Pila TCP/IP

- **Application Layer**
  - DHCP (Dynamic Host Configuration Protocol)
  - DNS (Domain Name System)
  - HTTP (HyperText Transfer Protocol)
  - SMTP (Simple Mail Transfer Protocol)
  - SNMP (Simple Network Management Protocol
  - Telnet/SSH (Secure Shell)
  - FTP (File Transfer Protocol)
  - URL (Uniform Resource Locator)
  - RTTs (Round-Trip-Times)
  - POP3 (Post Office Protocol Version3)
  - IMAP (Internet Mail Access Protocol)
  - BitTorrent (Architettura P2P)
- **Transport Layer**
  - UDP (User Datagram Protocol)
  - TCP (Transmission Control Protocol)
  - GBN (Go Back N)
  - Selective Repeat
- **Network Layer**
  - IP (Internet Protocol)
- **Link Layer**
  - MAC (Medium Access Control)

Application Layer

FTP File Transfer Protocol
Uno dei più vecchi per trasferimento file.

Telnet/SSH
Telnet non sicuro per accedere a terminali virtuali remoti
ssh versione sicura

HTTP HyperText Transfer Protocol
Collezioni di informazioni mandate tramite questo protocollo come documenti, immagini ecc e' basato su una architettura tipicamente client-server dove abbiamo un client che effettua una richiesta http e il server che esegue la richiesta e invia una risposta http.
http definisce la struttura del messaggio e come client-server devono scambiarsi questo messaggio. Utilizza il protocollo TCP per il livello di trasporto.
Il client inizia una connessione TCP con il server e successivamente comunicano con i rispettivi Socket (interfaccia software). HTTP utilizza TCP per i suoi servizi offerti, il più importante è l'affidabilità, infatti i messaggi arriveranno intatti.
HTTP è molto importante per la sua semplicità in quanto deve gestire numerose richieste al secondo. E' stateless (non mantiene informazioni). Persistent default connection.
Durante le richieste e' possibile avere una pipeline (default nowadays) dove possiamo richiedere più risorse contemporaneamente e risparmiare RTT.
I messaggi sono scritti in testo ASCII.

Request message format:
1. Status line
    a. Il metodo (comando da effettuare)
        i. GET (read)
        ii. POST (read/write)
        iii. HEAD (GET ma senza oggetto di ritorno)
        iv. PUT
        v. DELETE
    b. URL
    c. HTTP version
2. Header Lines
3. Body

Response message format:
- Status Line
    - version
    - status code
        - 100-199 informational
        - 200-299 successful

        - 300-399 redirection
        - 400-499 client error
        - 500-599 server error
    - phrase (risultati)

Cookie sono dei token digitali utilizzati dai server per identificare uno specifico client, alla prima richiesta HTTP da parte di un client, nella risposta HTTP viene aggiunto un set cookie nella header line e successivamente verra' utilizzato nelle richieste HTTP per identificare il client (problematiche di privacy).

Web Cache (Proxy server) sono dei server intermedi che vengono utilizzati come "cache" per poter rispondere alle richieste HTTP piu' velocemente se queste ultime sono state gia' soddisfatte recentemente. Se il Proxy non possiede la risorsa, creera' una connessione TCP con il server avente l'oggetto e la salvera' in memoria successivamente per poter essere velocemente reperibile, di fatto i Proxy sono client e server allo stesso tempo. Essi quindi riducono il traffico e il tempo di risposta per le richieste.
Una problematica del web caching e' l'update, un oggetto puo' diventare obsoleto nella memoria del Proxy e per risolvere questo problema si utilizza il conditional GET per verificare se il file e' stato modificato in un determinato intervallo di tempo.
Il caching e' effettuato anche localmente dai browser per migliorare le prestazioni.

URL Uniform Resource Locator

e' una stringa che identifica una risorsa sul web composta da:

- un protocollo
- user info
- host/ip
- port (optional)
- path
- query
- fragment (identifica un elemento ex. form)

RTTs Round-Trip-Times

E' il tempo che impiega un pacchetto ad effettuare un viaggio client -> server -> client

SMTP Simple Mail Transfer Protocol

E' il protocollo utilizzato per le mail e ci sono due soggetti principali coinvolti: lo **user agent** (outlook, gmail) e il **mail server**. Il protocollo e' utilizzato principalmente dai mail server per scambiarsi mail. I mail server possono essere client side e server side in base al mittente e al destinatario e utilizzano il protocollo TCP per trasferire le mail.

mittente -> mittente mail server -> destinatario mail server -> destinatario

mittente -> mittente mail server (POP3, IMAP, HTTP solitamente)
mittente mail server -> destinatario mail server (SMTP layer applicazione, TCP layer trasporto)
destinatario mail server -> destinatario (POP3, IMAP, HTTP solitamente)

Utilizzare dei mail server rispetto alla connessione diretta tra due client e' piu' affidabile ed efficiente, inoltre i mail server contano sull'affidabilita' del TCP

POP3 Post Office Protocol Version3

Connessione TCP tra il client e il mail server, 3 fasi:

1) **Autorizzazione**
2) **Transazione**
3) **Update**

I messaggi possono essere solo scaricati o eliminati

IMAP Internet Mail Access Protocol

Permette ai server:

- Gestire e creare Cartelle
- Ricerca
- Molteplici client connessi allo stesso server

DNS Domain Name System

Traduce gli Hostname in IP, due attori coinvolti: **DNS Client, DNS Server** (Sistemi UNIX che eseguono il **BIND (Berkeley Internet Name Domain)** software. Utilizzano connessioni UDP. Il DNS e' distribuito e organizzato gerarchicamente per distribuire il traffico, per una migliore manutenibilita', evitare un singolo punto di rottura e migliore raggiungibilita'.
Abbiamo 3 Classi di DNS:

1) **Root**
2) **Top Level Domain** (.org/.net.com.it)
3) **Authoritative DNS** (server gestiti da organizzazioni)

Per ottenere un indirizzo IP un client:

- contatta uno dei root server
- contatta uno dei Top Level Domain server
- contatta uno degli Authoritative server

Il Local DNS Server e' un server gestito solitamente da un ISP e funzionano come proxy per ridurre il traffico sulla rete e migliorare le performance. Il DNS inoltre fornisce un servizio di Aliasing per i server così da poter accorciare i nomi lunghi

Architettura P2P Peer to Peer

I peer si scambiano dati tra di loro direttamente (sono client e server allo stesso tempo). E' piu' complesso da implementare ma scala meglio nell'ambito del File Sharing e ha problematiche di sicurezza

BitTorrent

E' un protocollo per il **File Sharing**. Un **Torrent** e' una collezione di tutti i peers partecipanti alla distribuzione di un particolare file. I peers scaricano una stessa grandezza di **chunks**. Ogni torrent ha un nodo chiamato **tracker** che tiene traccia di tutti i peers partecipanti al torrent. Quando un nuovo Peer entra in un torrent:

- si registra al tracker e lo informa periodicamente del suo status
- il tracker fornisce gli indirizzi IP di un gruppo casuale di Peers dal torrent

Con questo sottogruppo di Peers il nuovo entrato cerca di stabilire delle connessioni TCP per poi iniziare a scaricare i chunks, viene utilizzato il principio del **"Rarest First"** per avere una ridistribuzione rapida dei chunk piu' rari. I peers che stanno condividendo risorse scelgono quali richieste soddisfare seguendo due principi:

1) Dando **priorita'** a chi sta trasferendo dati con il rateo piu' alto
2) Tramite una **selezione casuale** per trovare "migliori vicini" con cui poter condividere le risorse

DHCP Dynamic Host Configuration Protocol

E' il protocollo utilizzato per assegnare gli indirizzi ip dinamicamente e automaticamente all'interno di una rete. E' un client-server protocol, solitamente e' il router a fornirlo.
I passi del DHCP sono 4:

1) **scoperta del DHCP server:** il nuovo host invia in broadcast un DHCP discovery message.
2) **DHCP server offer:** il server risponde con un broadcast message offrendo una possibile configurazione
3) **DHCP request:** l'host rinvia la configurazione
4) **DHCP ACK:** ack finale che conferma la transazione

Possiamo avere un DHCP relay agent che gestisce più sottoreti.

Transport Layer

Il layer di trasporto fornisce la comunicazione logica tra i processi applicativi. Esso converte i messaggi del layer applicativo in pacchetti di trasporto chiamati segmenti o datagrammi.
Le responsabilita' principali dei protocolli di trasporto sono:

1) delivery process to process
2) controllo dell'integrita'
3) affidabilita' del trasferimento dati
4) controllo della congestione e del flusso

L'affidabilità dei trasferimenti e' ancora uno dei problemi principali del networking poiche' bisogna assicurare che i bits non sono corrotti, non sono persi o ripetuti e che vengano consegnati nello stesso modo di come sono stati inviati.

Packet Corruption

Per risolvere il **Packet Corruption** viene utilizzato l'approccio **Stop and Wait** e si basa sui riconoscimenti:

1) **Positive acknowledgments ACK** (messaggio arrivato intatto)
2) **Negative acknowledgments NCK** (messaggio con errori, deve essere rinviato)

Protocolli basati sulla **ritrasmissione** sono chiamati **ARQ (Automatic Repeat reQuest)**.
Cosa succede se il messaggio ACK/NCK e' corrotto? Viene aggiunto un campo nell'header dei pacchetti contenente un numero 0/1 chiamato **Sequence Number**

Packet Loss

Se un pacchetto viene perso gli host non sapranno mai quando mandarne uno nuovo, per questo viene introdotto il **Timeout** da parte del mittente che temporalizza dopo quanto tot bisogna rinviare il messaggio in caso di non risposta e viene utilizzato anche nel ACK loss.
Timeout lunghi rallentano la comunicazione, troppo corti causano overlap dei pacchetti.
Un timeout ragionevole deve essere piu' lungo dell'RTT

Pipelining

Il pipelining e' utilizzato per mandare più pacchetti senza aspettare la conferma. Abbiamo bisogno dei **buffer** per immagazzinare i pacchetti che arrivano per poi poter controllare la loro correttezza. Abbiamo **due protocolli** che utilizzano il pipelining

- **Go-Back-N (GBN)**

- **Selective Repeat**

GBN Go-Back-N

Il mittente puo' mandare molteplici pacchetti senza aspettare la conferma ma non puo' avere piu' di N pacchetti non confermati e abbiamo due elementi:

1) **Base:** sequenza di numeri del pacchetto piu' vecchio non confermato
2) **Nextseqnum:** la piu' piccola sequenza di numeri non usata, la prossima da inviare

Il destinatario scarta tutti i pacchetti fuori ordine, lo svantaggio e' che scarta anche i pacchetti arrivati intatti.

Selective Repeat

Rispetto al GBN non scarta i pacchetti arrivati intatti fuori ordine, ma li immagazzina in un buffer. Utilizza sempre una finestra di N pacchetti non confermati. Un problema dell SR e' che la sequenza dei numeri e' finita, quindi la sequenza dei numeri deve essere almeno il doppio della window size

UDP

UDP fornisce solo i primi 2. Per risolvere il problema dei processi multipli utilizziamo i **socket** come end-point.
Dato che abbiamo molteplici socket abbiamo bisogno di un **identificatore univoco** e un **processo di multiplexing e demultiplexing** da parte di mittente/destinatario.
Demultiplexing: Processo di indirizzamento del segmento verso il giusto socket
Multiplexing: Processo di creazione dei segmenti per processi diversi.
Una porta e' un numero a 16-bit, da 0-1023 sono chiamate porte ben note.

Il socket UDP e' formato solo dall'ip del destinatario e dalla porta di destinazione, mentre quello TCP anche dall'ip del mittente e dalla porta mittente per poter permettere uno scambio continuo di messaggi.
UDP non garantisce l'arrivo dei messaggi ma e' piu' veloce, e' **minimale**. Un esempio di utilizzo e' la richiesta DNS. Utilizzare troppe connessioni UDP per la visualizzazione di streaming puo' creare overflow nella rete.
Il segmento UDP e' composto da 5 elementi:

- **Header**
    - Porta di origine
    - Porta di destinazione
    - Lunghezza
    - Checksum (utilizzato dal destinatario per verificare l'integrita')
- **Data**

Il Checksum e' un numero a 16 bit utilizzato per **l'Error Detection** e funziona come segue.

- Il mittente effettua il complemento ad 1 della somma di tutte le parole a 16 bit nel datagramma e il risultato viene posto nel campo checksum
- Il destinatario controlla se nel checksum sono presenti degli zero dato che si aspetta di trovare 1111111111111111, se trova uno zero allora c'e' un errore.

TCP

Protocollo basato sul **3 Way Handshake** per permettere l'affidabilita'.
Le connessioni possono essere:

- **Persistenti**
    - Risparmiamo gli RTT di creazione connessione
    - Difficile da implementare
- **Non Persistenti**
    - Chiusa ogni volta che la richiesta viene soddisfatta
    - Facile da implementare
    - Consuma piu' risorse ed e' lenta +RTTs

La 3 way handshake viene effettuata in questo modo:

1) client -> server "vorrei iniziare una connessione"
2) server -> client "va bene sono pronto" (1 RTT)
3) client -> server "va bene fai qualcosa per me"

Nel protocollo HTTP il passo 3 e' combinato con la richiesta client HTTP
Le connessioni TCP sono Full-Duplex e point to point (1 mittente, 1 destinatario), inoltre sono strettamente dipendenti dai buffer. I messaggi lunghi vengono spezzettati in segmenti più piccoli.
Il segmento TCP e' composto da:

- Header Fields
    - Sequence Number per l'affidabilita' del trasferimento dati
    - Acknowledgment Number per l'affidabilita' del trasferimento dati
    - Receive Window numero di bytes che il destinatario si aspetta di accettare
    - Header Length
    - Option Field
    - Flags
    - Checksum
    - Urgent data pointer field
- Data Field

Il Sequence Number e il ACK Number sono molto collegari, il Sequence serve anche per gestire la segmentazione e il ACK Number e' calcolato dalla Sequence.
Il Sequence Number per un segmento e' il primo numero della stream di byte.
Quindi per stimare un **tempo ottimale** per un **Timeout** abbiamo bisogno di stimare l'RTT che ovviamente sara' un valore **fluttuabile**, per calcolarlo viene utilizzata la **Media Mobile Ponderata Esponenziale.**

3 Way Handshake

1) Il client manda un **SYN Segment** al server per effettuare una richiesta di connessione.
2) Il server risponde con un **SYN-ACK Segment.**
3) Il client risponde con un **ACK Segment.**

Nella fase 2 e 3 rispettivamente il server e il client allocano i buffer e le variabili TCP.
La chiusura di una connessione TCP avviene sempre con un 3 way handshake

Fast Retransmit

Il periodo di **Timeout** può essere relativamente lungo, per limitare questo problema il TCP applica il meccanismo di **fast retransmit** che utilizza gli **ACK di errore**. Se al Destinatario arrivano dei segmenti con un **sequence number molto maggiore rispetto a quello**

previsto, viene rimandata una copia dell ACK del pacchetto previsto e **dopo N duplicati il mittente rinvia il pacchetto mancante**.

Maximum Segment Size (MSS)

E' il limite dei dati all'interno di un segmento che e' dato da

MSS = **Maximum Transmission Unit** - **Header Size**

MTU: massima lunghezza di un frame accettabile dal link layer
Header Size: tipicamente l'header del tcp + header ip sono 20+20 bytes.

Flow Control

E' un servizio di corrispondenza della velocita' in base ai dati scambiati tra il mittente e il destinatario

Congestion Problem

E' un problema collegato all'**infrastruttura della rete**, quando il link raggiunge la massima capacita' i pacchetti inizieranno ad accumularsi all'interno del **buffer del router**, visto che il buffer del router e' limitato, i pacchetti in eccesso verranno **scartati**.
Ci sono 2 approcci per il controllo della congestione:

1) **End to end Control** vengono osservati i packet loss e il delay
2) **Network assisted Congestion Control**: approccio recente dove il layer di trasporto lavora in sinergia con il network layer. I router forniscono feedback espliciti riguardanti lo stato della congestione della rete.

Il TCP fa affidamento principalmente all **end to end control** e ci sono **3 problemi** da considerare:

1) **Rate Regulation:** come il mittente deve regolare il rateo
2) **Congestion Detection:** come il mittente percepisce la congestione?
3) **Rate Adjustment:** quale algoritmo il mittente deve utilizzare per modificare il suo rateo

Rate Regulation

Viene gestito **aumentando o diminuendo la congestion window**.

Congestion Detection

Viene percepito attraverso i Timeout e gli ACK duplicati.

Rate Adjustment

E' effettuato tramite una sonda della larghezza di banda, Il TCP utilizza il **Jacobsen congestion control algorithm**. Esso ha **3 fasi:**

1) **Slow Start:** parte da 1 RTT e lo incrementa esponenzialmente
2) **Additive Increase:** aumenta linearmente
3) **Fast Recovery (opzionale):** dimezza il rateo della slow start e procede con un incremento additivo

Network Layer

Il ruolo del layer di rete e':

1) All'host mittente quello di prendere i segmenti dal layer di trasporto e incapsularli nei **datagrammi**
2) Al destinatario estrarre i segmenti del layer di trasporto dal **datagramma** e passare il compito al layer di trasporto

Nelle reti ci sono dei nodi che inoltrano i datagrammi ai nodi adiacenti. Quindi il Network layer si occupa principalmente dell **host to host delivery**.

I servizi che devono essere offerti sono:

1) consegna garantita
2) consegna garantita entro tot tempo
3) consegna nello giusto ordine
4) possibilita' di specificare una larghezza di banda minima
5) sicurezza

Ma **nessuno di questi servizi e' garantito dalla rete**. All'atto pratico la rete offre il **best effort service**, ovvero prova del suo meglio a consegnare i pacchetti dal mittente al destinatario.

Il Routing e' meccanismo che decide il path giusto da prendere per un pacchetto.

Edge Routers sono dei router che distribuiscono i dati tra più reti.

Core routers sono router che distribuiscono pacchetti all'interno della stessa rete.

Il Forwarding si riferisco all'azione di trasferire pacchetti dall'input link all'output link del router.

Le forwarding table sono tabelle che specificano quale output un pacchetto deve prendere per arrivare alla destinazione. Il contenuto delle tabelle deve essere determinato **collezionando le informazioni** dagli altri router, questo processo può essere **decentralizzato** o **centralizzato.**

Nel caso centralizzato possiamo avere dei **data center remoti** che potrebbero essere gestiti da ISP, questo e' alla base del **software defined network SDN.**

I **principali componenti** di un router sono:

- Input port
- Switching fabric (collega porte input-output)
- output port
- Routing Processor

Un indirizzo ip e' un numero a 4 Byte e il forwarding segue la **regola del longest prefix matching**, ovvero nella comparazione quello che piu' somiglia viene scelto e viene effettuata dall'hardware per essere il più veloce possibile.

L'azione di trovare l'ip e inoltrare il pacchetto e' chiamato match plus action.

Nella **switching fabric** lo switching puo' essere effettuato in 3 modi:

1) **via memory:** metodo abbastanza lento utilizzato nei primi router
2) **via bus:** una porta alla volta puo' essere servita ed e' utilizzato nelle reti piccole
3) **via interconnection network:** piu' pacchetti mandati in parallelo, tipico dei router moderni

Utilizziamo dei buffer all'interno dei router per immagazzinare i pacchetti dato che la procedura di switching può essere più o meno veloce della ricezione.

Le code nei buffer vengono gestite in 3 modi:

1) first come first served
2) priority queuing basato sull'importanza dei pacchetti

3) round robin

Se non c'e' abbastanza spazio all'interno dei buffer i pacchetti vengono scartati e un pacchetto viene rimosso dalla coda quando viene completamente trasmesso

IP

abbiamo due versioni ipv4 e ipv6, identifica gli host. In un datagramma IP abbiamo:

- version number
- header length
- tipo di servizio
- datagram length
- identifier
- flags e fragmentation offset
- time to live
- upper layer protocol
- header checksum
- ip di mittente e destinazione
- opzioni
- dati

Il primo **problema** dei datagrammi IP è quello della **frammentazione al link layer**. Il router deve dividere il carico di lavoro in due o più piccoli datagrammi (frammenti) che vengono incapsulati in frame del link layer. Quando un router frammenta un datagramma esso copia lo stesso identificatore, indirizzo mittente-destinatario, setta il fragment offset con dei numeri progressivi e pone il flag dell'ultimo frammento a 1, ovviamente dovranno essere riassemblati. Ipv4 riassembla negli end system così da alleggerire il carico sui router.
l'ip e' scritto nella **dotted decimal notation.** Gli IP sono associati alle interfacce che sono il confine tra il link fisico e l'host. Le reti sono gerarchiche, possiamo avere delle sotto reti. L'IP e' diviso in due parti, la parte di sinistra e' riferita a quale nodo e' collegato e la parte di destra identifica la singola interfaccia. Le due porzioni non sono fisse. Ad un ip e' associata una subnet mask che specifica quali bit fanno parte della sottorete. Slashed notation /24.
Gli indirizzi ip devono essere accuratamente assegnati e l'approccio utilizzato e' quello di dividere internet in delle sottoreti.
Problematica delle classi di reti: se ho bisogno di 300 ip devo prendere una classe b e vengono buttati 63mila indirizzi.
L'approccio moderno e' il **Classless InterDomain Routing (CIDR)**
IPv6 ha 3 vantaggi, piu' indirizzi ip disponibili dato che utilizza 128 bit invece di 32, un header di 40 byte e un flow labeling, raggruppamento di pacchetti di particolari applicazioni che vengono gruppato in un unico flow avente un'identificazione specifica.
Cose che mancano dall'IPv4:

- Frammentazione
- Header checksum
- Options

Uno dei grandi problemi di IPv6 e' che non e' compatibile con le versioni precedenti, i sistemi ipv4 non riescono a gestire i datagrammi ipv6.

NAT

E’ un **servizio** che permette il **remap** degli indirizzi IP usando delle tabelle di traduzione associando **ip locali a ip globali della WAN**. Per le reti private utilizziamo degli ip riservati per evitare ambiguità nel routing.

Routing

I router quindi possono effettuare **3 azioni**:

- **inoltrare pacchetti, inoltrare pacchetti modificati e scartare pacchetti**.

Un **buon path** solitamente e’ uno che ha il minimo costo, path piu’ veloce. Possiamo formalizzare una rete come un **grafo avente un insieme di nodi e un insieme di edges**, rappresentati come **coppie di nodi.**

Ci sono due famiglie di algoritmi:

- Distance-Vector
- Link-State

Attualmente sono entrambi usati.

Distance Vector

E’ un approccio decentralizzato che sfrutta la conoscenza locale della rete. Ogni nodo riceve qualche informazione da uno piu’ vicini, effettua calcoli e distribuisce i risultati ai suoi vicini. Questo algoritmo e’ asincrono.

2 fasi

1) Inizializzazione: sono settate le distanze con i vicini.
2) Online phase: le news dai vicini sono ricevute, vengono aggiornati i distance vector e vengono comunicate le modifiche ai vicini.

Pro

E’ **asincrono**

Contro

Convergenza lenta: gli aggiornamenti si diffondono lentamente.

Count to infinity: Il problema del "count to infinity" si verifica quando un router riceve informazioni obsolete su un **percorso** che ora è diventato **inaccessibile**, ma a causa del **ritardo nella propagazione** delle nuove informazioni, il router continua a utilizzare il percorso obsoleto e inizia a incrementare il numero di "**hop**" (salti) nel vettore di distanza. In una situazione estrema, questo conteggio può aumentare fino all'infinito, da cui deriva il nome del problema.

Link State

E’ un algoritmo **centralizzato** e sfrutta la **completa conoscenza della rete** per trovare il miglior percorso. E’ basato sul dijkstra.

Per raggiungere questo livello di conoscenza bisogna inviare dei **pacchetti link-state** da ogni nodo per far conoscere i costi dei link vicini.

La versione base computa il path piu’ corto da un nodo di partenza per tutte le possibili destinazioni.

2 fasi.

1) **Inizializzazione**: Tutti i costi sono conosciuti, settiamo le distanze con i vicini\
2) **Construction Phase**: seleziona il nodo più vicino, esplora il vicino controllando se il path e’ migliore del corrente, esce quando tutti i nodi sono stati esplorati.

Pro

- **Convergenza veloce**
- **No count to infinity**

Contro

- **Sincrono**, i nodi devono ricevere informazioni dall'intera rete prima dello start.

Link Layer

I link e il physical layers forniscono la comunicazione tra due host connessi:

- **Link layer**: trasmissione dei pacchetti sul link.
- **Physical layer**: regola le strutture dei link.

I datagrammi IP vengono incapsulati in link-layer frames.
4 servizi offerti:

1) Framing: datagrammi IP in Link Layer Frames
2) Link Access: regolare l'accesso al link attraverso il MAC protocol
3) Consegna affidabile
4) Individuazione di errori e correzione

Il protocollo Link Layer dipende dalla tecnologia del link fisico ad esempio Ethernet, WiFi, Bluetooth.
Ci sono **2 tipi di link**:

- **Point to point link** (using point to point protocol): esempio collegamento ethernet tra due pc.
- **Broadcast link** esempio rete wifi.

Il problema principale dei broadcast sono le collisioni, i frames possono sovrapporsi e diventare incomprensibili.

Bit level error Detection and Correction

Parity Check: e' la forma piu' facile di riconoscimento degli errori, aggiungiamo un bit di parita' aggiuntivo che indica se il numero di 1 e' pari o dispari, il destinatario deve contare solo il numero di 1 nel messaggio ricevuto per controllare se e' arrivato intatto.
Two Dimensional Parity: tecnica che divide il messaggio in n righe e m colonne e ognuna di essa ha un parity bit.
Cyclic Redundancy Check (CRC): e' un metodo di error detection molto usato, mittente-destinatario si accordano su un pattern di bit chiamato **generatore**, se il **messaggio + CRC** bit aggiunti dal mittente sono **modulo 2 divisibili per il generatore** e danno **resto zero OK** altrimenti **ERRORE**

MAC

L'indirizzo MAC e' utilizzato per identificare le NICs (Network Interface Cards).
E' un indirizzo composto da 6 bytes ed e' rappresentato con una notazione esadecimale.
I MAC sono locali. I MAC sono inseriti nel frame per verificare se MAC inserito e MAC destinatario sono uguali, se si viene accettato. Esiste un MAC di Broadcast tutte F. Il ruolo del MAC e' filtrare pacchetti non desiderati.

Switches

Gli switch sono l'**equivalente dei router nel link layer**, utilizzano **solo MAC** e inoltrano i pacchetti nei link di output, hanno delle **forwarding tables** che associano MAC alle **interfacce**.

Differentemente dai router, **gli switch sono più veloc**i e plug and play. **I loop sono difficili da evitare.**

ARP (Address Resolution Protocol)

Traduce gli indirizzi **IP** nei **MAC**, ogni **interfaccia** ha un **modulo ARP** avente una **tabella ARP** che associa gli IP della lan ai MAC address. Nel caso in cui un IP non sia locale, il **router che collega le due reti ha una tabella ARP** e farà riferimento a quella.

Multiple Access Protocol (MAP)

E' utilizzato per regolarizzare le trasmissioni broadcast, le collisioni sono gestite e i nodi non monopolizzano i link.

I **3 principali approcci** per gestire le collisioni sono:

- **Partizionamento dei canali**
- **Accesso Random**
- **Prendendo i Turni**

Un MAP deve fornire queste **caratteristiche**:

- **Massimizzare l'utilizzo del canale**
- **Essere decentralizzato**
- **Semplice e leggero**

Time Division Multiple Access (TDMA)

Divide il tempo in time frames, questi time frames sono divisi in n time slot. Ogni slot e' assegnato ad uno degli N nodi. Le grandezze degli slot sono scelte in modo tale da poter trasferire un intero pacchetto durante uno slot time

Frequency Division Multiple Access (FDMA)

Divide il canale in **diverse frequenze** e assegna ogni frequenza ad uno degli N nodi.

FDMA e TDMA

Pro

- Evitano collisioni.

Contro

- Limitano la larghezza di banda a R/N.

Code Division Multiple Access (CDMA)

Assegna un **codice differente ad ogni nodo** che viene utilizzato per codificare e decodificare i bit di dati. Se i codici sono scelti con **accuratezza** le reti CDMA permettono di **trasmettere simultaneamente** senza causare interferenze. E' Principalmente utilizzato nei canali **wireless**.

Random Access Protocols

I nodi trasmettono sempre al **full rate**, se trovano una collisione aspetteranno un **delay random** per provare a rinviarli, e' meglio se utilizzano un delay differente per evitare di iterare il processo.

Slotted ALOHA

Il protocollo funziona come segue:

- Il tempo e' diviso in **slot**
- I nodi sono **sincronizzati**, quindi ogni nodo trasmette i frame **all'inizio** di uno slot
- Se nessuna collisione e' trovata la comunicazione continua.
- Se viene trovata una collisione, ogni nodo che sta collidendo ha una **probabilita' di rimandare il frame**.

Funzionalita' offerte:

- Se e' presente **un solo nodo** esso userà il **full rate**.
- E' **decentralizzato**.
- **Facile da Implementare.**
- E' **improbabile** avere **collisioni consecutive**.

Carrier Sense Multiple Access (CSMA)

Una **debolezza** di ALOHA e' che noi iniziamo una trasmissione anche se il canale e' gia' **occupato**, una soluzione e' **monitorare il canale.**

Il CSMA o il CSMA/CD (collision detection) e' basato su due principi:

- **Carrier Sensing**: i nodi ascoltano il canale **prima di trasmettere.**
- **Collision Detection**: i nodi ascoltano il canale **mentre sta trasmettendo.**

Ma le collisioni avvengono lo stesso a causa del **ritardo del segnale di trasmissione**.

Il funzionamento del **CSMA** e' semplice:

- Controlla se il canale e' **occupato.**
- Se il canale e' libero, invia il frame.

Il funzionamento del **CSMA/CD** e' piu' evoluto e usato nell'**Ethernet**:

- Controlla se il canale e' occupato.
- Se il canale e' **libero**, invia il frame.
- Mentre trasmette, **controlla per possibili collisioni.**
- Se viene trovata una collisione, ferma la trasmissione e aspetta un **periodo random per ritrasmettere.**

Polling Protocol

C'è un **master node** che seleziona in **round robin** un nodo alla volta.

- Non ci sono collisioni
- C'e' un **delay di polling**
- E' **centralizzato**

Token Passing Protocol

Non c'e' un master node, I nodi si scambiano un frame speciale chiamato token, chi lo ha puo' trasmettere.

- Non ci sono collisioni.
- E' **decentralizzato**.
- Ci sono problemi se qualche nodo **dimentica di rilasciare il token**.

Programmazione di Reti

2 tipi di protocolli utilizzati:

1) **Standard**
2) **Proprietary** (custom)

I Socket sono l'elemento centrale e gestiscono il layer di trasporto (TCP/UDP) e quello di rete (IP).
Vengono utilizzati degli **APIs** base in quasi tutti i linguaggi di programmazione. Nel dominio UNIX viene usato il **Berkeley socket**.
In C vengono utilizzate delle **strutture** che definiscono gli indirizzi e le porte dei mittenti/destinatari.

- Socket() //utilizzata per creare socket
- Bind() //utilizzata per associare indirizzo locale e porta al socket specialmente sui server
- Sendto(), Recvfrom() //utilizzate per mandare e ricevere messaggi UDP
- Close() //utilizzata per chiudere un socket

Per le **connessioni TCP** vengono utilizzati due socket dal server

1) socket di benvenuto //sempre on
2) client-specific socket //creato una volta stabilita la connessione

- Connect() //utilizzato per creare una connessione TCP
- Listen(), Accept() //utilizzate dal server per gestire le richieste di connessioni
- Send(), Read() //per mandare o ricevere messaggi in una connessione TCP

Il Berkeley Socket utilizza solo gli indirizzi IP quindi dobbiamo utilizzare il protocollo DNS per tradurre gli Hostname

- Gethostbyname() //per contattare il DNS server e ritorna ip, alias ecc
- Gethostbyaddr() //ritorna una struttura hostent

Risposte orale

1. .
    1.1. HTTP: protocollo situato al layer applicativo, stateless poiche' il server non ricorda le risorse che invia, funziona su una struttura client-server dove il client fa richiesta di una risorsa al server e il server la invia, utilizza il tcp protocol come trasporto, quindi l'affidabilita' e' garantita.
    1.2. TCP SYN ACK bit: durante il 3-way handshake il primo pacchetto inviato dal client setta il SYN bit a 1, nel secondo lo fa il server e nel terzo il client lo setta a 0 settando il ACK number come successivo.
    1.3. HTTP stateless: perche' il server non ricorda l'invio di una risorsa, se viene richiesta una stessa risorsa la mandera' come se fosse nuova.
    1.4. PROXY: sono dei server particolari che agiscono come se fossero una memoria cache, immagazzinano le risorse richieste cosi nel caso in cui dovessero essere richieste verranno subito ritrasmesse, nel caso in cui passi tanto tempo nel pacchetto di richiesta puo' essere inserito un if modified since, se si procedera' alla richiesta della risorsa al server originale.
    1.5. Servizi di Trasporto:
2. .
    2.1. Architettura Client-Server: struttura in cui c'e' un server che fornisce risorse a molteplici client, esempi: mailing, http, dns, dhcp.
    2.2. Tabelle NAT: sono delle tabelle gestite dal router che tengono traccia della sostituzione dell'ip locale con l'ip pubblico di una rete all'interno di un pacchetto quando viene mandato verso l'esterno.
    2.3. Indirizzi IP: sono dei numeri a 32 bit che identificano un host nella rete, possono esserci due versioni di ip, Ipv4 e ipv6, sono divisi in una parte che identifica la rete e una che identifica l'host, per capire quali bit si riferiscono alla rete si utilizza una subnet mask, inoltre e' usata per poter creare delle sottoreti e ottimizzare la distribuzione degli indirizzi ip dato che sono limitati.
3. .
    3.1. Server DNS: e' un server che fornisce la traduzione degli hostname in ip, viene utilizzato il BIND software per la traduzione. E' una truttura gerarchica in cui si fa riferimento prima al livello piu' alto di dns e poi si scende root->top level domain-> authoritative
    3.2. Algoritmo Jacobson per controllo della congestione: e' utilizzato dal protocollo TCP per il controllo della congestione, esso regola il rate a cui vengono trasferite le informazioni per evitare la congestione, ha uno slow start in cui viene aumentato il rateo esponenzialmente, se si verifica packet loss o rallentamenti del trasferimento viene dimezzato il rateo e itera questo processo finche' non si stabilizza
    3.3. UDP: e' un protocollo utilizzato per il livello di trasporto, non garantisce l'affidabilita' ed e' molto minimale, a differenza del tcp non ha un controllo della congestione ma e' piu' veloce e leggero, viene utilizzato nelle richieste dns, mailing streaming e video conferenze, e' connectionless, performa solo il process to process delivery e l'error checking tramite il campo checksum nei pacchetti. Il checksum e' il campo utilizzato per il controllo dell'integrita', si effettua il complemento ad 1 della somma di tutte le parole nel datagramma e

viene messo nel campo checksum, il ricevente deve ricevere tutti 1 in quel campo, se trova qualche zero allora si e' verificato un errore.

4. .
    - 4.1. Sicurezza della rete: e' il campo che studia i possibili attacchi all'interno di una rete e i modi per prevenirli. Un malware e' un programma maligno che puo' essere trasferito in un computer attraverso la rete, possono essere di vari tipi: adware, scareware, ransomware, spyware, rootkits, zombie.Ci possono essere dei virus che entrano nei pc senza l'interazione umana e sono chiamati worm. DOS deny of service rendono inutilizzabili rete, infrastrutture e host. packet sniffing ruba i pacchetti magari contenenti informazioni sensibili, ip spoofing ci sta un intermediario che si finge il ricevente, per prevenirlo si usa il checking dell'integrita' dei messaggi e end point authentication. Crittografia simmetrica problema, la chiave deve essere trasmessa, in asimmetrica abbiamo due chiavi una pubblica e una privata fornita da un CA certification authority. Per l'integrita' vengono usate le funzioni hash. SSL versione avanzata del tcp che fornisce i servizi di sicurezza così diventa HTTPS. SSL come tcp ma dopo il 3 way handshake viene mandata la chiave pubblica per criptare
    - 4.2. TCP handshake
5. .
    - 5.1. MAIL protocol: protocollo a livello applicativo, client-server, smtp viene usato tra i mail server per scambiarsi le mail. pop3, imap e http per comunicazioni tra il client e il mail server.
    - 5.2. perche' si usa tcp in http: per garantire la sicurezza della consegna, abbiamo un controllo della congestione, possiamo avere delle connessioni persistenti
    - 5.3. Creazione app con richieste tcp: viene creato il socket nel client, viene bindato per settare ip e porta, richiesta di tcp al welcoming socket del server, dopo l'handshake viene creato un socket specifico per il client cosi possono comunicare.
    - 5.4. Differenza tra richieste udp e tcp: in tcp abbiamo delle connessioni 1 a 1 quindi non e' possibile avere piu' socket che comunicano sulla stessa coppia ip-porte locali, in udp non ci sta il concetto di connessione quindi possiamo avere piu' socket che comunicano sulla stessa porta.
    - 5.5. Porte per la comunicazione quante e quali: sono gli identificatori dei socket, e' un numero a 16 bit quindi 65mila porte possibili, da 0 a 1023 sono chiamate porte ben note, esempi:
        - 5.5.1. 20 ftp
        - 5.5.2. 22 ssh
        - 5.5.3. 25 smtp
        - 5.5.4. 53 dns
        - 5.5.5. 80 http
        - 5.5.6. 110 pop3
        - 5.5.7. 443 https
6. .
    - 6.1. Architettura REST: insieme di principi architetturali che guidano i programmatori a definire web based application modulari, scalabili e flessibili, gli elementi rest sono definiti dalla risorsa e dal URI, forma generalizzata dell'url.

6.2. Composizione dell'url: uniform resource locator e' formato dal protocollo, user info, host, port, path, query e fragment che identifica un elemento della risorsa
6.3. Uso del dns per convertire hostname in IP: richiesta al dns server root poi tdl poi autoritative dove esegue BIND software per tradurre.
6.4. DHCP con utilizzo di udp: dhcp e' utilizzato per assegnare dinamicamente un indirizzo ip ad un client, utilizza udp come trasporto, client fa richiesta broadcast per scoprire il server dhcp, il server dhcp risponde con un offerta, client effettua richiesta dhcp e il server risponde con un ACK.
6.5. Router cos'e' e a cosa serve: sono dei dispositivi che inoltrano i datagrammi ai nodi adiacenti e performano la consegna host to host
6.6. Tabelle di routing locali: sono delle tabelle che specificano come instradare il pacchetto in una rete locale