

# Appunti di SETI

Luca Bianchi e Marco Zucca

Settembre 2020

# Indice

|          |   |          |
|----------|---|----------|
| <b>1</b> | <b>Rete</b>                               | <b>3</b> |
| 1.1      | Modello TCP/IP . . . . .                  | 3        |
| 1.1.1    | Layers . . . . .                          | 3        |
| 1.2      | TCP . . . . .                             | 3        |
| 1.2.1    | 3-Way Handshake . . . . .                 | 3        |
| 1.3      | UDP . . . . .                             | 4        |
| <b>2</b> | <b>Protocolli Applicativi</b>             | <b>5</b> |
| 2.1      | DNS . . . . .                             | 5        |
| 2.1.1    | Organizzazione dei server . . . . .       | 5        |
| 2.1.2    | Algoritmi . . . . .                       | 6        |
| 2.1.3    | Risposta DNS . . . . .                    | 8        |
| 2.2      | Network Time Protocol . . . . .           | 8        |
| 2.2.1    | Classificazione Server . . . . .          | 9        |
| 2.2.2    | Funzionamento . . . . .                   | 9        |
| 2.2.3    | Possibili problemi di sicurezza . . . . . | 10       |
| 2.2.4    | SNTP . . . . .                            | 10       |
| 2.2.5    | NTPsec . . . . .                          | 10       |
| 2.3      | File Transfer Protocol . . . . .          | 10       |
| 2.3.1    | Sicurezza . . . . .                       | 11       |
| 2.3.2    | SFTP . . . . .                            | 11       |
| 2.4      | Simple Mail Transfer Protocol . . . . .   | 11       |
| 2.4.1    | Sicurezza . . . . .                       | 12       |
| 2.4.2    | Post Office Protocol 3 . . . . .          | 12       |
| 2.4.3    | Internet Mail Access Protocol 4 . . . . . | 12       |
| 2.5      | Hyper Text Transfer Protocol . . . . .    | 12       |
| 2.5.1    | Risposta server . . . . .                 | 13       |
| 2.5.2    | HTTP 1.0 . . . . .                        | 13       |
| 2.5.3    | HTTP 1.1 . . . . .                        | 14       |
| 2.5.4    | HTTP 2.0 . . . . .                        | 14       |
| 2.5.5    | Cookie . . . . .                          | 14       |
| 2.5.6    | Caching HTTP . . . . .                    | 14       |

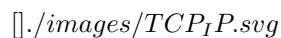
|          |   |           |
|----------|---|-----------|
| <b>3</b> | <b>Protocolli di Trasporto</b>                              | <b>15</b> |
| 3.1      | TCP . . . . .   | 15        |
| 3.1.1    | Controllo di Flusso . . . . .                               | 15        |
| 3.1.2    | Controllo di congestione . . . . .                          | 15        |
| 3.1.3    | Max Transfer Unit . . . . .                                 | 16        |
| <b>4</b> | <b>IP</b>   | <b>17</b> |
| 4.1      | Router . . . . .  | 17        |
| 4.1.1    | Architettura Router . . . . .                               | 17        |
| 4.1.2    | ICMP . . . . .  | 17        |
| 4.1.3    | Algoritmi di Routing . . . . .                              | 17        |
| 4.2      | IP . . . . .  | 19        |
| 4.2.1    | Type of Service . . . . .                                   | 19        |
| 4.2.2    | Datagram Length . . . . .                                   | 19        |
| 4.2.3    | Identifier e Offset . . . . .                               | 19        |
| 4.2.4    | Time To Live . . . . .                                      | 19        |
| 4.2.5    | Checksum . . . . .  | 19        |
| 4.2.6    | Next Level Protocol . . . . .                               | 19        |
| 4.2.7    | IPv4 . . . . .  | 20        |
| 4.2.8    | Network Address Translation . . . . .                       | 20        |
| 4.2.9    | IPv6 . . . . .  | 20        |
| 4.2.10   | Tunneling . . . . .   | 20        |
| <b>5</b> | <b>Data Link</b>  | <b>21</b> |
| 5.1      | Ethernet . . . . .  | 21        |
| 5.1.1    | Media Access Control . . . . .                              | 21        |
| 5.1.2    | Carrier Sense Multiple Access/Collision Detection . . . . . | 21        |
| 5.2      | Hub . . . . .   | 22        |
| 5.3      | Switch . . . . .  | 22        |
| 5.3.1    | 802.3 . . . . .   | 22        |
| 5.4      | LAN . . . . .   | 23        |
| 5.4.1    | Address Resolution Protocol . . . . .                       | 23        |
| 5.4.2    | ARP cache poisoning . . . . .                               | 23        |
| 5.4.3    | Dynamic Host Control Protocol . . . . .                     | 23        |

# Capitolo 1

## Rete

### 1.1 Modello TCP/IP

#### 1.1.1 Layers

 *./images/TCP\_IP.svg*

**Data Link** Protocollo Ethernet 802.3 (cavo)

**Rete** IPv4 e IPv6 (128 bit di indirizzo)

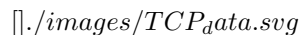
**Trasporto** TCP e UDP

Si effettua un'opera di multiplexing/demultiplexing per permettere a più applicativi di comunicare in rete, virtualizzando la rete tramite un sistema di porte

**Applicativo** FTP, HTTP, SMTP, etc.

Attraverso l'interfaccia dei socket è possibile accedere ad ogni layer.

### 1.2 TCP

 *./images/TCP\_data.svg*

#### 1.2.1 3-Way Handshake


Client → Server: Syn, Seq#

Server → Client: Syn, Ack (Seq# + 1)

Client → Server: Ack, Ack#, Seq#

Se una delle due parti non riceve l'ack entro la finestra temporale di timeout, si

effettua la ritrasmissione del messaggio.

./images/TCP<sub>h</sub>andshake.svg

## 1.3 UDP

**Datagram Stream** Invio di messaggi di lunghezza specificata dal mittente. Header contiene indirizzo e porta del destinatario. Il messaggio viene chiamato Payload. Alcuni messaggi potrebbero andare persi durante il trasferimento, di conseguenza i datagram non sono affidabili

## Capitolo 2

# Protocolli Applicativi

### 2.1 DNS

Si usa per la traduzione da URL a indirizzo IP (generalmente v4). Lo scopo è di rendere fruibile agli utenti umani la rete internet. Utilizza un sistema di server distribuiti su scala globale. Usa UDP.

#### 2.1.1 Organizzazione dei server

I server DNS sono divisi in

1. Root Name Server  $\leftarrow$  A livello logico si tratta di un solo server, in pratica è replicato in varie aree geografiche per ridurre i tempi di latenza ed il carico sui singoli server. Le richieste al RNS vengono effettuate da altri server DNS.
2. Top Level Domain  $\leftarrow$  (.it, .ch, .us) Ne esiste circa uno per nazione, si occupano di gestire la risoluzione dei DNS associati al loro stato.
3. Autoritativi  $\leftarrow$  Fanno riferimento ad un top level domain (unige.it)
4. Local Server  $\leftarrow$  Il client vi si connette, inviando un datagramma di richiesta ed ottenendo un datagramma di risposta, contenenti, rispettivamente, l'URL di cui si chiede la risoluzione e l'indirizzo IP associato ad esso.

Lo scopo del local server è interrogare la gerarchia fino a trovare l'indirizzo richiesto.

Sono stati individuati due algoritmi per la ricerca delle informazioni, uno iterativo ed uno ricorsivo

### 2.1.2 Algoritmi

#### Ricorsivo

Prima si invia la richiesta al RNS, che procede al contattare il Top Level Domain adeguato, che procederebbe a sua volta a contattare il server Autoritativo adeguato, che andrà a sua volta a contattare un subserver autoritativo contenente finalmente l'indirizzo richiesto, cominciando quindi a ritornare la risposta con una catena di inoltri.

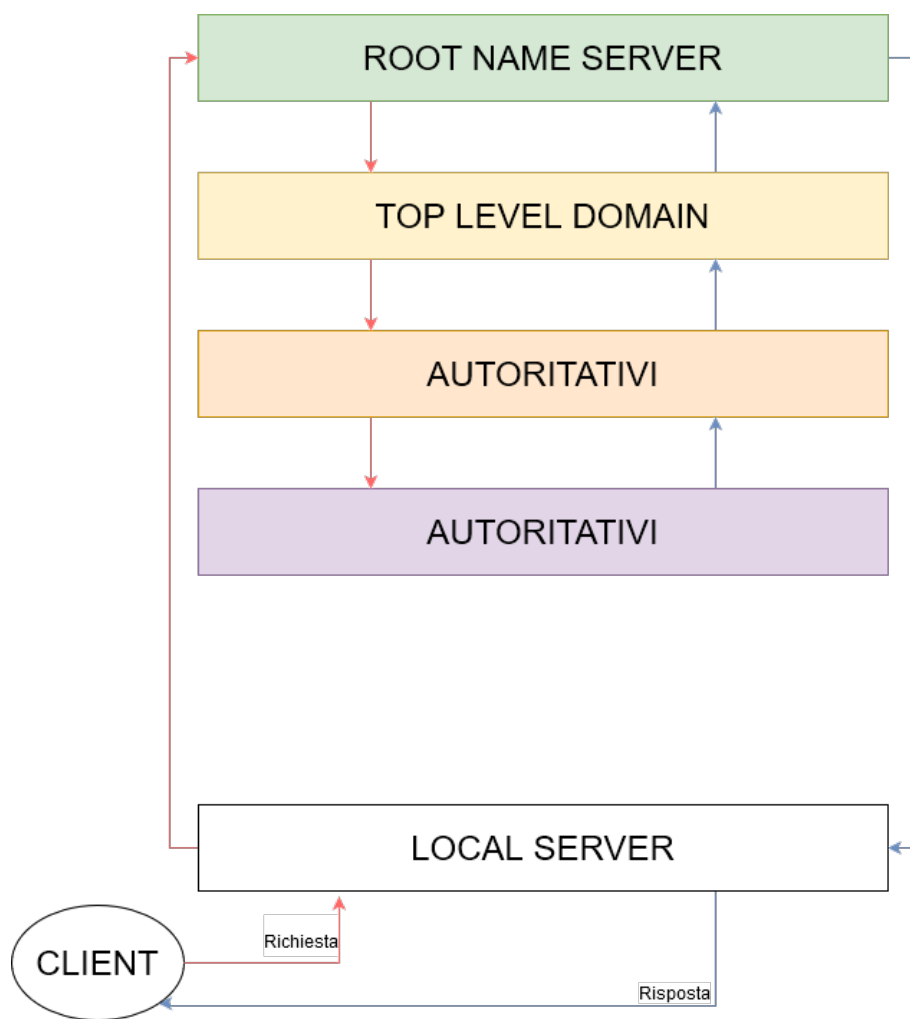


Figura 2.1: Schema di funzionamento del sistema ricorsivo

### Iterativo

Più efficiente rispetto al ricorsivo dal punto di vista dei server, in quanto non perdono tempo aspettando risposte dagli altri server. Viene definito stateless, in quanto si inviano le risposte senza mantenere memoria di quanto è stato fatto.

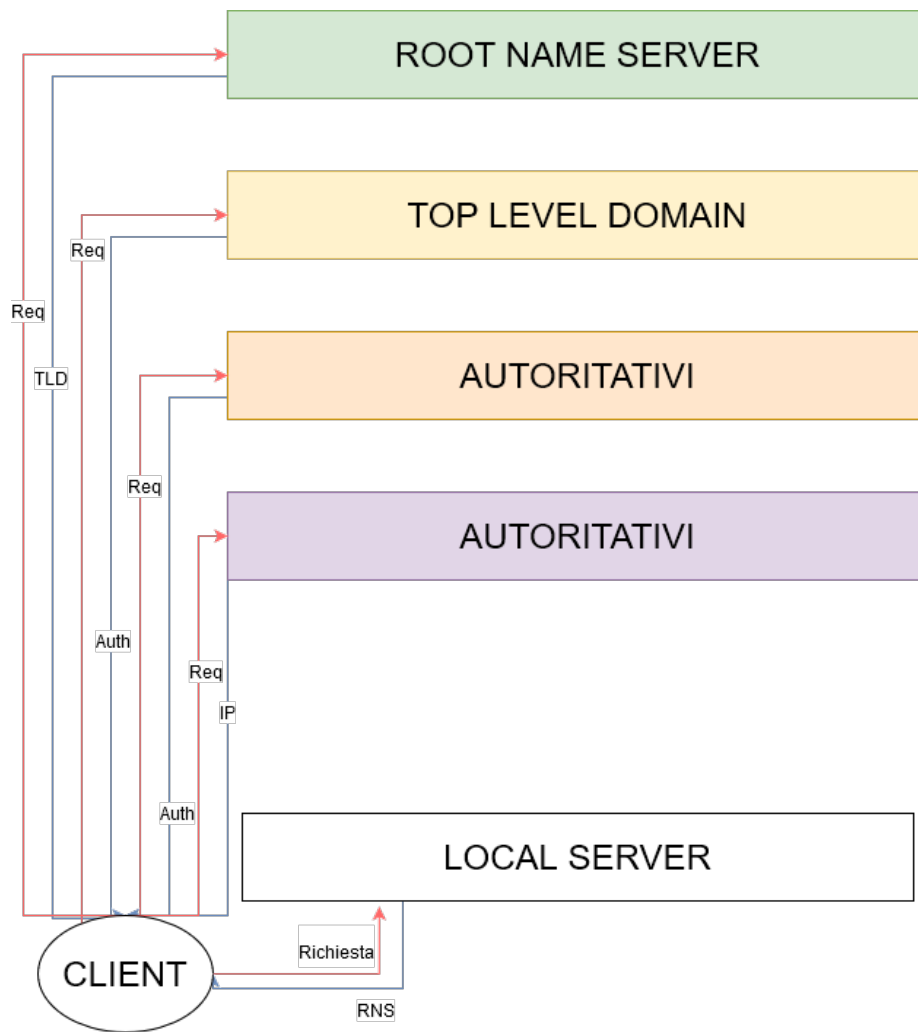


Figura 2.2: Schema di funzionamento del sistema iterativo

Generalmente, si utilizzano i server remoti in modalità iterativa ed il server locale in modalità ricorsiva. Il server locale generalmente utilizza un sistema di cache, per rispondere più velocemente a richieste successive del client in caso replichi la stessa richiesta. Il TTL della cache serve a determinare quando cancellare l'indirizzo contenuto in memoria.



### 2.1.3 Risposta DNS

| ID                                    | FLAG                 |
|---------------------------------------|----------------------|
| # Requests                            | # Answers            |
| # Auth. Answers                       | # Additional Answers |
| Richiesta (stringa)                   |                      |
| Risposta non autoritativa (IPv4, TTL) |                      |
| Risposta autoritativa                 |                      |
| Risposta addizionale                  |                      |

Figura 2.3: Header DNS

È suscettibile ad attacchi di cache poisoning, tramite il crafting da parte di  $2^{16}$  risposte nel caso in cui la porta sia fissa. L'attacco si fa più complesso nel caso la porta sia variabile, rendendo necessario generare circa  $2^{32}$  richieste per avere la certezza che l'attacco vada a buon fine. Nella versione TCP del protocollo, si aggiunge anche la necessità di indovinare il SEQ#, rendendo quasi impossibile che un attaccante riesca a portare l'attacco a termine.

## 2.2 Network Time Protocol

Abbiamo un orologio principale implementato a livello hardware per monitorare lo scorrere del tempo a macchina spenta. Questo viene interpellato in fase di bootstrap per aggiornare l'ora di sistema e passare all'orologio software. Abbiamo un contatore che viene aggiornato ad ogni interrupt che si genera, e che va a "contare" il tempo passato dal 1/1/1970. tv\_sec e tv\_nsec contano rispettiva-

mente il numero di secondi e di nanosecondi. Nel momento in cui il contatore nsec conta un secondo, si azzerà e si aggiorna il contatore sec.  
La versione UTC usa come data di partenza l'1/1/1900 ed usa un contatore a 32bit senza segno.  
NTP usa la rappresentazione UTC, e si basa sul protocollo UDP.

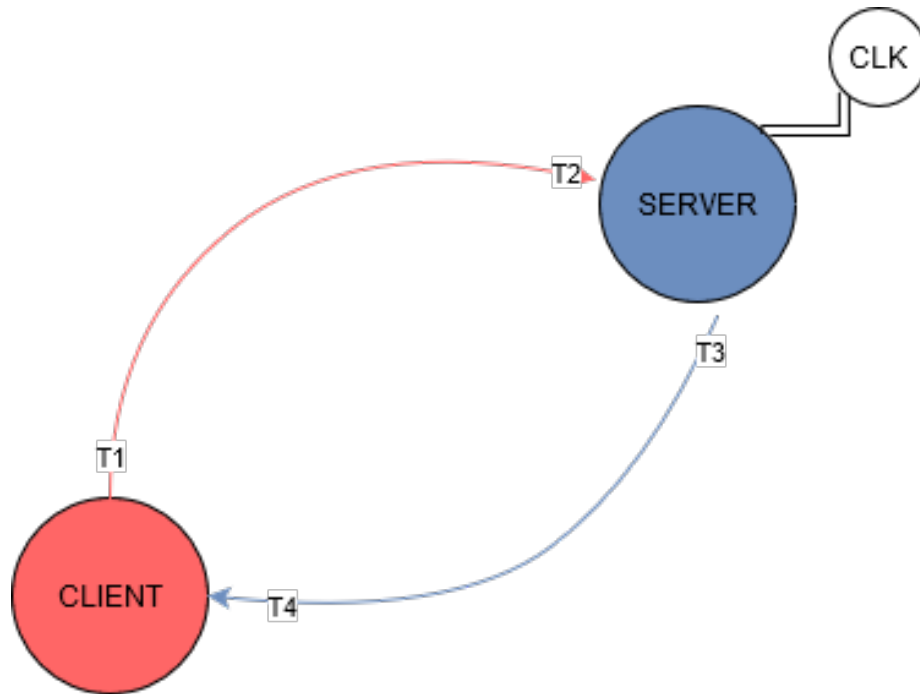


Figura 2.4: Schema NTP

### 2.2.1 Classificazione Server

1. Orologio Atomico
2. Server Principale
3. Server di strato 2 (si sincronizza con un server principale)
4. Server di strato 3 (si sincronizza con un server s2)

### 2.2.2 Funzionamento

Siccome UDP è un protocollo non affidabile, per essere sicuro di ricevere almeno una risposta il protocollo NTP interpella diversi server, per poi selezionare i server migliori (ossia, che rispondono E hanno i tempi di latenza medi più

stabili). Alcuni server di strato 1 si interfacciano con gli orologi presenti sui satelliti geostazionari del GPS, in quanto orologi atomici.

### **2.2.3 Possibili problemi di sicurezza**

È possibile, in caso di errore nella sincronizzazione dell'orario, magari per colpa di un MITM, che vengano accettati certificati digitali in realtà scaduti. Altre problematiche posso insorgere a causa dell'utilizzo del protocollo UDP, che permette in genere una più facile intromissione di terzi, permettendo attacchi di spoofing. NTP è inoltre utilizzabile per effettuare attacchi DDoS, fingendo che la richiesta parta dal dispositivo vittima.

### **2.2.4 SNTP**

Versione semplificata del protocollo NTP, principalmente per dispositivi IoT

### **2.2.5 NTPsec**

Versione sicura del protocollo NTP, richiede l'autenticazione con il server

## **2.3 File Transfer Protocol**

Usato per lo scambio di file. RFC 959, utilizza la porta 20 per lo scambio di comandi di controllo e la porta 21 per lo scambio di dati. Il protocollo utilizzato è il TCP. Nel momento in cui ci si connette al server FTP, sono visibili tutti i file visibili dall'utente con cui ci si è loggati. Quando si è connessi, bisogna tener conto di 2 current directory, una locale ed una remota. Il comando per cambiare la directory remota è cd, quello per la directory locale lcd. Si può ottenere una lista di file tramite il comando dir.

In modalità attiva il server regola il trasferimento di dati, agendo quindi da client.

In modalità passiva il client regola entrambe le connessioni ed il server si limita a rispondere, e, per evitare problemi di sicurezza, il client notifica continuamente al server da che porta proviene la richiesta e su quale porta accettarla, in modo da rendere più difficile attacchi.

L'autenticazione viene effettuato tramite username e password, veniva utilizzato un account anonymous solo per il download o l'upload, generalmente utilizzato per dare la possibilità agli utenti di inviare bug report. È possibile connettersi alla porta 20 in telnet e scambiare messaggi col server.

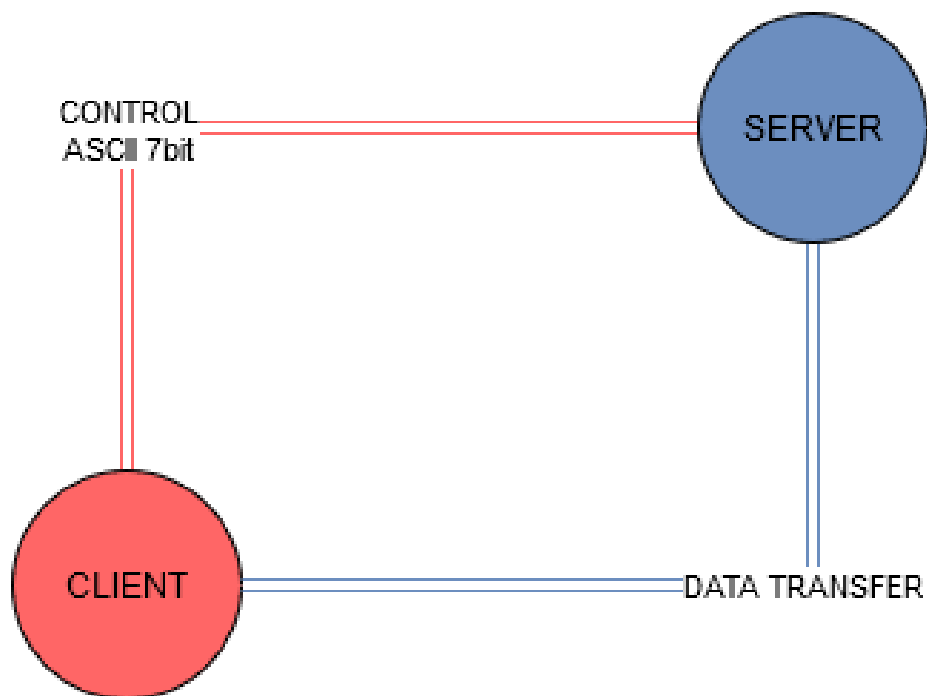


Figura 2.5: Schema FTP

### 2.3.1 Sicurezza

Il protocollo è suscettibile a spoofing, in quanto è possibile richiedere al server di inviare i file ad una macchina vittima.

Inoltre, in modalità attiva, il protocollo permette al server di contattare il client per la connessione, il che va contro mille milioni di regole di sicurezza. Il protocollo è soggetto ad attacchi di tipo sniffing.

### 2.3.2 SFTP

Versione attualmente in uso, utilizza cifratura TLS per evitare sniffing.

## 2.4 Simple Mail Transfer Protocol

Usato per stabilire una connessione di tipo asincrona, in modo da poter mandare messaggi anche quando il destinatario non è online. La comunicazione tra gli utenti è mediata da un server, che si occupa di salvare i messaggi in una mailbox. Nello specifico, SMTP si occupa della fase di invio dei messaggi. Utilizza il protocollo TCP, porta 25. È possibile delegare ad un server mail (Mail User Agent) l'invio di un messaggio ad un altro server mail (Mail Transfer Agent), e

la catena può continuare fino ad arrivare al MUA di destinazione. Le trasmissioni sono codificate in ASCII 7 bit, i campi dell'intestazione sono SUBJECT, FROM, TO e DATE. I caratteri terminatori di linea utilizzati sono <CR> e <LF>. La ripetizione dei caratteri indica la terminazione dell'header e l'inizio del body, terminato da un punto ad inizio riga. Non è possibile dunque inviare un messaggio composto solo da un punto. Per permettere l'invio di caratteri non ASCII 7bit, il messaggio può essere ricodificato in radix64.

DNS contiene dei record specifici (MX) che indicano, per ogni dominio, quale server di posta elettronica contattare.

Opera in modalità best effort.

### **2.4.1 Sicurezza**

L'utente root sul MUA può leggere tutti i messaggi sul server. Non è prevista alcuna autenticazione per l'invio dei messaggi, permettendo attacchi flood e spam. In una revisione successiva del protocollo, è stato inserito l'obbligo di login per inviare mail.

### **2.4.2 Post Office Protocol 3**

Le mail vengono scaricate dal server e copiate in locale, ed infine cancellate dal server, rendendo quindi impossibile accedere da più dispositivi alla stessa mailbox.

### **2.4.3 Internet Mail Access Protocol 4**

Le mail vengono mantenute sul server ed effettua copie parziali tramite caching sulla macchina locale.

## **2.5 Hyper Text Transfer Protocol**

TCP, porta 80.

Prima riga composta da method, resource, version.

METHOD : tipo di richiesta.

- GET
- POST
- HEAD
- PUT
- DELETE
- PATCH
- TRACE

- OPTION
- CONNECT

Seconda riga: OPTIONS.

Esempi di opzioni (tutte terminate da <CR><LF>):

- Host (v 1.1) → Nome simbolico dell'host a cui si vuole riferire. Utile con virtual host (domini virtualizzati, server multihost).
- User-Agent → Specifica il client che manda la richiesta.
- Cookie → Memorizza dati, lato client.
- If-Modified-Since

### 2.5.1 Risposta server

Prima riga: STATUS (codificati):

- 200 OK
- 3xx Redirect
- 301 Moved Perm, il contenuto è su un altro sito
- 4xx Errori
- 400 Bad Request
- 404 Not Found
- 5xx Errori Server
- 500 Internal Server Error
- 505 HTTP version not implemented.

OPTIONS:

- Server
- Content-Type (MIME)

BODY.

La codifica ASCII si limita all'header, il file di risposta non deve necessariamente essere ASCII.

### 2.5.2 HTTP 1.0

Le connessioni non sono permanenti, vengono aperte quando servono e chiuse a fine risposta. Inefficiente perchè bisogna attendere 2RTT (Uno per 3WHS con TCP/80 ed uno per richiesta HTTP e risposta).

### 2.5.3 HTTP 1.1

Connessioni persistenti. Si usa la prima connessione per le richieste e la si mantiene aperta anche per le risposte, fino ad un timeout. Il client si accorge di aver bisogno di altre richieste dopo la prima richiesta. Non c'è overhead, il Time To Live porta alla chiusura automatica della connessione se non la si chiude manualmente.

#### Richieste in pipeline

Permette al client di mandare più richieste in parallelo, ed il server risponderà alle richieste in ordine.

### 2.5.4 HTTP 2.0

Multiplexing per gestire in parallelo comunicazioni diverse. Identificatori in cui ogni richiesta corrisponde ad un flusso di informazioni. Ogni richiesta avrà un ID univoco e ogni risposta sarà associata all'ID, in modo da poter rispondere in modo non ordinato.

### 2.5.5 Cookie

Set-Cookie: Nome - Valore - Scadenza.

Quando il server manda un set-cookie, il client salva il valore e si occuperà di ritrasmetterlo ad ogni richiesta. Serve per aggirare l'impostazione stateless di HTTP, così che il server possa utilizzare un sistema di sessioni.

### 2.5.6 Caching HTTP

#### Proxy

Il proxy funziona da intermediario. Copia la richiesta ricevuta e la risposta ottenuta dal server. In questo modo, se la richiesta si ripete, può inviare in risposta il file salvato. Per capire se la copia è consistente, il server specifica una data in last-modified. Il client può usare l'header if-modified-since per ottenere una risposta aggiornata nel caso questa sia cambiata dalla data last-modified. Nel caso questa non sia cambiata la risposta del server consisterà in un 304 NOT MODIFIED, senza il contenuto del file.

Il browser può essere configurato per fare richieste condizionali.

- Proxy trasparente → proxy che filtra, con le richieste HTTP che vengono redirette forzatamente al proxy per poter effettuare monitoraggio del traffico.
- Proxy esplicito → dal browser si sceglie esplicitamente un proxy a cui connettersi.

## Capitolo 3

# Protocolli di Trasporto

### 3.1 TCP

#### 3.1.1 Controllo di Flusso

Serve a ridurre quanto più possibile le perdite di messaggi. Nel momento in cui si riceve un messaggio si effettua un controllo di integrità. Se fallisce, il messaggio viene scartato e si attende il reinvio. Nel caso in cui il destinatario sia più lento a processare i dati di quanto non sia il mittente ad inviarli, si rallenta la trasmissione per evitare overflow nel buffer di ricezione. Durante il 3 way handshake ci si scambia la dimensioni dei buffer di ricezione. I dati rimangono salvati nel buffer finchè non vengono processati, e ad ogni messaggio inviato si notifica la receive window, ossia quanti byte si possono ancora ricevere. Nel momento in cui la recv window si azzerà, il mittente si ferma, attendendo che il destinatario liberi parte della memoria e procedendo in seguito ad inviare un numero di byte pari a quello che si è liberato. C'è il rischio che il mittente non sappia quando il destinatario si è liberato, di conseguenza invia periodicamente un messaggio, in modo da ricevere la nuova recv window se il destinatario si è liberato, o causare un reset della connessione (nel caso peggiore). Il messaggio inviato in questa situazione contiene un solo byte.

#### 3.1.2 Controllo di congestione

Il problema precedente si ripropone sui buffer dei router. Per evitare che si congestionino, si stima la congestion window del percorso, pari alla dimensione del buffer più piccolo sul percorso. Il protocollo per effettuare la stima si può dividere in 4 fasi

1. Slow Start → si parte con un valore del messaggio più piccolo della congestion window, generalmente 1.
2. Lo si fa crescere in modo esponenziale fino a quando smette di ricevere risposte dal destinatario.



3. Dopo aver trovato il range di valori in cui risiede la c.w., ed aver fatto smaltire alla rete il sovraccarico, mandando pacchetti da 1 per ottenere un ack che notifichi lo sblocco della rete, aumento il numero di byte inviati in modo lineare fino ad arrivare al valore critico.
4. Tengo come valore della c.w. il minimo tra il valore appena calcolato e la recv. window.

Tutto ciò non tiene conto della possibilità che ci siano altri client connessi alla rete, che potrebbe quindi stare intasando i buffer. Calcolare la c.w. può essere utile quando la recv. window del destinatario è particolarmente spaziosa. Il tutto viene effettuato tramite protocollo ICMP (ping).

### 3.1.3 Max Transfer Unit

Dipende dal protocollo usato. Nel caso di IP è  $\leq 2^{16}$ , nel caso di Ethernet è  $< 1500$ . Se un datagramma IP venisse frammentato in più parti a causa di Ethernet, l'acknowledge si invia solo alla ricezione di tutte le sue parti. Nel caso anche solo una parte venga persa, l'ack non viene inviato.

TCP divide i propri pacchetti in dimensioni più piccole dell'MTU della rete, in modo da poter ricevere un ack per ogni pacchetto inviato. Il numero di messaggi inviati dal Congestion Avoidance non indica un messaggio, quanto più un datagramma.

# Capitolo 4

## IP

### 4.1 Router

Determinano il percorso da seguire dal mittente al destinatario. Un router si può interfacciare con tanti dispositivi quante interfacce di comunicazione ha.

#### 4.1.1 Architettura Router

Ogni porta (fisica) del router ha un proprio buffer di ricezione ed uno di trasmissione. Ad ogni indirizzo associa una porta di destinazione per sapere dove instradare il pacchetto (tabella di forwarding). Per risparmiare sullo spazio usato per la memorizzazione degli indirizzi, si memorizza solo un sottoinsieme dell'indirizzo. Generalmente si associano tutti i pacchetti la cui destinazione è distante diversi hop ad una singola uscita verso un router con più informazioni. Più vicino è l'arrivo, più bit di informazione il router ha salvato. (Generalmente, possiamo dire che il router salvi i bit di informazione relativi alla rete).

#### 4.1.2 ICMP

Serve ai router per dialogare tra loro e costruire le tabelle di routing. Segnala gli errori tra router ed ha funzioni per il debug della rete. IP e ICMP possono essere considerati allo stesso livello di rete e lavorano quasi insieme: ICMP non può funzionare senza IP e viceversa.

Completa le funzionalità di IP per la segnalazione degli errori e si colloca tra il quarto ed il terzo livello.

#### 4.1.3 Algoritmi di Routing

La precisione dell'instradamento aumenta all'aumentare del numero di bit di prefisso salvati nella tabella

|  |        |   |   |              |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |    |
|--|--------|---|---|--------------|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|----|
|  | 0      | 1 | 2 | 3            | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34    | 35 |
|  | Prefix |   |   | Network Mask |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | #Host |    |

Nel caso dovessi avere due indirizzi salvati, di cui uno è prefisso dell'altro, nel caso venga richiesto un indirizzo con quel prefisso, si controllerà prima se esiste il match con l'indirizzo più preciso per poi scalare, in caso, a quello meno preciso

### Approccio Globale

Ogni router, a ciclo continuo, comunica con tutti gli altri router raggiungibili e notifica le sue connessioni. Dopo un momento di setup, tutta la rete sa le connessioni degli altri, riuscendo a ricostruire un grafo e permettendo, dunque, di utilizzare algoritmi per trovare i percorsi minimi (ex. Dijkstra).

### Approccio Locale

Ogni router comunica ai vicini le sue connessioni. Ad ogni passo, ogni router comunica i vicini a distanza 1, fino al passo N, dove otterremo tutte le informazioni sui router a distanza N + 1, contattando i router che già conosciamo a distanza N. L'algoritmo termina quando otteniamo tutte le informazioni fino alla profondità richiesta N, o termina lo spazio disponibile sulla tabella. Si riesce quindi a costruire una tabella di forwarding in maniera abbastanza rapida.

### Scelta del path

I rami del grafo possono quindi essere pesati, ed i percorsi vengono scelti di conseguenza. Il peso può essere determinato da costo, latenza, distanza ecc. in base ai parametri che si preferiscono.

### Tipi di instradamento

**Link State** Approccio Globale, conveniente in reti piccole ma inutilizzabile su reti di ampia estensione. Generalmente applicato alle reti di host.

**Distance Vector** Approccio Locale, perde precisione sulla distanza in quanto limitato dal numero di hop di cui ha informazioni. Generalmente applicato alle reti di reti.

## 4.2 IP

|                |      |          |    |                                   |                       |    |  |  |  |
|----------------|------|----------|----|-----------------------------------|-----------------------|----|--|--|--|
| 0              | 3    | 7        | 15 | 18                                | 23                    | 31 |  |  |  |
| Version        | HLEN | ToS      |    | Datagram Length ( $\leq 2^{16}$ ) |                       |    |  |  |  |
| Identifier     |      |          |    | Flag                              | Offset frammentazione |    |  |  |  |
| TTL            |      | Protocol |    | Header Checksum                   |                       |    |  |  |  |
| Source IP      |      |          |    |                                   |                       |    |  |  |  |
| Destination IP |      |          |    |                                   |                       |    |  |  |  |
| Options        |      |          |    |                                   |                       |    |  |  |  |
| Payload        |      |          |    |                                   |                       |    |  |  |  |

### 4.2.1 Type of Service

Normalmente, non viene usato, ma dovrebbe servire per identificare il tipo di datagramma.

### 4.2.2 Datagram Length

La dimensione del messaggio è compresa tra 20 e 65535 byte, ma può dipendere anche dai protocolli di livello 2 (ad esempio Ethernet).

### 4.2.3 Identifier e Offset

L'ID serve ad identificare ogni datagramma, per riassemblarli in ordine ci si basa sul campo Offset.

### 4.2.4 Time To Live

Massimo numero di salti del datagramma prima di scadere. Ogni hop decrementa di 1. Nel momento in cui il TTL raggiunge lo 0, viene automaticamente eliminato dal router che lo ha in carico, per evitare che un pacchetto continui a circolare nella rete all'infinito in caso di errori nelle tabelle di routing.

### 4.2.5 Checksum

Verifica di integrità dell'header in CRC16, calcolato ad ogni hop. A livello 2 esiste il CRC32 per controllare l'integrità del livello precedente.

### 4.2.6 Next Level Protocol

Indica il protocollo di trasporto utilizzato nel payload

### 4.2.7 IPv4

Indirizzi a 32 bit, netmask da 32 bit. Le classi predefinite A, B e C usano netmask con i primi 8, 16 e 24 bit a 1. È anche possibile usare netmask personalizzate.

### 4.2.8 Network Address Translation

Permette la conversioni di indirizzi da privati a pubblici al momento dell'instradamento del messaggio. All'indirizzo del mittente, viene sostituito quello del primo router di collegamento alla rete pubblica, che si occuperà di ricordarsi a chi deve restituire poi la risposta andando ad impostare un valore specifico per la porta sorgente.

### Port Forwarding

È possibile specificare manualmente su quale indirizzo (e porta) privato inviare i messaggi ricevuti su una determinata porta. Questo permette di inserire dei server accessibili dalla rete pubblica in una rete privata.

### 4.2.9 IPv6

|                              |    |   |            |                |           |    |
|------------------------------|----|---|------------|----------------|-----------|----|
| 0                            | 3  | 7 | 15         | 18             | 23        | 31 |
| Version                      | TC |   | Flow label |                |           |    |
| Payload Length               |    |   |            | Net. L. Header | Hop Limit |    |
| Source Address (128bit)      |    |   |            |                |           |    |
| Destination Address (128bit) |    |   |            |                |           |    |
| Payload                      |    |   |            |                |           |    |

Header di dimensioni fisse, supporta la possibilità di mandare messaggi di tipo diverso, "marchia" tutti i pacchetti come se fossero nello stesso flusso. Non esiste la frammentazione, che viene delegata al mittente. Se riceve un datagramma troppo lungo, invia un errore in ICMP. Il Next Level Header compensa la mancanza del campo options permettendo di specificare il protocollo di livello superiore, come IPv4, ma lasciando la possibilità di avere più header di livello IP. Non viene effettuato il controllo del checksum, in quanto se ne occupano già i livelli inferiori. IPv6 non è da considerarsi strettamente di livello 3, in quanto permette già di creare una connessione end-to-end tra 2 host. IPv4 ed IPv6 possono essere usati insieme.

### 4.2.10 Tunneling

Si incapsula IPv6 come payload IPv4 per poter transitare in reti senza supporto ad IPv6.

## Capitolo 5

# Data Link

### 5.1 Ethernet

IEEE 802.3 per la versione via cavo, IEEE 802.11 per la versione wireless. Inizialmente utilizzo di un cavo coassiale in banda base (senza modulazione). Un host manda un messaggio sulla rete, ma, essendo tutti gli host collegati senza forma di indirizzamento alcuna, i messaggi dovevano essere inviati in broadcast e tutti potevano potenzialmente essere i ricevitori.

#### 5.1.1 Media Access Control

Di conseguenza, si sono introdotti gli indirizzamenti per dispositivo (Media Access Control), da 6 byte ed univoco per ogni dispositivo, di cui la prima parte identifica il produttore della scheda, e la seconda identifica la scheda stessa. La modalità promiscua permette di ricevere tutti i messaggi che transitano sulla rete, a prescindere dal destinatario.

#### 5.1.2 Carrier Sense Multiple Access/Collision Detection

Tutti gli host rimangono permanentemente in ascolto sul canale. Nel momento in cui si deve trasmettere qualcosa, si attende che il canale si liberi, per poi iniziare la trasmissione. Nel caso due host inizino a trasmettere in contemporanea, si ha un momento in cui, per motivi fisici, nessuno può riconoscere la collisione. Nel momento in cui la collisione viene rilevata, si seguono due step

1. Jamming → Inviamo una sequenza di bit facilmente rilevabile come collisione, per una durata di tempo sufficiente alla propagazione del segnale su tutto il cavo.
2. Stop → Terminiamo la trasmissione, liberando il canale.

Per evitare nuove collisioni, si attende un'unità di tempo casuale in base a quante collisioni ho avuto in precedenza, con backoff esponenziale. Considerando  $n =$

numero di collisioni dall'ultima trasmissione riuscita, possiamo definire l'insieme dei possibili tempi di attesa come un range  $[0, 2^n - 1]$ , ognuno dei quali avente probabilità che venga scelto pari a  $\frac{1}{2^n}$ , fino ad un numero di tentativi massimi pari a 10. In questo modo si può avere la quasi certezza della risoluzione del problema.

## 5.2 Hub

La struttura di rete che si va creare è a stella, ed è come se ci fosse un solo cavo che connette tutti gli host. L'hub funziona da amplificatore di segnali, rendendo il mezzo di trasmissione più affidabile, ed aumentando la velocità di trasmissione grazie al cambio di tecnologia (da cavo coassiale a doppino telefonico, 100Mb/s). Nella NIC è presente un buffer per salvarsi il messaggio nel caso sia necessario ritrasmetterlo.

|        |         |       |
|--------|---------|-------|
| Header | Payload | CRC32 |
|--------|---------|-------|

## 5.3 Switch

Topologicamente simile ad un hub, permette canali full-duplex (invio e ricezione contemporanei, con un doppino per trasmissione ed uno per ricezione). Ha il meccanismo di store & forward, per cui ogni switch salva i messaggi ricevuti su ogni porta in buffer dedicati. Il mittente invia il frame ETH allo switch, che lo memorizza e procede ad inviare il messaggio sulla porta collegata al dispositivo col MAC di destinazione. Le connessioni non sono quindi in broadcast, e non è possibile che avvengano collisioni. Il meccanismo dello store and forward aumenta la latenza.

### 5.3.1 802.3

Lo switch guarda il contenuto del frame e lo inoltra solo ad destinatario

|     |            |                 |         |       |
|-----|------------|-----------------|---------|-------|
| PRE | Source MAC | Destination MAC | Payload | CRC32 |
|-----|------------|-----------------|---------|-------|

Lo switch deve conoscere lo stato della rete e, di conseguenza, i vari mac address dei dispositivi connessi. Si utilizza un algoritmo di autoapprendimento per costruire una tabella che relazioni ogni MAC address alla porta su cui è connesso. Quando uno switch riceve un messaggio su una porta, mappa la porta al MAC del mittente. Mano a mano che gli host inviano messaggi, lo switch ripete l'operazione ottenendo una mappatura completa ed aggiornata. Non impedisce lo sniffing, è possibile effettuare DDOS riempiendo la tabella di MAC falsi.

## 5.4 LAN

### 5.4.1 Address Resolution Protocol

Associa ad un IP un MAC. In LAN l'host chiede in broadcast di chi è l'IP, specificando il proprio MAC. L'unico a rispondere con una ARP reply dovrà essere chi ha l'IP richiesta, inviando il proprio MAC address. ARP usa una cache per salvare il risultato ottenuto e continuare a contattare gli altri host per velocizzare. Essendo stateless, può arrivare una risposta senza sapere a quale domanda è associata

### 5.4.2 ARP cache poisoning

Basta mandare delle ARP reply errate. Non essendoci controlli, non c'è modo di verificare la correttezza della risposta. In questo modo, si può essere soggetti ad attacchi Man In The Middle, con un host che si finge un altro host, andando a compromettere la tabella dello switch e, di conseguenza, tutta la rete. Non si è fatto nulla per risolvere il problema, di conseguenza si cerca, per quanto possibile, di non usare ARP, mantenendo una tabella statica sull'host con le associazioni di IP e MAC. Con i file di configurazione diventa impossibile la gestione di una rete di grosse dimensioni.

### 5.4.3 Dynamic Host Control Protocol

Assegna un IP ad un host connesso. Nel momento in cui si disconnette, l'IP rimane assegnato a quel dispositivo fino a quando non scade o la tabella si riempie. DHCP per funzionare deve avere almeno un server DHCP di riferimento. Nel caso ci siano più server, l'host appena connesso fa una richiesta UDP con IP sorgente 0.0.0.0 in broadcast sulla rete. I server a questo punto ricevono la richiesta e inviano le risposte con un possibile IP da utilizzare, con MAC sorgente quello del server e MAC destinatario quello dell'host. Il nuovo dispositivo riceve quindi le risposte con gli IP e manda un messaggio di accettazione al server da cui ha accettato l'indirizzo, sempre in broadcast, in modo da far sapere anche agli altri server quale indirizzo è stato accettato. Infine, il server accetta l'IP dato all'host, confermandolo.

Una volta assegnato l'IP si può usare normalmente sulla rete. Lo stesso host, in momenti diversi, può avere IP diversi. Con i file di configurazione posso dire al server DHCP di associare in modo statico un MAC ad un IP.