Introduzione

Le reti di computer sono l'espressione principe dell'informatica distribuita, che comporta come vantaggi riduzione dei costi, flessibilità, scalabilità ed espansibilità, distribuzione del lavoro da svolgere, maggiore portabilità e riusabilità delle applicazioni e forse anche maggiore affidabilità. Si definiscono in maniera ricorsiva come reti di computer e reti di reti, e sono basate sul layering, ossia astrazione e suddivisione della complessità in più livelli adiacenti ma indipendenti tra loro, ognuno dei quali offre servizi al superiore e riceve servizi dal sottostante (tramite opportune interfacce). La comunicazione avviene grazie a protocolli, ossia insiemi di regole e convenzioni seguite dai computer che intendono comunicare tra loro (protocolli formati da una sintassi, una semantica e dalle conseguenze della temporizzazione). I protocolli sono raggruppati in blocchi (protocol stack), e ogni livello è caratterizzato da una service interface (interfaccia col livello superiore) e da una p2p interface (interfaccia col pari livello nel calcolatore di destinazione). Mentre la comunicazione concettuale avviene quindi tra peer entity (tra sorgente e destinazione), la comunicazione effettiva è indiretta: il messaggio proveniente dal livello superiore della sorgente deve passare tutti i livelli sottostanti, incapsulato con i rispettivi header, passare il mezzo fisico, per poi essere disassemblato seguendo il procedimento inverso nella destinazione. A ciascun livello, il messaggio si compone di un PCI (Protocol Control Information, ossia un header) e da un SDU (Service Data Unit, ossia l'informazione), per formare il PDU (Protocol Data Unit).

Protocolli di comunicazione

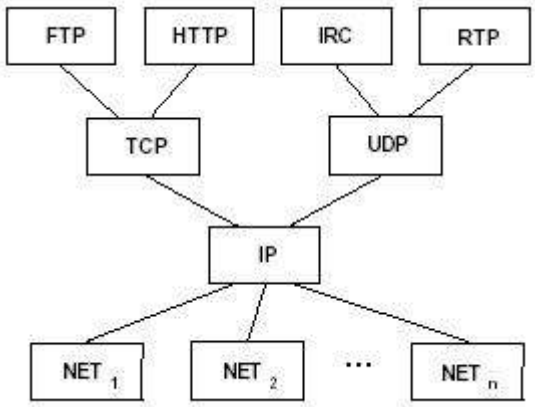
Il protocollo di comunicazione de iure è stato l'**ISO/OSI**:

1. ***livello fisico***: gestisce i particolari meccanici ed elettrici della trasmissione fisica di un flusso di bit
2. ***livello di collegamento dati***: gestisce il formato del messaggio (suddivisione in pacchetti, controllo e correzione di errore e della giusta sequenza)
3. ***livello di rete***: gestisce l'instradamento dei pacchetti (indirizzo, informazioni di instradamento ecc…)
4. ***livello di trasporto***: controllo end-to-end della sessione di comunicazione
5. ***livello di sessione***: consente a utenti su macchine diverse di stabilire sessioni, implementando funzioni di coordinamento, sincronizzazione e mantenimento dello stato di sessione
6. ***livello di presentazione***: risolve le differenze di formato delle informazioni, ma anche compressione e sicurezza dei dati
7. ***livello di applicazione***: interfaccia standard per gli applicativi che utilizzano la rete (mascherano la complessità sottostante)

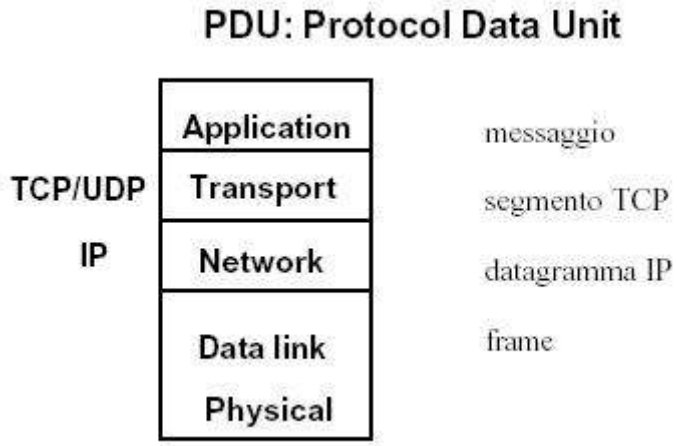
Il protocollo di comunicazione de facto è invece il **TCP/IP**:

1. ***livello host2network*** : livello fisico e data link strettamente dipendenti (ad es., Ethernet per le lan o PPP per le connessioni via modem)
2. ***livello internet***: caratterizzato dal protocollo **IP**, un protocollo per la consegna dei pacchetti da mittente a destinatario; comprende la possibilità di identificare univocamente ciascun host (indirizzo IP) e la comunicazione logica tra host, ma è privo di connessione (ogni pacchetto è mandato in maniera indipendente dagli altri), non è affidabile (consegna e ordine di consegna non garantiti) e presenta la consegna con impegno (tenta comunque di consegnare ogni pacchetto).

3. ***livello di trasporto***: estende la consegna con impegno dell'IP tra due host terminali ad un servizio di consegna a due processi applicativi in esecuzione sugli host; in aggiunta all'IP, presenta la possibilità di multiplazione e demultiplazione dei messaggi e il rilevamento dell'errore (checksum). I protocolli del livello di trasporto sono l'**UDP** (User Datagram protocol) e il **TCP** (Transmission Control Protocol)
    I. ***UDP*** : fornisce un livello di trasporto dell'informazione connectionless
    II. ***TCP*** : fornisce un livello di trasporto affidabile, orientato alla connessione (instaurazione, utilizzo e chiusura della connessione), orientato al flusso di dati (considera interamente il flusso di dati dall'host mittente fino al destinatario), caratterizzato dal trasferimento con buffer (bufferizzazione e spedizione quando il buffer è pieno) e da una connessione full-duplex (bidirezionale e contemporaneo)
4. ***livello applicativo***: utilizza il livello di trasporto dell'informazione tra processi in esecuzione su host terminali per realizzare applicazioni di rete (ftp, telnet, http, smtp, irc ecc…)
   NB: applicazioni di rete ≠ protocolli applicativi



Comunicazione in internet



Internet è una rete di router interconnessi mediante una topologia non regolare. Le tipologie di trasferimento sono ***circuit-switching*** e ***packet-switching***.
Nel primo caso, esiste un circuito virtuale dedicato per ogni comunicazione (sistema telefonico). Ha quindi la necessità di riservare risorse end-to-end prima di trasmettere, non ha possibilità di condivisione (se una risorsa è utilizzata da una chiamata, nessuno la può utilizzare fino al suo termine), necessita di una fase di setup per ogni chiamata e ha prestazioni dipendenti dalle risorse riservate. Tali risorse sono suddivise in parti (Frequency Division Multiplexing o Time Division Multiplexing).

La tipologia usata da internet è invece la seconda: ogni comunicazione è suddivisa in pacchetti, che condividono perciò le risorse della rete e utilizzano tutta la capacità trasmissiva di un link; inoltre, le risorse sono utilizzate sulla base della necessità e non della prenotazione. Si usa il multiplexing deterministico (TDM o FDM), oppure quello statistico (TDM ma su richiesta anziché ad intervalli prefissati): i pacchetti vengono mischiati e bufferizzati, e poi mandati a destinazione. Possibilità di congestione: i pacchetti possono ritardare o essere persi, quindi serve un protocollo che gestisca il trasferimento affidabile dei dati e il controllo della congestione (appunto il TCP).

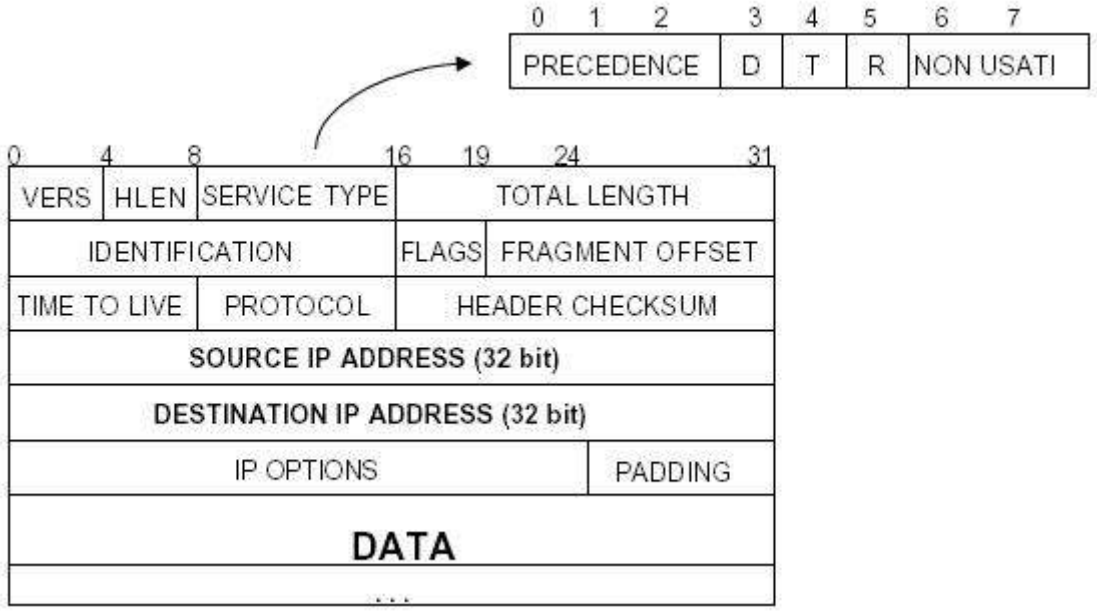
Tipologie di servizio di trasporto

I servizi affidabili utilizzano le primitive response/confirm per gestire la ritrasmissione dei messaggi in caso di errore, garantiscono la ricezione completa e corretta di tutti i messaggi, ma sono più lenti. Il TCP è progettato appunto per consentire affidabilità (trasferimento ordinato di uno stream di dati), controllo della congestione (diminuzione del tasso di trasmissione del mittente quando la rete è congestionata) e controllo di flusso (il mittente non deve sovraccaricare il ricevente).

Il livello IP

Permette *l'indirizzamento univoco degli host*, definisce l'unità di trasferimento dati di internet (*pacchetti fino a 64kb*), ha funzioni di *routing* (scelta del percorso nella rete) e ha una *consegna non affidabile* dei pacchetti (consegna *non garantita però con impegno, e privo di connessione*).
PS: X.25, Frame Relay e ATM: esempi di protocolli con servizio di consegna affidabile, che forniscono un "circuito virtuale"

Datagramma IP



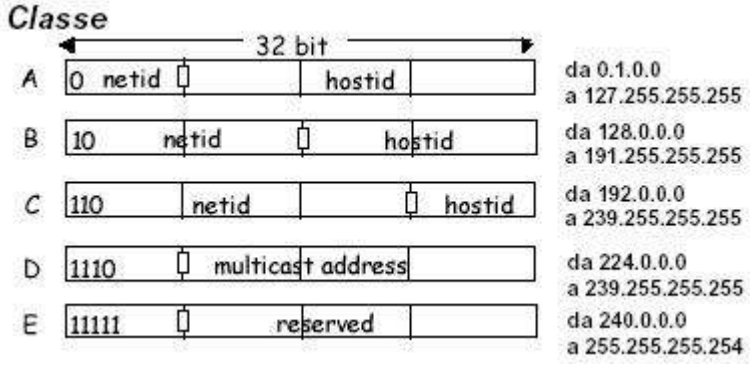
dove
VERS = versione del protocollo IP
HLEN = lunghezza dell'header
TYPE OF SERVICE = specifica come deve essere trattato il datagramma (tipo di trasporto)

PRECEDENCE = importanza
D = basso delay
T = alto throughput
R = alta affidabilità

TOTAL LENGTH = lunghezza dell'intero datagramma
IDENTIFICATION = intero che identifica il datagram
FLAGS = controllo della frammentazione
FRAGMENT OFFSET = la posizione del frammento nel datagram originale
TTL = fissa un limite di tempo entro il quale un pacchetto può restare nella rete (tempo di vita rimanente); ciascun router decrementa il TTL di una unità prima di instradare il pacchetto: in questo modo, il TTL può essere considerato il numero di hop che il pacchetto può fare prima di giungere a destinazione, pena l'essere scartato.
NB: i router controllano se TTL = 0 dopo il decremento, subito prima dell'inoltro; se il destinatario del pacchetto è il router stesso, questo non viene mai scartato
PROTOCOL = indica quale protocollo applicativo può utilizzare i dati contenuti nel datagram
HEADER CHECKSUM = serve per controllare l'integrità dei dati trasportati nell'header
SOURCE IP ADDRESS = IP del mittente, non viene mai modificato dai router
DESTINATION IP ADDRESS = IP del destinatario, non viene mai modificato dai router
IP OPTIONS = campo opzionale e variabile, serve per il testing e il debugging della rete
PADDING = campo opzionale che serve per fare in modo che l'header abbia lunghezza multipla di 32 bit (per il byte stuffing)

Indirizzo IP

Formato da una parte *netID* (prefisso che identifica la rete) e da una parte *hostID* (che identifica l'host di quella rete)



Tale indirizzo è univoco per ogni host collegato; può essere statico o assegnato dinamicamente, può essere impostato manualmente (dall'amministratore del sistema) o dinamicamente (dal server DHCP, Dynamic Host Configuration Protocol). *Tali indirizzo sono logici, non fisici*. In genere, il netID è assegnato alle organizzazioni da un'unica autorità centrale (IANA), gli hostID invece sono assegnati localmente dall'amministratore. Entrambi vengono utilizzati per il routing. Gli indirizzi IP vengono in realtà *assegnati ad una interfaccia di rete*, quindi un singolo host può averne più di uno.

Indirizzi IP speciali

- network address: 155.185.**0.0** (hostID a 0)
- directed broadcast address: 155.185.**255.255** (hostID a 1)
- limited broadcast address: **255.255.255.255**
- this host address: **0.0.0.0**
- loopback address: **127.0.0.1** (usato per il test di applicazioni di rete)

Subnetting
Divisione dello spazio di indirizzi IP in sottoreti, per raggruppare host basati sulla topologia fisica della rete, ma soprattutto per facilitare il routing dei pacchetti all'interno della rete

Indirizzi privati
Per consentire a macchine non appartenenti ad internet di scambiarsi pacchetti (intranet)
Classe A : 10.0.0.0 to 10.255.255.255
Classe B : 172.16.0.0 to 172.16.255.255
Classe C : 192.168.0.0 to 192.168.255.255

Architettura organizzativa di internet

Dal punto di vista organizzativo, internet è costituito gerarchicamente da ISP (Internet Service Provider) locali (ai cui POP si collegano gli utenti), a loro volta collegati a ISP regionali (nazionali). Questi ultimi sono collegati a ISP internazionali detti NBP (National Backbone Provider). Infine, questi NBP sono collegati tra loro mediante NAP (Network Access Point) o Peereing Point.
I NAP sono pubblici, costosi e (visto il traffico che devono saper sopportare) dotati di elementi di switching molto potenti e affidabili. I Peering Point invece sono gestiti da privati.

Bande di alcune tecnologie trasmissive

GSM	9.4-14.4 Kbps
POTS	fino a 56 Kbps
GPRS	56-114 Kbps
ISDN	64-128 Kbps
IDSL	128 Kbps
Satellite	382 Kbps
Frame relay	56 Kbps – 1.544 Mbps
T-1	1.544 Mbps
UMTS	fino a 2 Mbps
IBM Token Ring	4 – 16 Mbps
T-2	6.312 Mbps
DSL	512 Kbps – 8 Mbps
Modem via cavo	512 Kbps – 52 Mbps
Ethernet	10 Mbps
T-3	44.736 Mbps
OC-1 (*ottica*)	51.84 Mbps
Fast Ethernet	100 Mbps
FDDI	100 Mbps
OC-3	155.52 Mbps
OC-12	622.08 Mbps
Gigabit Ethernet	1 Gbps
OC-256	13.271 Gbps

NOTA: bande di picco teoriche

I router sono aggregati in regioni dette AS (Autonomous Systems), gestite da una stessa amministrazione. Tutti i router all'interno dello stesso AS usano lo stesso algoritmo di instradamento dei messaggi e hanno informazioni su tutti gli altri. Nessuna di questa rete gestisce più del 5% del traffico globale, anzi, la stragrande maggioranza ne gestisce molto meno dell'1%. Ogni AS ha un numero identificativo (da 1 a 65535). Usa un unico IGP (Interior Gateway Protocol) per indirizzare i pacchetti all'interno dell'AS (come ad esempio RIP e OSPF), e un EGP (Exterior Gateway Protocol) per indirizzare i pacchetti verso altri AS (ad esempio BGP). Ogni AS appare come un unico AS agli altri.

Protocollo ICMP

E’ un *protocollo di supporto* (porta dati di controllo, non offre direttamente servizi), usato da host, router e gateway per comunicare *informazioni riguardati il livello rete*: congestione e controllo di flusso, comunicazione periodica dei cambiamenti nelle tabelle di routing, determinazione di cammini circolari o eccessivamente lunghi, stima del tempo di trasmissione da mittente a destinatario, ma soprattutto controllo e notifica di situazioni di errore o anomalie; inoltre, supporta il debugging interattivo della rete. Il messaggio ICMP è *incapsulato in un datagramma IP* (è considerato protocollo di livello 3).

Regole

1. nessun messaggio ICMP viene generato a seguito ad eventuali errori rilevati su messaggi ICMP
2. se il pacchetto viene frammentato, solo il primo frammento può generare messaggi di errore ICMP
3. i broadcast e i multicast non generano ICMP

Messaggi di errore
TIME_EXCEEDED = pacchetto con TTL=0
DESTINATION_UNREACHABLE = un gateway vede la rete di destinazione a distanza infinita, oppure un host non risponde ad una ARP, oppure l’host non conosce il protocollo nel pacchetto, oppure il pacchetto non può essere frammentato
PARAMETER_PROBLEM = il gateway non riesce ad interpretare il pacchetto a causa di un valore errato, problema generalmente software
SOURCE_QUENCH = rallentamento della sorgente, host di destinazione lento

Messaggi di informazione:
ECHO_REQUEST e ECHO_REPLY = controllo di raggiungibilità di un host
TIMESTAMP e TIMESTAMP_REPLY = echo con informazioni su orario di invio
REDIRECT = il router inoltra il pacchetto specificando un percorso migliore per la sua destinazione

Ping
Invia una successione di ICMP echo_request e attendo la relativa risposta, misurando il tempo che intercorre tra l’invio e la ricezione e riportando semplici statistiche

Traceroute
Permette di capire per quali router passano i pacchetti IP quando sono diretti ad una data destinazione: manda un echo_request con valori di TTL bassi, se riceve un time_exceeded (da parte del router) allora rimanda il pacchetto incrementando il TTL, altrimenti se riceve un echo_reply (dalla destinazione) conclude che l’host è raggiungibile con n hop (visualizzandoli).
PS: manda 3 pacchetti per ogni TTL n-esimo, se non osserva un time_exceeded entro un tempo prestabilito, allora visualizza un “ * “.

Routing IP

Il routing è il meccanismo per la scelta del percorso in internet attraverso il quale inviare un datagramma. E’ effettuato dai router, per mezzo di alcune tabelle che memorizzano le informazioni per raggiungere le possibili reti di destinazione (il router usa solamente il netID dell’IP del destinatario). Si distingue in

- *routing diretto* (quando mittente e destinatario appartengono alla stessa rete fisica)
- *routing indiretto* (quando il mittente deve individuare un router a cui inviare il datagramma)

Si parla in genere di *next-hop routing*, in quanto il router possiede l'informazione sul salto successivo che il datagramma deve compiere per giungere a destinazione.

Caratteristiche

- routing statico: la tabella di routing non viene modificata dal router, ma solo dall'amministratore (non può reagire dinamicamente ai cambiamenti topologici)
- routing dinamico: la tabella di routing viene modificata dal router al variare delle condizioni della rete
- indipendenza dal mittente o dal cammino del datagramma fino a quel momento (il router estrae dal datagramma solo l'IP del destinatario
- routing universale: la tabella di routing deve avere un next-hop router per ogni destinazione
- routing ottimo: questo next-hop router deve essere quello che minimizza il cammino verso la destinazione
- routing di default: router comune a più indirizzi di destinazione, per ridurre la tabella

Funzionamento del router

1. estrae dell'IP destinatario e del rispettivo netID
2. se tale netID corrisponde ad una rete connessa direttamente, risolve dell'indirizzo fisico consegna il frame
3. altrimenti consulta la tabella e manda al next-hop router se ne ha uno per quel netID
4. altrimenti lo manda al next-hop router
5. altrimenti dichiara un errore

Algoritmi di routing

Determinano il percorso ottimale (costo minimo) da mittente a destinatario, considerando la rete un grafo. Si può distinguere tra cammino di costo minimo (minima somma dei costi dei link del cammino) o cammino minimo (minimo numero di archi dal mittente al destinatario).
Si suddividono in:

- algoritmi di routing globale: il cammino di costo minimo è calcolato avendo un'informazione globale sulla rete, il calcolo è centralizzato, e l'algoritmo deve conoscere lo stato (costo) di ogni link
  Link state algorithm (Dijkstra): costi degli link conosciuti, input dell'algoritmo (ogni nodo li trasmette agli adiacenti)
- algoritmi di routing distribuito: nessun nodo ha un'informazione completa, il calcolo è distribuito e interattivo, ogni nodo sa solo il peso dei link adiacenti
  Distance vector algorithm: il router tiene in una tabella tutti i percorsi di instradamento conosciuti, ognuno dei quali con la relativa distanza misurata in hop (numero di router da attraversare). Periodicamente, ogni router invia agli adiacenti una copia di questa tabella. Ricevuta la tabella, ogni router controlla: se trova una destinazione che non conosceva, o se trova un percorso più breve per una certa destinazione, o se alcune distanze conosciute sono cambiate, il router provvede ad aggiornare la propria tabella.

NB: entrambi sono algoritmi di routing dinamici

Indirizzamento di subnet e supernet

Per risolvere i problemi di esaurimento dello spazio di indirizzamento, si può modificare l'interpretazione degli indirizzi con il *subnet addressing*, utilizzando una maschera di subnet (i bit a 1 corrispondono alla parte di rete, quelli settati a 0 alla parte locale, AND bit a bit). Ne consegue il cosiddetto routing gerarchico: i router esterni usano solo il netID, mentre quello interno usa il netID della sottorete. In questo caso, dunque, le tabelle di instradamento sono nella forma

< maschera della sottorete, indirizzo di rete, indirizzo del next-hop router >

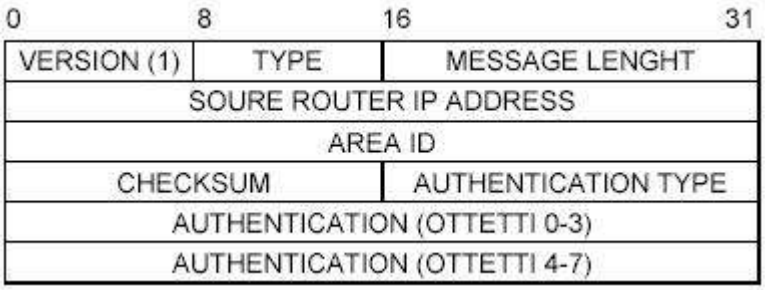
L'approccio opposto per risolvere lo stesso problema è invece il *supernet addressing*: una organizzazione utilizza più indirizzi di rete per la sua rete. Tale sistema aumenta però il numeri di ingressi nella tabella di routing: problema risolvibile con il meccanismo CIDR, che presenta due campi "network address" (il più piccolo indirizzo del blocco) e "count" (il numero di indirizzi contigui).

Routing AS

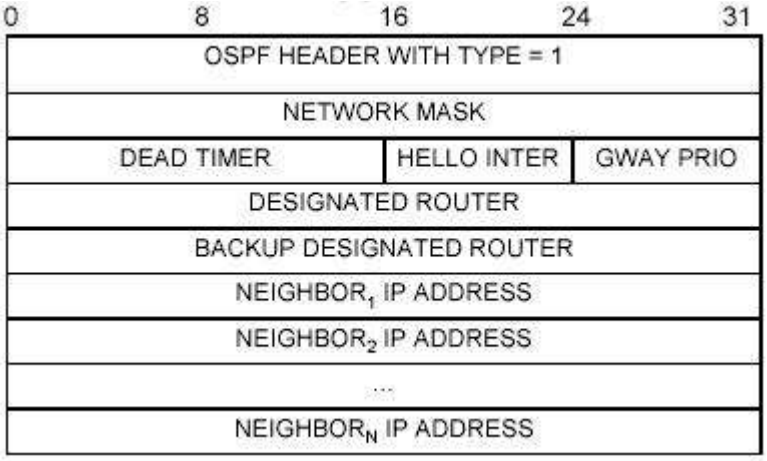
Come già detto, i router sono raggruppati in AS che utilizzano gli stessi protocolli di routing e sono gestiti dalla stessa amministrazione. Ogni router sa come instradare pacchetti verso il suo AS, ma non conosce nulla riguardo alla struttura interna di altri AS (per questo, i cammini sono mediamente più lunghi).

Protocolli intra-AS

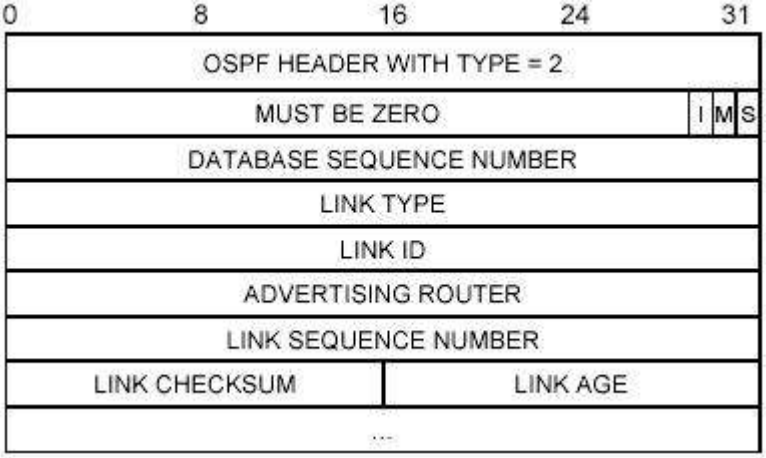
- **RIP** (Routing Information Protocol)
  Protocollo distance vector, a conteggio di hop, massimo costo di un percorso = 15, tabelle di instradamento scambiate ogni 30 secondi, dead time di un vicino settato a 180 secondi
- **OSPF** (Open Shortest Path First)
  Protocollo link-state che permette di specificare più instradamenti verso una specifica destinazione (se sono allo stesso costo, distribuisce il carico); gestisce inoltre l'autenticazione del router, le gerarchie all'interno del dominio di un singolo router, e supporta varie metriche.
  Ogni messaggio ha una intestazione di 24 byte:



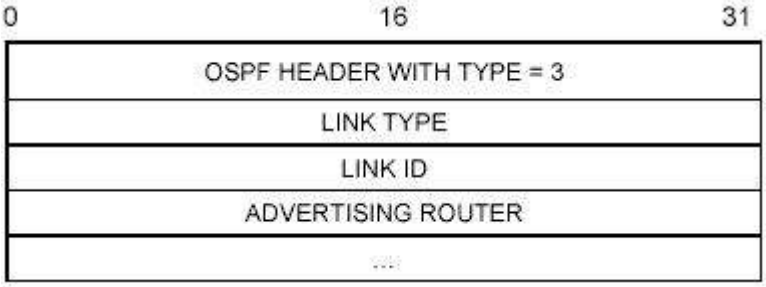
  OSPF prevede 5 tipi di messaggi:
  1. *Hello* (usato per verificare la raggiungibilità)



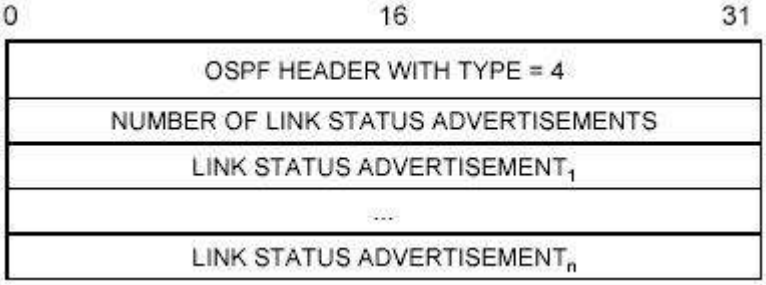
2. *Descrizione del database* (topologia)
   Scambio di informazioni tra un router master e uno slave, suddivisione in più messaggi (uso di I e M e sequenze number)



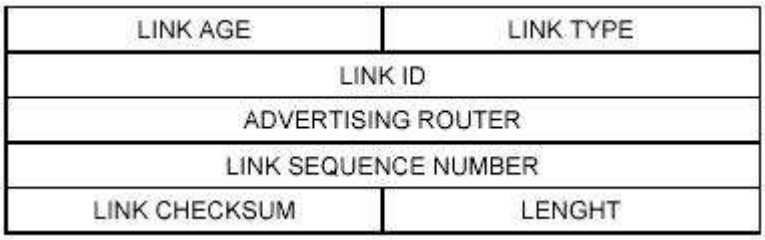
3. *Richiesta dello stato del collegamento*



4. *Aggiornamento dello stato del collegamento*



   in cui ogni annuncio ha una intestazione del tipo

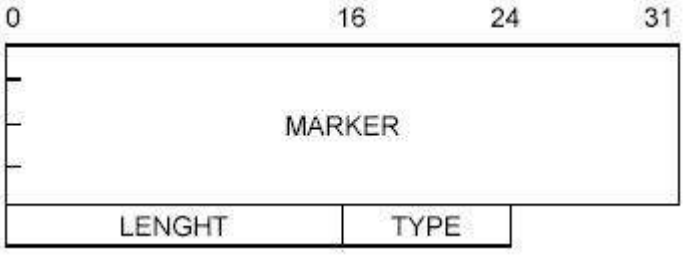


5. *Conferma di ricezione dello stato di collegamento*

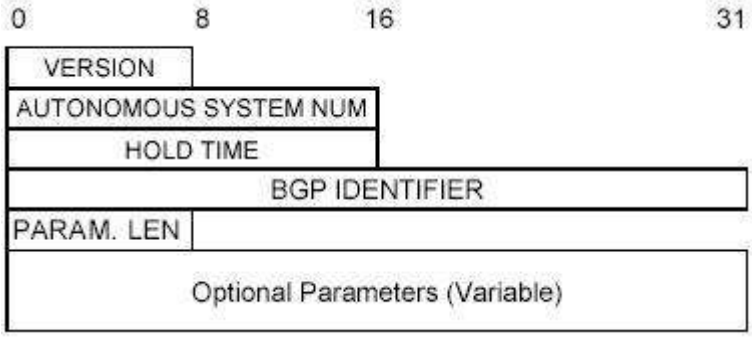
Protocolli inter-AS

- **BGP** (Border Gateway Protocol)
  Algoritmo *path vector*: ogni aggiornamento della tabella di instradamento contiene tutto il cammino (propaga le informazioni sul percorso, non sui costi), se è già nella tabella rifiuta il cammino, altrimenti aggiunge e informa gli altri AS del cambiamento. Il vantaggio sta nel fatto che l'AS sceglie il cammino, mentre il protocollo assicura che non vi siano cicli. Usa un trasporto affidabile (TCP), l'aggiornamento è incrementale (informazioni incomplete all'inizio, poi diffusione dei delta), supporta l'indirizzamento CIDR e l'autenticazione. Permette inoltre di scegliere fra varie politiche (che non gli appartengono), facendole successivamente rispettare: gestisce insomma il routing nel caso di AS *stub* (un AS che ha una sola connessione con un altro AS), *multi-horned* (un AS con più connessioni ma che non trasporta traffico in transito) e *transit* (un AS con più connessioni che trasporta sia traffico locale che in transito).
  Il BGP ha 3 funzioni di base:
  1. autenticazione e acquisizione del partner iniziale
  2. invio di informazioni di raggiungibilità
  3. verifica che le connessioni funzionino correttamente

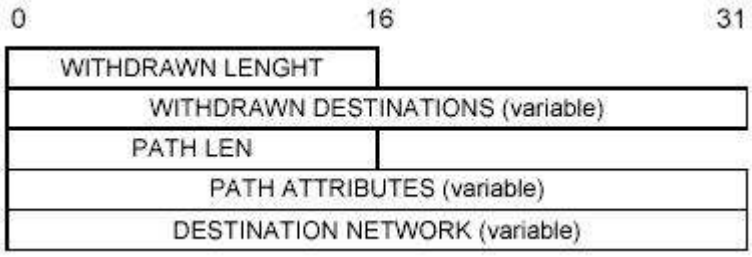
  Ogni messaggio ha un'intestazione di 19 byte:



  Permette 4 tipi di messaggi:
  1. *OPEN* (inizializza la comunicazione)

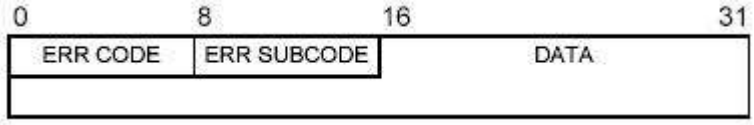


     a cui il destinatario risponde con un KEEPALIVE (conferma ricezione)
  2. *UPDATE* (annuncia o ritratta gli instradamenti)



La prima parte riguarda le destinazioni precedentemente annunciate che sono state eliminate, la seconda riguarda le nuove destinazioni diventate raggiungibili.

3. *NOTIFICATION* (risponde ad un messaggio errato)



A seguito di un errore, la connessione viene chiusa.

4. *KEEPALIVE* (verifica attivamente la connessione)
   Intestazione standard di 19 byte senza ulteriori dati. Necessario perché una funzione keepalive non è presente nel TCP.

NB: un EGP non interpreta le unità metriche, neanche se sono disponibili… quindi possono essere indicati con BGP solo i percorsi che il traffico dovrebbe seguire

DNS (Domain Name System)

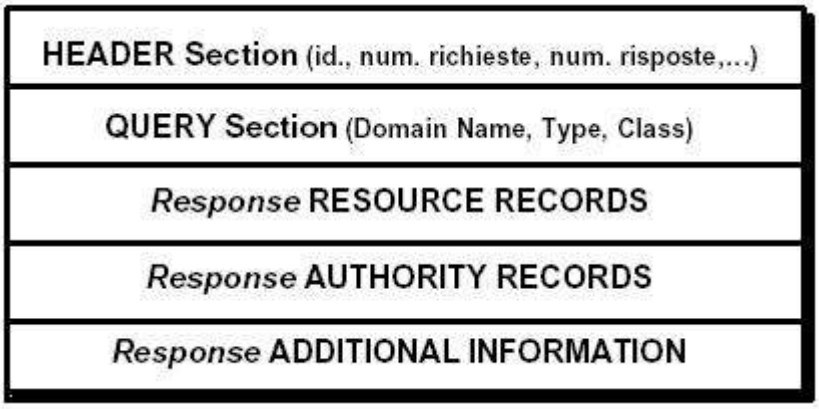
Agli indirizzi IP corrispondono degli hostname: sequenza di label separate da punti, ogni label al massimo lunga 63 caratteri, intero hostname al massimo di 255 caratteri, hostname bindato all'IP fino a tempo di esecuzione. Per mappare e quindi risolvere un hostname (hostname2address e viceversa) si usa il DNS, una soluzione distribuita su scala geografica con uno spazio gerarchico dei nomi. E' un sistema dotato di uno spazio dei nomi consistente (e quindi affidabile), con elevata tolleranza ai guasti, scalabile (partizionamento, decentralizzazione della registrazione degli indirizzi, distribuzione e caching dell'informazione) e funzionante in reti amministrativamente e tecnicamente eterogenee.
I domini sono organizzati gerarchicamente. Si parte dalla Root, e dai TLD (Top Level Domain), suffissi assegnati in maniera univoca dall'Internet Authority. L'onere dell'amministrazione del dominio e della registrazione di domini inferiori passano nelle mani dell'organizzazione alla quale il dominio è registrato.

Componenti del DNS

- *Domain Name Space e Resource Records*
  Il domain name è la sequenza di label che va dall'hostname al TLD.
  Il dominio (riferito alla struttura gerarchica dei nomi) è un sottoalbero dell'albero di naming.
  La zona (riferito all'organizzazione dei name server) è l'insieme di dati relativi ai nomi di un dominio. I name server della zona forniscono dati autoritativi (consistenti e ragionevolmente aggiornati).
  I descrittori di risorsa (legati ai nodi nell'albero del DNS) contengono il nome del dominio, TTL, class (ad esempio INternet), type (Start Of Authority, Name Server, host Address, Mail eXchanger, CNAME ecc…) e Data (indirizzo).
- *Name Server*

Sono i possessori e gestori dell'informazione, con le funzionalità di server abilitati a rispondere alle query dei client. Non hanno i dati di tutti i nomi: ovviamente però devono conoscere i root NS, i NS della zona immediatamente superiore e/o i NS di altre zone. Per ogni zona, c'è un master server (che legge i dati autoritativi di una zona direttamente dal master file) e 1 o più secondary server (che scaricano dati dal master, aggiornando periodicamente i dati.
Ogni NS può effettuare il caching dei dati, in modo da evitare delle risoluzioni nel caso vengano richieste frequentemente. Naturalmente, ogni dato in cache ha un certo TTL, e il client viene informato che il dato ricevuto proviene dalla cache.
La gerarchia presenta root NS, TLD NS, intermediate NS e infine local NS. Ci sono 13 root NS, il server A ha il file originale, che viene propagato con il protocollo DNS (UDP) agli altri 12.

- *Resolvers*
  Sono i primi client del DNS, che sottomettono query per informazioni su hostname e indirizzi IP per conto delle applicazioni di rete. Ogni revolver deve ovviamente conoscere il riferimento ad almeno un NS locale.
  Il messaggio è nella forma:



  al quale, in risposta, viene dato domain name, response type, classe (IP), TTL, dimensione dei dati del RR di risposta e dati del RR di risposta. Viene usato il TCP per il trasferimento di interi DB, altrimenti l'UDP per piccoli aggiornamenti.
  C'è una libreria di funzioni del DNS che funge da resolver:
  - gethostbyname() ↙ prende in input la stringa hostname, e restituisce il puntatore ad una struttura con dentro anche l'IP dell'host; in caso di fallimento, ritorna 0 (gestione dell'errore)
  - gethostbyaddr()
  - gethostbyname2() ↙ IPv6
  - uname() ↙ per ottenere l'hostname di un host locale
  - getservbyname() ↙ per ottenere la porta relativa ad un certo applicativo di rete
  - getservbyaddr() ↙ per ottenere il nome dell'applicativo data la porta

Meccanismo distribuito di risoluzione dei nomi
Dato che nessun NS ha tutte le corrispondenze tra hostname e indirizzo IP, un host che debba risolvere un nome contatta il suo NS locale, e lascia poi a lui eventualmente il compito di interpellare altri NS per risolvere il nome. Ogni NS può effettuare una query ricorsiva (propaga l'onere della ricerca in modo client/server ad altri NS) oppure iterativa (il NS che non sa rispondere alla risoluzione può rispondere mandando una lista di server da contattare). I root NS, per evitare il sovraccarico, usano la query iterativa; gli altri in genere usano la ricorsiva.

Consistenza ed efficacia del DNS
Nel NS master di una zona possono essere effettuate modifiche sui nomi della zona. Il mantenimento del database è effettuato dai NS slave che periodicamente interpellano il NS master per controllare se vi sono stati cambiamenti. Un NS può possedere dati autoritativi per 0..n zone.

Livello Host2Network

Protocollo LAN: Ethernet, token-ring
Protocollo per modem: PPP
Protocollo per wireless: 802.11x
Si occupa del collegamento tra host terminale e rete (host terminale – router) e su di un singolo link (router – router e host terminale – host terminale), e consegna pacchetti (frame) solo all'interno di una stessa lan.
Comprende servizi di framing (incapsulamento in frame), accesso al link, controllo di flusso, recapito affidabile, ricerca e correzione di errori (con ritrasmissione), half o full duplex

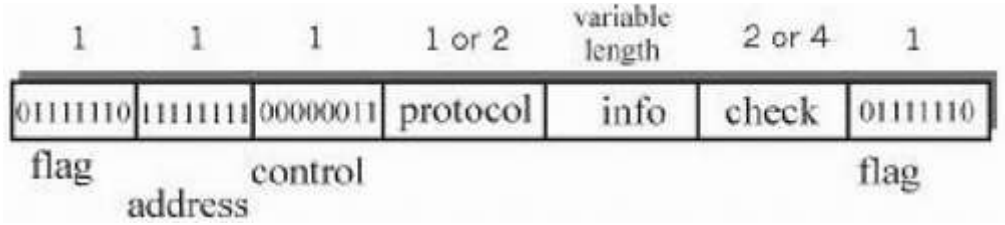
Accessi ad internet residenziali

Accessi di tipo point2point:

- modem dialup
  l'utente si collega, tramite la Public Switched Telecommunication Network (PSTN), all'ISP; la rete interna PSTN è col tempo passata da segnale analogico a segnale digitale
- ISDN
  sostituisce la linea telefonica analogica commutata con una linea digitale commutata; la rete della compagnia telefonica è interamente digitale
- xDSL
  eliminano completamente le apparecchiature in banda fonica per utilizzare al meglio il doppino telefonico; richiede l'installazione di apposite apparecchiature nelle centrali telefoniche
- HFC (Hybrid Fiber Coax)
  rete di cavi coassiali e fibra che collegano le case ai router dell'ISP

ISP
Il provider possiede uno o più server sempre attivi e collegati ad Internet in modo permanente, e una batteria di modem collegati a numeri di telefono.

Protocollo PPP
Protocollo di comunicazione point2point, senza indirizzamento MAC. Utilizza il packet framing, data transparency, error detection, connection liveness (individua e segnala problemi di connessione) e address negotiation (gli endpoint possono apprendere o configurare l'indirizzo di rete dell'altro). Non prevede correzione di errore né controllo di flusso, accetta la consegna di pacchetti disordinati e non ha la necessità di supportare link multipoint (non ha bisogno di effettuare il polling).
Il frame PPP è nella forma:



Dove il datagram è contenuto all'interno del campo info, e i flag fungono da delimitatori del frame.
Viene utilizzato, come detto, il data transparency, che permette di inviare qualsiasi dato, compreso il byte 01111110 usato come flag (ripetendolo due volte se è un dato).
Lo scambio dati avviene configurando prima il link PP, poi apprendendo/configurando il network layer (con messaggi IPCP)

Accessi istituzionali

Lan wired o wireless, client connessi tramite bridge e switch ai router.

Livello fisico

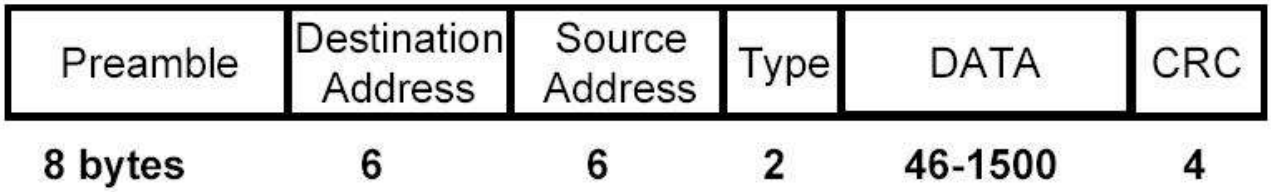
- Mezzo fisico wired
  - doppino telefonico (2 fili di rame schermati)
  - cavo coassiale (cavo segnale all'interno di un cavo schermo, bidirezionale)
  - fibra ottica (alta velocità e basso tasso di errore)
- Mezzo fisico wireless
  Segnale trasportato nello spettro elettromagnetico, bidirezionale, problemi dovute alle interferenze con l'ambiente (micro-onde, lan, wide-area, satellite)

Ethernet

Protocollo del livello data-link, multi-accesso (mezzo condiviso); ogni interfaccia ethernet ha un indirizzo hardware univoco di 48 bit (assegnati alle schede di rete dalla IEEE).
Utilizza il CSMA/CD:

- *Carrier Sense* (rilevazione della portante): un adattatore non può trasmettere un frame quando rileva che alcuni altri adattatori stanno trasmettendo
- *Multiple Access*: più host su 1 solo filo
- *Collision Detection*: un adattatore che sta trasmettendo abortisce la sua trasmissione non appena rileva che anche un altro adattatore sta trasmettendo

Un frame ethernet è nella forma:



dove il preambolo è usato per la sincronizzazione, gli indirizzi sono MAC, il type è il tipo di protocollo e il CRC è il controllo di ridondanza ciclico.
Ogni interfaccia guarda ogni frame e controlla il campo di destinazione: se l'indirizzo non corrisponde al proprio MAC address o all'indirizzo di broadcast (FF:FF:FF:FF:FF:FF), allora il frame viene scartato.
Ogni rete è caratterizzata da una quantità massima di dati trasportabili in un frame, detta *Maximum Transfer Unit (MTU)*. Se si supera tale quantità, è necessario suddividere il datagram in frammenti

(ognuno avente il proprio fragment header). Il riassemblamento avverrà poi soltanto alla destinazione finale (non nei router). I bit dell'IP header sono usati per identificare i frammenti. Inoltre, se almeno uno dei frammenti viene perso, tutto il datagram viene scartato (con conseguente invio di un ICMP al mittente); lo stesso vale in caso di errori di altro genere.

Protocollo ARP (Address Resolution Protocol)
Incluso nella suite TCP/IP, usato per *tradurre (risolvere) un indirizzo logico (IP) nel corrispondente indirizzo fisico (MAC)*. La traduzione avviene o tramite una tabella, oppure tramite computazione in forma chiusa (funzione matematica) oppure tramite scambio di messaggi (messaggi inviati a server noti oppure a tutti i nodi). Il protocollo ARP si compone in un messaggio di richiesta (contenente l'indirizzo IP) mandato in broadcast, e un messaggio di risposta (contenente il MAC); il messaggio viene incapsulato in un frame. E' possibile utilizzare anche il caching delle risposte ARP: per ridurre il traffico sulla rete causato dallo scambio di messaggi ARP, ciascun host effettua un caching temporaneo delle risoluzioni ottenute nella propria ARP table. Un'ulteriore ottimizzazione può essere fatta inserendo nella richiesta il proprio indirizzo IP e fisico, in modo tale da permettere ai riceventi di mettere direttamente in cache il mapping del mittente
Conversione RARP: risoluzione inversa (MAC 2 IP).

Componenti Ethernet

Tipologie di ethernet

- 10base2 : 10mbps, cavo < 200m, coassiale sottile in topologia a bus
- 10baseT / 100baseT : fast ethernet, presenza di hub, topologia a stella, cavo < 100m
- Gbit ethernet

Componenti ethernet

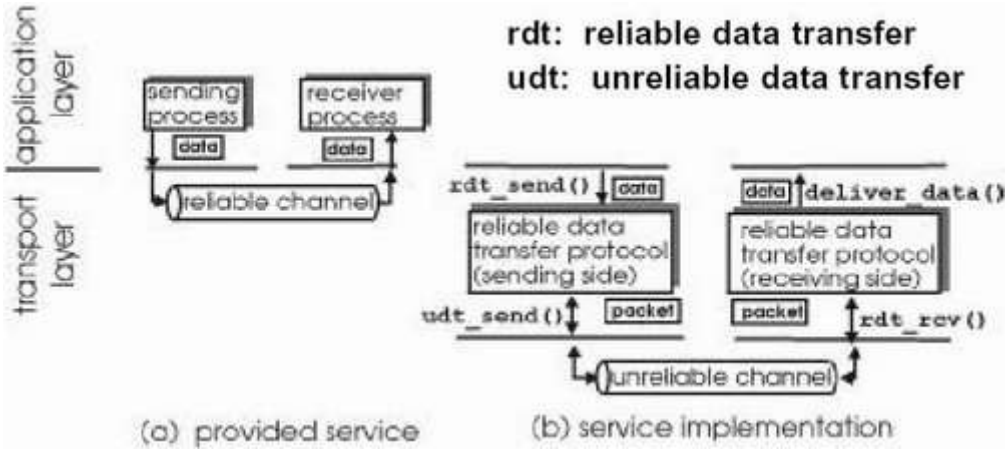
- ***Hub***
  Dispositivo semplice e poco costoso che estende la massima distanza fra coppie di nodi e permette ad una lan di continuare a funzionare anche se uno degli hub non funziona; tuttavia, non fa aumentare il throughput, non può connettere diversi tipi di ethernet, e ogni ethernet rimane comunque limitata dal suo raggio d'azione geografico.
- ***Bridge***
  Dispositivo store&forward di livello h2n che opera con frame ethernet analizzando l'header dei frame e inoltrandoli selettivamente, basandosi sulla loro destinazione. I bridge isolano i domini di collisione dal momento che bufferizzano i frame: prima di essere inoltrato, infatti, usa il CSMA/CD.
  Vantaggi
  - isola i domini della collisione, producendo un aumento del massimo throughput totale, e non limitando né il numero dei nodi né la copertura geografica
  - può connettere diversi tipi di ethernet
  - trasparente, ossia non necessita di alcun cambiamento degli adattatori lan
  - operazioni semplici
  - plug & play (non serve la configurazione di indirizzi IP)

  Filtraggio
  I bridge filtrano i pacchetti: i frame destinati ad host dello stesso segmento della lan non sono inoltrati sugli altri segmenti. Tale filtraggio avviene per mezzo di tabelle di filtraggio (i cui record sono composti da < indirizzo lan, interfaccia bridge, tempo >), costruite mediante apprendimento: se il frame è sulla lan di destinazione viene eliminato, altrimenti guarda

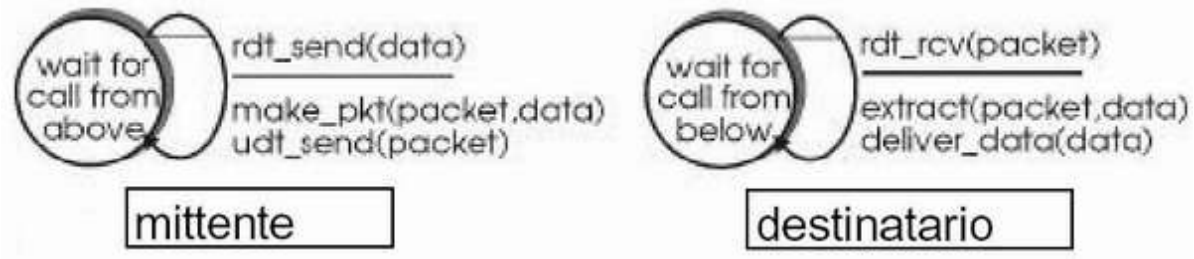
nella tabella di filtraggio; se trovi un record di destinazione instradalo sull'interfaccia indicata, altrimenti flodda (inoltra su tutte le interfacce tranne quella da cui il frame è arrivato).
Spanning tree
Per aumentare l'affidabilità, è desiderabile avere cammini alternativi, ma in questo modo si aumenta la probabilità del verificarsi di cicli. Per risolvere il problema, si possono organizzare i bridge in modo da formare uno spanning tree, disabilitando qualche interfaccia.
Svantaggi
- le topologie sono ristrette
- non offrono protezione dal broadcast selvaggio

- ***Switch***
Sono bridge ad alte prestazioni con molte interfacce, lavorano a livello h2n, permettono di switchare simultaneamente e senza collisioni.
Cut-through switching: il pacchetto è inoltrato dalla porta di input a quella di output senza aspettare che tutto il pacchetto sia arrivato al commutatore (a differenza degli altri dispositivi store&forward)

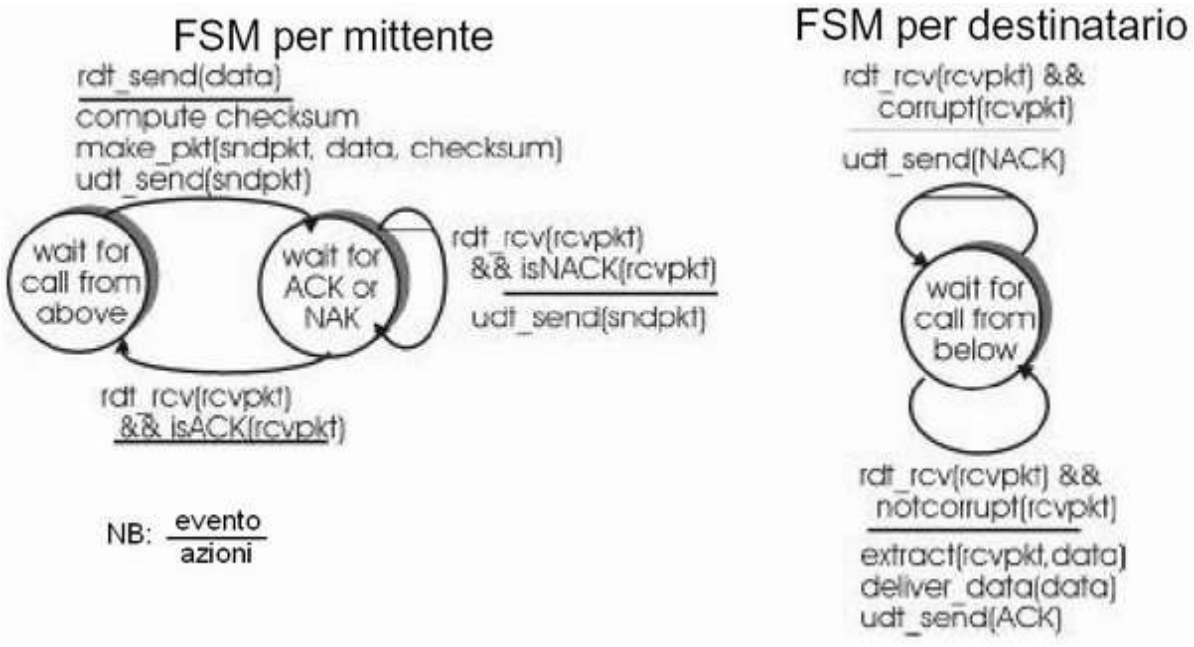
Teoria del trasferimento affidabile



1. Protocollo rdt1.0 : trasferimento su canale completamente affidabile

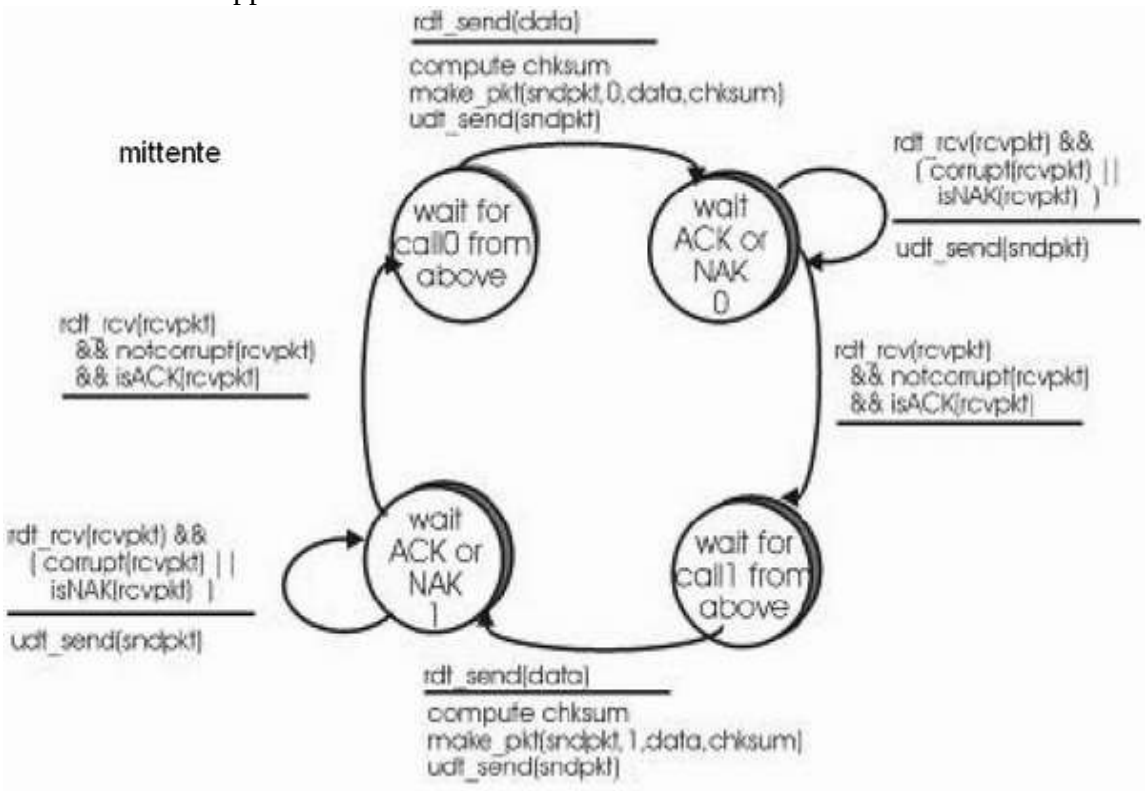


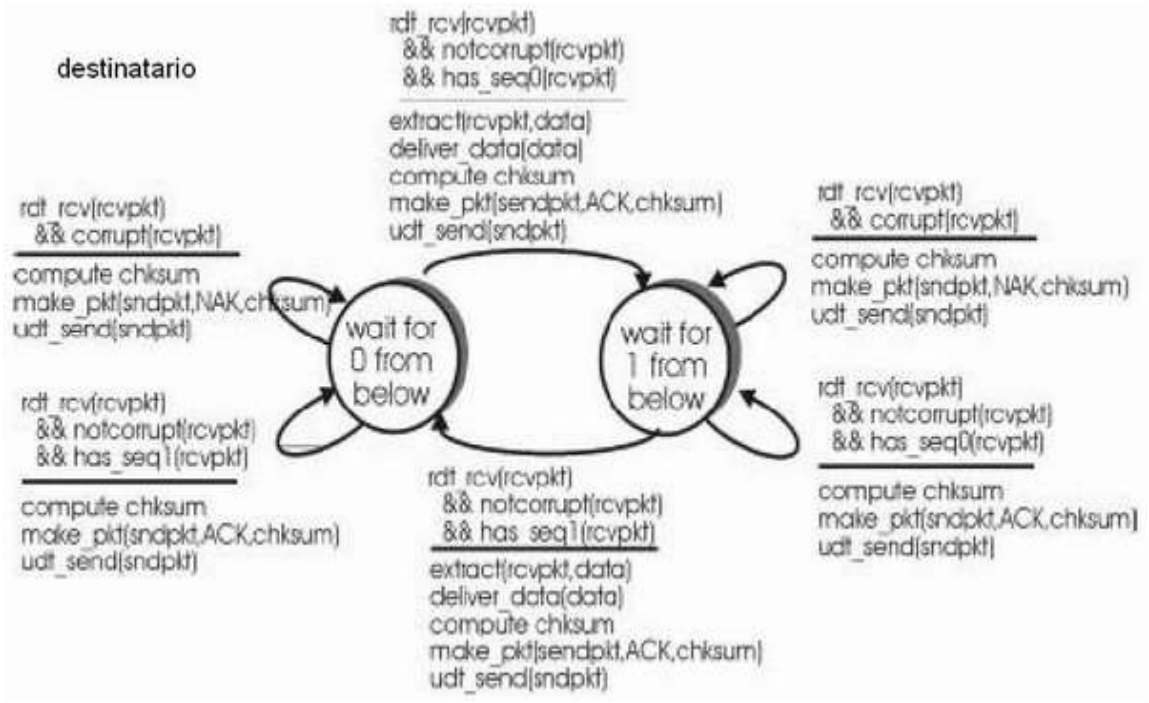
2. Protocollo rdt2.x : trasferimento su un canale con errore sui bit
il mittente invia un pacchetto (uso di checksum) e si mette in attesa di un ACK o NACK, ciclando finché non riceve un ACK.
…Ma cosa succede se ACK/NACK sono danneggiati?? Pericolo di duplicazione di pacchetti…



2.1 versione base

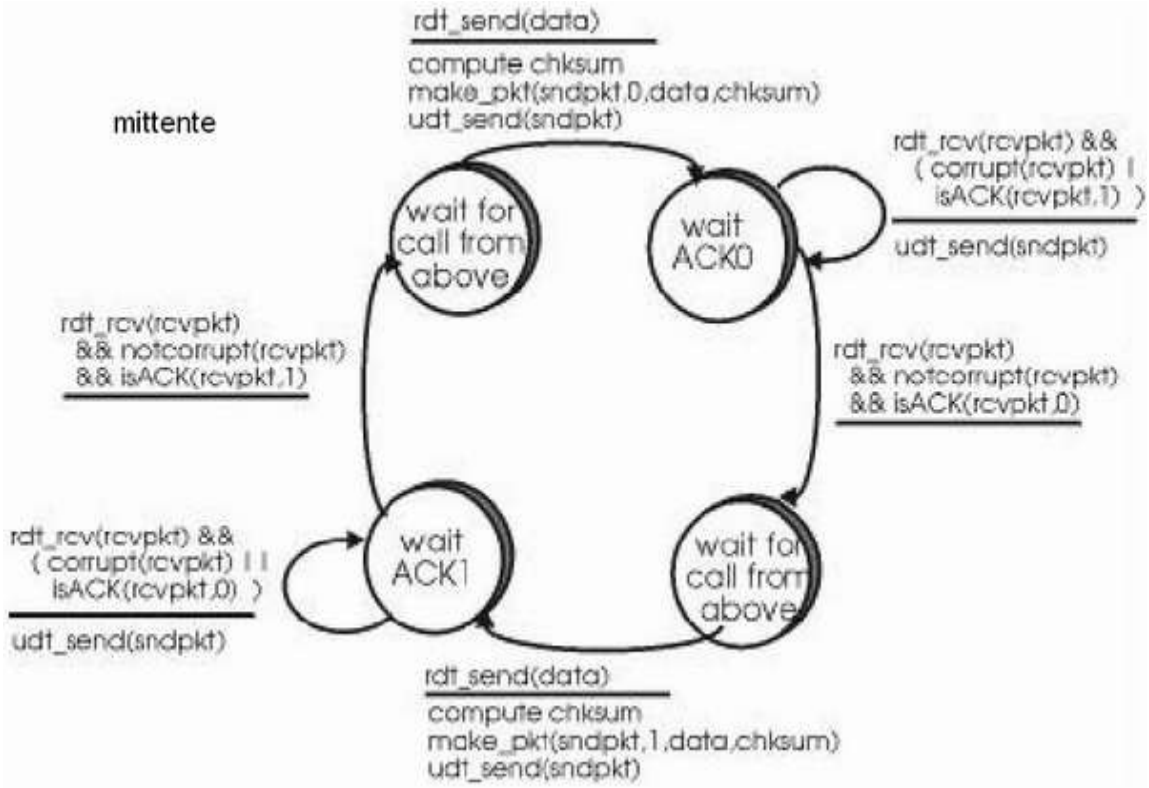
- il mittente aggiunge un numero di sequenza (0,1) a ciascun pacchetto
- il mittente verifica se ACK/NACK sono danneggiati, e in questo caso ritrasmette
- il destinatario scarta i duplicati.
- Il destinatario può non sapere se il suo ultimo ACK/NACK è stato ricevuto correttamente
- serve il doppio di stati

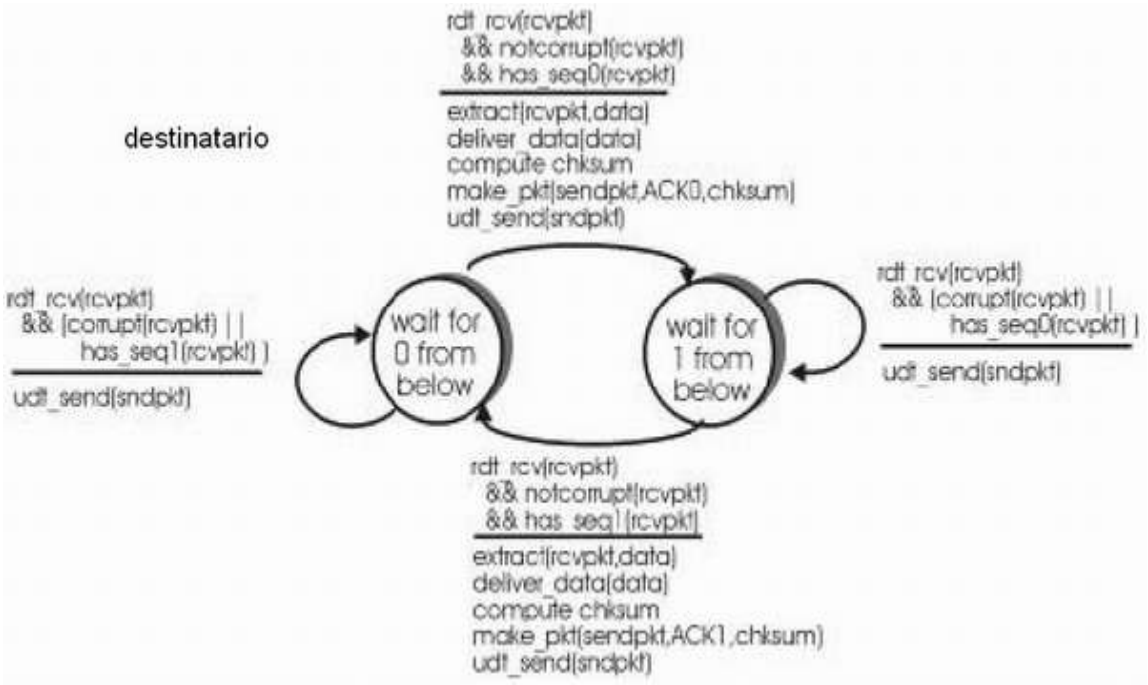




2.2 risolve i problemi di duplicazione

invece di inviare NACK, inviare un ACK per l'ultimo pacchetto ricevuto correttamente: quindi se il mittente riceve due ACK per lo stesso pacchetto, sa che il destinatario non ha ricevuto correttamente il pacchetto successivo

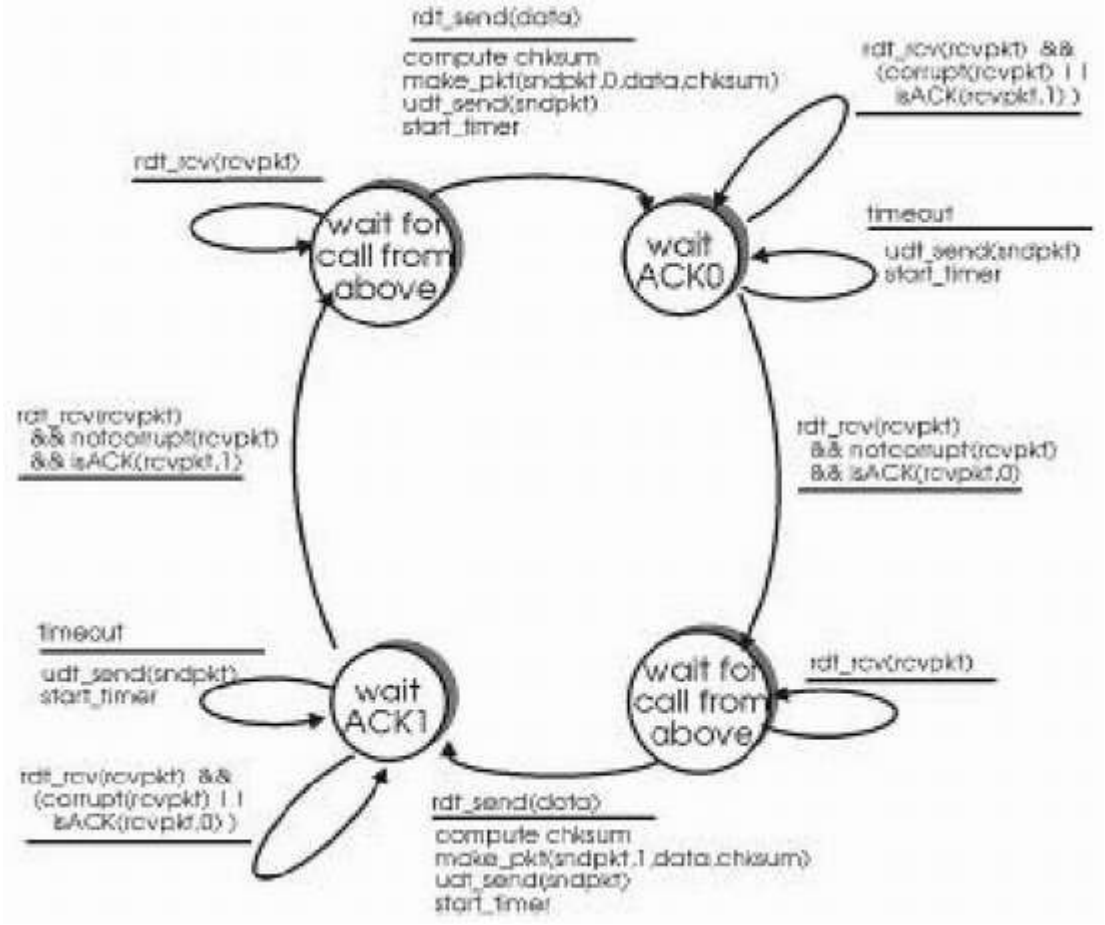




2.3 evita due tipi di ACK

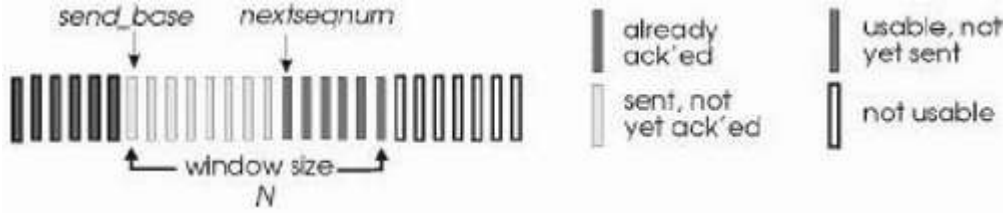
mittente attende l'ACK per un intervallo di tempo "ragionevole", poi ritrasmette il pacchetto se non ha ricevuto l'ACK; serve un countdown timer.

3. Protocollo rdt3.0 : trasferimento su un canale con errore sui bit e perdita di pacchetti il mittente attende l'ACK per un intervallo di tempo "ragionevole", poi ritrasmette il pacchetto se non ha ricevuto l'ACK; serve un countdown timer.



Funziona correttamente, ma la prestazione del meccanismo stop&wait è scadente. Per combinare affidabilità ed efficienza, si ricorre al pipelining: il mittente invia un numero multiplo di segmenti prima di ricevere ACK (serve un buffer).
Due possibili meccanismi di pipelining:

- *Go-Back-N*
  finestra di n pacchetti, spediti ma non ACKati e non spediti; ACK cumulativo per n pacchetti; se il pacchetto n non è arrivato, ritrasmette n e tutti i successivi.



- *Ritrasmissione selettiva*
  il destinatario invia ACK di tutti i pacchetti ricevuti correttamente (bufferizzazione dei pacchetti); il mittente ritrasmette soltanto i pacchetti per i quali non ha ricevuto ACK dal destinatario; mittente e destinatario gestiscono due finestre di sequenze di n pacchetti consecutivi

Livello di trasporto, protocolli UDP e TCP

Questo livello estende il servizio di consegna con impegno proprio del protocollo IP tra due host terminali ad un servizio di consegna a due processi applicativi in esecuzione sugli host. E' composto da due protocolli, TCP e UDP, che sono dotati della possibilità di multiplazione e demultiplazione di messaggi e del rilevamento (non correzione) dell'errore. Il TCP permette un trasferimento affidabile dei dati (controllo di flusso, numeri di sequenza, ack e timer) e il controllo della congestione (regolazione del tasso d'invio).

Multiplazione e demultiplazione
Rispettivamente, creazione dei segmenti provenienti dai messaggi di diversi processi applicativi (avviene nel mittente) e determinazione del processo a cui deve essere consegnato il segmento. Tale procedimento necessita del numero di porta del mittente e del destinatario, in quanto permettono di identificare univocamente i due processi applicativi comunicanti (magari infatti dello stesso tipo e operativi nello stesso istante). A livello di trasporto, è quindi necessaria la coppia (indirizzo IP, numero porta).
PS: 65535 porte, le prime 1023 riservate a protocolli applicativi noti.

Protocollo UDP

Il segmento UDP è incapsulato nella parte dati di un datagramma IP. E' nella forma:

Per effettuare il checksum: il mittente tratta i segmenti come interi a 16 bit, ne somma i contenuti con complemento a 1 e invia il valore del checksum nel segmento; il destinatario controlla il checksum e controlla se è uguale. Per aumentare la sicurezza, si usa lo pseudo-header UDP

Il checksum viene calcolato sull'intero segmento UDP, e lo pseudo-header non è trasmesso dal mittente (inserito, calcolato e eliminato dal livello IP).

Protocollo TCP

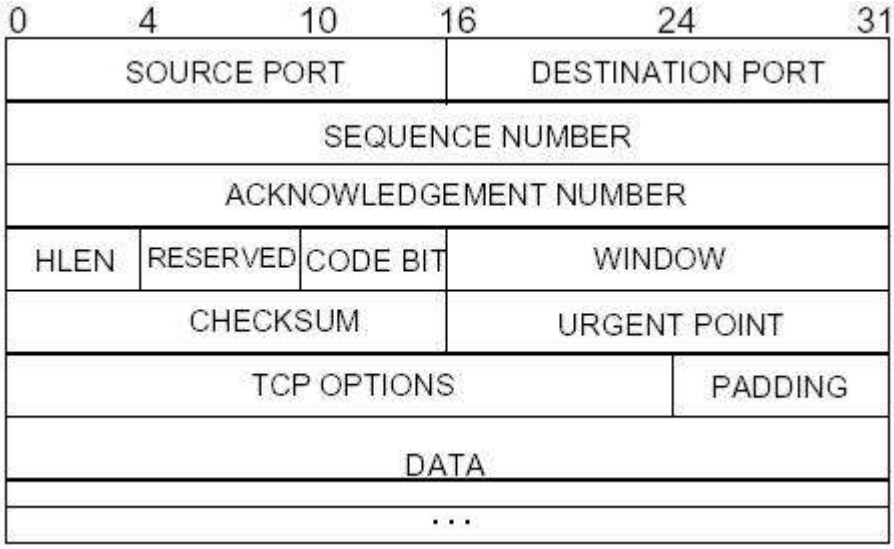
1. *Connection oriented*: è stabilita prima una connessione, chiusa alla fine del trasferimento di dati; l'applicazione di rete viene avvisata se non si riesce a stabilire la connessione o se essa viene interrotta
2. *Affidabile*: TCP gestisce un trasferimento ordinato di uno stream di dati (con ack, eventualmente con ritrasmissione in caso di nack o time-out). I dati vengono bufferizzati e trasferiti quando il buffer è pieno (dove MSS = Maximum Segment Size). Vengono usate politiche di controllo della congestione (il mittente diminuisce il tasso della trasmissione quando la rete è congestionata) e controllo di flusso (il mittente non deve sovraccaricare il ricevente).
3. *Trasmissione byte stream*: la connessione viene trattata come un flusso di byte dal mittente al destinatario (unità di misura = byte)
4. *Connessioni full duplex*: possibilità di effettuare trasferimenti in entrambe le direzioni contemporaneamente; tali trasferimenti possono essere sovrapposti, fornendo le informazioni di controllo assieme ai dati utente (piggybacking)

Il TCP funziona in 3 fasi:

1. *handshaking*: fase di setup (3 segmenti), dopo la quale si arriva ad uno stato di connessione riconosciuto da entrambe le parti
2. *trasmissione*
3. *chiusura connessione*

PS: non è possibile il multicast in TCP

L’affidabilità del TCP è garantita dalla tecnica dell’*acknowledgement positivo con ritrasmissione*: il destinatario, quando riceve i dati, invia un ACK al mittente, che attende di ricevere un ACK prima di inviare il pacchetto successivo. Tale tecnica può essere migliorata con l’implementazione di timeout; tuttavia, dato che questo metodo causa la perdita di banda della rete (per il timeout), si usa una finestra scorrevole (che si muove man mano che gli ACK sono stati ricevuti).
Il segmento TCP è nella forma:



dove:
- acknowledgement number: ACK relativo ad un numero di sequenza (piggybacking)
- code bit: scopo e contenuto del segmento
- window: dimensione della finestra

Anche nel TCP, il checksum avviene con la creazione di uno pseudo-header, con le stesse modalità di funzionamento viste nell’UDP.

L’instaurazione della connessione avviene con il modello client/server, inizializzando numeri di sequenza e buffer. Il client invia il segmento SYN, specificando solo la porta (l’IP viene deciso dopo). Il server deve essere ovviamente in uno stato di wait. Il client manda nel SYN segment (solo header):

- ISN (Initial Sequenze Number) : numero pseudo-casuale
- MRW (Maximum Receive Window) : massimo numero di byte che il client può ricevere
- MSS (Maximum Segment Size) : massima dimensione del segmento

Di seguito, il SYN segment del server ha:

- ISN del server
- ACK del server: client ISN+1
- MRW del server
- MSS del server

Il client manda il suo segmento di controllo con SYN=1; il server risponde con SYN=1 e ACK = client_isn +1; il client risponde con SYN=0, ACK = server_isn+1 e numero di sequenza client_isn+1.

La chiusura avviene con un segmento di controllo con FIN=1: il client manda un segmento di controllo con FIN=1, il server riceve e invia ACK; il server chiude la connessione ed invia FIN=1, il client manda ACK, e dopo il timeout la connessione viene chiusa.
PS: è presente nel TCP anche un meccanismo di reset, che fa chiudere al server la connessione immediatamente
Il numero di sequenza è il numero sequenziale del byte del flusso di dati; il numero iniziale di sequenza per un segmento TCP è scelto casualmente, con l'obiettivo di minimizzare la probabilità che sia presente un segmento identificato con lo stesso numero appartenente ad una connessione precedente con identici numeri di porta. Il numero di acknowledgement è il numero di sequenza del successivo byte attesa (con tutti i precedenti già ricevuto).

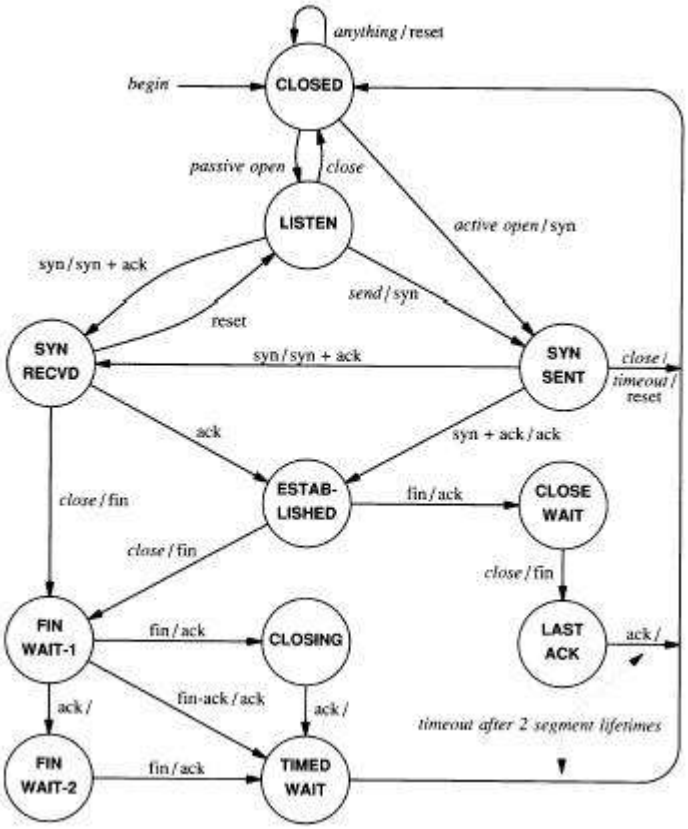
Buffering
Il TCP fa parte del sistema operativo, non del livello applicativo che gestisce l'invio e la ricezione dei dati: è necessario quindi che i dati ricevuti vengano memorizzati temporaneamente in un buffer (in modo da poterli trasmettere al livello applicativo quando gli vengono richiesti, e viceversa).

Controllo di flusso
Il mittente non deve riempire il buffer del destinatario inviando una quantità eccessiva di dati ad un tasso di trasmissione troppo elevato. Per far questo, il destinatario informa esplicitamente il mittente della quantità di spazio libero nel buffer di ricezione TCP (finestra che varia dinamicamente).
L'ACK contiene quanti byte sono stati ricevuti e quanti il destinatario ne può ancora ricevere (istantaneamente).

Round Trip Time e Timeout
Il RTT è il tempo che intercorre tra la spedizione di un segmento TCP e la ricezione del successivo ACK. Varia dinamicamente in base alle condizioni della rete.
Il timeout può essere misurato in maniera "grezza" (SampleRTT), oppure si può stimare con media pesata (EstimatedRTT), successivamente moltiplicato per un margine di errore.

Controllo della congestione
La congestione provoca perdita di pacchetti e lunghi ritardi (buffer nei router pieni). Può essere controllata end2end (approccio usato dal TCP, congestione determinata analizzando le perdite e i ritardi ai nodi terminali), oppure il controllo può essere assistito dalla rete (feedback, diretto o aggiornando un campo del pacchetto, dei router con informazioni riguardanti lo stato della congestione, ricavato dalla lunghezza della coda del buffer).
Nello stato stazionario (senza congestione), la dimensione della finestra di connessione è pari a quella indicata dal ricevente (per il controllo di flusso); al crescere del traffico, il TCP riduce il tasso di trasmissione, e la finestra viene ridotta, scegliendo come dimensione la minima tra quella di congestione e quella di ricezione.
Permette inoltre di ottimizzare la banda disponibile (per trasmettere più velocemente possibile senza perdere segmenti), grazie al meccanismo di slow-start. Tale ottimizzazione consiste in un incremento progressivo (esponenziale) della finestra di congestione fino ad una threshold, alla quale segue un incremento lineare; questo prosegue fino al verificarsi di un episodio di congestione, giunto il quale la finestra viene diminuita, ricominciando poi ad incrementarla nuovamente.

FSM del TCP



UDP vs TCP

I vantaggi dell'UDP stanno nel fatto che non c'è instaurazione di connessione (nessun ritardo dovuto alla connessione); inoltre non deve mantenere la connessione, quindi un server può sopportare più connessioni attive se sono UDP. Inoltre, l'UDP provoca un overhead minore (header più piccolo), e nelle applicazioni in real-time l'assenza del controllo della congestione risulta un vantaggio. L'UDP si può quindi usare quando

- Si opera su rete locale
- L'applicazione mette tutti i dati in un singolo pacchetto
- Non è importante che tutti i pacchetti arrivino a destinazione
- L'applicazione gestisce i meccanismi di ritrasmissione

Applicazione	Prot. strato applicativo	Prot. trasporto sottostante
posta elettronica	SMTP	TCP
accesso terminale remoto	Telnet	TCP
Web	HTTP	TCP
trasferimento file	FTP	TCP
file server remote	NFS	solitamente UDP
multimedia streaming	proprietario	solitamente UDP
Telefonia Internet	proprietario	solitamente UDP
Gestione della rete	SNMP	solitamente UDP
Protocollo di routing	RIP	solitamente UDP
Traduzione dei nomi	DNS	solitamente UDP

Livello applicazioni

Utilizza il livello di trasporto dell'informazione tra processi in esecuzione su host terminali per realizzare applicazioni di rete (DNS, SMTP, Telnet, FTP, WWW).

Meccanismi di naming (URI, Uniform Resource Identifiers)
Si compone di URL, URN e URC, ma di questi il più utilizzato è l'***URL*** (Uniform Reosurce Locator): specifica la locazione fisica delle risorse e le modalità di accesso. E' il meccanismo standard per fare riferimento a tutte le risorse presenti nel Web: pagine, risultati di esecuzioni, programmi eseguibili.
L'URL è nella forma:
schema://host.domain/pathname
dove:

- *schema* : indica il modo con cui accedere alla risorsa, cioè quale protocollo bisogna usare per interagire con il server che controlla la risorsa. Il metodo di accesso più comune è il protocollo HTTP
- *host.domain* : è l'hostname del nodo nel quale risiede la risorsa web
- *pathname* : identifica la risorsa presso il server web

HTML (Hyper Text Markup Language)
Ogni pagina è composta da vari oggetti (file) detti *embedded objects*. Costituiscono l'*ipermedia*, usato mediante point&click. Il linguaggio di markup fornisce delle *linee guida per la rappresentazione* (formattazione della pagina): per questo motivo due browser potrebbero visualizzare lo stesso documento in modo differente. Le istruzioni sono racchiuse in marcatori <tag> </tag>
Schema generale:

```
<HTML>
      <HEAD>
            <TITLE>
                  Titolo del documento
            </TITLE>
      </HEAD>
      <BODY>
            Corpo del documento
      </BODY>
</HTML>
```

L'ancora permette poi di trasformare un normale testo in ipertesto multimediale.

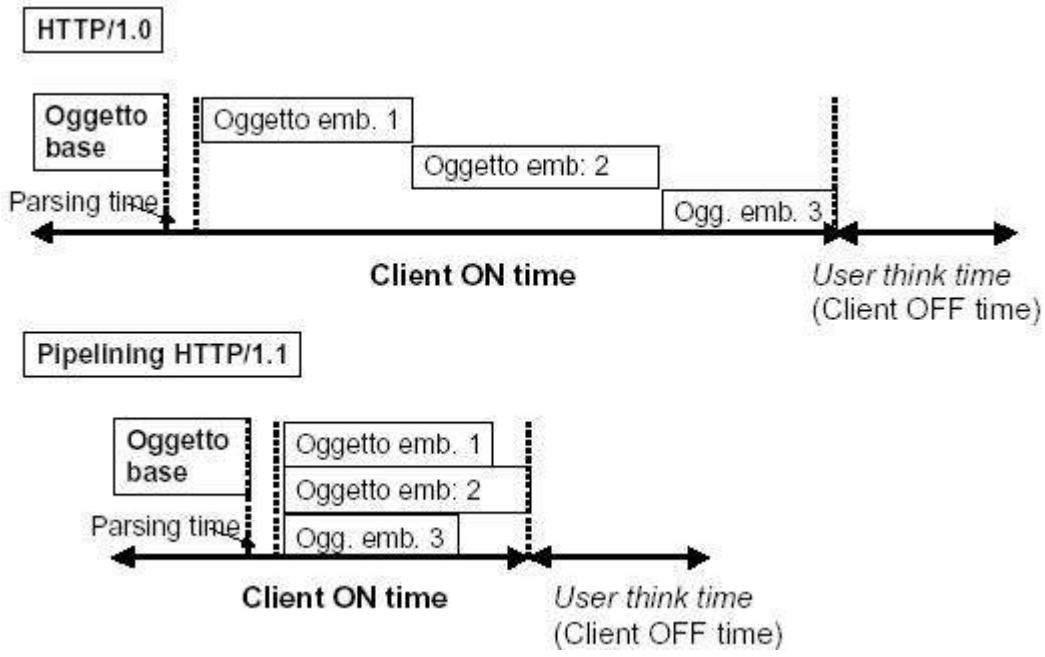
Protocollo HTTP (Hyper Text Trasmission Protocol)

Usa il meccanismo del request/reply, è basato sulla suite di protocolli TCP/IP.

- Il client inizia la connessione TCP verso il server sulla porta 80
- Il server accetta la connessione TCP dal client
- Messaggi http di tipo testuale scambiati tra browser e web server
- Chiusura della connessione TCP

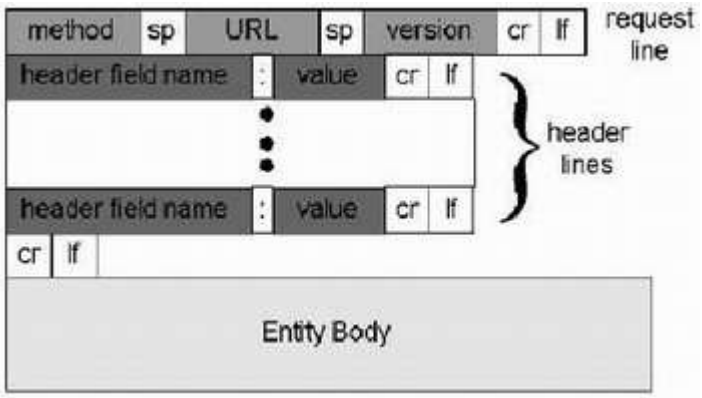
Essendo basati sul TCP, i messaggi di richiesta/risposta sono consegnati integri al destinatario. Inoltre, HTTP è stateless: il server non conserva nessuna informazione riguardante le richieste dei client passati.

- *HTTP/1.0* : 3-way handshake per ogni oggetto da trasferire, slow-start del TCP ↙ risolto con la creazione di connessioni TCP multiple.
- *HTTP /1.1* : 3-way handshake solo all'inizio, controllo della congestione a regime, connessione persistente durante la quale vengono trasferiti più oggetti, pipelining



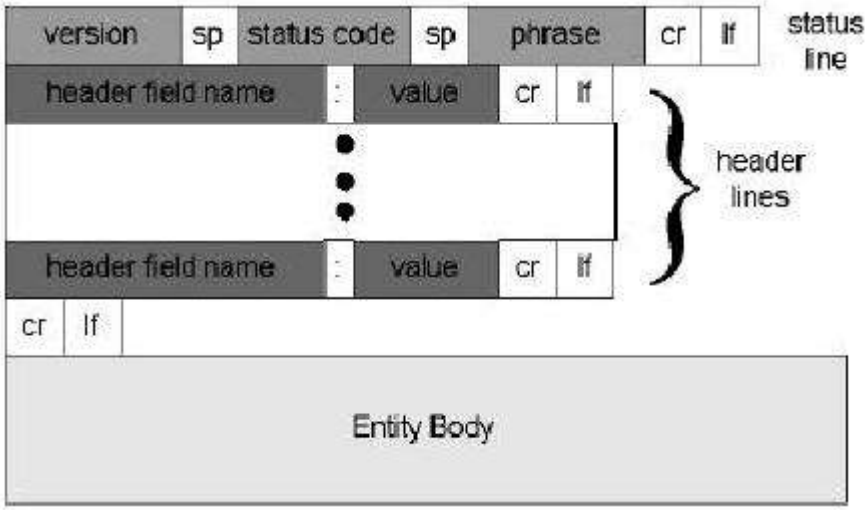
Una richiesta HTTP comprende metodo (operazione richiesta al server dal client), URL, identificativo della versione e extension headers. I messaggi http sono testuali e possono essere di richiesta o risposta.

Messaggio di richiesta:



Metodo	*Richiesta*	*Versione protocollo*
GET	Ricevere una risorsa dal server	Dalla HTTP/0.9
HEAD	Ricevere il solo header di una risorsa	Dalla HTTP/1.0
POST	Appendere un oggetto ad un altro sul server	Dalla HTTP/1.0
PUT	Inviare un oggetto al server	Dalla HTTP/1.1
DELETE	Cancellare un oggetto dal server	Dalla HTTP/1.1
LINK e UNLINK	Creare o eliminare collegamenti fra oggetti del server	Dalla HTTP/1.1
TRACE	Individuare la catena dei server proxy	Dalla HTTP/1.1

Messaggio di risposta



Se la pagina richiesta, oltre al testo HTML, contiene altri oggetti, ciascuno di essi sarà identificato da un URL differente, per cui è necessario che il browser invii un esplicito messaggio di richiesta per ognuno degli elementi collegati alla pagina.
Status code e phrase indicato l'esito della richiesta (codice e frase):

1xx: Informazioni

2xx: Successo

3xx: Redirezione

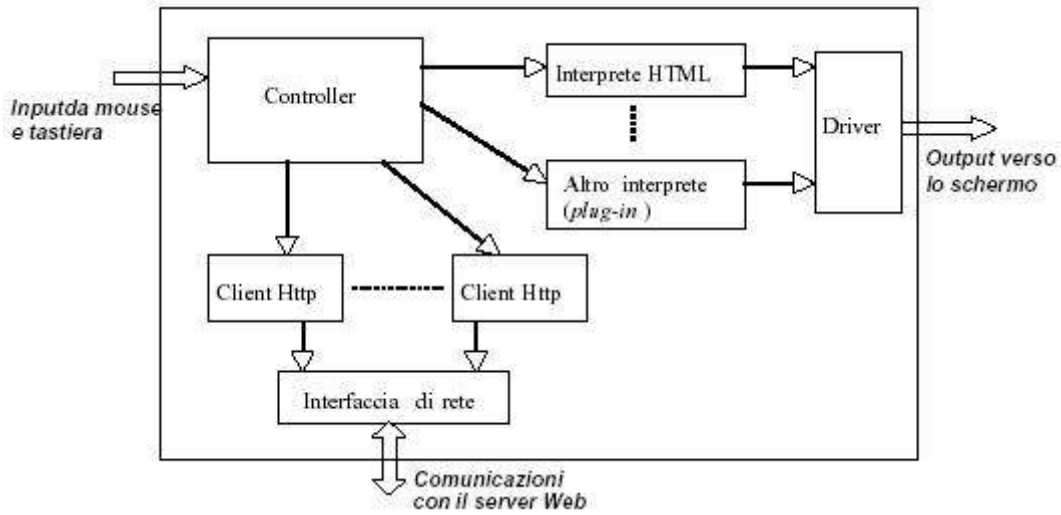
4xx: Errore del client

5xx: Errore del server

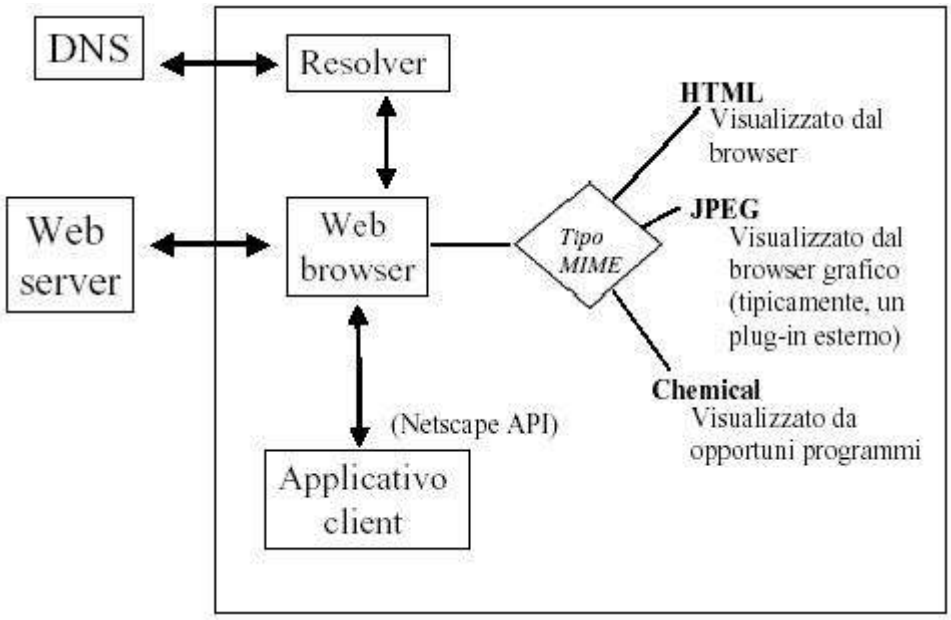
Autenticazione: per controllare gli accessi ai documenti del server; l'autorizzazione viene chiesta con una linea di header nella richiesta, in caso di rifiuto, il server invia una WWW authenticate.
Cookie: usati per l'autenticazione, o la memorizzazione delle preferenze degli utenti e delle scelte effettuate in precedenza; il server invia un set-cookie, il client lo richiede, poi il server confronta il cookie presentato con i cookie da lui memorizzati.

WWW, lato client: client http
Il *browser* è un'applicazione software che svolge il ruolo di interfaccia fra l'utente ed il WWW, mascherando la complessità di internet. Diventa quindi un client web per recuperare informazioni dai server web.

- Instaura una connessione TCP con il server tramite cui invia opportuni messaggi al server web per ottenere le risorse richieste
- Interpreta il codice ipertestuale HTML
- Elabora il codice allo scopo di visualizzare in modo appropriato il contenuto delle pagine sullo schermo



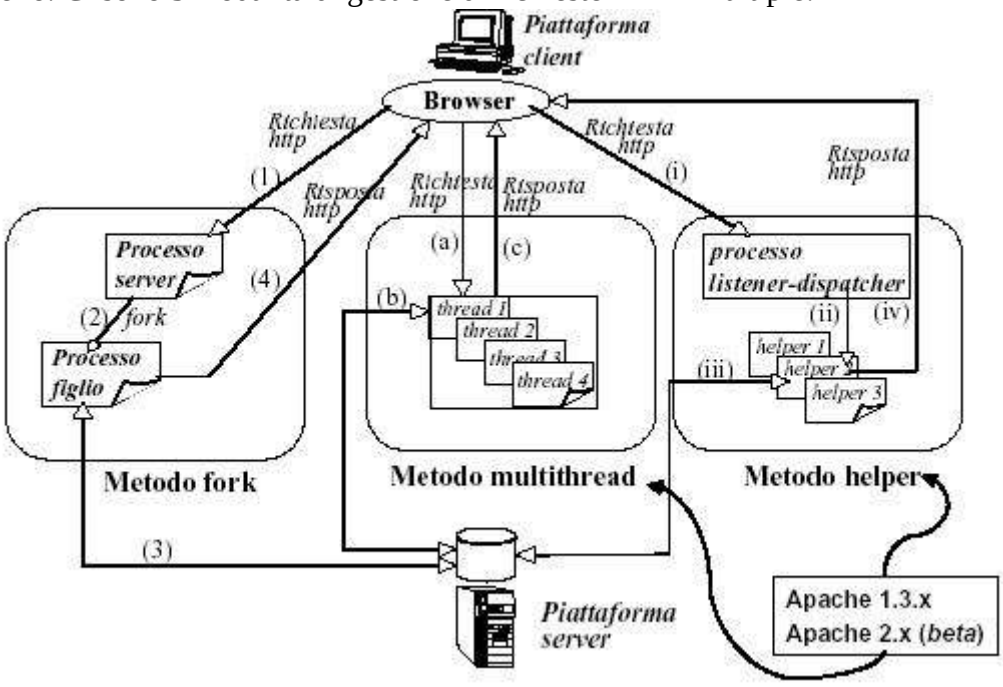
1. *Fase iniziale*: analizza l'indirizzo, cerca nella cache disco (valutando il timestamp degli oggetti presenti), se non c'è passa alla fase successiva
2. *Fase di lookup*: invoca il resolver per conoscere (tramite il DNS) l'IP dell'URL cercato; se esistente, il DNS lo restituisce (fase bloccante)
3. *Fase di richiesta*: il browser avvia la connessione con l'host, richiede le risorse (parsing, ossia analisi degli oggetti allegati alla pagina)
4. *Fase di visualizzazione*



WWW, lato server: server http

Organizzazione delle pagine gerarchica ad albero. L'albero delle pagine web è organizzato in modo simile ad un file system gerarchico con una radice e directory che contengono altre directory e file. Naturalmente, non significa che il file system del server sia organizzato effettivamente allo stesso modo… Spesso viene anche utilizzato il *mirroring* dell'albero o di una sua parte (copia replicata altrove). Le risorse si suddividono poi in *statiche*, *volatili* (in base alla permanenza o meno dei contenuti nel tempo), *dinamiche* (contenuto creato dinamicamente sulla base della richiesta del client) o *attive* (esecuzione del contenuto sul client).

L'interazione tra l'HTTP (esterno) e il TCP (interno al sistema operativo) avviene tramite ***Application Program Interface (API)***, una parte del sistema operativo che consente ai processi applicativi di utilizzare i protocolli di rete. Si ricorre quindi alla definizione di *socket*: API che consentono ai programmatori di gestire le comunicazioni tra processi, anche residenti su macchine diverse. In altre parole, permettono di effettuare trasmissioni TCP e UDP senza curarsi dei dettagli di basso livello. Ci sono 3 modalità di gestione di richieste HTTP multiple:



Una connessione viene identificata dall'IP e dalla porta sia del server che del client.

Richiesta lato server:

1. attesa di una richiesta sulla porta 80
2. arrivata la richiesta, viene avviata la connessione TCP (3-way handshake)
3. stabilita la connessione, il client invia le richiesta per i file desiderati; il server riceve, le decodifica secondo le regole dell'HTTP. La richiesta contiene il metodo *GET* (specifica le azioni), il cammino della pagina da individuare, il protocollo utilizzato dal browser (e anche informazioni sul browser); inoltre la connessione deve essere chiusa dal server.
4. il server esegue il GET, ricerca nel suo albero di risorse la risorsa desiderata, invia informazioni riguardo alla risorsa, dopodichè preleva i file e li scrive sulla porta della connessione di rete
5. suddivide i dati in più pacchetti e li invia
6. PS: se il server non trova i dati, invia il *404 – Not found*
7. se il protocollo HTTP è l'1.0, ogni embedded object riserva una nuova connessione, se è l'1.1 invece la connessione TCP rimane aperta e gli oggetti vengono inviati in pipeline
8. il server chiude la connessione

Istruzioni da eseguire in una comunicazione socket:

- *inizializzazione*: usata per inizializzare le strutture dati per gestire la connessione, ritorna un file descriptor
- *apertura connessione*: chiamata dal client, con server in attesa
- *trasmissione dati*
- *ricezione dati*
- *chiusura*: close() usata sia nel server che nel client
- *assegnazione di un nome*: bind() usata tipicamente dal lato server, collega un socket ad un indirizzo visibile dall'esterno
- *specifica del backlog*: specifica al sistema operativo la dimensione della coda delle richieste in attesa su un socket, solo per le connessioni
- *accept()* : istruzione bloccante di attesa di una connessione, ritorna un socket

Web server

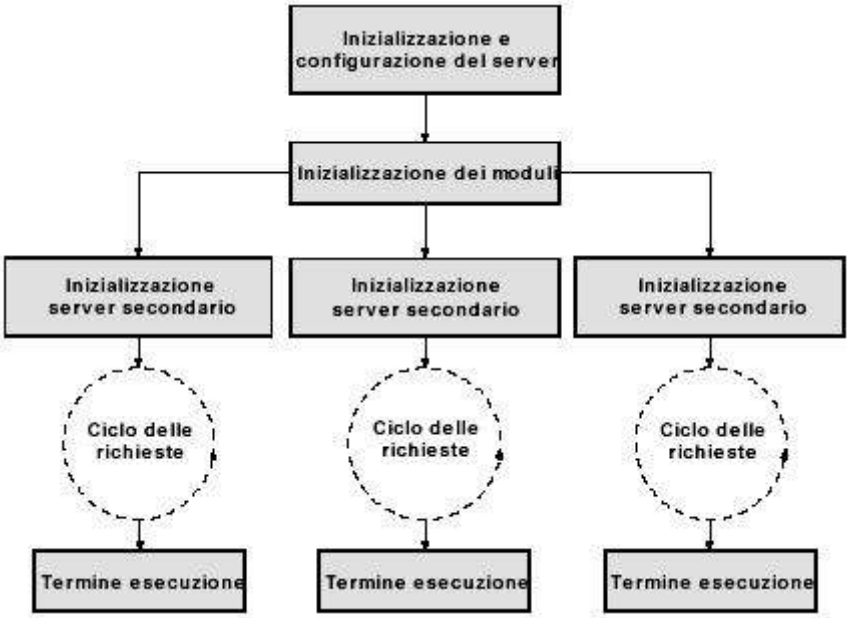
I più diffusi sono *Apache* e *Microsof t IIS*.
Apache è free, portabile, efficiente, supporta HTTP 1.1, stabile e sicuro, modulare (modifica funzionalità web server caricando gli opportuni moduli).
E' caratterizzato da

- Autenticazione
- Personalizzazione dei messaggi d'errore
- URL rewriting e alias
- Negoziazione contenuti
- Virtual hosts
- Logfile personalizzabili

Le richieste possono essere processate in diversi modi:

- Server monoprocesso
- Servizio parallelo mediante fork()
  ogni richiesta genera un nuovo processo server che serve la richiesta, mentre il vecchio si rimette in ascolto; sistema però molto oneroso

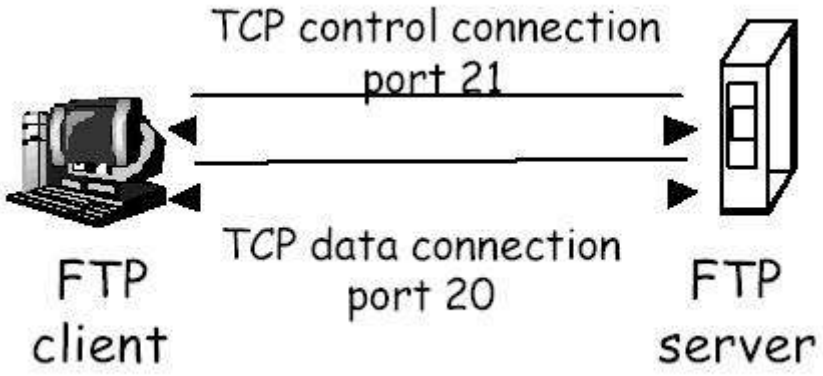
- Servizio parallelo mediante processi helper (Apache 1.3)
  quando arriva una richiesta, il processo principale (master) la accetta e la smista la richiesta verso un processo helper (slave) che la serve
- Servizio parallelo con thread
  più leggere, ma la programmazione è più difficile e il sistema è più vulnerabile a crash



Altri protocolli applicativi storici

FTP (File Transfer Protocol)
Il client FTP contatta il server FTP sulla porta 21, specificando TCP come protocollo di trasporto. Vengono aperte due connessioni TCP parallele:

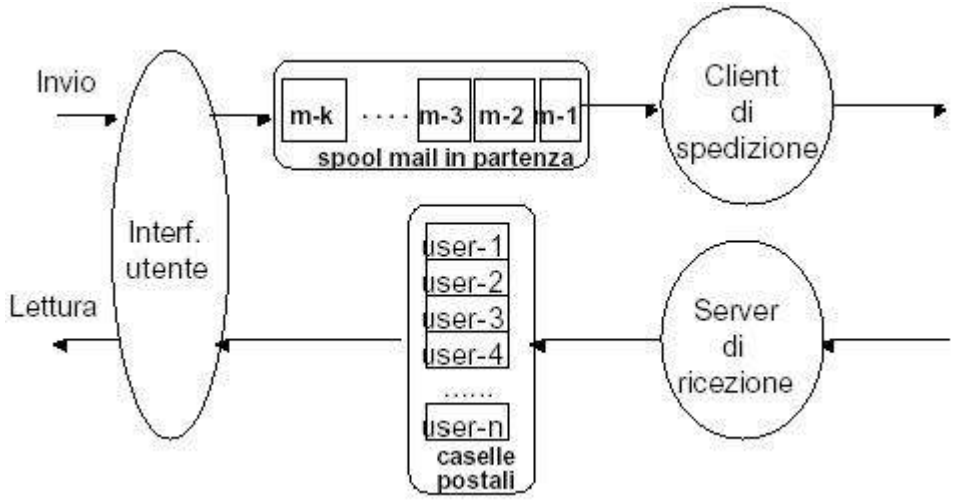
- Connessione per controllo: scambio dei comandi, delle risposte tra client e server
- Connessione per dati: file di dati da/verso il server (una per ogni file)



SMTP (Simple Mail Transfer Protocol)

- User agent: utente
- Mail server : mailbox (messaggi in ingresso dell'utente, ancora da leggere), coda di messaggi (in uscita, da inviare)
- Protocollo SMTP : per inviare i messaggi; paradigma client/server tra mail server del mittente a mail server del destinatario. Usa il trasferimento diretto con TCP sulla porta 25,

usa connessioni persistenti per trasferire più messaggi in una sola volta. Una mail consiste di un header e un body (file testuale ASCII a 7 bit)



MIME (Multipurpose Internet Mail Extension)
Si interpone tra l'interfaccia utente e il mailer automatizzando la procedura di conversione.
Introduce un ulteriore livello di incapsulamento, estendendo l'header del messaggio con 5 campi specifici: questo permette di aggiungere il supporto multimediale (testo, immagini, video, audio, applicazioni)

POP3 (Post Office Protocol), IMAP (Internet Mail Access Protocol) o HTTP
Servono per accedere alla posta.
Il protocollo POP3 ha:

- Fase di instaurazione della connessione (connessione TCP nella porta 110 con il mail server)
- Fase di autorizzazione
- Fse di transazione
- Fase di aggiornamento (della mailbox del client)

Il protocollo IMAP permette invece di modificare la mailbox come se fosse locale, permette di gestire una gerarchia di mailbox per ogni utente e permette all'utente di ottenere alcune parti del messaggio (come gli allegati).

- Fase di instaurazione della connessione
- Fase di autorizzazione
- Fase di transazione (stati: non autenticato, autenticato, selezionato, logout)

Accesso alla posta tramite web: il browser è lo user agent.