Appunti di Reti

Luca Facchini
Matricola: 245965

Corso tenuto dal prof. Casari Paolo
Università degli Studi di Trento

A.A. 2024/2025

Sommario

Appunti del corso di Reti, tenuto dal prof. Casari Paolo presso l'Università degli Studi di Trento. Corso seguito nell'anno accademico 2024/2025.

Indice

Capitolo 1

Introduzione

1.1 Cos'è internet?

Internet
HostUna rete di milioni di dispositivi collegati detti "Host"
Applicazioni di reteun insieme di applicazioni di rete
Collegamentiuna rete di collegamenti fisici (rame, fibra ottica, onde elettromagnetiche, ecc...). Frequenza di trasmissione è uguale a ampiezza di banda
RouterInstrada i pacchetti verso la destinazione finale
ISPInternet Service Provider

Protocollo Un protocollo definisce il formato e l'ordine dei messaggi scambiati fra due o più entità in comunicazione

Standard
RFCRequest for Comments
IETFInternet Engineering Task Force

1.2 Ai confini della rete

Sistemi Terminali (Host)

1. Fanno girare i programmi applicativi (Web, email, ecc...)
2. Sono situati ai margini della rete

Architettura client-server Host client richiede e riceve i servizi da un programma server in esecuzione su un host server

Peer-to-peer Nessun server fisso, i peer sono client e server allo stesso tempo (es. BitTorrent, Skype)

1.2.1 Reti d'accesso e mezzi fisici

1.2.2 Accesso residenziale: punto-punto

Modem dial-up Fino a 56 kbp/s (mai raggiunti) su linea telefonica, non si può telefonare e navigare contemporaneamente

DSL - Digital Subscriber Line Installazione da parte di un ISP, fino a 1-5 Mbps in upstream e 10-50 Mbps in downstream, linea dedicata

1.2.3 FTTH - Fiber To The Home

Fibra ottica Fino a 2.5 Gbps in upstream e 2.5 Gbps in downstream

1.2.4 Accesso aziendale: reti locali (LAN)

LAN Una Local Area Network, o LAN, collega i sistemi terminali di aziende e università all'edge router

Ethernet
Velocità10 Mbps, 100 Mbps, 1 Gbps, 10 Gbps
Moderna Configurazione: sistemi terminali collegati a switch, switch collegato a router

1.2.5 Acceso wireless

Wireless LANs
Wi-Fi2.4 GHz, 5 GHz, 802.11b/g/n/ac
Cellular networks3G, 4G, 5G

1.3 Al nucleo della rete

La rete solitamente è composta da una maglia di router interconnessi, questi lavorano per trovare la migliore strada per arrivare alla destinazione più velocemente possibile

Commutazione di circuito Esistono delle risorse punto-punto riservate alla comunicazione
Time Division Multiplexing Il tempo è diviso in slot, ogni slot è assegnato ad una connessione diversa
Frequency Division Multiplexing La banda è divisa in frequenze, ogni frequenza è assegnata ad una connessione diversa.

Capitolo 2

Il Livello Applicazione

2.1 principi delle applicazioni di rete

2.1.1 Architetture di rete

Esistono varie architetture per le applicazioni di rete, tra le quali:

- Client-Server
- Peer-to-Peer
- Architetture ibride
- Cloud Computing

Client - Server

In questa architettura esistono due ruoli principali:

Server Il server è un host **sempre attivo** con un **indirizzo permanente** e molto spesso difficile da scalare

Client Il client **comunica col server**, inoltre a differenza del server può **disconnettersi temporaneamente** e inoltre può avere un **indirizzo IP dinamico**. Generalmente i client **non comunicano tra di loro**.

Architettura P2P pura

In questa architettura **non c'è** sempre un server attivo, vengono eseguire **coppie arbitrarie** di host che comunicano tra di loro. Infine i **peer** non devono necessariamente essere sempre attivi e possono avere un **indirizzo IP dinamico**.

Architetture ibride

In queste architetture si ha una combinazione tra client-server e P2P, ad esempio un server con peer che comunicano tra di loro. Un esempio di architettura ibrida è Skype, oppure un applicativo di messaggistica istantanea dove le chat sono P2P ma l'individuazione degli utenti è fatta tramite un server centrale.

Cloud Computing

In questa architettura si ha un insieme di tecnologie che permettono **di memorizzare archiviare e/o elaborare dati** tramite l'utilizzo di risorse distribuite. La creazione di copie di sicurezza dette **backup**

è automatica e l'operabilità si trasferisce online. I dati sono memorizzati in **server farm** generalmente localizzate nei paesi di origine del service provider.

2.1.2 Struttura delle applicazioni di rete

Processi del sistema operativo

Processo: Programma in esecuzione su un host.
All'interno di uno stesso host due processi comunicano utilizzando **schemi interprocesso** (definiti dal S.O.)
Processi su host diversi comunicano tramite **messaggi** scambiati tramite la rete.

Processo client processo che inizia la comunicazione

Processo server processo che attende di essere contattato

Socket

Socket Un processo che invia/riceve messaggi a/da il suo **socket**.
Un socket è analogo ad una porta

2.1.3 Indirizzamento

IP Per identificare un host in modo univoco si usa un **indirizzo IP** che è formato da 32 bit.

Numeri di porta L'indirizzo IP però non è sufficiente ad identificare un processo all'interno dell'host per questo definiamo dei **numeri di porta**.

2.1.4 Protocolli a livello applicazione

Definizioni

I protocolli a livello applicazione definiscono:

- Tipi di messaggi scambiati
- Sintassi dei messaggi
- Semantica dei campi dei messaggi
- Regole per determinare quando e come i processi inviano e ricevono messaggi

Protocolli dominio pubblico Alcuni protocolli sono di pubblico dominio definiti nelle **RFC** (Request for Comments) della **IETF** (Internet Engineering Task Force). Questi consentono interoperabilità tra diversi host, esempi di protocolli a pubblico dominio sono: **HTTP**, **SMTP**...

Protocolli proprietari Altri protocolli sono proprietari, ad esempio Skype.

2.1.5 Servizi di trasporto

Come segliere il protocollo di trasporto

Perdita di dati Applicazioni che richiedono trasmissione affidabile dei dati (es. file transfer) richiedono un protocollo di trasporto affidabile

Temporizzazione Applicazioni che richiedono bassa latenza (es. VoIP) richiedono un protocollo di trasporto con bassa temporizzazione

Throughput Applicazioni che richiedono alto throughput richiedono un protocollo di trasporto con alto throughput

Sicurezza Applicazioni che richiedono sicurezza (es. trasferimento di file) richiedono un protocollo di trasporto sicuro

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

TCP / UDP

TCP **Transmission Control Protocol** è un protocollo di trasporto **affidabile** e **orientato alla connessione**. TCP ha un **controllo di flusso** e **controllo di congestione**, **non offre** temporizzazione e garanzie su un'ampiezza di banda minima, sicurezza.

UDP **User Datagram Protocol** è un protocollo di trasporto **inaffidabile** fra i processi d'invio e di ricezione. UDP **non offre** controllo di flusso, controllo di congestione, temporizzazione, garanzie su un'ampiezza di banda minima, sicurezza.

Applicazione	**Protocollo a livello applicazione**	**Protocollo di trasporto**
Posta elettronica	SMTP [RFC 2821]	TCP
Accesso a terminali remoti	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
Trasferimento file	FTP [RFC 959]	TCP
Multimedia in streaming	HTTP [RFC 2616], RTP [RFC 3550]	TCP, UDP
Telefonia Internet	SIP [RFC 3261], RTP [RFC 3550], Proprietario	Tipicamente UDP

2.2 Web e HTTP

2.2.1 Terminologia

Pagina Web Una **pagina web** è costituita da **oggetti**

Oggetto Un **oggetto** può essere una **pagina HTML**, un'**immagine**, un'**applet**, un'**audio**, un'**video**, ...

un file HTML è un **file base** per formare una **pagina web**. Suddetto file è scritto tramite l'**HyperText Markup Language** che include diversi oggetti referenziati

URL Ogni oggetto è referenziato tramite un **URL** (Uniform Resource Locator)

Esempio di URL

`http://www.sito.com/folder/file.html`

http Protocollo di trasferimento

www.sito.com Nome del server

folder Cartella in cui si trova il file

file.html Nome del file

2.2.2 Introduzione a HTTP

Overview L'**HTTP** (HyperText Transfer Protocol) è un protocollo di livello applicazione del web. Sfrutta il modello **client-server** dove il **client** invia una **richiesta** al **server** che risponde con una **risposta** contenente il **contenuto richiesto** e il client visualizza il contenuto.

Usa TCP Il client inizializza una connessione **TCP** con il server sulla porta 80, il server accetta la connessione **TCP** del client e si scambiano messaggi HTTP tra il browser e il web-server. Quando il trasferimento è completato la connessione **TCP** viene chiusa.

Si noti come il protocollo HTTP sia **stateless**, ovvero non mantiene informazioni sullo stato del client.

Connessioni HTTP

Connessioni non persistenti Almeno un oggetto viene trasmesso su una connessione `TCP`.

1. Il client `HTTP` inizializza una connessione `TCP` con un server `HTTP` sulla porta 80
2. Il server `HTTP` sul host in attesa di una connessione `TCP` alla porta 80
3. Il client `HTTP` trasmette un *messaggio di richiesta* con l'*URL* nella socket della connessione `TCP`. Il messaggio indica che oggetto si vuole
4. Il server `HTTP` trasmette un *messaggio di risposta* con l'oggetto richiesto nella socket della connessione `TCP`
5. Il server chiude la connessione `TCP`
6. Il client riceve l'oggetto e visualizza l'oggetto richiesto e all'arrivo del messaggio di risposta chiude la connessione `TCP`

- Il metodo di connessione non persistente richiede 2 round-trip time (`RTT`) per ottenere un oggetto.
- Overhead di connessione `TCP` per ogni oggetto richiesto
- I browser moderni spesso in caso di connessioni non persistenti aprono richieste parallele per ottenere più oggetti contemporaneamente

Connessioni persistenti Più oggetti vengono trasmessi su una connessione TCP

Tipi dei metodi

GET Il client richiede un oggetto al server

POST Il client invia dati al server

HEAD Il client richiede solo l'intestazione dell'oggetto

PUT Il client invia un oggetto al server (da HTTP/1.1)

DELETE Il client cancella un oggetto dal server (da HTTP/1.1)

Messaggio di risposta HTTP

```
HTTP/1.1 200 OK ⇒ Versione del protocollo, codice di stato, frase di stato
Connection close ⇒ Connessione chiusa
Date:  Thu, 06 Aug 1998 12:00:15 GMT ⇒ Data e ora
Server:  Apache/1.3.0 (Unix) ⇒ Server web
Last-Modified:  Mon, 22 Jun 1998 ...  ⇒ Data ultima modifica
Content-Length:  6821 ⇒ Lunghezza del contenuto
Content-Type:  text/html ⇒ Tipo di contenuto
dati dati dati dati dati ...  ⇒ Dati
```

Codici di stato

200 OK ⇒ La richiesta è stata completata con successo l'oggetto richiesto è stato trasmesso

301 Moved Permanently ⇒ Il documento richiesto è stato spostato in un'altra locazione

400 Bad Request ⇒ La richiesta non può essere soddisfatta di errori client

404 Not Found ⇒ Il documento richiesto non è stato trovato sul server

505 HTTP Version Not Supported ⇒ La versione HTTP usata non è supportata dal server

2.2.3 Cookies

I **cookies** sono composti da quattro componenti:

1. Una riga di intestazione nel messaggio di *risposta HTTP*
2. Una riga di intestazione nel messaggio di *richiesta HTTP*
3. Un file mantenuto sul *client*
4. Un database mantenuto sul *server*

Come vengono usati cookies

Cosa contengono

- Autorizzazione
- Carta per acquisti
- Raccomandazioni
- Stato della sessioni dell'utente

Lo Stato

- Mantengono lo stato del mittente e del ricevente per più richieste
- I messaggi HTTP trasportano lo stato

Privacy

- I cookies possono essere usati per tracciare la navigazione dell'utente
- L'utente può fornire al sito nome e l'indirizzo

2.2.4 Cache web

Obbiettivo: soddisfare le richieste degli utenti senza coinvolgere il server d'origine

Cache è una copia di un oggetto mantenuta da un'entità più vicina all'utente

Il Procedimento Il client invia una richiesta al server proxy, il server proxy invia la richiesta al server d'origine se l'oggetto non è in cache, altrimenti il server proxy invia l'oggetto al client.

Vantaggi Riduzione del tempo di risposta, riduzione del traffico di rete, riduzione del carico sui server d'origine

Perchè viene usata Viene usata per ridurre il tempo di risposta e il traffico di rete, in certe situazioni delle istituzioni si possono dotare di un cache interna per ridurre il traffico di rete verso l'esterno e per ridurre il tempo di risposta.

`GET` **condizionale** Il client può chiedere al server proxy di inviare l'oggetto solo se è stato modificato, in caso contrario il server proxy invia un messaggio di risposta con codice 304 (Not Modified) e l'oggetto non viene inviato. Il controllo viene eseguito tramite un **header** `If-Modified-Since` che contiene la data dell'ultima modifica dell'oggetto.

2.2.5 HTTP/1.0 e HTTP/1.1

HTTP/1.0

- Connessioni non persistenti
- Ogni oggetto richiede una connessione TCP separata
- Non supporta proxy
- Non supporta cache

HTTP/1.1

- Connessioni persistenti
- Pipelining
- Host Virtuale
- Cache
- Cookies
- Connessioni persistenti

- Pipelining
- Host Virtuale
- Cache
- Cookies

2.2.6 HTTP/2.0

HTTP/2 rappresenta una evoluzione di **HTTP/1.1**, il protocollo è focalizzato sulle prestazioni, specificatamente sulla latenza percepita. Obbiettivo di `HTTP/2` è di avere una unica connessione per browser.

Framing binario

Nuovo livello di framing binario per incapsulare i messaggi `HTTP`, in questo modo la semantica `HTTP` rimane invariata ma la codifica in transito è differente. Tutte le comunicazioni `HTTP/2` sono suddivise in messaggi più piccoli, ognuno dei quali codificano un formato binario, inoltre sia il client che il server possono inviare messaggi in qualsiasi momento.ù

Stream, messaggi e frame

Tutte le comunicazioni vengono eseguite all'interno di una connessione TCP bidirezionale, ogni **stream** ha un identificativo univoco con priorità. Ogni messaggio è un messaggio `HTTP` logico (richiesta/risposta). Il frame è la più piccola unità di comunicazione di un certo tipo specifico di dati.

Multiplexing di richieste e risposte In **HTTP/1.x** se il client esegue più richieste in parallelo per migliorare le prestazioni deve usate `TCP` multiple (`HTTP/1.1 o HTTP/1.2`) oppure aprire una nuova connessione (`HTTP/1.0`). Grazie al **framing binario** di `HTTP/2` è possibile rimuovere queste limitazioni consentendo il **multiplexing** di richieste e risposte.

Priorità degli streamL'ordine nel quale i frame vengono inviati dal client o dal server influenza le prestazioni, per questo motivo `HTTP/2` supporta di associare a ciascun `stream` una priorità e delle dipendenze. Infatti ogni stream può avere un peso tra 1 ovvero il peso minimo e 256 ovvero il peso massimo, inoltre uno stream può avere un elenco di dipendenza su altri stream. Grazie a questa funzionalità il client costruisce un "**albero di priorità**" in modo da ottimizzare il caricamento della pagina.

Server Push Il server può inviare più risposte per una singola richiesta (se ad esempio è necessaria una dipendenza per il caricamento della pagina) in modo da ridurre il tempo di caricamento della pagina senza dover attendere la richiesta del client.

2.2.7 Transport Layer Security (TLS)

Il `TLS` ovvero **Transport Layer Security** è un protocollo crittografico che permette una comunicazione sicura da sorgente a destinatario fornendo: **Autenticazione**, **Integrità dei dati** e **Confidenzialità**. [1]
Il funzionamento del `TLS` può essere riassunto in tre fasi:

1. Negoziazione fra client e server per stabilire l'algoritmo di crittografia da usare
2. Scambio delle chiavi per la crittografia e autenticazione della comunicazione
3. Cifratura simmetrica dei dati e autenticazione dei dati

[1] Più sulla sicurezza di computer e reti in: "Appunti di Introduction to Computer and Network Security" di Luca Facchini

2.2.8 HTTPS

`HTTPS` è un protocollo di comunicazione sicura che estende `HTTP` aggiungendo una crittografia tramite `TLS`. Il protocollo `HTTPS` usa la porta 443 e permette tutti i vantaggi di `TLS` come l'autenticazione, l'integrità dei dati e la confidenzialità. Questo però non significa che tutto il traffico dei livelli inferiori sia crittografato, infatti solo il traffico (header e dati) del livello applicazione è crittografato.

2.3 FTP - File Transfer Protocol

FTP Il **File Transfer Protocol** è un protocollo di trasferimento di file che permette di trasferire file tra un host e un server. FTP è un protocollo **stateful** che mantiene lo stato del client e del server durante la sessione. Lo standard FTP è definito nella **RFC 959** e usa una porta standard di **21**.

2.3.1 Connessione di controllo

Connessione di controllo La connessione di controllo è usata per inviare comandi tra il client e il server. I comandi sono inviati in **ASCII** e i comandi sono **case-insensitive**. La connessione di controllo è **stateful** e mantiene lo stato del client e del server durante la sessione. La connessione di controllo usa la porta **21**, mentre la connessione dati usa la porta **20**, questo è un esempio di protocollo con **controllo fuori banda**.

2.3.2 Comandi & Risposte FTP

Comandi FTP

USER *username* Autenticazione con l'username

PASS *password* Autenticazione con la password

LIST Mostra i file nella directory corrente

RETR *filename* Recupera un file dalla directory corrente

STOR *filename* Memorizza un file nella directory corrente

Risposte FTP

331 Username OK, password richiesta

125 Connessione dati aperta, inizio trasferimento

425 Connessione dati non aperta

452 Errore di memorizzazione

2.4 Posta Elettronica

Introduzione Per la gestione della posta elettronica esistono 3 componenti principali:

- Agente utente
- Server di posta
- Simple Mail Transfer Protocol (SMTP)

Agente utenteè detto anche "mail reader" e permette di comporre, modificare e leggere i messaggi di posta elettronica. I messaggi in uscita o in arrivo vengono memorizzati sul server di posta che è sempre attivo.

Server di posta Contiene la **Casella di posta** che contiene i messaggi in arrivo, ha una **coda di messaggi** in uscita ed usa il **protocollo SMTP** tra server di posta per inviare messaggi di posta elettronica, in quanto il protocollo **SMTP** richiede che il server ricevente sia sempre in ascolto.

2.4.1 SMTP

Il protocollo **SMTP** (Simple Mail Transfer Protocol) è un protocollo di livello applicazione che permette di inviare messaggi di posta elettronica tra server di posta. Il protocollo **SMTP** usa la porta **25** ed è un protocollo **stateless**.

Fasi del trasferimento Il trasferimento di un messaggio di posta elettronica avviene in tre fasi:

Handshaking Il client apre una connessione **TCP** con il server di posta, il server risponde con un messaggio di benvenuto

trasferimento Il client invia il messaggio, il server accetta il messaggio e lo deposita nella casella di posta del destinatario

Chiusura Il client chiude la connessione

Iterazione comando/risposa I comando usano **ASCII** a 7 bit e sono **case-insensitive**, le risposte sono codificate con un codice a tre cifre.

Note finali

- Il protocollo usa connessioni **persistenti**
- Il protocollo richiede che il messaggio (intestazione e corpo) sia nel formato **ASCII** a 7 bit
- Il protocollo prevede che `<CR><LF>.<CR><LF>` sia usato per terminare il messaggio

Formato dei messaggi di posta elettronica

Intestazione contiene i mittenti, i destinatari, il soggetto, la data e l'ora

riga vuota separa l'intestazione dal corpo

Corpo contiene il testo del messaggio

2.4.2 POP3

Il protocollo **POP3** (Post Office Protocol 3) è un protocollo di livello applicazione che permette di scaricare i messaggi di posta elettronica dal server di posta. Il protocollo **POP3** usa la porta **110** ed è un protocollo **stateful**.

Fasi del trasferimento Il trasferimento di un messaggio di posta elettronica avviene in tre fasi:

autorizzazione Il client apre una connessione **TCP** con il server di posta, il client si autentica con il server

trasferimento Il client scarica i messaggi di posta elettronica

Chiusura Il client chiude la connessione

Comandi POP3

USER Autenticazione

PASS Password

LIST Lista dei messaggi

RETR Recupera un messaggio

DELE Cancella un messaggio

QUIT Chiude la connessione

2.4.3 IMAP

Il protocollo **IMAP** (Internet Message Access Protocol) è un protocollo di livello applicazione che permette di scaricare i messaggi di posta elettronica dal server di posta. Il protocollo **IMAP** usa la porta **143** ed è un protocollo **stateful**.

Fasi del trasferimento Il trasferimento di un messaggio di posta elettronica avviene in tre fasi:

autorizzazione Il client apre una connessione **TCP** con il server di posta, il client si autentica con il server

trasferimento Il client scarica i messaggi di posta elettronica

Chiusura Il client chiude la connessione

Comandi IMAP

LOGIN Autenticazione

SELECT Seleziona una casella di posta

FETCH Recupera un messaggio

STORE Modifica lo stato di un messaggio

LOGOUT Chiude la connessione

2.5 DNS

Introduzione

Domain Name Sysyem Il **DNS** consiste in un *database distribuito* implementando una gerarchia di *server DNS*. Il *DNS* è un protocollo a livello applicazione che consente agli host e ai router di comunicare per *risolvere* i nomi degli host in indirizzi IP.

2.5.1 Servizi DNS

- Traduzione degli hostname in indirizzi IP
- Host aliasing - Un host può avere più nomi
- Mail server aliasing - Un host può avere più server di posta
- Payload distribution - Distribuzione del carico tra i server

Perchè non centralizzare DNS

- Singolo punto di fallimento
- Traffico di rete
- Database centralizzato distante
- Manutenzione

2.5.2 Struttura del DNS

In generale i server `DNS` sono organizzati in una struttura gerarchica a **albero** dove il nodo radice è il server `DNS` radice (13 al mondo) esistono dei server di `DNS` di nomi di primo livello (com) (TLD) e infine i server di `DNS` autoritativi usati per un dominio di secondo livello (google.com)

Server `DNS` locali Ogni ISP ha un server `DNS` locale che si occupa di tradurre i nomi degli host in indirizzi IP

2.5.3 Resource Record `RR`

Resource Record Un `RR` è una tupla che contiene i seguenti campi:

- `Name` - Il nome del dominio
- `Value` - Il valore del campo
- `Type` - Il tipo di record
- `TTL` - Il tempo di vita del record

Tipi di `RR`

- `A` - Indirizzo IP - **name**: `hostname` **value**: `IP`
- `NS` - Server di nomi - **name**: `dominio` **value**: `hostname`
- `CNAME` - Nome canonico - **name**: `alias` **value**: `hostname`
- `MX` - Mail server - **name**: `dominio` **value**: `hostname`

2.5.4 Inserire un record

Esempio Abbiamo avviato la nuova società

- Registriamo il nome "foo.com" presso un `registrar`
- Otteniamo un indirizzo IP per il nostro server web (host)
- Diamo al nostro registrar l'indirizzo IP del nostro server web e il nome del nostro server web. Esempio records: `(foo.com, dns1.foo.com, NS), (dns1.foo.com, 211.211.211.211, A)`