



UNIVERSITÀ DI PISA

Dipartimento di Informatica
Corso di Laurea in Informatica

Reti di Calcolatori

Giacomo Trapani
Anno Accademico 2021/2022

Indice.

1	Tipi di Reti	5
1.1	LAN (Local Area Network)	5
1.2	WAN (Wide Area Network)	5
2	Tecniche di commutazione.	5
2.1	Circuit switching.	5
2.2	Packet switching.	6
3	Internet.	6
3.1	Metriche.	6
4	Modelli stratificati.	7
4.1	Modello ISO/OSI.	7
4.2	Stack protocolare TCP/IP.	8
4.2.1	Application Layer.	8
HTTP	9	
TELNET	10	
SMTP	10	
DNS	11	
FTP	12	
4.2.2	Transport Layer.	13
TCP	13	
UDP	16	
4.2.3	Network Layer.	16
IP	16	
Indirizzi IP	18	
DHCP	18	
Routers	19	

Elenco delle figure.

1	Esempi di LAN.	20
2	Esempi di WAN.	20
3	Tipi di reti.	20
4	Internet.	20
5	Modello ISO/OSI.	21
6	Comunicazione in una internet.	21
7	Esempi di applicazioni Internet.	21
8	Messaggi HTTP.	21
9	TELNET.	22
10	SMTP.	22
11	SMTP: esempi di sessioni.	22
12	DNS: servizio di risoluzione.	23
13	Gerarchia dei name server.	23
14	Query DNS.	23
15	Struttura di un messaggio DNS.	24
16	DNS hijacking.	24
17	FTP.	24
18	Demultiplexing.	25
19	TCP: trasferimento bufferizzato.	25
20	Formato dei segmenti TCP.	25
21	TCP: handshake.	26
22	TCP: ritrasmissione veloce.	26
23	TCP: FSM mittente.	27
24	TCP: FSM destinatario.	27
25	TCP: operatività normale.	28
26	TCP: riscontro smarrito.	28
27	TCP: riscontro smarrito corretto con ritrasmissione.	28
28	TCP: segmento smarrito.	29
29	TCP: finestra di trasmissione.	29
30	TCP: finestra di ricezione.	29
31	TCP: esempio di comunicazione con rwnd.	30
32	TCP: slow start.	30
33	TCP Reno: macchina a stati.	30
34	TCP Reno: esempio.	31
35	TCP Tahoe: macchina a stati.	31
36	TCP Tahoe: esempio.	31
37	TCP: AIMD.	32
38	TCP: half-close.	32
39	TCP: chiusura con handshake a 3 vie (comune nel caso di modelli client-server del tipo richiesta/risposta).	33
40	Apertura e chiusura della connessione.	33
41	Chiusura della connessione.	34
42	UDP: struttura del campo checksum.	34

43	UDP: formato di un datagramma.	34
44	IP: inoltro.	35
45	IP: instradamento.	35
46	IP: inoltro e tabella di inoltro.	35
47	IP: frammentazione.	36
48	IP: formato di un datagramma.	37
49	IP: indirizzamento senza classi.	37
50	IP: tabella di traduzione NAT.	38
51	DHCP: esempio di sessione.	38

Premessa.

Si danno le seguenti definizioni:

- **Rete**: interconnessione di dispositivi in grado di scambiarsi informazioni, quali sistemi terminali, router, switch e modem.
- **Host (sistema terminale)**: sono macchine appartenenti agli utenti finali, dedicate all'esecuzione di applicazioni (e.g. computer, tablet, smartphone) oppure "server" che forniscono servizi a diverse applicazioni utente (e.g. posta elettronica, web).
- **Dispositivi di interconnessione**: si distinguono in "router" (i.e. dispositivi che interconnettono reti) e "switch" (i.e. dispositivi che collegano fra loro molteplici host a livello locale).
- **Collegamenti (link)**: mezzi trasmissivi (cablati, wireless).

Tipi di Reti.

1.1 LAN (Local Area Network).

Le LAN vengono definite come reti di computer circoscritte a un'area geograficamente limitata (si estendono al più per alcuni chilometri, sono circoscritte ad aree limitate come un ufficio, una scuola, un edificio etc.). Si cita Ethernet come tecnologia LAN popolare.

Le LAN sono tipicamente di proprietà di una qualche organizzazione (sono dunque reti private), connettono sistemi terminali (stampanti, PC, workstations) mediante cavi di rame o connessioni wireless.

Si distinguono LAN con cavo condiviso (obsolete) e LAN con switch (moderne); il maggiore vantaggio della seconda sulla prima è che non ha bisogno di far passare un pacchetto da ogni dispositivo connesso nel percorso dal router al (dispositivo) target.

Si veda la figura 1.

1.2 WAN (Wide Area Network).

Una WAN (o rete geografica) è una rete il cui compito è di interconnettere LAN o singoli host separati da distanze geografiche. Viene tipicamente gestita da un operatore di rete che fornisce servizi ai clienti. Un esempio di WAN è la rete GARR.

Si distinguono WAN punto-punto (collegano direttamente due dispositivi attraverso un mezzo trasmissivo (e.g. cavo in fibra ottica, ponti radio)) e WAN a commutazione (collegano più di due punti di terminazioni (e.g. dorsali Internet)); le seconde sono dotate di elementi di commutazione (i.e. elaboratori specializzati utilizzati per connettere tra loro più linee di trasmissione).

Si veda la figura 2.

Tecniche di commutazione.

Si definiscono "tecniche di commutazione" le modalità con cui viene determinato il percorso sorgente-destinazione e vengono dedicate ad esso le risorse della rete. Si distinguono due tecniche: circuit switching (si parla di "reti a commutazione di circuito") e packet switching ("reti a commutazione di pacchetto").

Si veda la figura 3.

2.1 Circuit switching.

La rete determina un percorso dalla sorgente alla destinazione, su questo viene riservato e garantito un rate di trasmissione costante per la durata dell'intera sessione di comunicazione (una frazione della capacità di trasmissione dei link).

Un esempio di rete a commutazione è la rete telefonica fissa tradizionale.

Svantaggi: le risorse rimangono inattive quando non utilizzate e non vengono mai condivise, necessaria una fase di instaurazione della comunicazione (detta anche di “setup”) nella quale si configurano le tabelle di switching, la scarsa flessibilità nell’uso delle risorse può portare a “overprovisioning” e/o a un sottoutilizzo di queste in presenza di rate di traffico variabile.

Vantaggi: performance garantita, overhead limitato, tecnologie di switching efficienti.

2.2 Packet switching.

Il flusso di dati punto-punto viene suddiviso in pacchetti che condividono le risorse di rete e instradati singolarmente e indipendentemente dagli altri. Si parla di trasmissione “store and forward”: il commutatore (tipicamente un router) deve aspettare di aver ricevuto per intero il pacchetto prima di poter trasmettere sul collegamento in uscita.

Non c’è un canale dedicato e la sequenza dei pacchetti non segue uno schema prestabilito (si parla di “multiplexing statistico”): i router possono memorizzare i pacchetti nelle code (“buffer”) nel caso in cui il collegamento sia già usato alla massima capacità, introducendo dunque dei “ritardi di coda” e il rischio di “congestione”; non si ha garanzia nelle prestazioni e si corre il rischio di avere una perdita di pacchetti poiché i buffer hanno dimensione finita.

Svantaggi: alto overhead, tecnologie di inoltro non efficienti (si ha la necessità di selezionare l’uscita per ogni pacchetto), tempo di elaborazione ai router (si parla di “routing table lookup”), accodamento ai router.

Vantaggi: risorse trasmissive usate solo su richiesta, segnalazione non necessaria.

Internet.

Si definisce “internet” una rete costituita da due o più reti interconnesse; un esempio di internet è Internet, una rete a commutazione di pacchetti composta da migliaia di reti interconnesse che seguono l’Internet Protocol e rispettano convenzioni precise per nomi e indirizzi, che fornisce servizi di comunicazione alle applicazioni e per le applicazioni.

Le reti dei sistemi terminali sono connesse a Internet attraverso una gerarchia di fornitori di servizi internet (Internet Service Provider); definisco “dorsale” una ISP di primo livello. Si permette l’esistenza di “Internet eXchange Point”, definiti come punti di incontro per il peering tra due o più ISP.

Si veda la figura 4.

3.1 Metriche.

Si definiscono i seguenti parametri, utilizzati per misurare le prestazioni della rete:

- **Larghezza di banda (Bandwidth):** larghezza dell’intervallo di frequenze utilizzato dal sistema trasmissivo misurato in Hertz.
- **Velocità di trasmissione (Bitrate o transmission rate):** quantità di dati (bits) che possono essere trasmessi (“inseriti nella linea”) nell’unità di tempo (bits/secondo o bps) su un certo collegamento; dipende alla larghezza di banda e dalla tecnica trasmissiva utilizzata.
- **Throughput:** quantità di dati che possono essere trasmessi da un nodo sorgente a un nodo destinazione in un certo intervallo di tempo al netto di perdite sulla rete, duplicazioni, protocolli etc.
- **Latenza:** tempo richiesto affinché un messaggio arrivi a destinazione dal momento in cui il primo bit parte dalla sorgente; è definito come somma di ritardo di elaborazione, di accodamento, di trasmissione e di propagazione.

Si definiscono dunque i ritardi sopra elencati:

- **Ritardo di elaborazione:** è dovuto al controllo di errori sui bit e alla determinazione del canale di uscita.
- **Ritardo di accodamento:** è il tempo che un pacchetto spende all’interno del buffer (tipicamente di un router), è una quantità aleatoria.
- **Ritardo di trasmissione:** è il tempo impiegato per trasmettere un pacchetto sul link; definendo L lunghezza del pacchetto in bit, R rate di trasmissione sul link in bps, vale $\frac{L}{R}$.

- **Ritardo di propagazione:** è il tempo che 1bit impiega per essere propagato da un nodo all’altro; definendo d lunghezza del collegamento fisico, s velocità di propagazione nel mezzo, vale $\frac{d}{s}$.

Si definisce inoltre il “ritardo end-to-end” come sommatoria del ritardo ai singoli nodi:

$$R_{end-to-end} = \sum_{i=1}^N R_{nodo_i}.$$

Si ricorda l’esistenza dei comandi “traceroute”, che traccia un pacchetto dal proprio dispositivo all’host mostrando anche il numero di passaggi (salti) necessari per raggiungerlo assieme al ritardo dal mittente per ogni passaggio, e “ping”, che calcola il ritardo per la trasmissione dal dispositivo corrente all’host.

Modelli stratificati.

Si definisce un “protocollo” un insieme di regole che permettono a due entità di comunicare; nei sistemi di comunicazione non è sufficiente un singolo protocollo, si ricorre a una organizzazione di protocolli.

Si definisce inoltre “aperto” un insieme di protocolli i cui dettagli sono disponibili pubblicamente e i cui cambiamenti sono gestiti da una organizzazione la cui partecipazione è aperta al pubblico; definisco ”sistema aperto” un sistema che implementa protocolli aperti.

Per la stratificazione si fa riferimento ai principi di “separation of concern” (i.e. separazione degli interessi e delle responsabilità, fare ciò che compete, delegando ad altri tutto ciò che è delegabile) e “information hiding” (i.e. nascondere tutte le informazioni che non sono indispensabili per il committente per definire completamente un’operazione).

In concreto:

- Il modello permette di scomporre il problema in sottoproblemi più semplici il cui sviluppo è indipendente (da quello degli altri sottoproblemi).
- Ogni strato scambia informazioni con quelli adiacenti, ma comunica logicamente con il proprio omologo; fornisce informazioni allo strato superiore e usa i servizi offerti da quello inferiore.
- Un singolo strato svolge una sola funzione e realizza uno e un solo livello logico.

A questo punto l’obiettivo diventa realizzare una rete di calcolatori in cui qualsiasi terminale comunica con un qualsiasi fornitore di servizi mediante qualsiasi rete; lo standard per l’interconnessione di sistemi aperti è il modello ISO/OSI.

4.1 Modello ISO/OSI.

Si danno le seguenti definizioni nei termini del modello ISO/OSI:

- **Livello** (o **strato**): modulo interamente definito attraverso i servizi, protocolli e le interfacce che lo caratterizzano.
- **Servizio**: insieme di primitive (operazioni) che uno strato fornisce ad uno strato soprastante.
- **Interfaccia**: insieme di regole che governano il formato e il significato delle unità di dati (es. messaggi, segmenti o pacchetti) che vengono scambiati tra due strati adiacenti della stessa entità.
- **Protocollo**: insieme di regole che permettono a due entità omologhe (stesso strato) uno scambio efficace ed efficiente delle informazioni, definiscono il formato e l’ordine dei messaggi inviati e ricevuti tra entità omologhe della rete e le azioni che vengono fatte per la trasmissione e ricezione dei messaggi; occorre specificare sintassi e semantica di un messaggio e quali azioni intraprendere una volta ricevuto.

Per uno schema, si veda la figura 5.

I layer dal livello 5 al 7 sono di supporto all’elaborazione e interazione con l’utente (vengono realizzati a livello software), gli altri 4 alla rete e all’infrastruttura trasmissiva (realizzati a livello hardware e software); tipicamente i nodi intermedi implementano solo i primi 4.

Il modello prevede inoltre l’incapsulamento: il flusso di informazioni - che ha origine al livello applicativo - discende verso i livelli inferiori e viene progressivamente arricchita mediante l’aggiunta di headers.

Si danno in proposito le seguenti definizioni:

- **Header**: qualificazione del pacchetto dati al livello corrente.
- **Data**: payload proveniente dal livello precedente.

- **Trailer:** usato in funzione di trattamento dell'errore.

4.2 Stack protocollare TCP/IP.

TCP/IP è una famiglia di protocolli attualmente utilizzata in Internet. Si tratta di una gerarchia di protocolli, ciascuno dei quali fornisce funzionalità specifiche. Prende di riferimento il modello ISO/OSI e si articola nei seguenti 5 livelli (elencati dall'alto al basso):

- **Application Layer:** è di supporto alle applicazioni di rete, permette un collegamento logico end-to-end; si usa per lo scambio di pacchetti di informazioni detti “messaggi”. Include i protocolli HTTP, FTP, SMTP...
- **Transport Layer:** permette il trasporto di messaggi tra sistemi terminali; un pacchetto a questo livello prende il nome di “segmento”. Include i protocolli TCP (connection oriented, garantisce la consegna dei messaggi) e UDP (connectionless, non garantisce la consegna dei messaggi).
- **Network Layer:** si occupa dell'instradamento di pacchetti detti “datagrammi” dalla sorgente alla destinazione tipicamente attraversando una serie di router. Include i protocolli IP, ICMP.
- **Link Layer:** si occupa di trasferire pacchetti detti “frame” attraverso il collegamento tra elementi di rete vicini. Include i protocolli Ethernet, PPP, WiFi...
- **Physical Layer:** si occupa di trasferire individualmente i bit contenuti nei singoli frame da un nodo al successivo.

Si veda la figura 6.

4.2.1 Application Layer.

Si danno in merito le seguenti definizioni:

- **Application Programming Interface:** insieme di regole che un programmatore deve rispettare per l'utilizzo delle risorse.
- **Socket:** API che funge da interfaccia tra gli application e transport layer; due processi comunicano mandando dati alla socket (che fa da connessione a livello logico) il cui invio e ricezione sono responsabilità dei restanti quattro livelli dello stack TCP/IP nel sistema operativo. È la API di Internet.
- **Uniform Resource Identifier:** stringa di caratteri che identifica una risorsa (i.e. qualsiasi cosa abbia una identità) presente sulla rete. Se ne distinguono di due tipi:
 - **Uniform Resource Locator:** sottoinsieme di URI che identifica le risorse attraverso il loro meccanismo di accesso. Segue la sintassi <scheme>://<user>:<password>@<host>:<port>/<path>; “scheme” identifica il protocollo, “user” e “password” sono opzionali e generalmente deprecati, “host” è il nome di dominio di un host, “port” è il numero di porta del server, “path” identifica la risorsa nel contesto specificato.
 - **Uniform Resource Name:** sottoinsieme di URI che devono rimanere globalmente unici e persistenti anche quando la risorsa cessa di esistere e diventa non disponibile.

A questo livello si distinguono due modelli architetturali:

- **Client-Server:** esiste un host sempre attivo (il “server”) che serve le richieste di altri host (i “clients”).
- **Peer-to-Peer:** la comunicazione avviene tra coppie di dispositivi non sempre attivi (i “peer”) che possono offrire servizi e inviare richieste.

Per poter inviare dati risulta necessario poter indirizzare opportunamente il processo destinatario: per Internet si usa la coppia indirizzo IP (i.e. un intero a 32bit che identifica univocamente una macchina host), numero di porta (un intero a 16bit che identifica univocamente un processo all'interno di una macchina).

Il livello di trasporto mette a disposizione due protocolli diversi per la comunicazione:

- **TCP:** ha le seguenti caratteristiche:
 - **Connection-oriented:** è necessaria una fase di setup iniziale tra client e server prima di poter trasmettere messaggi (si parla di “handshaking”).
 - **Controllo del flusso:** il mittente non “inonderà” di dati il destinatario.
 - **Controllo di congestione:** “strangola” il mittente (client o server) quando la rete è sovraccarica.

- Garantisce il trasporto affidabile tra processo mittente e destinatario.
- Non offre garanzie di timing né di ampiezza minima di banda.

- **UDP:** ha le seguenti caratteristiche:

- **Connectionless:** non orientato alle connessioni, non prevede una fase di setup iniziale.
- Trasporto non affidabile.
- Non implementa meccanismi di controllo di flusso.
- Non implementa meccanismi per il controllo di congestione.
- Non offre garanzie di timing né ampiezza minima di banda.

Per degli esempi di applicazioni Internet, si veda la figura 7.

Si passa dunque alla trattazione dei protocolli implementati a questo livello:

- **HTTP:** ha le seguenti caratteristiche:

- **Stateless:** non ha memoria di stato: le coppie richiesta, risposta sono indipendenti; ogni richiesta viene eseguita indipendentemente da quelle che la han preceduta.
- **Richiesta/Risposta:** La connessione viene iniziata dal client, che invia un messaggio di richiesta detto “request” a cui il server risponde con un messaggio di risposta detto “response”.
- **Content negotiation:** permette di selezionare la rappresentazione appropriata per una risorsa (e.g. formato, lingua etc) quando viene servita richiesta.
- **Web caching:** si memorizzano copie temporanee di risorse Web (es. pagine HTML, immagini) e vengono servite al client per ridurre l’uso di risorse (e.g. banda, workload sul server) e il tempo di risposta al client.
- Implementa i “cookie”: un pacchetto di informazioni assegnato dal server alla prima richiesta del client che, a sua volta, lo legge dal messaggio di risposta (in cui trova una linea “set-cookie”) e lo salva in un file; il cookie viene poi aggiunto dal client a tutte le successive richieste a quel sito (conterranno la linea “cookie:<cookie>”).
- Usa TCP.
- HTTP/1.0 permetteva solo connessioni non persistenti, HTTP/1.1 prevede che - di default - le connessioni siano invece persistenti e vengano mantenute fino a una richiesta di chiusura. I server che rispettano HTTP/1.1 supportano inoltre il pipelining (i.e. consiste nell’invio da parte del client di molteplici richieste senza aspettare la ricezione di ciascuna risposta, il server deve inviare le risposte nello stesso ordine in cui sono state ricevute; riguarda solo metodi idempotenti); implementato in pochi browser, introduce il rischio di Head of Line Blocking (i.e. se una richiesta richiede tempo per essere processata le risposte a quelle successive sono bloccate).

Per la struttura dei messaggi HTTP, si veda la figura 8.

I metodi messi a disposizione dal protocollo sono:

- **OPTIONS:** richiede informazioni sulle opzioni di comunicazione (i.e. quali metodi) sono associati a una URL o al server stesso.
- **GET:** richiede il trasferimento di una risorsa identificata da una URL o operazioni associate all’URL stessa; la semantica può essere modificata in un “conditional GET” se l’header contiene uno dei campi “If-Modified-Since”, “If-Unmodified-Since”, “If-Match”, “If-None-Match” o “If-Range”, “partial GET” se contiene il campo “range”.
- **HEAD:** identico al metodo GET con l’eccezione che la risposta non contiene il campo “message-body”.
- **POST:** serve per inviare dal client al server informazioni inserite nel body del messaggio, la funzione effettiva determinata è dal server e dipendente tipicamente dalla Request-URI; tipicamente non è abilitato sui server pubblici.
- **PUT:** il client chiede al server di creare/modificare una risorsa.
- **DELETE:** il client chiede di cancellare una risorsa.
- **TRACE.**

Si definiscono “safe” i metodi che non modificano la rappresentazione di una risorsa (i.e. GET, HEAD, OPTIONS, TRACE), “idempotent” quelli che godono della proprietà di idempotenza (i.e. GET, HEAD, PUT, DELETE, OPTIONS, TRACE).

- **TERminaL NETwork:** ha le seguenti caratteristiche
 - Prevede un programma server che accetta le richieste e un programma client che effettua le richieste; quest'ultimo interagisce con il terminale utente sull'host locale e scambia messaggi con il Telnet server.
 - Usa TCP e persiste per l'intera durata della sessione, il client si connette alla porta 23 del server.
 - Il client accetta le battute di tasti del terminale e le invia al server nel quale un software chiamato “pseudo terminal driver” fa da entry point del sistema operativo: consente di trasferire caratteri a un processo come se provenissero dal terminale. Il client accetta i caratteri che il server manda indietro e li visualizza sul terminale utente.
 - I messaggi di controllo iniziali sono usati per scambiare informazioni sulle caratteristiche degli host (si parla di “Telnet option negotiation”).
 - Il client invia (in chiaro) un login identifier e una password (non è sicuro, è stato sostituito da “SSH”). Il terminale in esecuzione sia sul client sia sul server è un “Network Virtual Terminal”: gli host traducono le loro caratteristiche locali così da apparire esternamente come un NVT e assumono che l'host remoto sia un NVT; si scambiano dati e comandi in formato 7-bit US-ASCII sullo stesso canale (prevede dunque “in-band signaling”), ogni carattere è inviato come un ottetto con il primo bit settato a 0 se si tratta di un dato, a 1 per i comandi.
- Si veda la figura 9.
- **Simple Mail Transfer Protocol:** si danno in merito le seguenti definizioni:
 - **User agent:** componente che permette l'invio, la modifica e la lettura di messaggi di posta elettronica.
 - **Mail server:** componente che trasferisce i messaggi di posta elettronica.
 - **Indirizzo email:** stringa che identifica un destinario, ha la struttura “<local-part>@<domain-name>”: la prima parte identifica il destinatario all'interno del mail server, la seconda (identifica) il mail server.
 - **Tipo MIME:** specifica la natura dei dati nel corpo di un'entità MIME.

Il protocollo ha le seguenti caratteristiche:

- Ha come obiettivo **trasferimento affidabile ed efficiente di email**.
- Implementa un meccanismo di **scambio formale di responsabilità**: i client hanno come responsabilità trasferire email a un server SMTP o comunicare un eventuale insuccesso.
- È indipendente dal sistema di trasmissione usato e richiede solo il **trasferimento di stream di byte ordinato e affidabile** (tipicamente, si usano connessioni TCP su porta 25).
- Ha la capacità di **trasportare mail attraverso più reti** (i.e. un messaggio di mail può passare attraverso server intermedi nel percorso da mittente a destinatario finale).
- I messaggi sono costituiti da una componente **header** (i.e. linee di intestazione, tra cui “To:”, “From”, “Subject”) e un **body** (i.e. il corpo del messaggio formato da soli caratteri ASCII 7bit).
- Le interazioni tra client e server sono del tipo **command/response**: a ogni comando (testo ASCII 7 bit) corrisponde una risposta formata da codice di stato e optionalmente una descrizione.
- È un protocollo di tipo **push**.

I mail server implementano una tecnica denominata “spooling”: al momento dell'invio di un messaggio, il sistema (client) copia messaggio, identificativo del mittente, del destinatario e della macchina di destinazione e tempo di deposito in un'area di memoria detta “spool”; il client stabilisce una connessione TCP con la macchina destinazione, se questo ha successo allora la copia locale viene eliminata; il processo di trasferimento scandisce periodicamente lo spool ed esegue la procedura di trasferimento dei messaggi non consegnati, dopo un intervallo di tempo definito dall'amministratore del server se un messaggio non è stato ancora consegnato allora l'utente mittente viene notificato.

Si veda la figura 10.

Si identificano tre fasi per il trasferimento SMTP:

- **Handshaking:** nella fase di setup iniziale, il client stabilisce la connessione e attende dal server il messaggio di risposta “220 READY FOR MAIL”, che indica la disponibilità a ricevere posta.
- **Trasferimento del messaggio.**
- **Chiusura della connessione.**

I mail server moderni permettono la negoziazione per l'invio di dati in codifica binaria (quindi a 8bit), se non

ha successo si inviano caratteri ASCII 7bit seguendo il **protocollo MIME**. MIME fornisce regole di codifica e decodifica per trasformare caratteri speciali e contenuti multimediali in caratteri ASCII 7bit (in questo modo, si garantisce la retrocompatibilità con i mail server e non con gli user agent), permette inoltre l'uso di più codifiche all'interno dello stesso messaggio con il tipo "multipart", che specifica il "boundary" (confine) tra ciascuna.

Per indicare la fine di un messaggio si usa la sequenza di caratteri "<CLRF>.<CLRF>". Vengono messi a disposizione i seguenti comandi:

- **HELO** <client identifier>;
- **MAIL FROM:<reverse-path><CRLF>;**
- **RCPT TO:<forward-path><CRLF>;**
- **DATA;**
- **QUIT.**

Si veda la figura 11.

SMTP è un protocollo di tipo "push", per la ricezione risulta necessario un protocollo di tipo "pull". Alcuni esempi sono: Post Office Protocol (**POP**), Internet Mail Access Protocol (**IMAP**), **HTTP** (quando user agent è un browser).

- **Domain Name Server:** si danno in merito le seguenti definizioni:

- **Nome:** identifica un oggetto con una sequenza di caratteri scelti da un alfabeto finito; si usa per motivi mnemonici e per disaccoppiare il livello applicativo da quello di rete.
- **Indirizzo:** identifica la locazione dell'oggetto (i dispositivi connessi alla rete vengono individuati mediante indirizzi da 4byte detti "indirizzi IP").
- **Dominio:** sottoalbero nello spazio dei nomi di dominio che viene identificato dal nome di dominio del nodo in cima al sottoalbero.
- **Sottodomini:** insieme dei domini in cui si può suddividere un dominio.
- **Name server:** programma che gestisce la conversione da nome di dominio a indirizzo IP.

DNS offre i seguenti servizi:

- **Risoluzione di nomi** di alto livello ("hostname") in indirizzi IP.
- **Host aliasing:** un host può avere più nomi, vengono tradotti nel nome canonico e - di seguito - nell'indirizzo IP.
- **Mail server aliasing:** sinonimi per mail server; e.g. nome identico per mail server e web server).
- **Distribuzione carico:** un hostname canonico può corrispondere a più indirizzi IP, il DNS restituisce la lista variandone l'ordine per ogni risposta.

Concretamente, DNS è costituito da:

- **Schema di assegnazione dei nomi** gerarchico e basato su domini.
- **Database distribuito** contenente i nomi e le corrispondenze con gli indirizzi IP implementato con una gerarchia di name server ("record DNS").
- **Protocollo per la distribuzione delle informazioni sui nomi tra name server:** host, router, name server comunicano per risolvere nomi (traduzione nome/indirizzo) utilizzando tipicamente UDP con porta 53.

Si vedano le figure 12 e 13.

A livello pratico, garantire una struttura gerarchica per i nomi permette di delegarne l'assegnazione a singoli organizzazioni e distribuire la conversione di nomi e indirizzi.

Lo spazio dei domini ha una struttura ad albero; ogni nodo dell'albero ha un **nome di dominio**, ossia una sequenza di diverse etichette (i.e. stringhe) separate da punti (e.g. "lab3"."di"."unipi"."it"). Inoltre, si definisce **zona** la regione (tipicamente una parte contigua dell'albero) di cui è responsabile un name server; per esempio, in "di.unipi.it" zona e dominio coincidono, lo stesso discorso non vale per "unipi.it".

Per esempi di query DNS, si veda la figura 14.

Si distinguono i seguenti ruoli:

- **Server Top-Level Domain:** mantiene le informazioni dei nomi di dominio che appartengono a un certo TLD, restituisce le informazioni sui name server di competenza dei sottodomini.

- **Server Radice**: responsabile dei record della zona radice e restituisce le informazioni sui name server di TLD.
- **Server di Competenza (Authoritative Name Server)**: autorità per una certa zona, memorizza nome e indirizzo IP di un insieme di host dei quali può effettuare traduzioni nome/indirizzo; per una certa zona ci possono essere server di competenza primari e secondari: quelli primari mantengono il file di zona, quelli secondari ricevono il file di zona e offrono il servizio di risoluzione.
- **Local Name Servers**: non appartengono strettamente alla gerarchia dei server, ogni ISP (università, società, ISP) ha il proprio; a questi vengono inizialmente rivolte le query DNS: operano da proxy e inoltrano la query in una gerarchia di server DNS.

Spesso, i server DNS implementano al loro interno un meccanismo di caching su cui salvano dei record corrispondenti ai risultati delle query; i record vengono definiti come una quadrupla formata dai campi “name”, “value”, “type”, “ttl” (time to leave).

Dal parametro “type” dipende il significato della coppia “name” e “value”; si citano alcuni esempi:

- **A**: “name” sta per hostname, value per indirizzo IP.
- **CNAME**: “name” sta per hostname (o un suo sinonimo), “value” per il nome canonico dell’host.
- **NS**: “name” sta per nome di dominio, “value” per l’authoritative name server per quel dominio.
- **MX**: “name” sta per nome di dominio, “value” per il nome canonico del server di posta associato a quel dominio.

Il protocollo DNS ha le seguenti caratteristiche:

- Usa connessione **UDP sulla porta 53**.
- Permette uso di **TCP** (tipicamente usata per messaggi di grandi dimensioni tra server DNS)
- **Messaggi di query e reply hanno lo stesso formato**. Per la struttura dei messaggi, si veda la figura 15.
- Ha un **header** formato dai seguenti campi:
 - * **identification**: intero a 16bit uguale per la coppia query, reply.
 - * **flags**: 0 se query, 1 se risposta.
 - * **autorevolezza** della risposta.
- **Domande**: campi per il nome richiesto e il tipo di query
- **Risposte**: RR nella risposta alla domanda
- **Competenza**: record relative ai server di competenza
- **Informazioni aggiuntive**

Essendo per natura DNS altamente centralizzato, una query attraversa una serie di server prima di restituire il risultato: in questo modo, si introduce il rischio di “DNS hijacking” (consiste nel restituire risposte non corrette alle query, si reindirizza il client verso siti malevoli). Si veda la figura 16.

Per interrogazioni a server DNS si usa il tool ”dig”.

- **File Transfer Protocol**: ha le seguenti caratteristiche:

- Permette control connection (i.e scambio di comandi e risposte tra client e server). Segue il protocollo Telnet.
Il client FTP contatta il server FTP alla porta 21, ottiene l’autorizzazione sulla connessione di controllo e invia i comandi su di questa (e.g. cambio directory, invio file, ecc.). La connessione di controllo è persistente
- Permette data connection (i.e. connessione su cui i dati sono trasferiti con modi e tipi specificati). I dati trasferiti possono essere parte di un file, un file o un set di file.
Il server apre una connessione dati TCP con il client (caveat: non è il client a farlo, si parla di “modalità attiva”), trasferisce il file sulla connessione dati e, terminata l’operazione, il server chiude la connessione. La connessione dati non è persistente, ne viene aperta e chiusa una per ciascun trasferimento.
- Si usa il protocollo di trasporto TCP, per cui si permettono due differenti modalità: quella “attiva” (descritta precedentemente, usata di default) e quella “passiva” (in questo caso il client chiede al server di mettersi in ascolto su una porta per una connessione dati, ottiene questo numero di porta - tipicamente 20 - e lo usa per aprire la connessione con il server).
- È un protocollo stateful.

- Fa uso di codici di ritorno.
- Il collegamento può essere anonimo (si parla di “anonymous FTP”) o reso sicuro con TLS.
- Implementa più modalità di trasmissione, come:
 - * **Stream mode**: FTP invia i dati a TCP con un flusso continuo di bit.
 - * **Block mode**: FTP invia i dati a TCP suddivisi in blocchi. Ogni blocco è preceduto da un header.
 - * **Compressed mode**: si trasmette il file compresso.

Si veda la figura 17.

I comandi messi a disposizione sono i seguenti:

- **USER <username>**.
- **PASS <password>**.
- **LIST**: elenca i file della directory corrente.
- **NLST**: richiede elenco file e directory.
- **RETR <filename>**: recupera filename dalla directory corrente.
- **STOR <filename>**: memorizza (PUT) filename nell’host remoto.
- **ABOR**: interrompe l’ultimo comando ed i trasferimenti in corso.
- **PORT**: indirizzo e numero di porta del client.
- **SYST**: il server restituisce il tipo di sistema.
- **QUIT**: chiude la connessione.

4.2.2 Transport Layer.

Si danno le seguenti definizioni:

- **Multiplexing**: provvede all’“accorpamento” dei flussi dati dai processi verso la rete, li “imbusta” con un preambolo. È basato sui socket address dei processi.
- **Demultiplexing**: provvede allo “smistamento” dei pacchetti fra la rete e le applicazioni (i.e. dirige i dati ricevuti al processo corretto). È basato sui socket address dei processi.
- **Porta**: intero senza segno a 16bit assegnato a un punto di demultiplexing dei protocolli TCP o UDP. Si distinguono tre differenti categorie di porte: “system ports” (da 0 a 1023, identificano processi server, assegnate da IANA), “user ports” (da 1024 a 49151, assegnate da IANA), “dynamic ports” (le restanti, non assegnate da IANA).
- **Indirizzo IP**: indirizzo di 32bit presente nello stack TCP/IP.
- **Socket address**: identificativo formato dalla coppia “porta”, “indirizzo IP”.
- **Socket TCP**: identificativo formato dalla quadrupla “indirizzo IP di origine”, “numero di porta di origine”, “indirizzo IP di destinazione”, “numero di porta di destinazione”.

Offre i seguenti servizi:

- **Multiplexing**.
- **Demultiplexing**: se ne distinguono due tipi:
 - **senza connessione (UDP)**: lo strato di trasporto dell’host ricevente consegna il segmento UDP alla socket identificata dal socket address destinazione; i datagrammi inviati da mittenti differenti e con stessa destinazione vengono consegnati alla stessa socket.
 - **con connessione (TCP)**: lo strato di trasporto dell’host ricevente usa i parametri della socket TCP per inviare il segmento alla socket appropriata (e.g. un server Web può creare una socket per ogni client connesso); un host server può supportare più socket alla volta.

Si veda la figura 18.

- **Protocollo TCP**: ha le seguenti caratteristiche:
 - **Orientato allo stream**: i dati sono un flusso di byte ordinati ma non strutturati; numeri i byte (NON i segmenti). Si definiscono in merito “numero di sequenza” associato a un segmento il numero (nel flusso) del primo byte del segmento e “numero di riscontro” il numero dell’ultimo byte correttamente ricevuto + 1.
 - **Orientato alla connessione**: i processi effettuano un handshake (i.e. informazioni preliminari per

preparare lo scambio dei dati) prima dello scambio dei dati; si definisce orientato poiché lo stato della connessione risiede sui punti terminali e non sugli elementi intermedi della rete.

- **Connessione full-duplex:** il flusso dati tra due host può avvenire contemporaneamente nelle due direzioni.
- Garantisce trasferimento dati ordinato e affidabile (i.e. corregge eventuali errori).
- Implementa multiplexing/demultiplexing.
- Implementa meccanismi di controllo di flusso (i.e. evita di spedire più dati di quanti il ricevitore possa elaborare).
- Implementa meccanismi di controllo di congestione (i.e. gestisce opportunamente situazioni di sovraccarico della rete).
- Suddivide il flusso di byte in segmenti in modo indipendente dal programma applicativo che li ha generati (si parla di “trasferimento bufferizzato”); si veda in merito la figura 19.

Per il formato dei segmenti TCP si veda la figura 20; un segmento TCP permette “selective acknowledgement” (nel campo “Opzioni”, non supportato da tutte le implementazioni) che permette di specificare quali segmenti sono stati ricevuti, e supporta i seguenti flags (in grassetto i più importanti):

- **URG:** Il campo Puntatore Urgente contiene dati significativi da trasferire in via prioritaria.
- **ACK:** Il campo Numero di Riscontro contiene dati significativi.
- **PSH:** Funzione Push (trasferimento immediato dei dati in un segmento dal trasporto al livello applicativo).
- **RST:** Reset della connessione, implica una chiusura forzata della connessione.
- **SYN:** Sincronizza il Numero di Sequenza. Messo a 1 durante l’handshake.
- **FIN:** Non ci sono altri dati dal mittente, implica la chiusura della connessione.

Si individuano tre fasi nella gestione di una connessione:

- **Handshake:** fase di setup iniziale, viene detto **a tre vie** per il numero di passaggi necessari. Si veda la figura 21.

I primi segmenti non hanno carico utile, non portano nessuna informazione. All’arrivo del primo segmento, il server inizializza due buffer (memorie di scambio) e le variabili, necessari per il controllo del flusso e della congestione; all’arrivo del riscontro del primo segmento, il client alloca due buffer e le variabili, necessari per il controllo del flusso e della congestione; alla ricezione del terzo segmento, la connessione è instaurata.

- **Trasferimento dati:** si distinguono due ruoli:

- * **Mittente:** TCP riceve i dati dall’applicazione, incapsula i dati in uno o più segmenti, assegna il numero di sequenza e avvia il timer di ritrasmissione. I dati vengono ritrasmessi se e solo se o si raggiunge il valore di **timeout** o si ricevono **tre ACK duplicati**: nel primo caso, si ritrasmette il segmento che non è stato riscontrato (e che ha causato il timeout), poi si riavvia il timer; nel secondo caso, il segmento successivo a quello riscontrato è andato perso, si fa “ritrasmessione veloce” (“fast retransmission”) prima della scadenza del timer (si veda in merito la figura 22).

Nel caso di segmenti fuori sequenza, i dati possono essere memorizzati dall’entità TCP destinataria (nota: TCP non dice come il destinatario deve gestire i pacchetti fuori sequenza) temporaneamente; nelle versioni più recenti si implementa “SACK”: i pacchetti ricevuti fuori sequenza vengono memorizzati e il riscontro di pacchetti fuori sequenza e duplicati viene inviato in OPTIONS.

Per uno schema riassuntivo delle funzionalità del mittente, si veda la figura 23.

- * **Destinatario:** si danno le seguenti regole:

- i. Tutti i segmenti inviati per trasmettere dati includono ACK.
- ii. Se il destinatario non ha dati da inviare e riceve un segmento “in ordine” allora ritarda l’invio di ACK di 500ms a meno che non riceva un nuovo segmento.
- iii. Se il destinatario riceve il segmento atteso e il precedente non è stato ancora riscontrato, allora invia immediatamente ACK.
- iv. Se il destinatario riceve un segmento fuori sequenza, allora invia immediatamente ACK indicando il prossimo numero atteso.

- v. Se il destinatario riceve un segmento che era mancante (un “buco” in una sequenza), allora invia immediatamente ACK indicando il prossimo numero atteso.
- vi. Se il destinatario riceve un segmento duplicato, allora invia immediatamente ACK indicando il prossimo numero atteso.

Per uno schema riassuntivo delle funzionalità del destinatario, si veda la figura 24.

Per alcuni esempi, si vedano le figure 25, 26, 27 e 28.

Si definisce **RTT** il tempo trascorso dall’invio di un segmento a quando se ne riceve il riscontro.

Il tempo di timeout **RTO** deve essere maggiore di RTT, dipende dalla sua stima e dalla stima della sua variabilità

$$RTT_{estimated} = (1 - \alpha) * RTT_{estimated} + \alpha * RTT_{sample} \text{ con tipicamente } \alpha = 0.125,$$

$$RTT_{dev} = (1 - \beta) * RTT_{dev} + \beta * |RTT_{sample} - RTT_{estimated}| \text{ con tipicamente } \beta = 0.25;$$

$$RTO = RTO_{estimated} + 4 * RTT_{dev}.$$

In molte implementazioni, dopo un errore (e.g. ACK non ricevuto) si raddoppia il timeout.

Si definisce inoltre **finestra di trasmissione** (o **sliding window**) una finestra sovrapposta alla sequenza da trasmettere, negoziata dinamicamente e fatta avanzare alla ricezione di un ACK; analogamente, per il processo destinatario di definisce la **finestra di ricezione**. Si vedano le figure 29, 30.

Concretamente, il meccanismo di controllo di flusso sopra menzionato consiste nella capacità del processo mittente di evitare di saturare il buffer del ricevitore; viene implementato utilizzando - lato mittente - la **finestra di ricezione** (o **rwnd**): viene comunicato nel campo “window” dell’header TCP e indica quanto spazio è ancora a disposizione nel buffer del destinatario. Quando vale zero, il mittente invia dei segmenti “sonda” da 1byte per ricevere un aggiornamento sulla sua dimensione. Si veda la figura 31.

Invece, il meccanismo di controllo di congestione consiste nella capacità dei sistemi terminali di stimare la capacità della rete, è basato sulla **congestion window** (o **cwnd**) - misurata in **Maximum Segment Size** (massima quantità di dati trasportabili da un segmento, dipende dalla massima unità trasmissiva e viene scelto in modo tale che il segmento stia dentro un singolo frame di collegamento quando incapsulato in un pacchetto IP (i.e. sottrago dalla massima unità trasmissiva la dimensione in bytes dell’header TCP e dell’header IP); la frequenza di invio dei dati non supera $\frac{cwnd}{RTT}$); di fatto, la dimensione della finestra di invio è il minimo tra rwnd e cwnd. Si osserva inoltre che - indicando con W il valore massimo in byte della finestra - TCP offre come throughput medio $\frac{0.75W}{RTT}$. L’algoritmo utilizzato per il controllo di congestione è costituito da tre passi:

- * **Partenza lenta** (o **slow start**): tipicamente, la finestra di congestione viene posta pari a 1MSS; per ogni ACK non duplicato si incrementa di 1MSS. Gli ACK ritardati rallentano la velocità di incremento della finestra (si veda la figura 32).
- * **Incremento additivo e decremento moltiplicativo** (o **AIMD**): il TCP del mittente aumenta la propria finestra di congestione a ogni ACK ricevuto in modo tale da avere per ogni RTT una crescita pari a 1MSS; tipicamente, la dimezza a ogni evento di perdita. Si veda la figura 37; si rimanda inoltre alle figure 33, 34), 35 e 36 per le differenze tra le varianti TCP Reno e TCP Tahoe.
- Questo meccanismo permette - nel caso in cui N connessioni abbiano stesso MSS e RTT, non ci siano altri protocolli sullo stesso link e che questo abbia capacità Rbit al secondo - che ogni connessione tenda a trasmettere $\frac{R}{N}$ bit al secondo.
- * **Ripresa veloce** (o **fast recovery**).
- * **Reazione ai timeout**.

- **Chiusura della connessione**: client e server chiudono ciascuno il proprio lato della connessione, viene inviato un segmento TCP con bit FIN uguale a 1; ciascuno risponde al FIN con un ACK (eventualmente, ACK può venire combinato col proprio FIN). Risulta possibile anche lo scambio simultaneo di FIN. Si implementa inoltre lo stato di “half-close”: uno dei due processi smette di inviare dati mentre ne sta ancora ricevendo; si veda in proposito la figura 38.

Per alcuni esempi, si vedano le figure 39, 40 e 41.

Durante una connessione TCP vengono attraversati più stati dai dispositivi terminali, tra questi si menziona

lo stato **TIME_WAIT**: indica che il dispositivo si trova adesso in attesa per un periodo di tempo lungo due volte la “Maximum Segment Lifetime” in modo tale da assicurarsi che l’altro endpoint abbia riconosciuto la richiesta di chiusura della connessione; questo stato viene utilizzato per consentire l’eliminazione di segmenti duplicati in rete e implementare in maniera affidabile la terminazione della connessione in entrambe le direzioni.

- **Protocollo UDP:** Si danno in merito le seguenti definizioni:

- **Checksum:** è un meccanismo di controllo dell’errore end-to-end; prevede l’eliminazione di pacchetti corrotti senza che il mittente riceva una notifica a riguardo.

Ha le seguenti caratteristiche:

- È orientato al messaggio: ogni datagramma risulta indipendente dal precedente.
- È un servizio di consegna a massimo sforzo: i datagrammi possono andare persi o essere inviati fuori sequenza.
- Non prevede l’uso di una connessione (quindi non si introduce ritardo - TCP prevedeva almeno 1RTT per l’handshake iniziale); l’header è lungo 8byte.
- Non prevede meccanismi di controllo di congestione o di flusso.
- Il checksum è facoltativo. Si faccia riferimento alla figura 42 per la struttura del campo checksum.

Viene usato spesso in applicazioni multimediali con tolleranza a piccole perdite e sensibilità alla frequenza, nel DNS e in SNMP.

Per la struttura dei datagrammi UDP si faccia riferimento alla figura 43.

Per l’uso del campo checksum si prendono in considerazione due ruoli:

- **Mittente:** viene inizialmente azzerato; si tratta il contenuto del datagramma UDP come una sequenza di parole da 16bit e si assegna al campo la somma delle parole in complemento a uno (i.e. viene negato).
- **Ricevente:** viene calcolata la checksum del segmento ricevuto; se non vale 0 allora il messaggio è corrotto e viene scartato.

4.2.3 Network Layer.

È implementato nei dispositivi intermedi e offre una astrazione che permette a host e reti eterogenee di funzionare (logicamente) come una singola rete; Mette a disposizione il **protocollo IP**. Ha le seguenti caratteristiche:

- È di tipo **connectionless**: non esiste un circuito né fisico né virtuale tra due sistemi terminali a livello IP.
- È di tipo **best effort**: non garantisce né la consegna dei pacchetti né l’ordine.
- Non è affidabile: non ci sono meccanismi di recupero di errore.
- Non prevede garanzie sulla qualità del servizio (QoS), sul tempo di consegna o sul controllo di flusso.
- Altri servizi sono aggiungibili mediante estensioni (e.g. DiffServ è un modello di QoS implementato dai router).

Mette a disposizione due funzioni: **inoltro** o **forwarding** (trasferimento del pacchetto sull’appropriato collegamento di uscita, usa la “tabella di inoltro”) e **instradamento** o **routing** (processo decisionale di scelta del percorso verso una destinazione, determina i valori da inserire nella tabella di inoltro tramite algoritmi di “routing”); si vedano in merito le figure 44 e 45.

Si distinguono due tipi di inoltro (si rimanda inoltre alla figura 46):

- **Inoltro diretto:** il pacchetto IP ha come destinazione un host appartenente alla stessa rete, non vengono coinvolte altre entità. Siano A l’host destinatario e B il mittente: B verifica che A appartiene alla stessa rete e consulta una tabella presente in ogni dispositivo terminale in cui vengono salvate le coppie (MAC, indirizzo IP) di ogni dispositivo (presente nella stessa rete).
- **Inoltro indiretto:** il pacchetto IP ha come destinazione un host appartentente a un’altra rete: si delega l’invio a un **router**. Siano A l’host destinatario e B il mittente: B verifica che A non appartiene alla stessa rete, invia il datagramma al router la cui coppia (MAC, indirizzo IP) è salvata nella tabella menzionata sopra.

La tabella viene detta **tabella di inoltro** e permette “aggregazione degli indirizzi” (i.e. riduce al minimo il numero di entries della tabella raccogliendo a fattor comune) e “routing gerarchico” (e.g. un ISP che divide i propri indirizzi in più blocchi non dichiara il loro indirizzo specifico bensì il proprio e, al momento opportuno, si occupa di inoltrare

alla sottorete corretta), segue la regola del “longer mask matching” (i.e. ordina le entries della tabella in ordine decrescente sulla lunghezza della maschera e - quando si deve fare il pattern matching - sceglie il primo pattern valido, ossia il più specifico).

Si distinguono due tipi di routing: **decentralizzato** (prevede che l’algoritmo di routing sia in esecuzione su ciascun router e che questi si scambino messaggi) e **centralizzato** (si parla di **Software Defined Networking**, un controller remoto interagisce con Control Agents locali: questi inviano informazioni su traffico e collegamenti, ricevono i valori da inserire nella tabella di inoltro); si vedano in merito le figure fig:ip-routing-decentralizzato e fig:ip-routing-centralizzato (per ulteriori informazioni riguardo ai routers, si rimanda alla [sezione dedicata](#)).

Si distinguono inoltre due meccanismi: **routing statico** (le righe della tabella vengono compilate manualmente da un operatore) e **routing dinamico** (protocolli specifici provvedono automaticamente alla compilazione della tabella).

Gli algoritmi di routing si distinguono in **globali** (si basano sulla conoscenza completa della rete in questione) e **decentralizzati** (nessun nodo conosce la topologia dell’intera rete, ogni nodo ha informazioni su se stesso e i propri vicini con cui scambia informazioni); un esempio di algoritmo decentralizzato è quello di **distance vector** basato sull’equazione di Bellman-Ford ($d_{i,j} = \min(d_{i,j}, (c_{i,k} + d_{k,j}))$) di cui si fornisce di seguito lo pseudocodice:

```
function DISTANCEVECTOR
    D [self] ← 0
    for i ← 1 to N do
        if Vicino(self, i) then
            D [i] ← c[self][i]
        else
            c[self][i] ← ∞
        end if
    end for
    for all n : Neighbor(self) do
        Send(D, n)
    end for
    while true do
        Wait(Receive(D), NewCost)
        for i ← 1 to N do
        end for
    end while
end function
```

Anche al livello di rete vengono implementati **multiplexing** e **demultiplexing** (ad esempio, l’intestazione dei datagrammi contiene un campo in cui viene codificato il protocollo a cui deve essere consegnato il payload).

Il modello datagram (senza connessione) per la consegna dei dati permette **segmentazione** e **riunificazione** dei pacchetti, **controllo degli errori** e **verifica del TTL**; i datagrammi IP prevedono l’uso dei seguenti campi (si veda anche la figura 48).

- **Versione** (4bit): specifica la versione di IP usata (attualmente IPv4 o IPv6).
- **Hlen** (lunghezza dell’header, 4 bit): lunghezza dell’intestazione espressa in parole da 32bit (tipicamente 0101, ossia 20bit).
- **Tipo di servizio** (8bit): serve per “colorare” il datagramma IP (basso ritardo, affidabilità, etc.); 6bit per **Differentiated Services** (i pacchetti vengono marcati in base alla classe di servizio, i router avranno una politica di accodamento specifica per ciascuna; alcuni esempi di classi di servizio sono telefonate, controlli, multimedia streaming etc.), 2bit per **Explicit Congestion Notification** (supporto a livello rete e trasporto per la notifica di eventi di congestione).
- **Lunghezza del datagramma** (16bit): è la lunghezza di tutto il datagramma in byte, header incluso; il massimo teorico è di 65535byte, nella pratica di 1500byte.
- **Identificatore** (16bit): è unico per ciascun datagramma, viene assegnato dall’host sorgente e usato dal desti-

natario per riconoscere i frammenti di un datagramma da riassemblare.

- **Flag** (3bit): il primo bit vale sempre 0, il secondo vale 1 se il pacchetto non deve essere frammentato (0 altrimenti), il terzo vale 1 se e solo se il pacchetto è l'ultimo frammento.
- **Offset** (13bit): indica la posizione relativa (come multiplo di 8byte) del frammento all'interno del datagramma.
- **Tempo di vita** (8 bit): ad ogni passaggio da un router viene decrementato, quando raggiunge lo zero viene scartato. Assicura che un pacchetto non rimanga in ciclo nella rete.
- **Protocollo** (8bit): in ricezione all'host destinatario indica quale protocollo dello strato superiore deve ricevere i dati.
- **Checksum dell'intestazione** (16bit): viene calcolato il checksum della sola intestazione (ponendo questo campo pari a zero) ad ogni router (il TTL cambia ad ogni hop!). Se si ottiene un errore si scarta il datagramma.
- **Indirizzi sorgente e destinazione** (32bit): indirizzi IP del mittente e del destinatario.
- **Opzioni** (lunghezza variabile, multiplo di 32bit): uso sporadico, da 0 a 40 byte.
- **Dati**: dati trasportati dal datagramma IP.

Nel caso in cui la MTU non risulta sufficiente per trasmettere per intero un datagramma, IP prevede la possibilità di frammentare (ved. figura 47); IPv6 non implementa questo meccanismo, in IPv4 è fortemente sconsigliato.

Per il meccanismo di indirizzamento, IP prevede l'uso di **indirizzi IP**, 32bit scritti in notazione decimale puntata formati da un **prefisso** (che indica la rete) e **suffisso** (che indica l'host all'interno della rete). Inizialmente si erano divisi gli indirizzi IP facendo **classful addressing**:

- **Classe A**: 7bit per reti IP, 24bit (0.0.0.0 - 127.255.255.255)
- **Classe B**: 14bit per reti IP, 16bit host id (128.0.0.0 - 191.255.255.255)
- **Classe C**: 21bit per reti IP, 8bit per host id (192.0.0.0 - 223.255.255.255)
- **Classe D**: riservata a multicasting (224.0.0.0 - 239.255.255.255)
- **Classe E**: riservata per usi futuri (240.0.0.0 - 255.255.255)

Questa soluzione risulta poco flessibile nell'utilizzo dello scarso range di indirizzi disponibili, IPv6 risolve con indirizzi IP da 128bit, IPv4 con **classless addressing** (ved. figura 49); per questi indirizzi si usa la notazione CIDR che prevede il formato “<byte.byte.byte.byte>/<lunghezza del prefisso>” dove il prefisso indica la parte dedicata all'indirizzo di rete, il resto identifica l'host all'interno di quella rete.

Si definisce inoltre la **subnet mask**, un numero composto da 32bit in cui i primi n bit a sinistra identificano la rete e sono impostati a 1, i rimanenti a 0; serve a distinguere quale parte di un indirizzo IP identifica la rete e quale l'host. Partendo da subnet mask e indirizzo IP, l'indirizzo di rete si calcola facendo l'AND.

Gli indirizzi IP vengono assegnati dall'ICANN agli ISP in blocchi di dimensione 2^k con $k \in \mathbb{N}$, che li dividono in blocchi di lunghezza 2^i con $i \in \mathbb{N} \wedge i \leq k$ contigui: in questo caso, la lunghezza del prefisso di rete vale $32 - i$; nel caso in cui le dimensioni dei blocchi non siano costanti, si assegnano in ordine decrescente (ordinando sulla dimensione del blocco). Si permette inoltre di aggregare blocchi di indirizzi per rendere efficiente l'instradamento.

Inoltre, si definiscono i seguenti indirizzi speciali:

- **This-host**: 0.0.0.0; usato quando un host ha necessità di inviare un datagramma ma non conosce il proprio indirizzo IP.
- **Limited-broadcast**: 255.255.255.255; usato quando un router o un host devono inviare un datagramma a tutti i dispositivi che si trovano all'interno della rete, i router bloccano la propagazione alla sola rete locale.
- **Loopback**: 127.0.0.1; il datagramma con questo indirizzo di destinazione non lascia l'host locale (localhost), tipicamente viene usato per testing e debugging.
- **Indirizzi privati**: 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16, 169.254.0.0/16; riservati per reti locali.
- **Indirizzi multicast**: 224.0.0.0/4.

Per l'assegnazione di un indirizzo IP si distinguono due modalità:

- **Configurazione manuale**: l'amministratore configura direttamente nell'host l'indirizzo IP e inserisce informazioni di servizio (indirizzo del gateway/router, netmask e indirizzo IP di almeno un server DNS).
- **DHCP**: l'host ottiene il proprio indirizzo IP e le altre informazioni (indirizzo del gateway, netmask e indirizzo IP di almeno un server DNS) in modo automatico. È un protocollo client-server che implementa i seguenti passaggi (si rimanda inoltre alla figura 51):

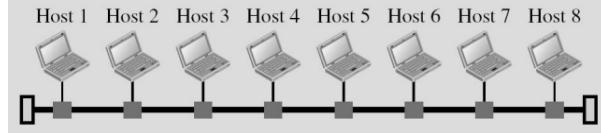
- L'host invia in broadcast un messaggio **DHCP discover** [opzionale].
- Il server DHCP risponde con un messaggio **DHCP offer** [opzionale].
- L'host richiede un indirizzo IP inviando il messaggio **DHCP request**.
- Il server DHCP invia un messaggio **DHCP ack** (se la richiesta va a buon fine).

Il protocollo di trasporto utilizzato è UDP.

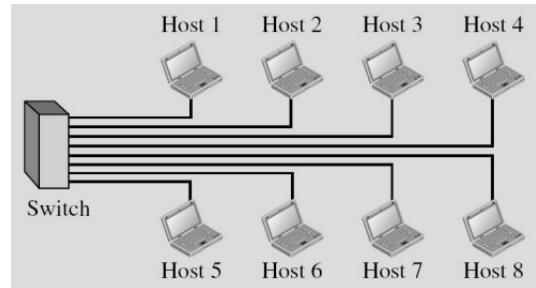
Il **protocollo NAT** permette di inoltrare su Internet il traffico proveniente da sistemi attestati su sottoreti private, in cui sono assegnati indirizzi IP privati; l'accesso di una rete privata su Internet è realizzato attraverso un router abilitato alla NAT con almeno un indirizzo IP pubblico (quindi, ogni datagramma in uscita dal router di accesso ha come indirizzo IP sorgente pubblico quello del router). Per inviare le risposte all'host corretto, il router di accesso ha in memoria una **tavella di traduzione NAT** le cui righe contengono le associazioni ((IP privato, porta), (IP pubblico, porta assegnata dal router)). Si veda la figura 50.

I router - fino ad adesso rappresentati come una scatola nera - hanno una struttura precisa per la quale si rimanda alla figura fig:routers-struttura. Si individuano le seguenti componenti nei routers:

- **Porte di input:** hanno una copia della tabella di inoltro, usano i valori dell'header per determinare la porta di output; prevede accodamento se la velocità con cui arrivano i datagrammi supera quella di inoltro nella struttura di commutazione.
- **Porte di output:** usano politiche di scheduling per definire l'ordine di trasmissione dei datagrammi, richiede buffering quando i datagrammi arrivano dalla struttura di commutazione ad una velocità maggiore di quella di trasmissione sul collegamento in uscita.
- **Processore di routing.**
- **Switching fabric (o struttura di commutazione):** mette in comunicazione porte di input e porte di output, per le tecnologie possibili si rimanda alla figure fig:switching-fabric-memory, fig:switching-fabric-bus e fig:switching-fabric-crossbar.



(a) LAN con cavo convidiso.

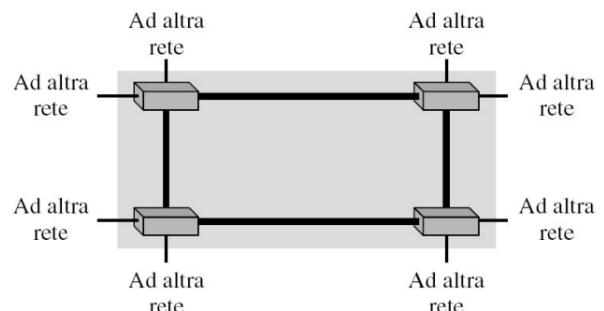


(b) LAN con switch.

Figura 1: Esempi di LAN.

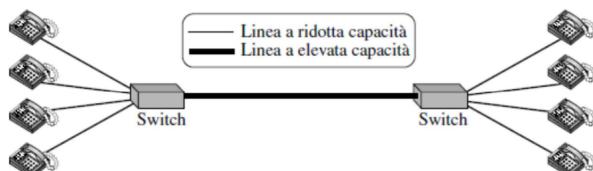


(a) WAN punto-punto.

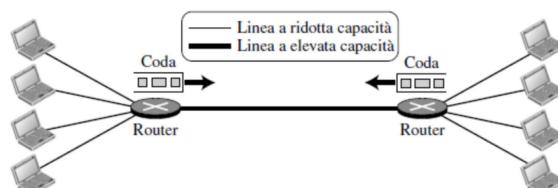


(b) WAN a commutazione.

Figura 2: Esempi di WAN.

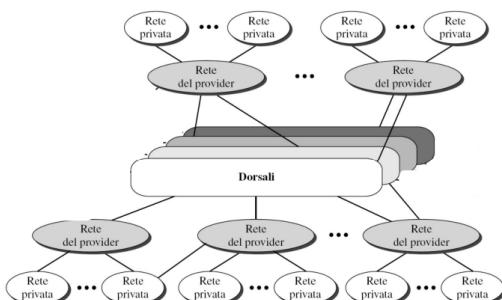


(a) Rete a commutazione di circuito.

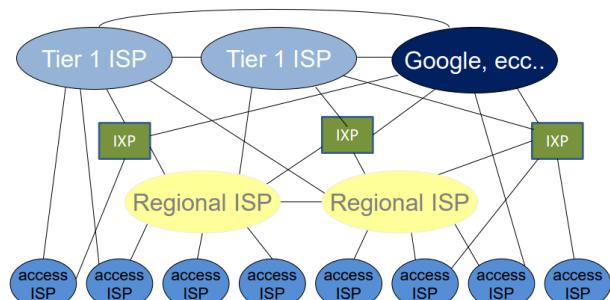


(b) Rete a commutazione di pacchetto.

Figura 3: Tipi di reti.



(a) Internet: rete di reti - versione semplificata.



(b) Internet: rete di reti - versione moderna.

Figura 4: Internet.

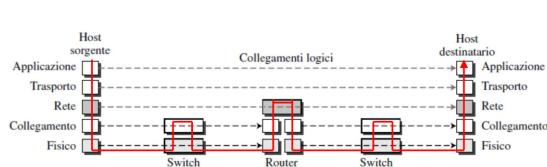


(a) Pila di protocolli.

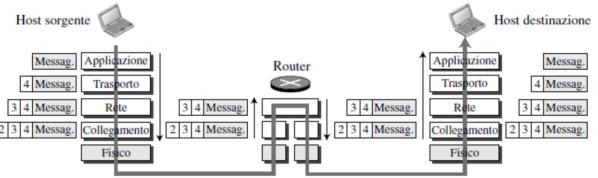


(b) Gerarchia di strati (o "layer").

Figura 5: Modello ISO/OSI.



(a) Comunicazione in una internet - livello fisico.



(b) Comunicazione in una internet - incapsulamento.

Figura 6: Comunicazione in una internet.

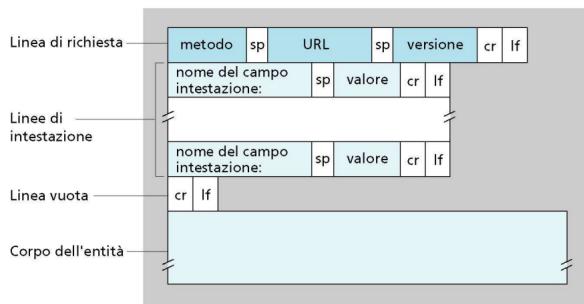
applicazione	Tolleranza alla Perdita di dati	throughput	Sensibilità al tempo
file transfer	no	elastico	no
e-mail	no	elastico	no
Documenti Web	no	elastico	no
Telefonia su Internet/ videoconferenza	si	audio: 5kbps-1Mbps video:10kbps-5Mbps	Si, centinaia di msec
audio/video memorizzato	si	come sopra	Si, pochi secondi
Giochi interattivi	si	Pochi kbps	Si, centinaia di msec
Messaggistica istantanea	no	elastico	Si e no

(a) Tipo di trasporto richiesto da applicazioni comuni.

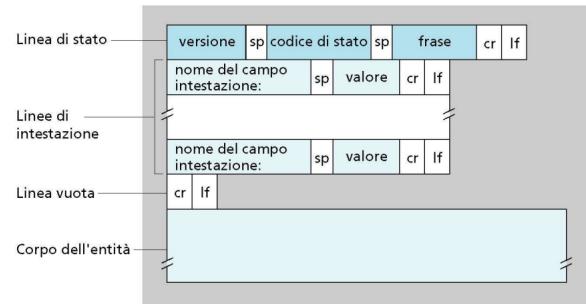
applicazione	protocollo di livello applicazione	Protocollo di trasporto
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

(b) Esempi di applicazioni Internet e i relativi protocolli.

Figura 7: Esempi di applicazioni Internet.



(a) HTTP Request Message.



(b) HTTP Response Message.

Figura 8: Messaggi HTTP.

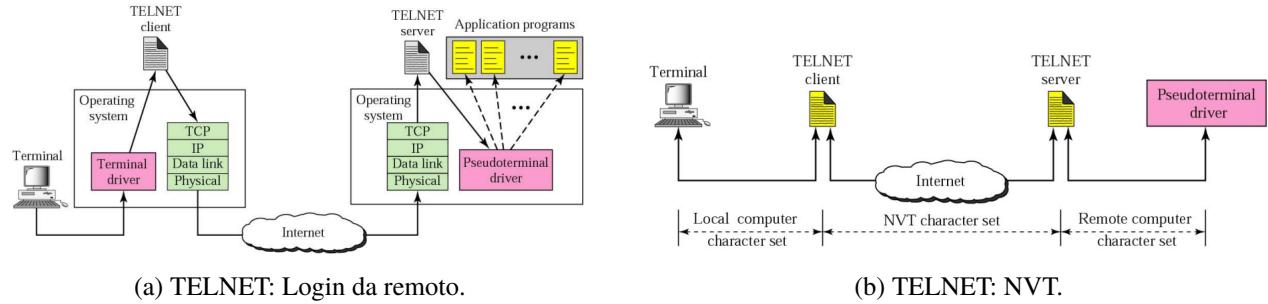


Figura 9: TELNET.

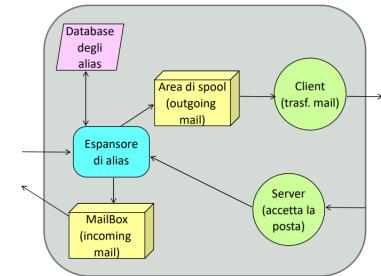
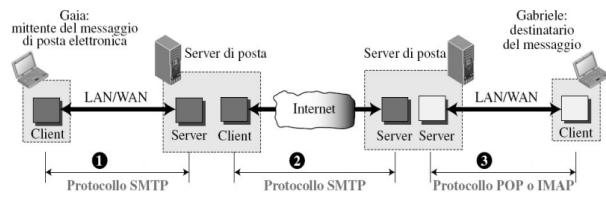


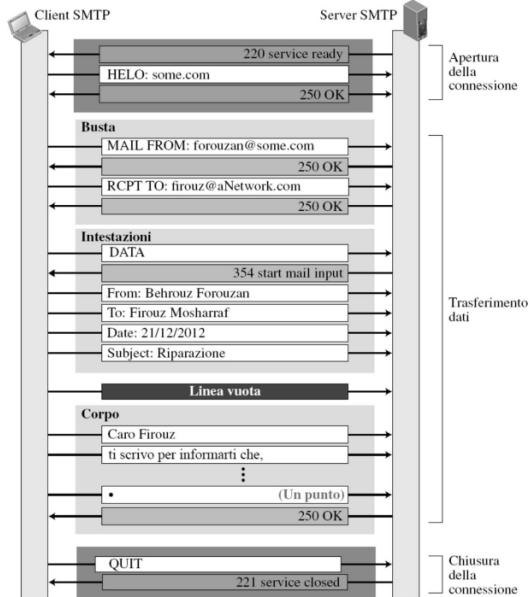
Figura 10: SMTP.

Versione MIME
metodo usato per codifica dati
Dati multimediali tipo, sottotipo, dichiarazione parametri
Dati codificati

```

From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg
base64 encoded data .....
.....base64 encoded data
  
```

(a) Esempio di uso del protocollo MIME.



(b) Esempio di sessione SMTP.

Figura 11: SMTP: esempi di sessioni.

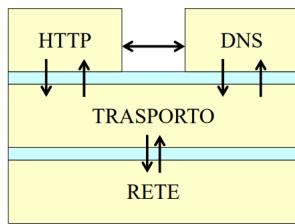


Figura 12: DNS: servizio di risoluzione.

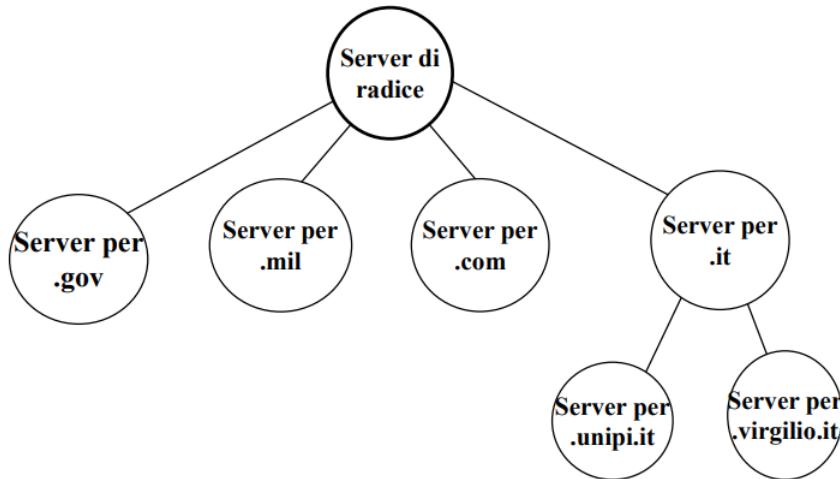


Figura 13: Gerarchia dei name server.

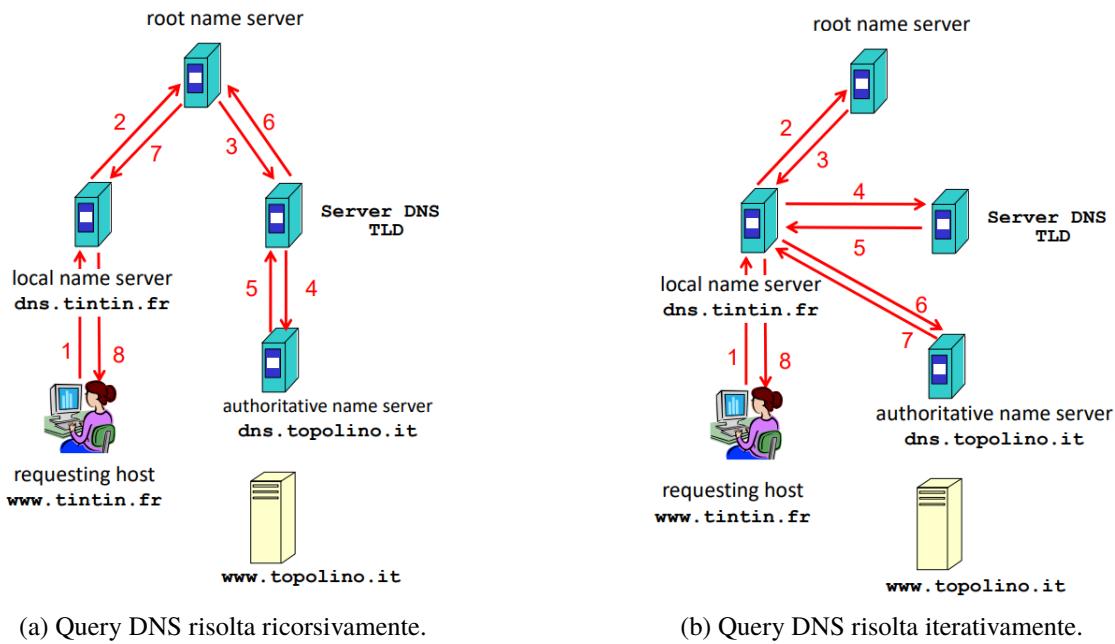


Figura 14: Query DNS.

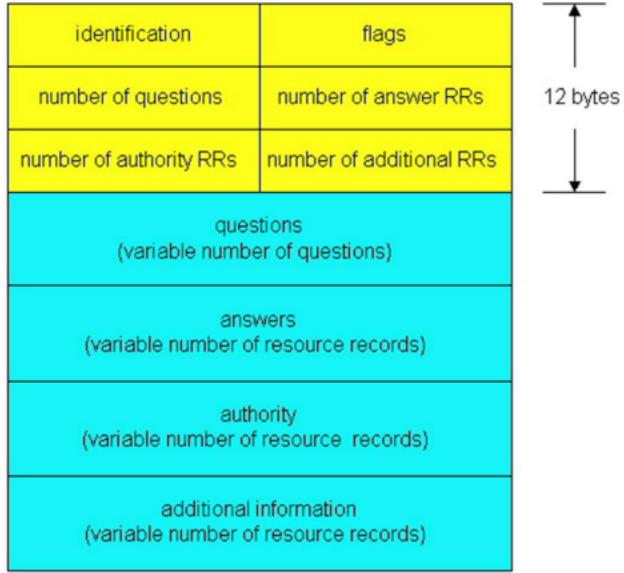


Figura 15: Struttura di un messaggio DNS.

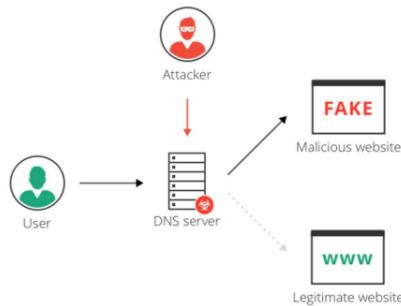


Figura 16: DNS hijacking.

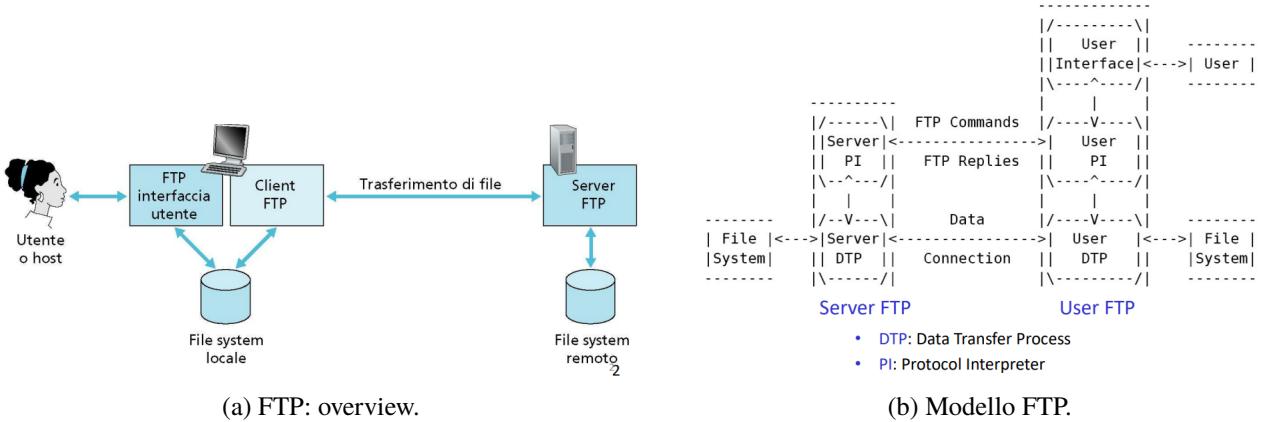


Figura 17: FTP.

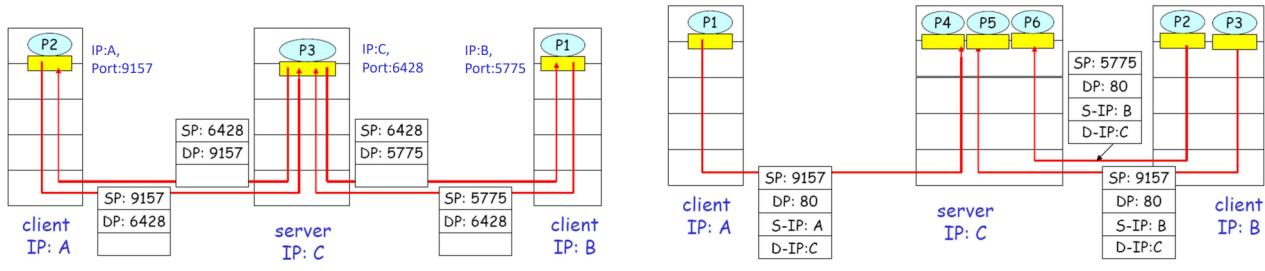


Figura 18: Demultiplexing.

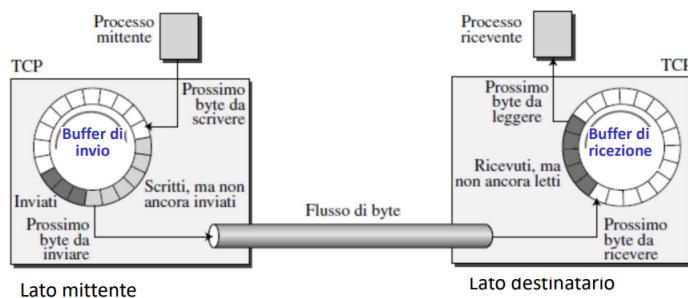


Figura 19: TCP: trasferimento bufferizzato.

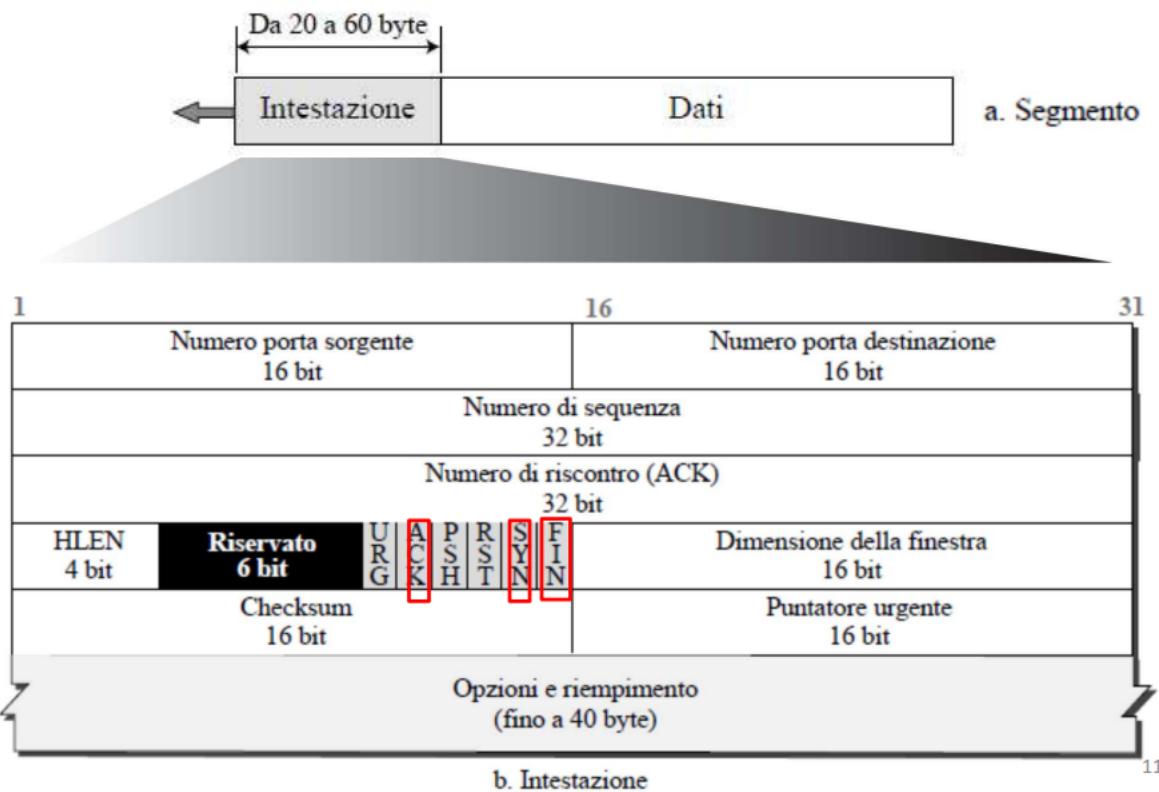


Figura 20: Formato dei segmenti TCP.

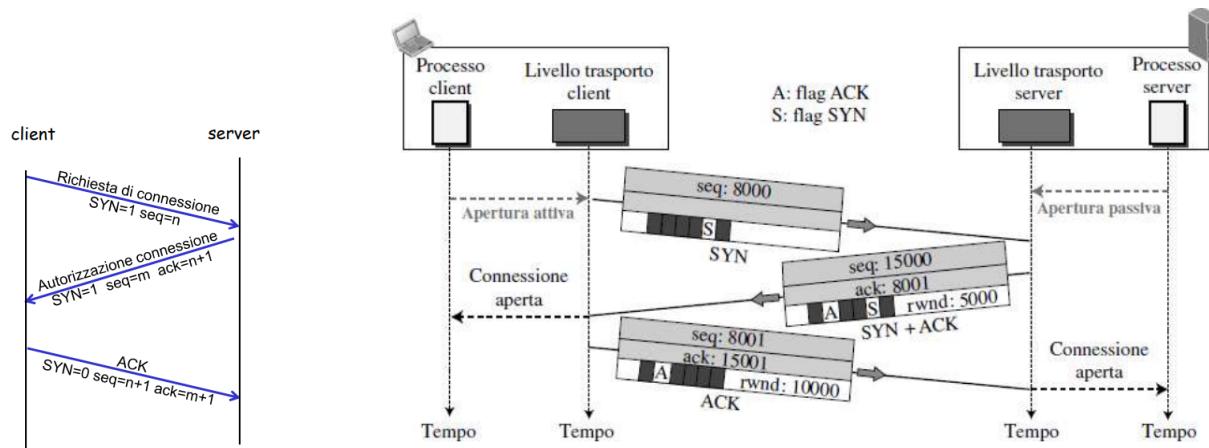


Figura 21: TCP: handshake.

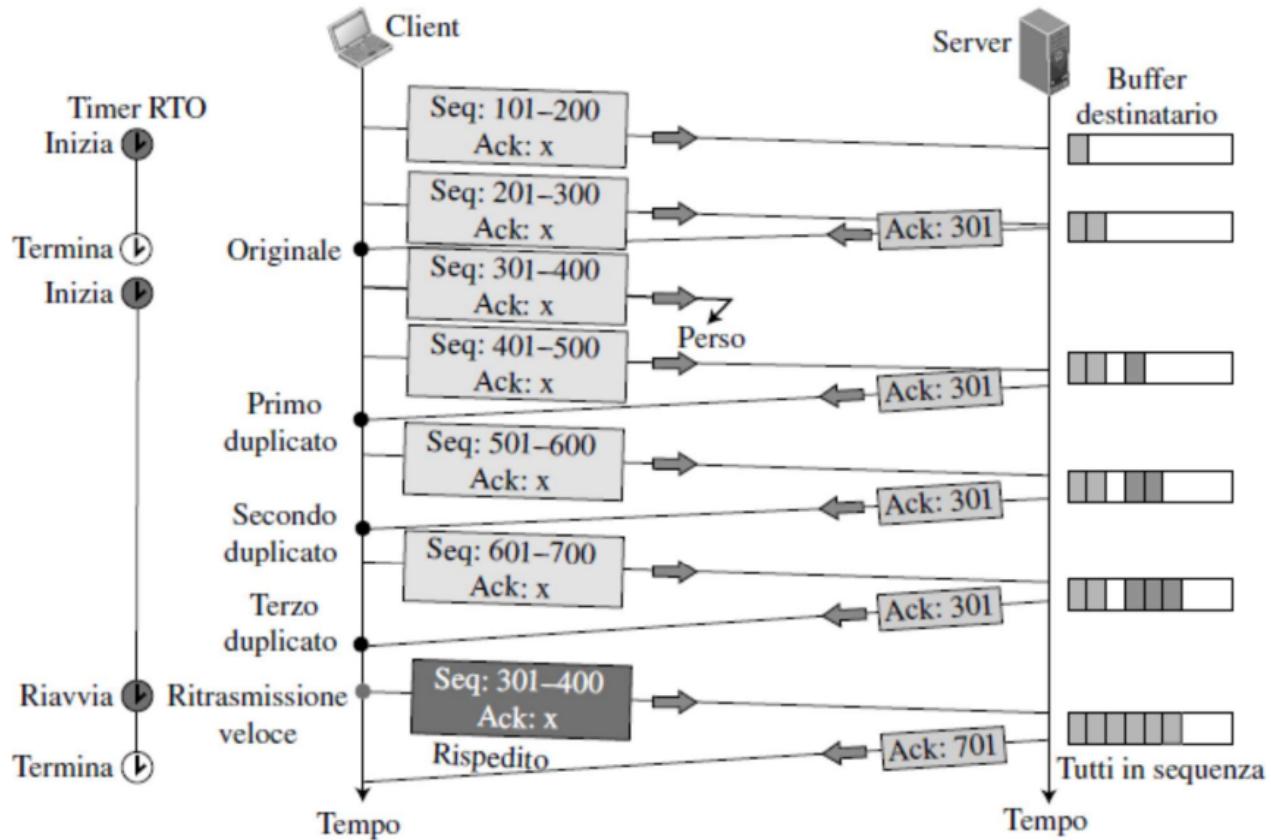


Figura 22: TCP: ritrasmissione veloce.

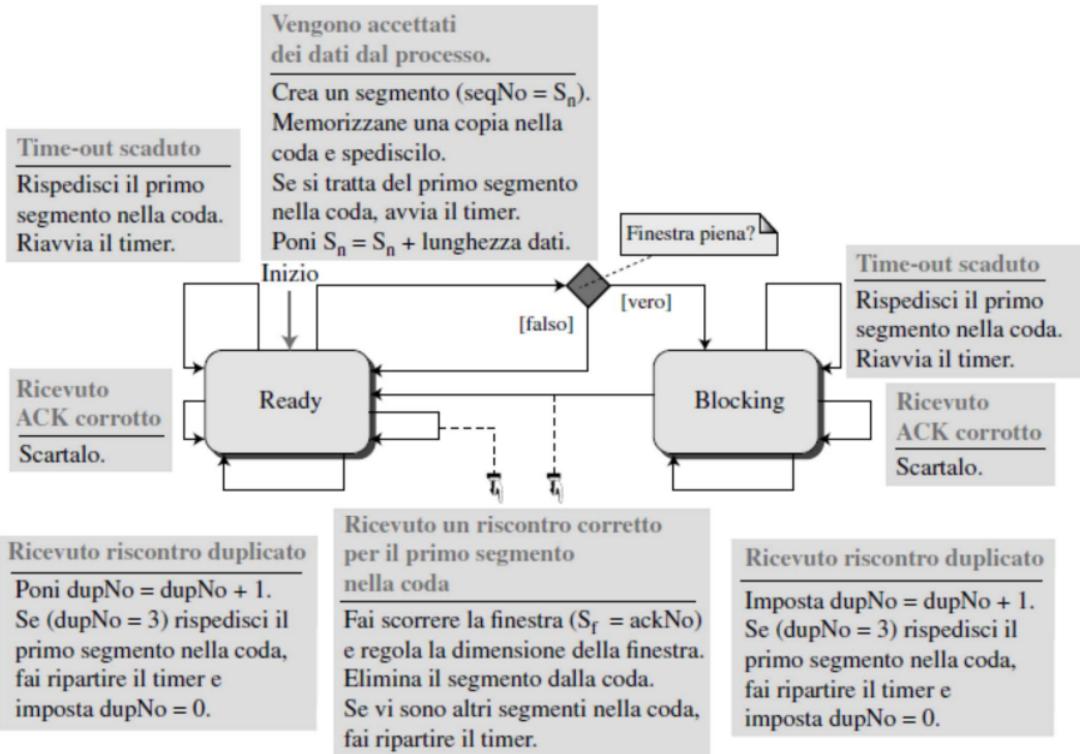


Figura 23: TCP: FSM mittente.

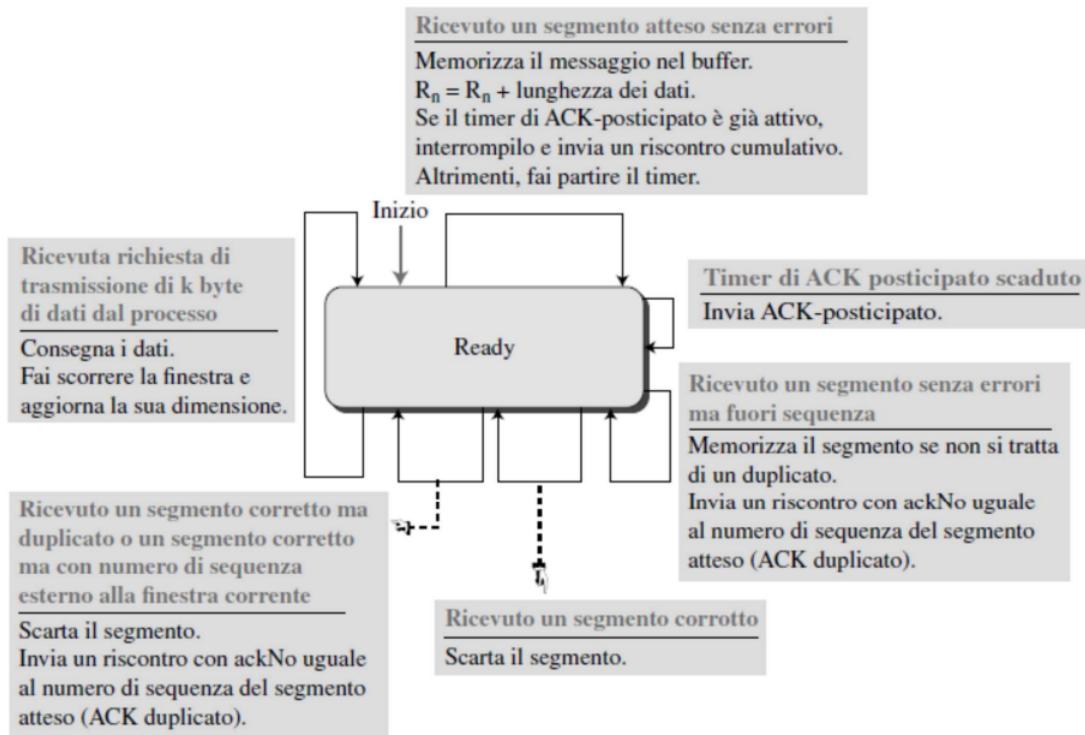


Figura 24: TCP: FSM destinatario.

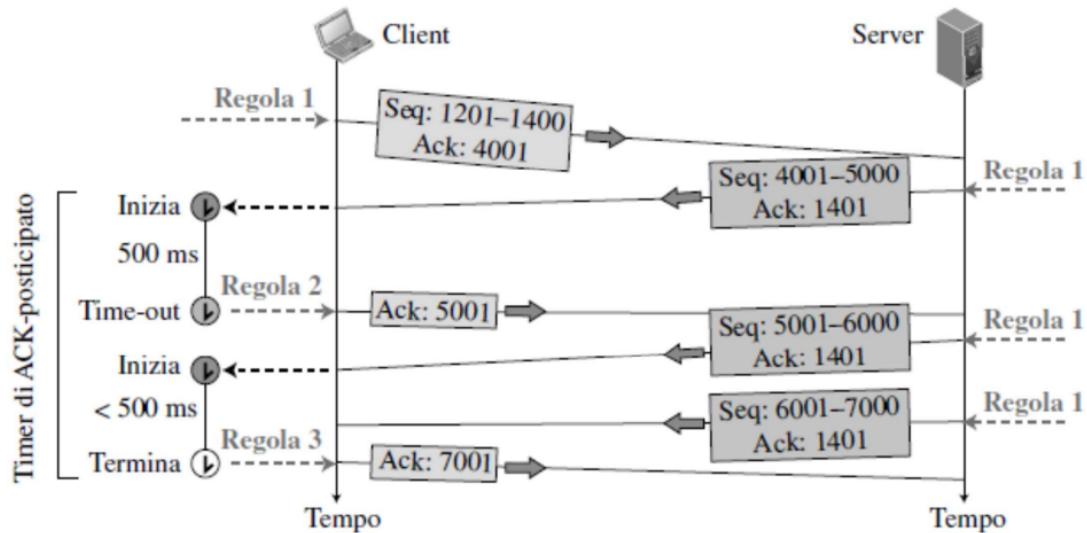


Figura 25: TCP: operatività normale.

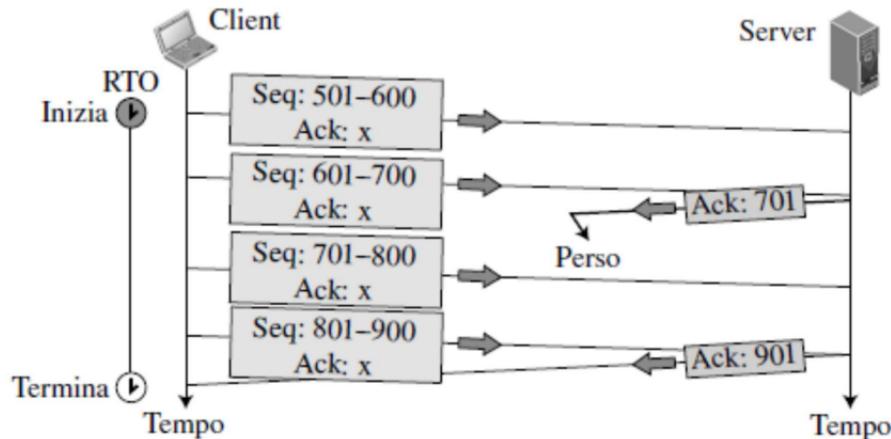


Figura 26: TCP: riscontro smarrito.

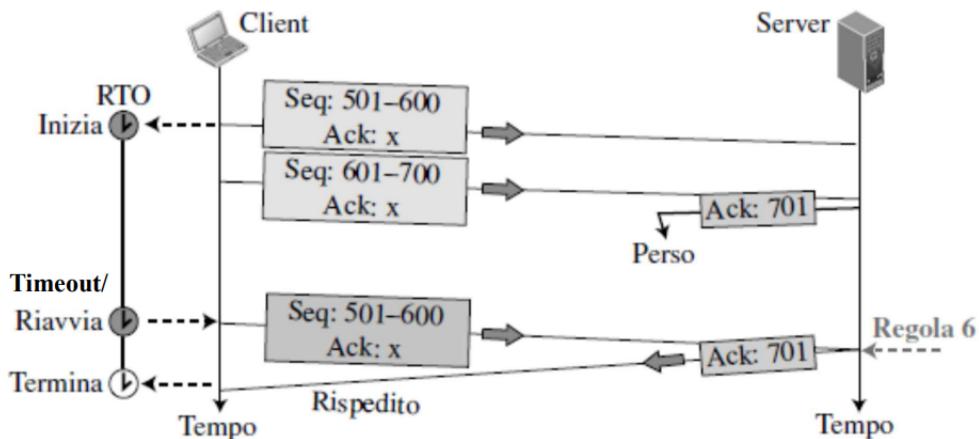


Figura 27: TCP: riscontro smarrito corretto con ritrasmissione.

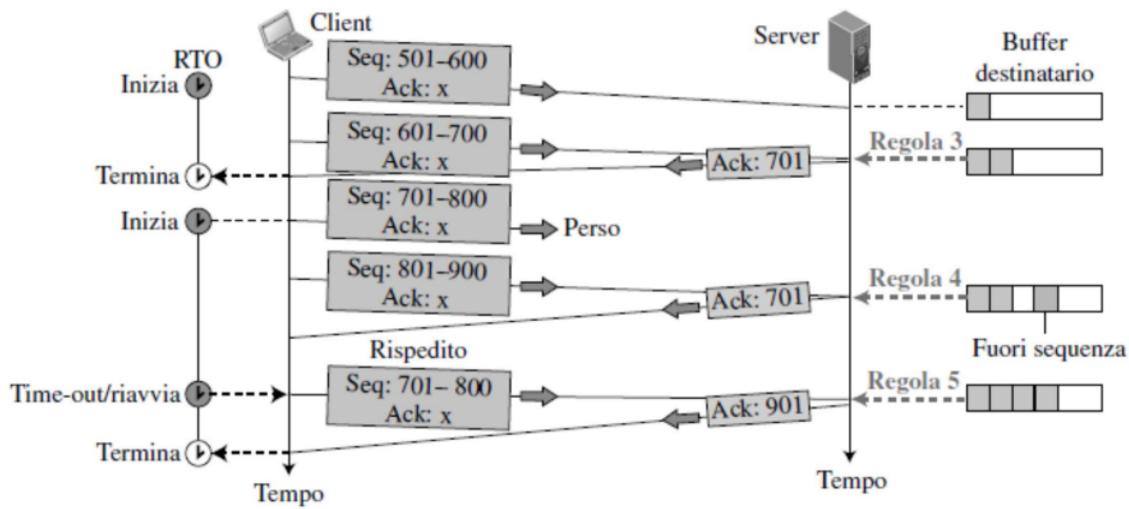


Figura 28: TCP: segmento smarrito.

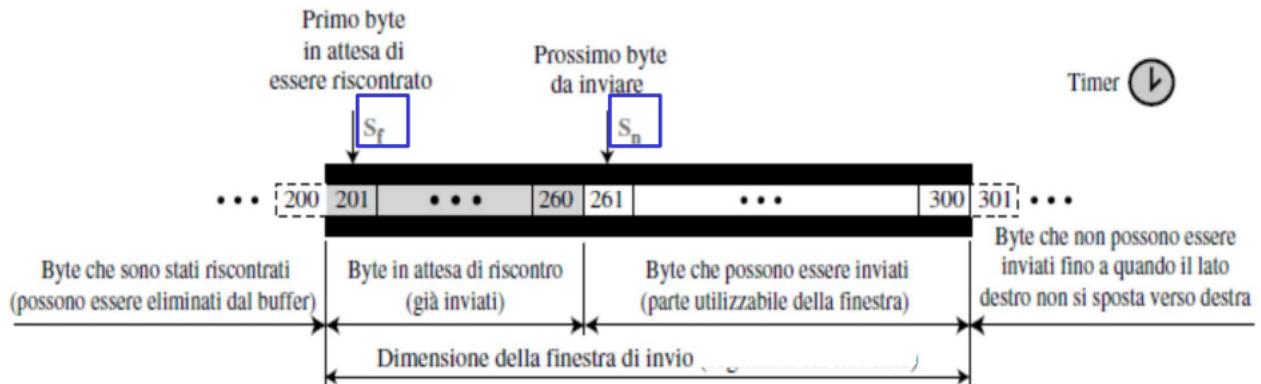


Figura 29: TCP: finestra di trasmissione.

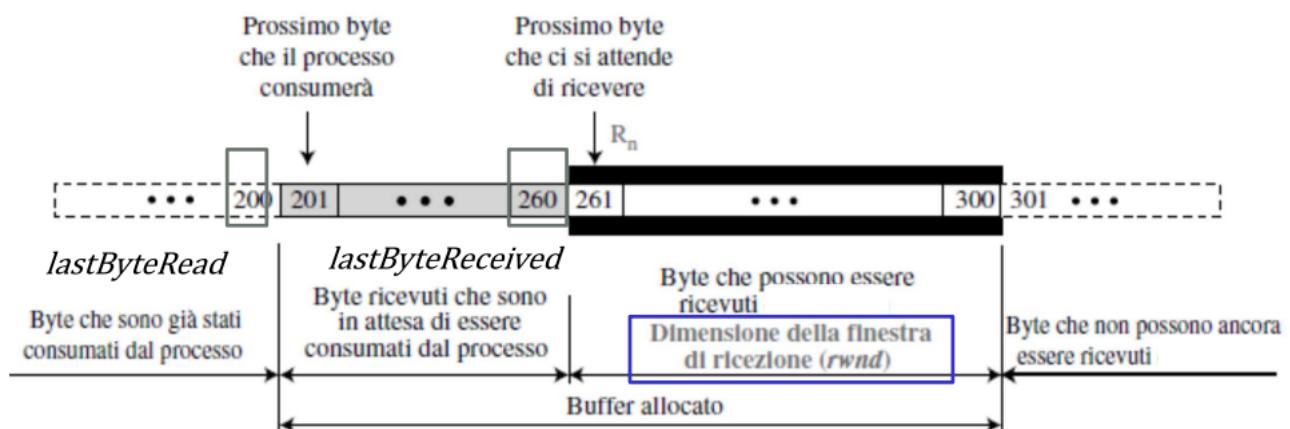


Figura 30: TCP: finestra di ricezione.

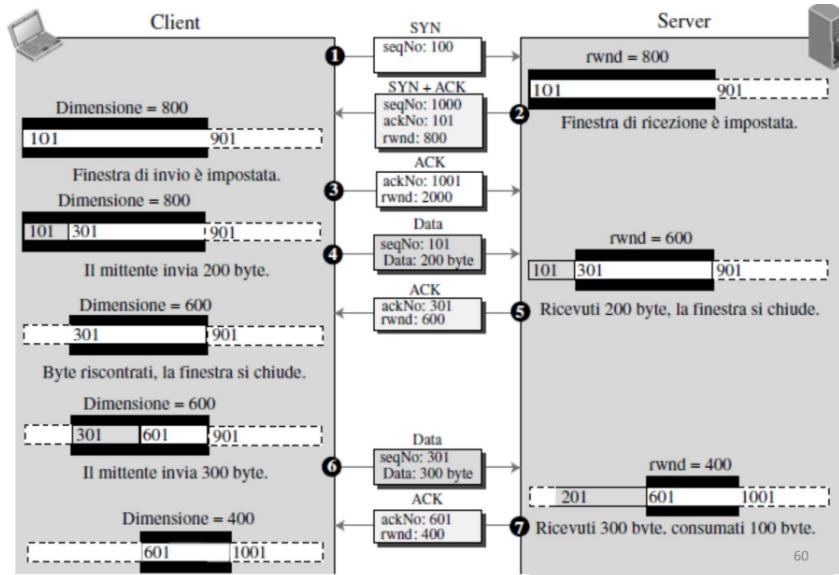


Figura 31: TCP: esempio di comunicazione con rwnd.

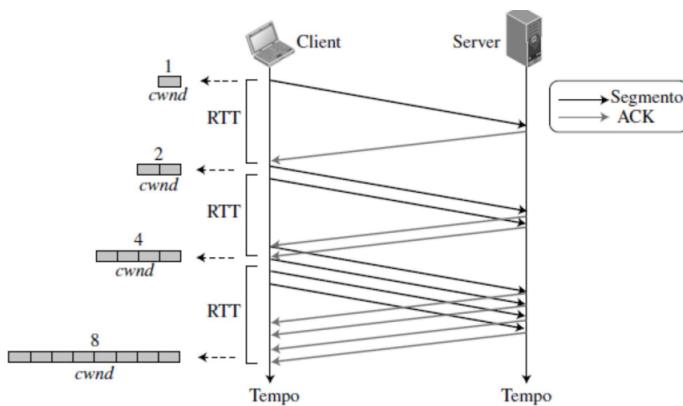


Figura 32: TCP: slow start.

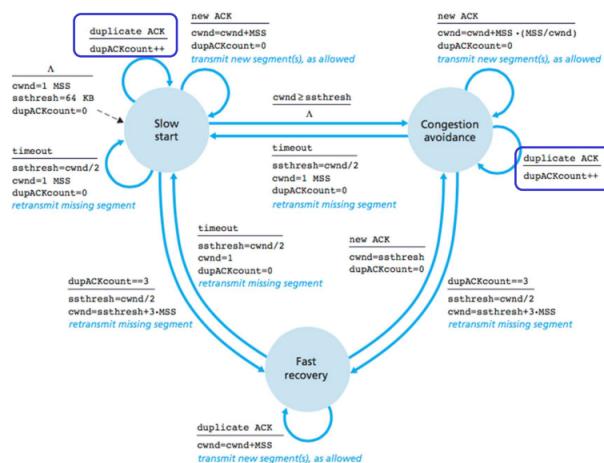


Figura 33: TCP Reno: macchina a stati.

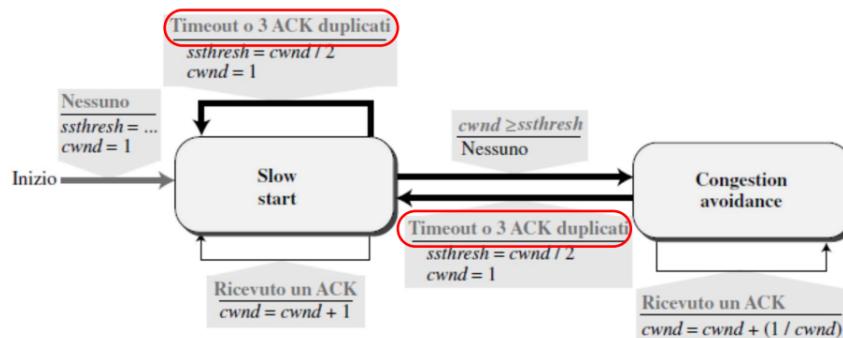
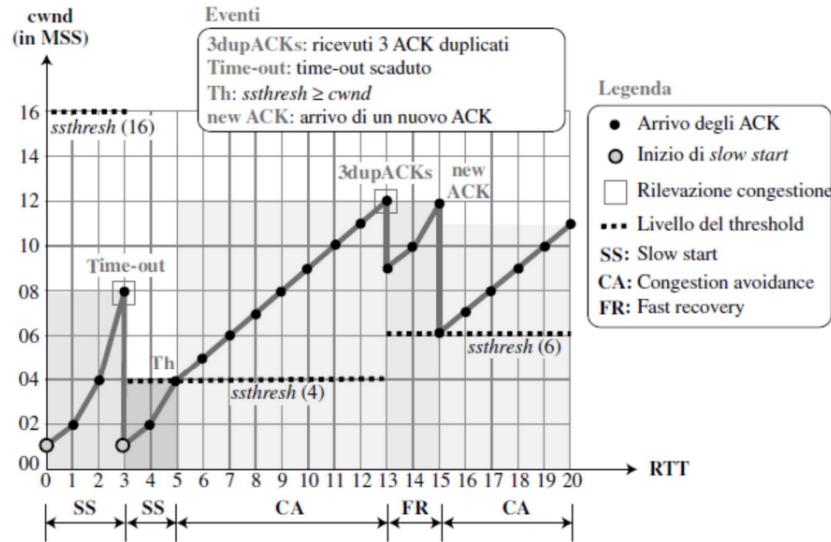


Figura 35: TCP Tahoe: macchina a stati.

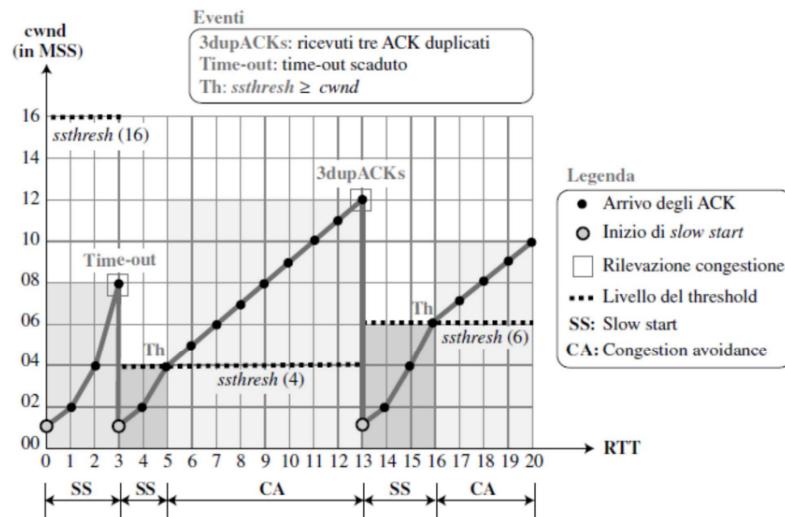


Figura 36: TCP Tahoe: esempio.

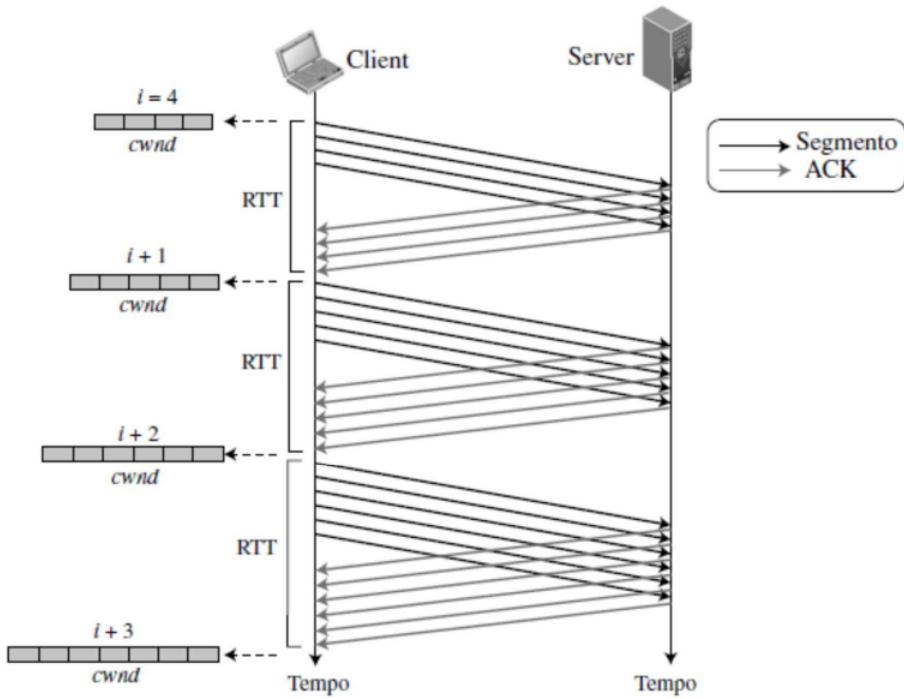


Figura 37: TCP: AIMD.

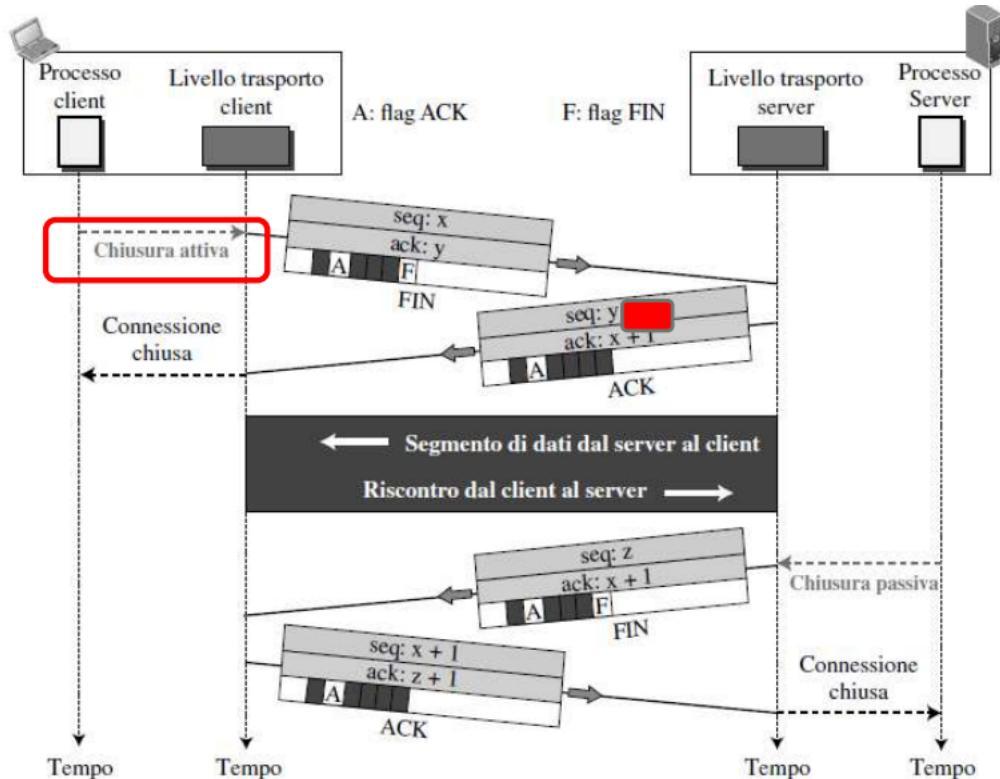


Figura 38: TCP: half-close.

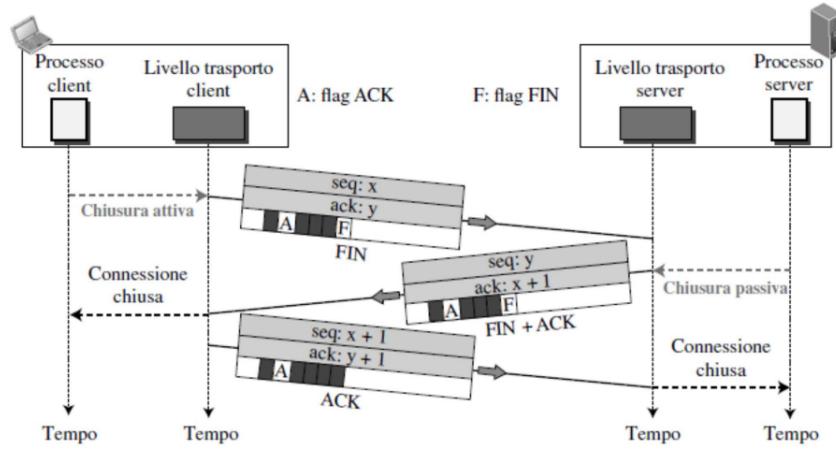


Figura 39: TCP: chiusura con handshake a 3 vie (comune nel caso di modelli client-server del tipo richiesta/risposta).

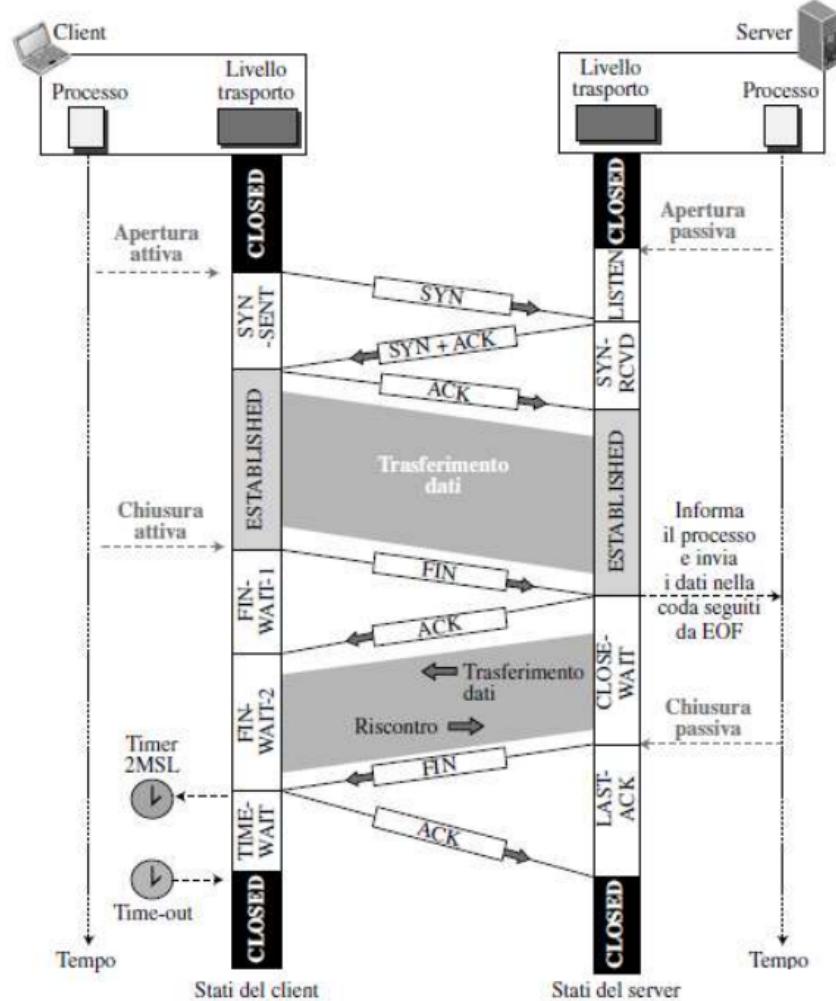


Figura 40: Apertura e chiusura della connessione.

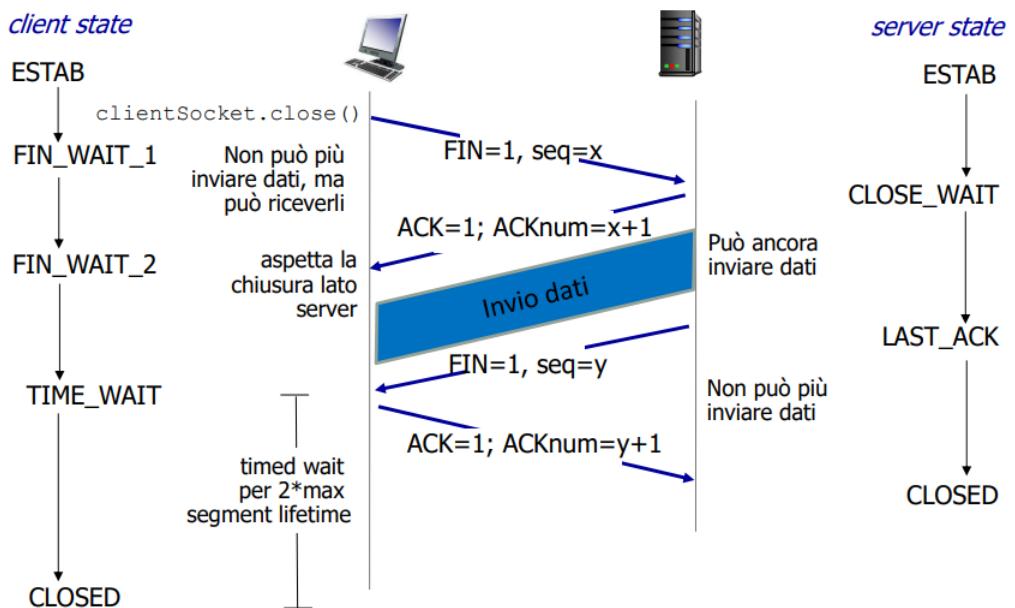


Figura 41: Chiusura della connessione.

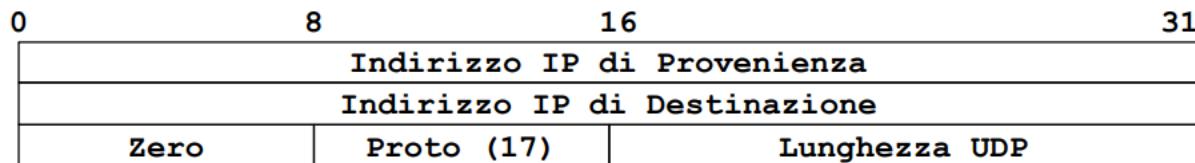


Figura 42: UDP: struttura del campo checksum.

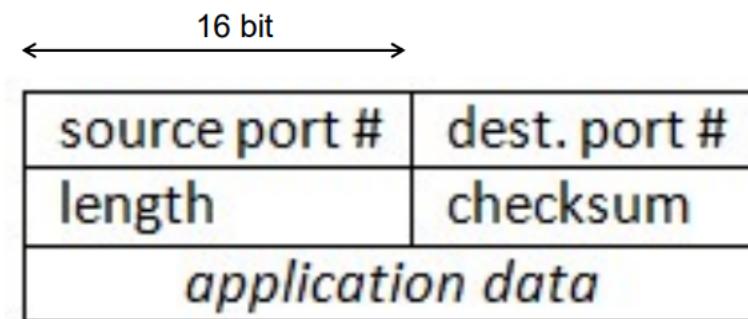


Figura 43: UDP: formato di un datagramma.

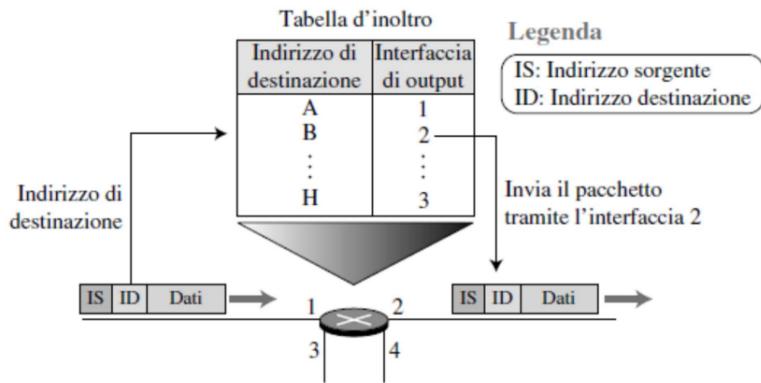


Figura 44: IP: inoltro.

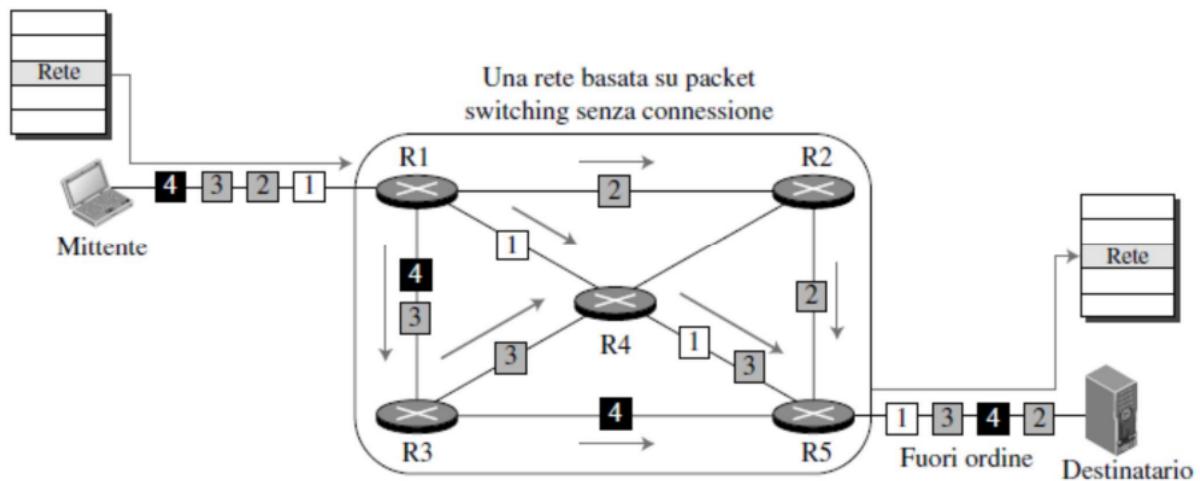


Figura 45: IP: instradamento.

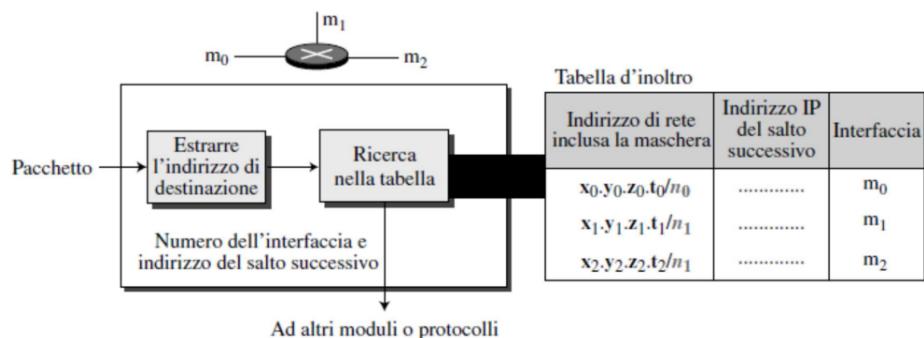


Figura 46: IP: inoltro e tabella di inoltro.

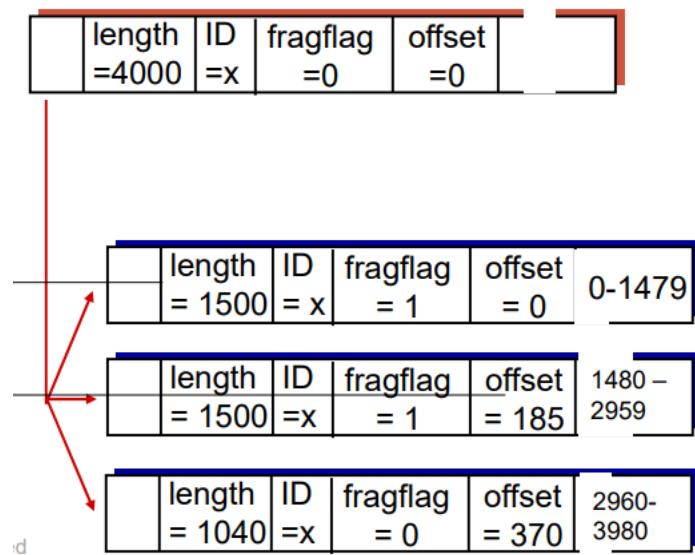


Figura 47: IP: frammentazione.

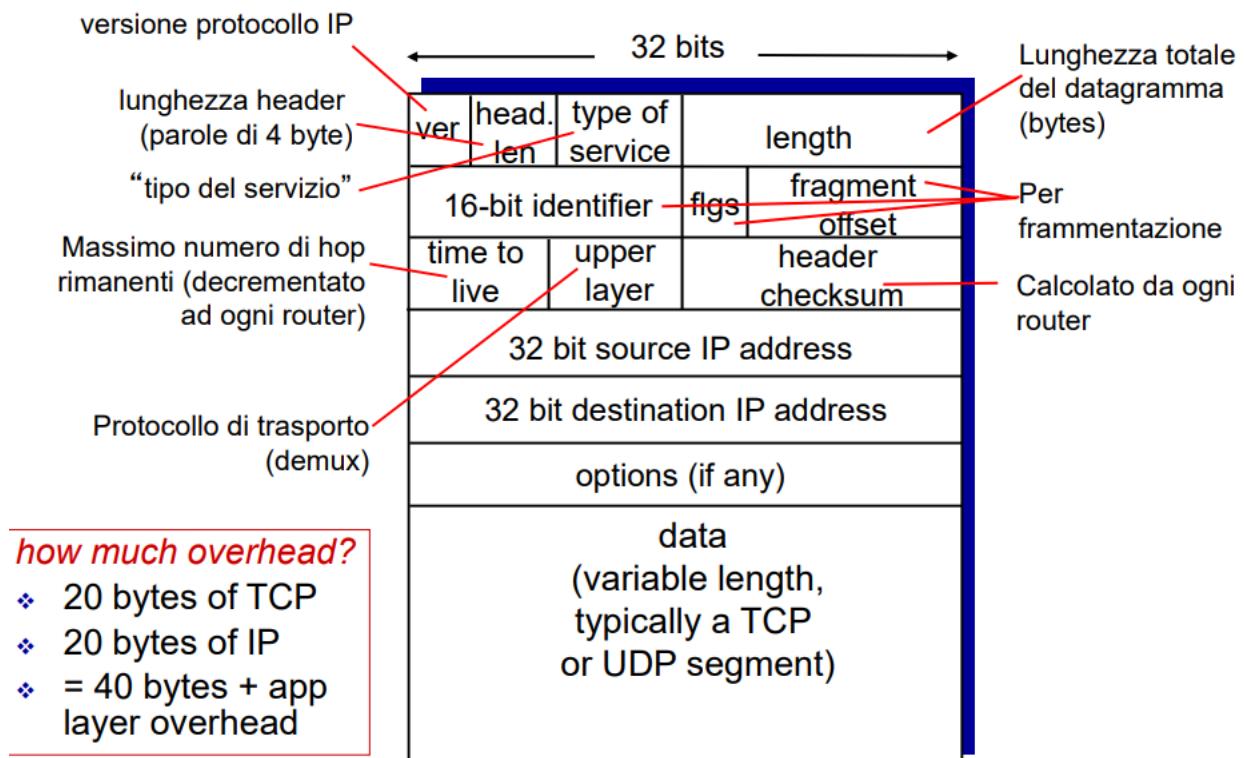


Figura 48: IP: formato di un datagramma.

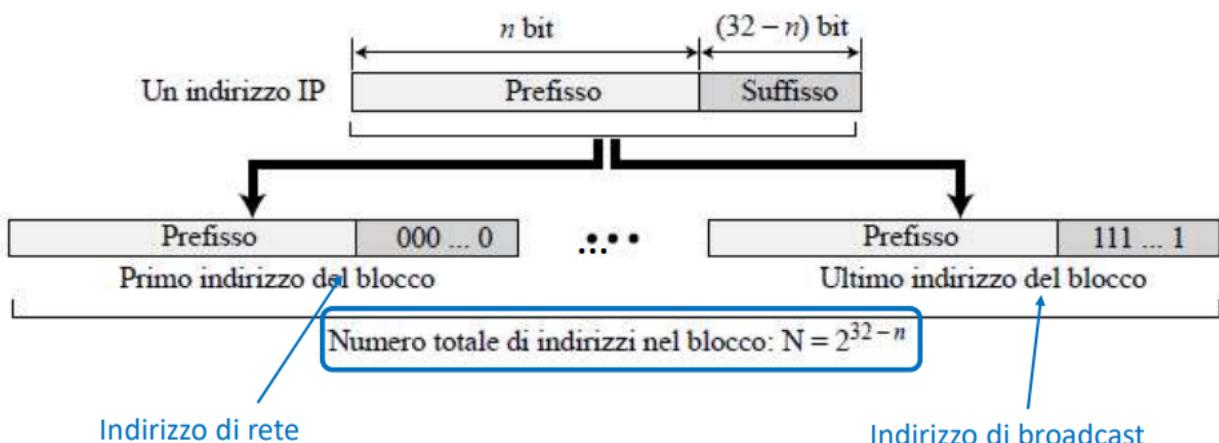


Figura 49: IP: indirizzamento senza classi.

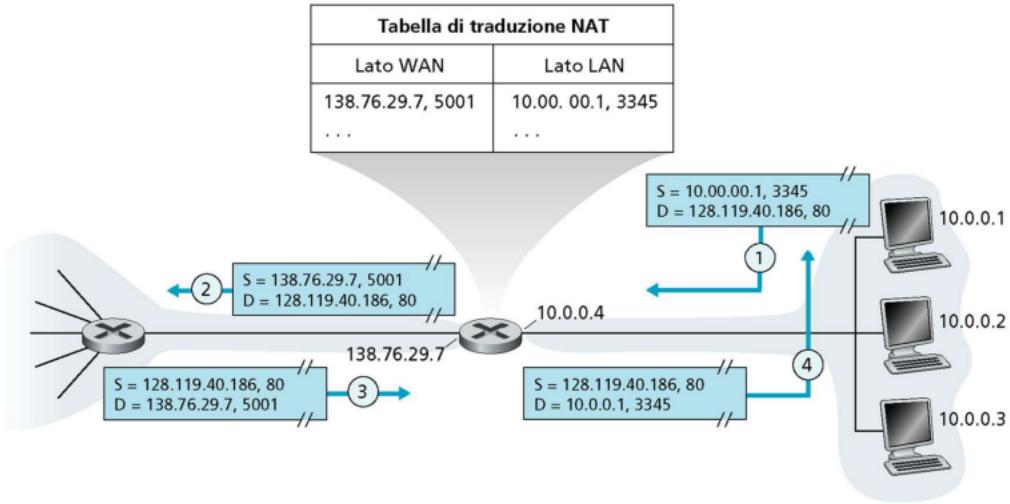


Figura 50: IP: tabella di traduzione NAT.

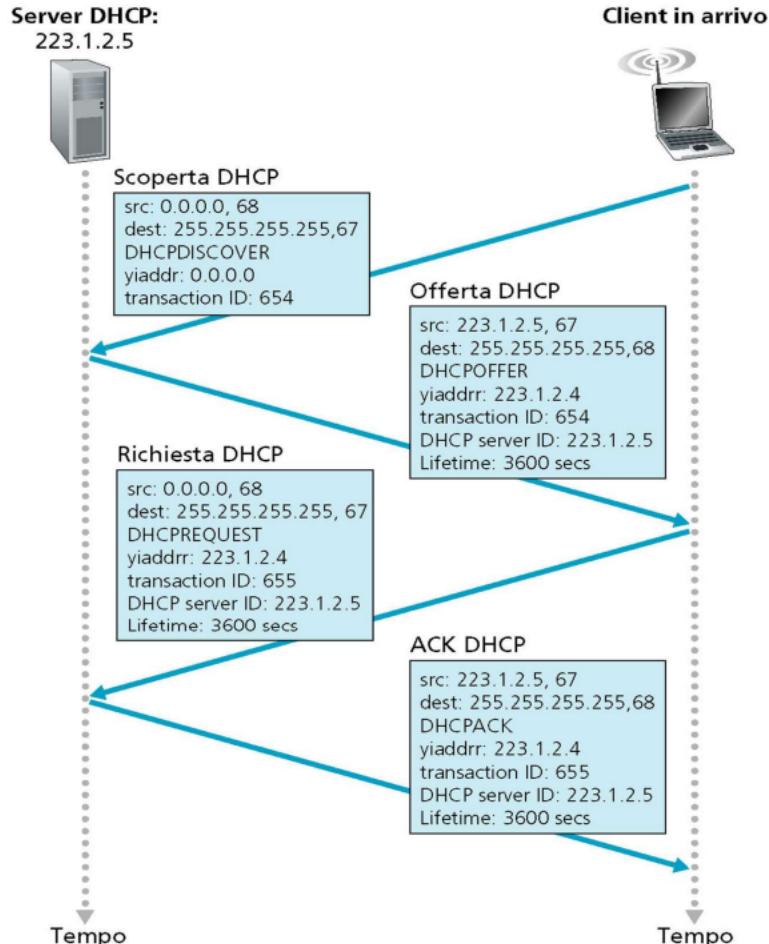


Figura 51: DHCP: esempio di sessione.