tagliava (cosa che rovina il segnale) e si inseriva il connettore. Non avere più il cavo unico è un problema e quindi ha segmenti massimi di 185 metri, sempre 5 segmenti massimo fra di loro, quindi arriva a stento a 1km.

Il cavo coassiale funziona meglio di quello intrecciato ma bisogna inserire un terminatore nel cavo , in assenza del quale la comunicazione non può avvenire .

- 10Base-T** (doppino) è un cavo a coppie intrecciate, ha 4 coppie di cavi intrecciati, i quali vengono intrecciati a loro volta. Questi cavi permettono comunicazione massimo 100 m con più nodi, questo perché lo schema di collegamento prevedeva un hub centrale, un concentratore, e la connessione era diretta tra scheda di rete e concentratore. Possiamo vederlo anche come un centro a stella.
- L'ultimo schema è **10Base-F**, la fibra ottica (la velocità è sempre la stessa). Permette una velocità di 10Mbit al secondo , prestazioni eccezionali e permette collegamenti in lunga distanza fino a 2km senza ripetitori.

Categorie dei cavi :

Trasmissione a Cavi

Category	Data Rate	Signal Frequency	Standard
Cat5	100 Mbps	100 MHz	TIA/EIA
Cat5e	100 Mbps / 1 Gbps	100 MHz	TIA/EIA-568-B
Cat6	1Gbps / 10 Gbps	250 MHz	TIA/EIA-568-B
Cat6a	1Gbps / 10 Gbps	500 MHz	ANSI/TIA/EIA-568-B.2-10

Fast ethernet

Quando ethernet è passato a 100Mbps sono scomparsi i cavi coassiali e sono comparsi i cavi a categoria. La codifica utilizzata è 4B5B.

Più sono regolari e intrecciate le coppie di cavi e più si ha una alta categoria .

Nel caso di cavo categoria 3 (T4) so tutte e 4 le coppie presenti nel cavo, una in andata, una in ritorno, alla velocità di 33 Mbps ciascuna, le ultime due coppie le utilizzo alternativamente (insieme) o in una direzione o nell'altra, in modo tale che ho 3x33Mbps in una direzione e 33 nell'altra. Quindi Fast Ethernet permette di ottenere circa **100Mbps Half Duplex** in una direzione e 33 nell'altra. In alternativa, nel caso di cavo categoria 5 (TX), le altre due coppie sono utilizzate per trasportare corrente continua **POE (power over ethernet)** per iniettare energia elettrica nel dispositivo. Non serve la codifica manchester e nemmeno il cavo coassiale.

Name	Cable	Max. segment	Advantages
100Base-T4	Twisted pair	100 m	Uses category 3 UTP
100Base-TX	Twisted pair	100 m	Full duplex at 100 Mbps
100Base-FX	Fiber optics	2000 m	Full duplex at 100 Mbps; long runs

Gigabit ethernet

Quando Ethernet passa alla velocità trasmissiva di 1000Mbps i cavi sono solo twisted (doppini) ed in fibra ottica. Cinque passi verso 1000Base-T (Cat5) :

- Rimuovere codifica 4B5B (100 -> 125 Mbps) : già solo rimuovendo questa codifica aumentiamo il throughput, ma ritorna il problema del framing.
- **Usare le 4 coppie simultaneamente (125 -> 500 Mbps)** : Fast Ethernet ne usa solo 2, una di andata e una di ritorno. Usando tutte e quattro le coppie simultaneamente possiamo quadruplicarlo.
- **Trasmissione full duplex (500 Mbps full duplex)** : conseguenza dell'uso delle 4 coppie simultanee.
- **Usare 5 livelli per baud invece che 3 (MLT-3) (500 Mbps -> 1Gbps full duplex)** : un livello viene usato per fare framing, gli altri 4 permettono di trasportare 2 bit al livello. Da 500 passiamo a 1 GBps. L'unico problema è che il tasso di errore è così alto che ogni frame sarebbe soggetto a errore.
- **Usare Forward Error Correction per recuperare 6 dB** : correzione errori a destinazione per sistemare il problema.

Name	Cable	Max. segment	Advantages
1000Base-SX	Fiber optics	550 m	Multimode fiber (50, 62.5 microns)
1000Base-LX	Fiber optics	5000 m	Single (10 μ) or multimode (50, 62.5 μ)
1000Base-CX	2 Pairs of STP	25 m	Shielded twisted pair
1000Base-T	4 Pairs of UTP	100 m	Standard category 5 UTP

Tecnologia	Massima lunghezza del link	Codifica	Topologia del mezzo		Bit rate (bps)
10Base5	500 m	Manchester	bus	50-ohm coax	10 M
10Base2	185 m	Manchester	bus	50-ohm coax	10 M
10BaseT	100 m	Manchester	star	2 pair UTP cat. 3,4,5	10
100BaseFL 100BaseT2	2000 m 100 m	Manchester PAM 5x5	star star	Multi-mode fiber* 2 pairs UTP cat. 3,4,5	10 M 100 M
100BaseT4	100 m	8B/6T	star	4 pairs UTP cat. 3,4,5	100 M
100BaseTX	100 m	4B/5B with MLT-3	star	2 pairs UTP cat. 5	100 M
100BaseFX 1000BaseT	412/2000 m 100 m	4B/5B with NRZI PAM 5x5	star star	Multi-mode fiber* 4 pairs UTP Cat 5	100 M 1000 M
1000BaseSX	275 m	8B/10B	star	Multi-mode fiber†	1000 M
1000BaseLX	316/550 m	8B/10B	star	Multi-modeFiber‡	1000 M
1000BaseCX	25 m	8B/10B	star	Twinax	1000 M

Codifica di Travis :

Si ha un automa a stati finiti che predefinisce un percorso univoco per ogni sequenza possibile di distanza 2 e si ricostruisce facilmente il messaggio di partenza basandosi sulla probabilità. Questo meccanismo è in grado di rilevare e correggere anche errori doppi per ogni byte (un tasso di errore enorme di un bit ogni 4).

A questa codifica segue un controllo CRC per garantire la correttezza dei dati. Questi metodi combinati forniscono una maggiore affidabilità nella trasmissione dei dati e la possibilità di rilevare e correggere una grossa tasso di errori.

Lo standard Ethernet 10Mbps prevede un **minimo di 64 byte** come dimensione della frame, ovvero payload ed header. Si necessita di avere una frame più lunga del periodo critico in quanto se il messaggio trasmesso viene rovinato il mittente deve accorgersene in tempo utile, ovvero prima che termini la trasmissione. Il tempo critico è di 51,2 microsecondi, questo tempo permette di trasmettere 512 bit a 10Mbps, ovvero 64 byte, ed ecco perchè il payload minimo in caso di collisione è di 64byte.

La lunghezza massima della frame ethernet è di 1500 byte, data dal payload. Il campo **E-type** nella frame indica o la quantità di dati effettivi all'interno del payload (se il valore è al di sotto di 1500) oppure (se il valore è sopra 1500) indica il tipo di protocollo che c'è nel payload (IPv4, IPv6).

Bridge

In informatica, un bridge è un dispositivo di rete che connette due segmenti di rete che utilizzano tecnologie di rete diverse. Il bridge opera a livello 2 (data link layer) del modello OSI e ha il compito di interconnettere due reti locali (LAN) utilizzando protocolli differenti a livello fisico e a livello di collegamento dati.

Il bridge interpreta i bit che arrivano da una rete e li trasmette all'altra rete, in modo che i dispositivi delle due reti possano comunicare tra loro. Se l'indirizzo di destinazione si trova sulla stessa rete del mittente, il bridge non fa nulla e il pacchetto viene trasmesso nella stessa rete. Se l'indirizzo di destinazione si trova su un'altra rete, il bridge inoltra il pacchetto sulla rete corretta.

A livello Ethernet il bridge si chiama **switch**. Un hub a 10Mbit lavora a 10Mbit sempre, uno switch a 10Mbit permette di avere 10Mbit su tutte le inteface quindi nel caso di due connessioni verso la stessa macchina contemporaneamente il traffico permesso è di 20Mbit (larghezza di banda elevata).

Ogni porta dello switch è una scheda di rete e dentro ha un processore che provvede a sistemare i vari collegamenti. Si definisce LAN tutto ciò che è indirizzabile a livello DLL, in pratica il router separa LAN.

Gli switch sono **plug&play**, ovvero non necessitano di configurazione ma capiscono in automatico come reindirizzare i pacchetti alle destinazioni

Il bridge utilizza una tabella di indirizzi MAC per memorizzare gli indirizzi fisici dei dispositivi connessi alle due reti. Quando un pacchetto viene ricevuto da una delle reti, il bridge controlla la sua destinazione confrontandola con la sua tabella di indirizzi MAC.

Affinchè la rete funzioni la topologia deve essere un albero e non un grafo, in quanto il flooding inonderebbe la rete, si usa STP per bloccare il flooding.

HUB vs Bridge

L'hub è un dispositivo di rete che connette diversi dispositivi di una rete locale, ma ripete tutti i segnali ricevuti a tutti i dispositivi connessi, creando traffico di rete inutile e riducendo le prestazioni della rete. Il bridge, d'altra parte, connette anche diversi dispositivi di una rete locale, ma inoltra i pacchetti solo ai dispositivi destinatari, riducendo il traffico di rete e offrendo prestazioni migliori rispetto all'hub.

VLAN (virtual LAN)

Quando si inizia la configurazione gli switch lavorano come hub, ovvero broadcast. Sarebbe utile suddividere la rete secondo necessità. Si pensa di suddividere la LAN in parti logicamente separate. Si suddividono le porte di uno switch in gruppi e si assegnano a LAN differenti. I dispositivi su reti differenti potranno comunicare solo tramite router. Il vantaggio in questo tipo di approccio è dato dai costi (un solo switch) e dalla facilità di manutenzione. Per supportare questo tipo di configurazione nella frame ethernet si necessita di specificare il formato **802.1Q** che aggiunge 4 byte al peso della frame, specificando il tag della vlan di appartenenza.