Reti e Laboratorio III

Giuseppe Di Palma

October 8, 2023

Appunti basati sulle lezioni e dispense delle professoresse Federica Paganelli [1] e Laura Ricci [2]

[1] http://pages.di.unipi.it/paganelli/
[2] https://pages.di.unipi.it/ricci/

Contenuti

0 Introduzione **3**

1 Commutazione e Ritardi **4**
1.1 Commutazione di Circuito 4
1.2 Commutazione di Pacchetto 5
1.3 Ritardi 6

2 Modelli Stratificati **7**
2.1 Perché e come stratificare 7
2.2 Open System Interconnection 7
2.3 Gerarchia degli strati ISO/OSI 9
2.4 Stack TCP/IP 12

3 Lo strato Applicativo **14**
3.1 Protocolli TCP e UDP 15
3.2 URI, URL E URN 15
3.3 HTTP 16
3.3.1 Formato messaggi HTTP 18
3.3.2 Metodi 20
3.3.3 Caching e Cookie 22
3.4 TELNET 23

0 Introduzione

Definizione (Rete)**.** Un'interconnessione di dispositivi in grado di scambiarsi informazioni, quali sistemi terminali (host), router, switch e modem

Definizione (Router)**.** Dispositivi che interconnettono reti.

Definizione (Switch)**.** Dispositivi che collegano fra loro più host a livello locale

Tipologie di reti Esistono varie tipologie di reti

- **LAN**: Local Area Network, sono reti di piccole dimensioni (al più qualche km). Connettono principalmente host, stampanti e workstation tra loro.
- **WAN**: Wide Area Network, è una rete il cui compito è di interconnettere LAN o singoli host separati da distanze geografiche.
- **MAN**: Metropolitan Area Network, rete di computer che collega i computer all'interno di un'area metropolitana, più grande di una LAN ma più piccola di una WAN.

Network of networks Gli host si collegano ad internet tramite Internet Service Provider (ISP) i quali devono a loro volta essere connessi tra loro. La risultante rete di reti è molto complessa.

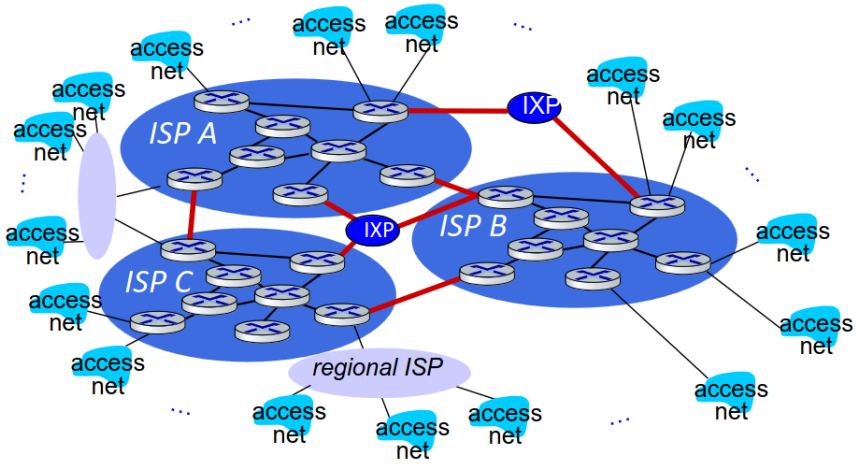

Figura 1: Struttura della rete Internet

1 Commutazione e Ritardi

Definizione (Commutazione)**.** Modalità con cui viene determinato il percorso sorgente-destinazione e vengono dedicate ad esso le risorse della rete.

Esistono due meccanismi:

- Commutazione di Circuito
- Commutazione di Pacchetto

Metriche Data una comunicazione tra due host, indipendentemente dal tipo di commutazione utilizzata, è possibile utilizzare delle metriche comuni per l'analisi delle prestazioni:

- Bandwidth: larghezza dell'intervallo di frequenze utilizzato dal sistema trasmissivo (si misura in Hz).
- Transmission Rate: quantità di dati (bits) che possono essere trasmessi per unità di tempo su un certo collegamento (dipende dal bandwidth ma anche dal mezzo trasmissivo, rumore, ecc...).
- Throughput: quantità di dati che possono essere trasmessi con successo dalla sorgente alla destinazione in un certo intervallo di tempo al netto di perdite sulla rete (duplicazioni, protocolli, ecc...). In un percorso da una sorgente a una destinazione un pacchetto può passare attraverso numerosi link, perciò il throughput dell'intero percorso è dato dal percorso intermedio con throughput minore.

NB: Throughput $<$ Transmission Rate

1.1 Commutazione di Circuito

Nella commutazione di circuito si instaura un cammino dedicato tra i due dispositivi che vogliono comunicare. Il percorso viene stabilito all'inizio della comunicazione (Setup) e vengono dedicate risorse alla comunicazione (canale logico o circuito) in modo esclusivo. Le risorse allocate sono garantite per tutta la durata della comunicazione, indipendentemente dall'utilizzo effettivo.

Canale logico Per quanto riguarda l'assegnazione di un canale di comunicazione logico esistono due principali metodi:

- **FDM**: Frequency Division Multiplexing, il canale di comunicazione viene suddiviso in bande di frequenze ognuna delle quali viene assegnata in modo esclusivo ad una certa connessione.

- **TDM**: Time Division Multiplexing, il tempo viene suddiviso in slot di tempo. Ogni comunicazione ha uno o più slot periodici assegnati nei quali può trasmettere alla velocità massima del canale.

Vantaggi:

- Performance garantite.
- Tecnologie di switching efficienti.

Svantaggi:

- Necessaria una fase di instaurazione della comunicazione.
- le risorse rimangono inattive se non utilizzate (non c'è condivisione).

1.2 Commutazione di Pacchetto

Nella commutazione di pacchetto il flusso di dati punto-punto viene suddiviso in pacchetti. Ogni pacchetto è instradato singolarmente e indipendentemente dagli altri pacchetti della stessa comunicazione (possono seguire lo stesso percorso o percorsi diversi in base alle necessità).
I commutatori (es. router) devono ricevere integralmente i pacchetti prima di poterne continuare l'instradamento. Quando ci sono più pacchetti in ingresso ripetto a quanto il canale può supportare, questi vengono messi in coda in dei buffer (se il buffer è pieno i pacchetti vengono persi).

Vantaggi:

- Risorse trasmissive usate solo se necessario.
- Fase di setup e segnalazione della connessione non richiesta.

Svantaggi:

- Tecnologie di inoltro poco effienti (calcolo del percorso indipendente per ogni pacchetto).
- Ritardi variabili nel percorso end-to-end (jitter).
- protocolli necessari per un trasferimento dati affidabile, controllo della congestione.

1.3 Ritardi

Definizione (Latenza)**.** Tempo richiesto affinché un messaggio arrivi a destinazione dal momento in cui il primo bit parte dalla sorgente.

Il valore della latenza in una rete a commutazione di pacchetto è determinato da 4 tipologie di ritardo:

- Ritardo di eleborazione, dovuto ai controlli di errore sui bit e dal calocolo del percorso di uscita del pacchetto.
- Ritardo di accodamento, dovuta all'attesa dei pacchetti nel buffer la quale dipende dal tipo e dall'intensità del traffico.
- Ritardo di trasmissione:, tempo impiegato per trasmettere un pacchetto sul canale.
  Dipende da due valori
  - R, rate di trasmissione sul canale (in bps).
  - L, lungezza del pacchetto (in bit).

  Si calcola come L/R.
- Ritardo di propagazione, tempo impiegato da 1 bit per essere propagato da un nodo all'altro
  Dipende anch'esso da due valori
  - d, lunghezza del collegamento fisico (in metri).
  - s, velocità di propagazione del mezzo.

  Si calcola come d/s.

La latenza è quindi la somma di tutti questi 4 ritardi. Nella pratica però i ritardi di elaborazione e accodamento vengono trascurati e la latenza end-to-end è data dalla somma dei ritardi di propagazione e trasmissione di tutti i collegamenti intermedi.

Prodotto Rate-Ritardo Con prodotto rate-ritardo indichiamo numero massimo di bit che un link può contenere ad un certo istante. Rappresentiamo il link come un tubo la cui sezione rappresenta il rate e la lunghezza il ritardo. Il volume di questo tubo fornisce il rapporto rate-ritardo

2 Modelli Stratificati

2.1 Perché e come stratificare

Nei sistemi di comunicazione non si utilizza un unico protocollo infatti, stratificare il problema permette di:

- scomporre il problema in sottoproblemi più semplici da trattare; il singolo strato è più semplice del sistema nel suo complesso.
- semplificare la progettazione, implementazione e manutenzione del software.
- rendere i livelli indipendenti: è possibile modificare l'implementazione di uno strato senza dover cambiare gli altri, a patto che l'interfaccia non cambi.

La separazione dei livelli non può ovviamente essere fatta in modo superficiale. Avere troppi livelli infatti rende il sistema poco scalabile, d'altro canto averne pochi rende i vari livelli complessi da realizzare.
Il processo di stratificazione avviene seguendo due principi:

- **Separation of Concern**: separazione degli interessi e delle responsabilità, fare ciò che compete, delegando ad altri tutto ciò che è delegabile.
- **Information Hiding**: nascondere tutte le informazioni che non sono indispensabili per il committente per definire compiutamente l'operazione.

Per garantire questi due principi inoltre la comunicazione tra livelli adiacenti deve essere ridotta al minimo indispensabile e ogni strato deve svolgere una sola e ben definita funzione.

2.2 Open System Interconnection

Sistemi chiusi Negli anni '60 nascono le prime reti di calcolatori come ARPANET[3], SNA (IBM)[4] e DNA (Digital) che utilizzano architetture di rete a strati. Ognuna di queste reti però utilizzava protocolli proprietari incapaci di operare in ambienti condivisi perché non in grado di interpretare i segnali provenienti dall'esterno.

[3] https://it.wikipedia.org/wiki/ARPANET
[4] https://it.wikipedia.org/wiki/Systems_Network_Architecture

Sistemi aperti L'obiettivo principale di questa tipologia di sistemi e quello di realizzare una rete di calcolatori in cui qualsiasi terminale comunica con un qualunque fornitore di servizi mediante una rete.
Per poter rendere possibile questo obiettivo è necessario stabilire delle regole comuni, degli standard.

Definizione. Un protocollo è detto **Aperto** se:

- i dettagli sono disponibili pubblicamente.
- i cambiamenti sono gestiti da un'organizzazione la cui partecipazione è aperta al pubblico.

Definizione. Un sistema che implementa protocolli aperti è un **Sistema Aperto** (Open System)

L'International Organization for Standards (ISO) ha specificato uno standard per l'interconnessione di sistemi aperti denominato Reference Model Open System Interconnection (OSI RM).

Definizione (Strato)**.** È un modulo interamente definito attraverso i servizi, protocolli e le interfacce che lo caratterizzano.

Definizione (Servizio)**.** Servizi che uno strato fornisce ad uno strato soprastante attraverso primitive di servizio.

Definizione (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.

Definizione (Protocollo)**.** Insieme di regole che 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; in modo:

- Efficace: Un sistema che riesce a raggiungere lo scopo prefissato con la maggior frequenza possibile.
- Efficiente: Un sistema che riesce a raggiungere lo scopo prefissato con il minor sforzo possibile.

Cosa deve fare un protocollo In un protocollo vanno specificati:

- La sintassi del messaggio: che campi contiene e in quale formato.
- La semantica del messaggio: come interpretare i vari campi e il messaggio stesso.
- Le azioni da intraprendere dopo la ricezione di un messaggio.

2.3 Gerarchia degli strati ISO/OSI

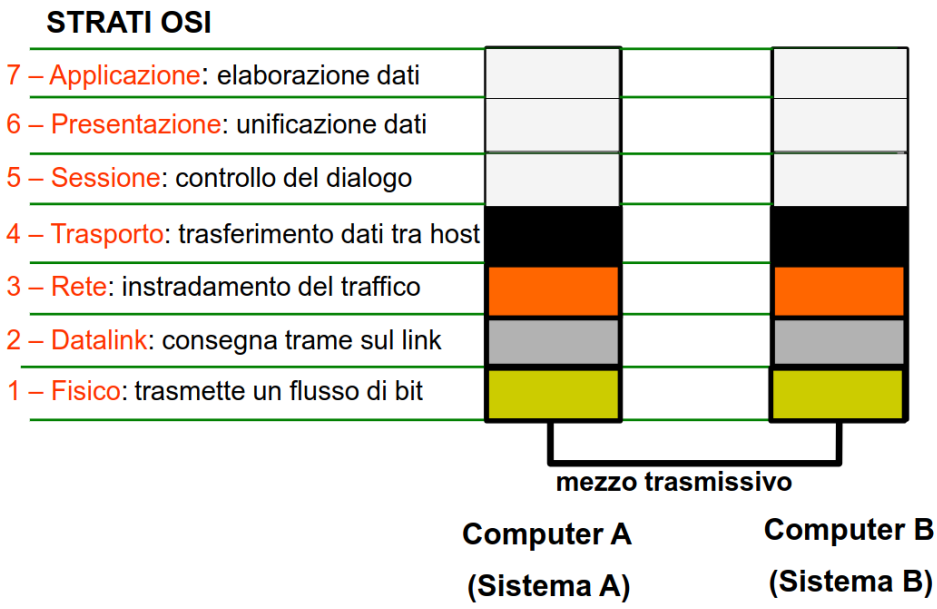

Figura 2: Strati definiti nel modello ISO/OSI

- **Livello Fisico:**
  - Comprende tutte le funzioni (procedure meccaniche ed elettroniche) che permettono una connessione a livello fisico.
  - Si occupa della trasmissione dei bit attraverso il mezzo trasmissivo e delle caratteristiche di cavi e connettori.
- **Livello di Collegamento:**
  - Definisce le regole per inviare e ricevere informazioni tra due sistemi in comunicazione.
  - Si occupa di formare i dati da inviare attraverso il livello fisico, incapsulando i dati in un pacchetto provvisto di header (intestazione) e tail (coda), chiamato frame.

- **Livello di Rete:**
  - Deve far giungere i "pacchetti" a destinazione.
  - Si occupa dell'istradamento ("routing") dei pacchetti, cioè di determinare la sequenza di collegamenti punto-punto necessari per trasmettere un pacchetto da un nodo generico della rete a un altro.
- **Livello di Trasporto:**
  - Questo strato fornisce un servizio di trasferimento dati end-to-end.
  - Si occupa di instaurare, mantenere e terminare una connessione.
  - Può offrire funzionalità per frammentare e riassemblare i dati, rilevare e correggere gli errori, controllare il flusso dei dati.
- **Livello di Sessione:** assembla il dialogo tra nodi in unità logiche.
- **Livello di Presentazione:** adatta la sintassi dei dati di ciascuna applicazione alla sintassi richiesta dalla sessione.
- **Livello di Applicazione:** protocolli a supporto di applicazioni distribuite.

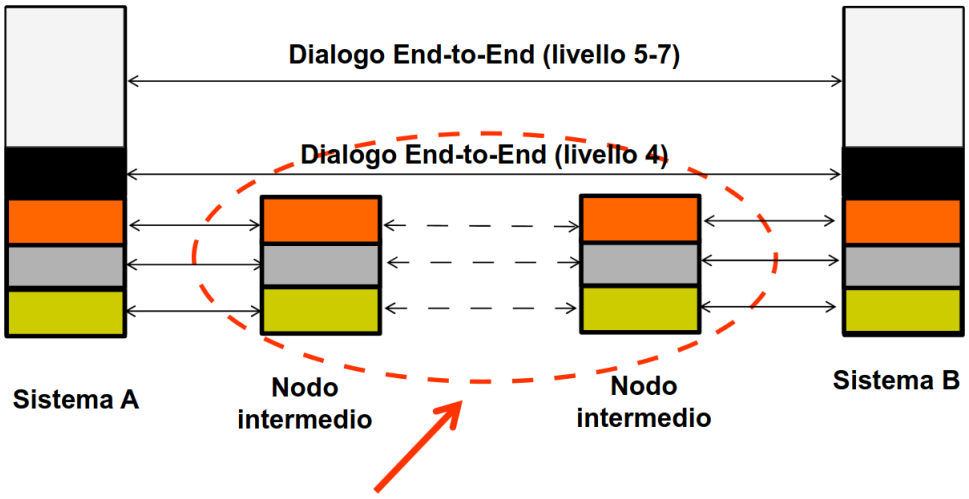

Figura 3: Collegamento tra End-System

Modalità di Servizio Esistono due principali metodologie di funzionamento per la comunicazione:

- **Connection-Oriented:**
  - Associazione logica tra due o più sistemi al fine di trasferire dati.
  - Gestione della connessione:
    - Instaurazione della connessione.
    - Trasferimento dati.
    - Chiusura della connessione
- **Connection-Less:** I dati vengono trasferiti senza l'instaurazione di una connessione.

Flusso dell'informazione Dal punto di vista delle rete, le informazioni hanno tutte origine dal livello Applicativo. L'informazione discende i vari livelli fino alla trasmissione sul canale fisico. Ogni livello aggiunge all'informazione del livello superiore una propria sezione informativa (o più) denominata header, che contiene informazioni riguardanti esclusivamente quel livello.
Per i dati ricevuti si segue il cammino inverso.

Incapsulamento Il processo di incapsulamento, ovvero quello nel quale ogni strato aggiunge dell'informazione a quella già presente, è un processo reversibile che garantisce l'estrazione durante la risalita delle informazioni dal livello di rete al livello applicativo (o fino ad un livello inferiore se ci troviamo in un nodo intermedio).
In questo processo abbiamo:

- Header: Contente informazioni relative a quel livello.
- Payload: Dati provenienti dal livello superiore.
- Tail: Generalmente utilizzato per l'individuazione e la correzione degli errori.

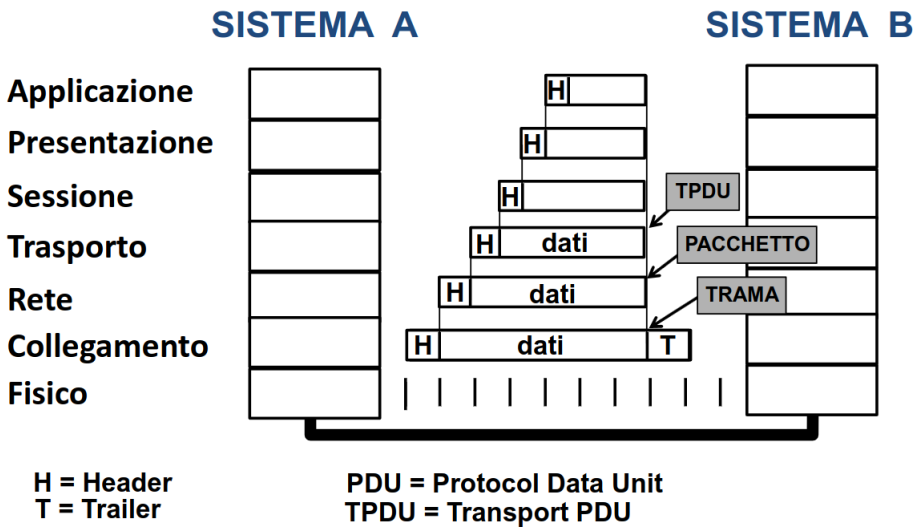

Figura 4: Processo di incapsulamento

2.4 Stack 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.
Definita in origine in termini di quattro livelli software soprastanti a un livello hardware, la pila TCP/IP è oggi intesa come composta di cinque livelli.

- **Applicazione:** supporta le applicazioni di rete, collegamento logico end-to-end; scambio di messaggi tra due processi (ftp, smtp, http).
- **Trasporto:** trasferimento dati end-to-end da un host sorgente all'host destinatario (Tcp, Udp).
- **Rete:** instradamento dei datagrammi dalla sorgente alla destinazione (Ip, ICMP).
- **Link:** trasferimento dati in frame attraverso il collegamento tra elementi di rete vicini (Point-to-Point Protocol, Ethernet, ...).
- **Fisico:** trasferimenti dei bit di un frame sul mezzo trasmissivo.

Differenze tra ISO/OSI e TCP/IP Nella pratica in Internet viene utilizzato lo stack protocollare TCP/IP. Questo accade perché ISO/OSI, a differenza di TCP/IP, fornisce una specifica generale, difficile da implementare e poco efficiente. ISO/OSI però viene utilizzato ancora come modello di riferimento.

3 Lo strato Applicativo

Applicazioni Le applicazioni di rete sono formate da processi distributi, su vari host. Ogni host può eseguire uno o più processi contemporaneamente. Questi processi comunicano tra loro mediante lo scambio di messaggi.
I livelli applicazione nei due lati della comunicazione agiscono come se esistesse un collegamento diretto su cui inviare questi messaggi.
I protocolli del livello applicativo definicono:

- Il tipo dei messaggi (es: di richiesta e di risposta).
- La sintassi dei vari tipi di messaggio (i campi del messaggio).
- La semantica dei campi (significato).
- Le regole per determinare quando e come un processo invia messaggi o risponde ai messaggi.

Paradigmi del livello applicativo Il modo in cui gli host devono comportarsi quando utilizzano una certa applicazione di rete dipende dal tipo di paradigma che questa applicazione utilizza:

- **Client-Server:** Un numero limitato di host, detti Server, offrono servizi e sono sempre in attesa di richieste; al contrario, i restanti host, detti Client, inviano richieste ai server per ricevere servizi.
- **Peer-to-Peer:** Tutti gli host che utilizzano l'applicazione sono allo "pari" tra loro, tutti chiedono e ricevono servizi da tutti gli altri.
- **Misto**

Definizione (API)**.** Application Programming Interface: insieme di regole che un programmatore deve rispettare per utilizzare delle risorse.

Interfaccia Socket L'interfaccia socket (l'API di internet per eccellenza) e quell'interfaccia che si frappone tra il livello di applicazione e il livello di trasporto. Questa interfaccia è messa a disposizione dal sistema operativo che implementa i 4 livelli inferiori dello stack protocollare TCP/IP. La socket è una struttura dati formata da:

- Indirizzo IP: identificativo della macchina con cui vogliamo comunicare (32bit per indirizzi ipv4, 128bit per indirizzi ipv6);
- Porta: numero a 16 bit che permette di identificare il processo sulla macchina con il quale si vuole comunicare.

3.1 Protocolli TCP e UDP

Nel livello di trasporto della pila di protocolli TCP/IP i due protocolli principali sono:

- **TCP** (Transmission Control Protocol): Protocollo connection-oriented, richiede una fase di setup tra client e server, che garantisce un trasporto affidabile dei dati. Fornisce inoltre meccaniscmi di controllo del flusso (il mittente non inonda il destinatario di dati) e della congestione (il mittente viene "rallentato" in caso di congestione della rete).
- **UDP** (User Datagram Protocol): Protocollo connection-less, non necessità di una fase di setup. Non garantisce trasporto affidabile e non presenta controlli di alcun tipo su flusso e congestione.

NB: entrambi non forniscono garanzie di timing e banda minima.

La scelta del protocollo da utilizzare dipende dalle esigenze dell'applicazione:

- Throughput: quanto è importante avere un certo livello di throughput per l'applicazione?
- Perdita dei dati: il 100% dei dati trasferiti devono necessariamente arrivare a destinazione?
- Sensibilità ai ritardi: l'applicazione è di tipo real-time?

3.2 URI, URL E URN

URI Uniform Resource Identifier, è una forma generale per identificare una risorsa presente in rete:

- Uniform: uniformità della sintassi dell'identificatore, anche con meccanismi diversi di accesso.
- Resource: qualsiasi cosa abbia un'identità (documento, servizio, immagine ...).
- Identifier: informazioni che permettono di distinguere un oggetto dagli altri.

Esistono due principali tipologie di URI:

- **URL:** (Uniform Resource Locator) sottoinsieme di URI che identifica le risorse attraverso il loro meccanismo di accesso.
- **URN:** (Uniform Resource Name) sottoinsieme di URI che devono rimanere globalmente unici e persistenti anche quando la risorsa cessa di esistere e diventa non disponibile.

URL Schema di una URL:

<scheme>://<user>:<password>@<host>:<port>/<path>

- <user> e <password> opzionale, in generale deprecato;
- <scheme> indica il protocollo di accesso alla risorsa;
- <host> nome di dominio di un host o indirizzo ip (in notazione decimale puntata);
- <port> numero di porta del server;
- <path> contiene dati specifici per l'host (o scheme) e identifica la risorsa nel contesto di quello schema e host.

A loro volta le URL possono essere di due tipologie differenti:

- URL assoluta: identifica una risorsa indipendentemente dal contesto in cui è usata;
- URL relativa: informazioni per identificare una risorsa in relazione ad un'altra URL (è priva dello schema e della authority).

Le URL relative non vengono utilizzate in rete, ma vengono interpretate dal browser in relazione al documento di partenza.

3.3 HTTP

HyperText Transfer Protocol Protocollo di tipo richiesta/risposta nel quale un client inizia la connessione, inviando al server una request, e ricevendo da esso una response. HTTP viene detto stateless inquanto ogni coppia richiesta/risposta è indipendente da tutte le altre. Per funzionare utilizza il protocollo di trasporto TCP con il quale viene instaurata una connessione tra client e server.

Definizione (Connessione)**.** Un circuito logico, al livello di trasporto, stabilito tra due programmi applicativi per comunicare tra loro.

Definizione (Connessione non persistente, da RFC 1945)**.** Viene stabilita una connessione TCP separata per recuperare ciascuna URL.

Definizione (Connessione persistente, da RFC 2616)**.** Se non diversamente indicato, il client può assumere che il server manterrà una connessione persistente.

- Lo standard specifica un meccanismo con cui client e server possono indicare la chiusura della connessione TCP (Connection header field).
- Dopo la chiusura, il client non deve più inviare richieste su quella connessione.

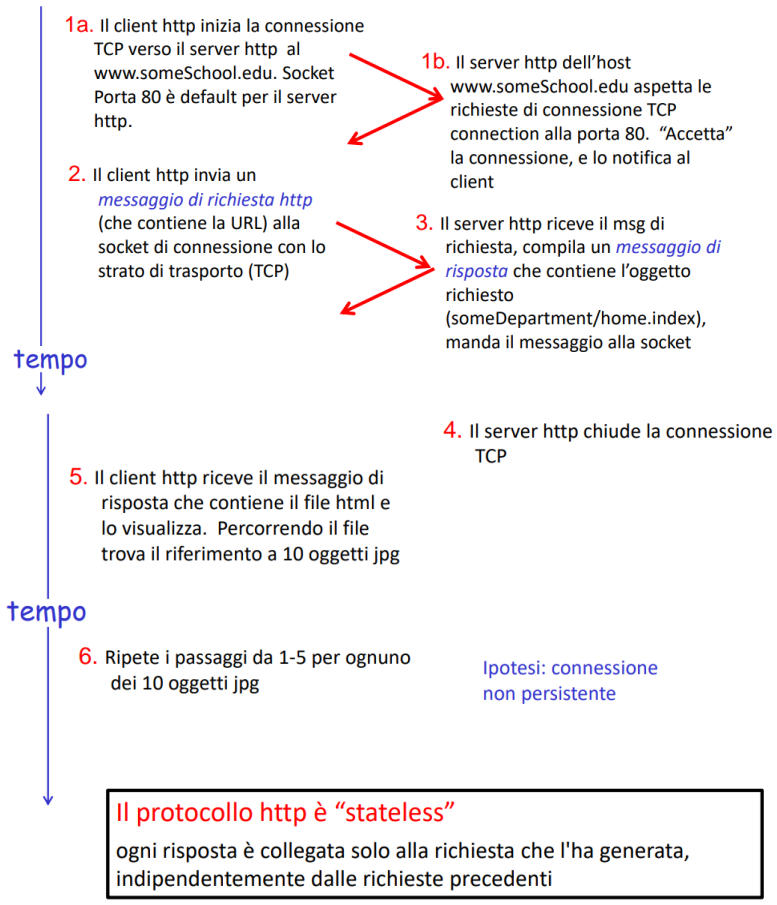

Figura 5: Esempio di interazione con protocollo HTTP.

L'utilizzo di una connessione persistente non permette di base ad un client di inviare una serie di richieste per ricevere una corrispondente serie di risposte; l'instaurazione di una connessione persistente ha lo scopo principale di migliorare le prestazioni della comunicazione.

Pipelining Il Pipelining è una tecnica di trasmissione delle richieste che 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 le richieste.
I server web che rispettano HTTP/1.1 devono supportare il pipelining, ma allo stesso tempo pochi browser web supportano questa tecnica. Questo accade perché se una richiesta necessita tempo per essere processata, le risposte alle richieste successive sono bloccate (Head

of Line Blocking) e in questi casi lo scopo principale di migliorare le prestazioni viene a mancare.

3.3.1 Formato messaggi HTTP

Come detto i messaggi HTTP possono essere di tipo richiesta o risposta, la struttura del messaggio è però la medesima:

- Start line
- Header
- Body

HTTP request line Nella start line denominata Request-Line per i messaggi di richiesta sono presenti tre campi:

- Method: operazione che il client richiede al server venga effettuata. I metodi più comuni sono GET, POST, PUT e DELETE;
- Request-URI: risorsa sulla quale il client vuole venga eseguita l'operazione;
- HTTP-Version: il mittente indica il formato del messaggio e la sua capacità di comprendere ulteriori comunicazioni HTTP.

HTTP status line Nella start line denominata Status-Line per i messaggi di risposta sono presenti tre campi:

- HTTP-Version
- Status-Code: intero a tre cifre che sta ad indicare l'esito della rispostsa (sono definiti dal protocollo).
- Reason-Phrase: descrizione testuale dello status code (pensata per l'utente umano).

Header Gli header sono coppie (nome: valore) che specificano alcuni parametri del messaggio trasmesso o ricevuto. Esistono vari tipi di header:

- General Header: relativi alla connessione (data, codifica, connessione,...);
- Entity Header: relativi all'entità trasmessa (content-type, content-length, data di scadenza,...);
- RequestHeader: relativi al messaggio di richiesta;
- Response Header: relativi al messaggio di risposta;

```
GET http://192.168.11.66/ HTTP/1.1
host: 192.168.11.66
Connection: close

HTTP/1.1 200 OK
Date: Sun, 14 May 2000 23:49:39 GMT
Server: Apache/1.3.9 (Unix) (Red Hat/Linux)
Last-Modified: Tue, 21 Sep 1999 14:46:36 GMT
ETag: "f2fc-799-37e79a4c"
Accept-Ranges: bytes
Content-Length: 1945
Connection: close
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
<HEAD><TITLE>Test Page for Red Hat Linux's Apache
Installation</TITLE></HEAD>
<H1 ALIGN="CENTER">It Worked!</H1>
<P>
If you can see this, it means that the installation of the
<AHREF="http://www.apache.org/">Apache</A> software
on this <ahref="http://www.redhat.com/">Red Hat Linux</a>
system was successful. You may now add content to this directory
and replace this page.
</P>
</BODY>
</HTML>
```

Content Negotation Le risorse che un client richiede possono essere disponibili in più rappresentazioni (lingua, formato di dati, dimensione, ecc..). La Content-Negotiation è un meccanismo per selezionare la rappresentazione appropriata quando viene servita una richiesta (uso di Request e Entity headers).

3.3.2 Metodi

OPTIONS Permette di richiedere al server di fornire le opzioni di comunicazione associate ad un URL o al server stesso (le sue capacità, metodi esposti,...).

```
OPTIONS http://192.168.11.66/manual/index.html
HTTP/1.1
host: 192.168.11.66
Connection: close

HTTP/1.1 200 OK
Date: Sun, 14 May 2000 19:52:12 GMT
Server: Apache/1.3.9 (Unix) (Red Hat/Linux)
Content-Length: 0
Allow: GET, HEAD, OPTIONS, TRACE
Connection: close
```

GET Permette di richiedere il trasferimento di una risorsa identificata da una URL o operazioni associate all'URL stessa.

```
GET http://192.168.11.66 HTTP/1.1
host: 192.168.11.66
Connection: close

HTTP/1.1 200 OK
Date: Sun, 14 May 2000 19:57:13 GMT
Server: Apache/1.3.9 (Unix) (Red Hat/Linux)
Last-Modified: Tue, 21 Sep 1999 14:46:36 GMT
ETag: "f2fc-799-37e79a4c"
Accept-Ranges: bytes
Content-Length: 1945
Connection: close
Content-Type: text/html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML> ...
```

Conditional GET Una variante del GET detta GET-Condizionale permette di modificare la risposta del server in base agli header utilizzati (Non modifica il numero totale di messaggi HTTP inviati).

```
GET http://192.168.11.66 HTTP/1.1
Host: 192.168.11.66
If-Modified-Since: Tue, 21 Sep 1999 14:46:36 GMT

HTTP/1.1 304 Not Modified
Date: Wed, 22 Sep 1999 15:06:36 GMT
Server: Apache/1.3.9 (Unix) (Red Hat/Linux)
```

HEAD Simile al GET, ma il server non trasferisce il message body nella risposta. Utile per controllare lo stato dei documenti (validità, modifiche, cache refresh).

POST Permette al client di inviare informazioni al server, le quali vengono inserite nel body.

Lo standard afferma che:
Il metodo POST è usato per chiedere che il server accetti l'entità (risorsa) nel corpo della richiesta come una nuova subordinata della risorsa identificata dalla Request-URI nella Request-Line.

Nella pratica la funzione effettivamente eseguita dal metodo POST è determinata dal server e dipendente tipicamente dalla Request-URI.
NB: con le API REST vengono rispettate le specifiche HTTP.

PUT Permette al client di chiedere al server di creare o modificare (se già esistente) una risorsa specificata nell'URI.

DELETE Permette al client di chiedere al server di cancellare una risorsa identificata dalla Request URI.

NB: I metodi PUT e DELETE sono normalmente non abilitati sui server web pubblici.

Metodi sicuri Con metodi sicuri si fa riferimento a quei metodi che non hanno "effetti collaterali" (per esempio modificare una risorsa). Questi sono: GET, HEAD, OPTIONS e TRACE.

Metodi idempotenti Con metodi idempotenti si fa riferimento a qeui metodi che presentano la proprietà per la quale $N > 0$ richieste identiche tra loro hanno lo stesso effetto di una singola richiesta. Questi metodi sono: GET, HEAD, PUT, DELETE, OPTIONS, TRACE.

3.3.3 Caching e Cookie

Web Caching Tecnica che ha lo scopo principale di soddisfare richieste del client senza contattare i server. Consiste nel memorizzare copie temporanee di risorse Web e servirle al client per ridurre l'uso di risorse e diminuire tempo di risposta. I due principali approcci sono:

- User Agent Cache: il browser mantiene una copia delle risorse visitate dall'utente per quando queste verranno nuovamente visitate.
- Proxy Cache: server proxy dedicati si intrappongono tra il client e il server. Quando un client esegue una richiesta questa passa per il server proxy e se è già stata memorizzata viene restituita al client; altrimenti il proxy si preoccupa di inoltrare la richiesta al server di competenza e una volta ricevuta la risposta la memorizza e la inoltra al client.

Cookie I Cookie sono un meccanismo con il quale è possibile memorizzare delle informazioni su un certo client. Come detto HTTP è un protocollo state-less perciò un server non ha modo di tenere traccia dei client (anche perché questi di norma non hanno un ip statico). La prima volta che un client invia un messaggio HTTP ad un server, questo lo "obbliga" a memorizzare un'insieme di informazioni che verranno utilizzate nelle successive comunicazioni.
Gli utilizzi principali dei cookie sono: autenticazione, memorizzazione dati form e in generale creare sessioni su un protocollo state-less.

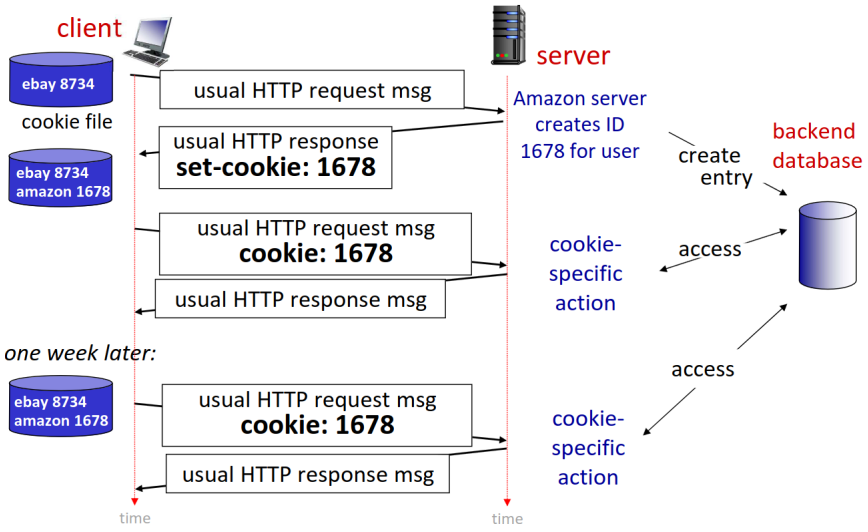

Figura 6: Esempio di interazione con protocollo HTTP e utilizzo di cookie

3.4 TELNET

Il protocollo TErminL NETwork ha lo scopo principale di permettere l'uso interattivo di macchine remote. Funzionamento:

1. la macchina locale stabilisce una connessione con un server login remoto;
2. tutte le battute dei tasti della macchina locale vengono inviati alla macchina remota, la quale esegue i comandi come se fossero stati battuti sulla macchina stessa.
3. l'output viene inoltrato dalla macchina remota alla macchina locale;

Per poter garantire questo funzionamento il modello di telnet include:

- Un programma server che accetta le richieste;
- Un programma client che effettua le richieste e interagisce direttamente con l'utente;

La connessione viene realizzata mediante il protocollo TCP sulla porta 23.

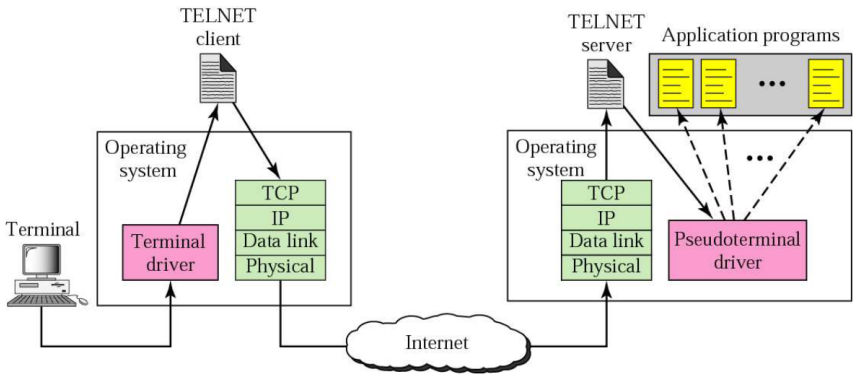

Figura 7: Esempio di interazione con protocollo TELNET

NVT Telnet deve poter operare con il numero massimo di sistemi e quindi gestire dettagli di sistemi operativi eterogenei (i quali possono differire per numerosi aspetti). Per risolvere il problema client e server devono eseguire un Network Virtual Terminal il quale effettua la codifica dei caratteri del sistema locale nel set di caratteri e comandi specifici della NVT.

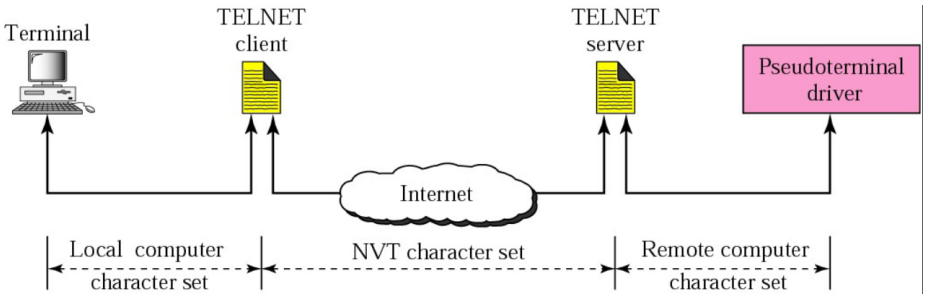

Figura 8: Esempio di interazione con protocollo TELNET e NVT

Gli NVT si scambiano dati in formato 7-bit US-ASCII. Ogni carattere è inviato come un ottetto con il primo bit settato a zero. I byte con il bit più significativo a 1 vengono usati per le sequenze di comandi e vengono preceduti da un ottetto speciale detto IAC (Interpret as Command).

I messaggi di controllo iniziali sono usati per scambiare informazioni sulle caratteristiche degli host (Telnet option negotiation).

Sicurezza Telnet non è un protocollo sicuro inquanto tutte le comunicazioni tra macchina locale e remota avvengono in chiaro.