Appunti di reti di calcolatori e programmazione di reti

Prime lezioni

Nozioni generali

Possiamo iniziare con una semplice definizione, ovvero cos'è l'**internet**? Bene, per rispondere a questa domanda bisonga prima avere dei concetti pregressi prima di poter rispondere concretamente a questa domanda.
Innanzitutto, partiamo dall'esterno con i dispositivi, ovvere la parte **finale** della nostra rete, loro infatti sono i responsabili dell'interfaciamento con gli utenti e per questo sono agli estremi e vengono chiamati **Internet's edge**. Questi dispositivi si interfacciano con l'esterno grazie a degli **switches** nel caso di connesione cablata oppure tramite degli **Access Point** nel caso della connessione wireless. In generale tutti i dispositivi si interfacciano a uno switch in una rete (domestica ed aziendale). A loro volta, gli switch si interfacciano con i **router** che si interfacciano con la rete centrale detta **ISP** (internet service provider, ma questo lo vedremo dopo). Tutto questo permette di creare dei **network**, ovvero un insieme di tutti questi apparecchi elencati sopra che lavarono per ricevere e spedire ***dati e informazioni***, questo è l'internet.

Altre nozioni

L'internet però è governato da delle regole rigide che permettono lo spostamento dei dati in tutto il mondo grazie alle infrastrutture interconnesse di ISP. Tali regole sono dettate dai **protocolli** i quali sono delle vere e proprie regole che definiscono il formato dei messaggi inviati e ricevuti tra le entità di un network.

Mezzi trasmissivi

Per inviare i dati utilizziamo diversi mezzi trasmissivi come quelli **guidati e non guidati** oppure con i **cavi incrociati** o twisted pair. I più usati rimangono ancora il cavo coassiale e la fibra ottica:

- Il primo lavora con dei conduttori in rame con un rivestimento molto spesso per evitare disturbi;
- Il secondo è una fibra di vetro che trasmette fasci di luce ad altissima velocità. Ha però bisogno di ripetitori per poter riacquisire la sua forza durante il tragitto.

Network Edge

Con **Network Edge** è già stato spiegato che tutti i dispositivi agli estremi di un network sono chiamati *host* (sia client che server). Questi si collegano tramite una connessione wireless o cablata a network

esterni che poi si andranno ad interfacciare al **Network core**, ovvero un grafo di routers conenssi tra di loro che possono collegare più ISP tra di loro tramite collegamenti chiamati **IXP** (Internet exchange point) oppure tra router come il collegamento di **peering** (o peer to peer che in italiano vuol dire tra simili).

Accesso alle reti

Gli accessi alla rete sono molteplici e, con il passare del tempo, sono cambiati sia nel modo trasmissivo che nei mezzi veri e propri. La **DSL** è stata una delle prime reti fruibili a tutti nelle case, questo perchè sfrutta i doppini telefonici già presenti nelle case. In linea di massima questo è stato possibile con la divisione del canale tramite uno *splitter* così da poter far passare sulla stessa linea sua la rete che la voce grazie a una divisione in frequenza.

Come avviene l'invio

Quando un'host deve inviare delle richieste il procedimento non risulta così semplice come pensiamo, infatti il nostro messaggio viene suddiviso in **pacchetti**. Ogni pacchetto ha una dimensione fissa e vengono instradati tramite un **link** (mezzo trasmissivo come una rete cablata) fino allo switch. Ogni rete ha una capacità trasmissiva che viene misurata generalmente con bits/sec. Una formula permette di calcolare il ritardo totale dell'instradamento dei pacchetti con la formula **L/R** dove L è la lunghezza in bit dei pacchetti e R è la capacità trasmissiva

---

Forwarding e Routing

Nell'instradamento dei pacchetti troviamo il **Forwarding e Routing** come funzioni necessarie per la ricerca di un giusto percorso dei pacchetti inviati. Con **Routing** definiamo un instradamento generale, ovvero il network sceglie il percorso più veloce ed efficiente tra un router e l'altro. Il **Forwarding** invece è quel sistema che permette la gestione in particolare dell'instradamento, è come se decidesse su quale router è meglio "svoltare" nel suo tragitto che, ricordiamo sempre dura centesimi di secondi.

L'instradamento

Nella fase di instradamento troviamo dei ritardi dovuti a diverse caratteristiche della comunicazione stessa, ecco qui un elenco:

- Durante l'instradamento il delay non è dato solo come quello totale citato prima, ma ne esistono di diversi tipi e forme. In generale 3 dei 4 tipi di delay nascono dal router. Il primo detto **store and forward (trasnmission)** dove i pacchetti vengono inviati dall'host al router fino al completamento totale del singolo pacchetto. Questo avviene per tutto il processo di instradamento dei pacchetti.
- Oltre a questo tipo di delay si pone il problema della fila o **queue** dove i pacchetti provenienti dagli host hanno una velocità di trasmissione dal link precedente molto maggiore di quello successivo. Questo forma nello switch un ritardo interno (**bottle neck**) dove una fila di pacchetti si ferma e deveono aspettare che quelli davanti siano sviluppati (questa coda è anche chiamata FIFO, ovvero first come first served).

Se la coda dovesse essere troppo lunga i pacchetti arrivati per ultimi vengono persi e rispoditi successivamente. Inoltre, una legge permette di calcolare l'intensità del traffico moltiplicando la lunghezza del pacchetto per la quantità media di pacchetti in arrivo fratto la capacità trasmissiva della rete. Se il risultato dovvesse essere maggiore di 1 allora si ha il bottleneck, altrimenti il delay può variare da molto piccolo con valore intorno allo zero fino a un delay medio con il vavlore che si approssima ad 1. Tutto questo venne teorizzato e poi sviluppato da **Kleinrock**

- Il tipo di delay principale è quello di **propagazione (propagation)**, ovvero quello che divide il nostro router a quello del destinatario. Questo delay è quello principale perchè nel caso di un trasferimento di dati trans-oceanico avremo molto delay, si parla di almeno 200ms. Ma nel caso la distanza sia molto piccola il delay creato da questa caratteristica è quasi inesistente.
- Un delay che esiste ma in linea di massima non influenza veramente la trasmissione è quello detto **nodal processing**, ovvere quel tempo di trasposto del pacchetto dal nostro host al router.

Throughput

Con questo termine si indica la capacità trasmissiva della rete misurata in bit/sec . Questa definizione viene spesso accompagnata da quella di bottleneck già vista in precedenza dove la quantità di dati in arrivo da un primo link ha un Throughput maggiore del link in uscita. Questo provoca ritardi e, nel peggiore dei casi perdita di pacchetti

Commutazione di circuito

La commutazione di circuito è un'alternativa a quella di pacchetto dove la comunicazione avviene sulla rete telefonica tradizionale. Questa rete viene divisa tra gli utenti che la utilizzano così da rendere un singolo canale valido per più utenti contemporaneamente. La divisione può essere **a divisione di frequenza** oppure a **divisione di tempo**. La differenza principale tra le due commutazioni sta nella divisione del canale trasmissivo, questo perchè nonostante quella di circuito abbia una migliore capacità trasmissiva non può essere usata contemporaneamente da molti utenti. Invece, quella di pacchetto permette a molti utenti di inviare informazioni grazie a una tecnica dove il canale si mette in **idle (riposo)** quando non trasmette. Inoltre, quella di pacchetto trasmette in un modo **bursty**, ovvere con un forte *colpo* e poi smette, questo avviene ad intervalli regolari per permettere a tutti sulla rete di poter inviare dati.

I Network

Come già citato prima, gli ISP sono coloro che formano la rete vera e propria, ma anche loro hanno bisogno di avere delle regole per poter funzionare a dovere. Prima di tutto devo avere una certa **topologia** (forma) che gli permetta di comunicare con il minor numero possibile di **hop** nella rete. Questi ISP sono suddivisi in reti che parlano tra di loro con degli **IXP (Internet Exchange Point)** che sono messi tra una rete di ISP e un'altra. Gli ISP devono essere concentrati perchè su di loro vengono inviati i dati e i servizi del **Content provider network**, ovvere di tutti quei servizi che offrono dei servizi come Netflix o Youtube. Poi questi servizi arrivano a noi utenti tramite delle regional ISP, che non sono altro che delle reti più piccole che poi arrivano a noi clienti finali.

Security

Introduzione

All'inizio di internet la sicurezza non era neanche stata presa in considerazione dai creatori stessi ,questo perchè internet doveva essere una risorsa resa disponibile ai soli governi, scienziati ecc...
Però con il passare del tempo e con l'avanzamento scientifico internet è arrivato nelle case dei cittadini e, quando una cosa diventa comune se ne pagano le consequenza se non viene gestita da delle regole. Internet è uno di questi casi dove i network erano constantemente attaccati nei suoi primi anni di vita.
Naturalmente ci sono diversi tipi di attacchi che un mal intenzionato può fare per provare a rubare informazioni al un utente, ecco un elenco:

- **Packet sniffing** è il nome utilizzato per il tipo di attacco non grave che permette a un malintenzionato di poter vedere il pacchetto durante il traffico dati e leggere delle informazioni al suo interno come la **header e i protocolli utilizzati**, ma anche l'ip del mittente e destinatario.
- La fase successiva è quella dell' **Ip spoofing** dove viene ruabta l'identità e si mandano pacchetti cammuffati nella sequenza di pacchetti del mittente, così da poter inserire codice malevolo.
- Il peggiore è il **Dos (Denial of service) e il DDos (Distributed denial of service)** dove, proprio come ci dice il nome, andiamo a negare un servizio bloccandolo con un overflow di pacchetti che portano a un malfunzionamento globale del sistema per colpa delle troppe richieste. Il DDos funziona allo stesso modo ma con una rete di botnet alle sue spalle, ovvero un'organizzazione ben gestita di attaccanti che inviano contemporaneamente richieste a un serve che alla fine non riesce a reggere il carico.

A tutti questi problemi le soluzioni elaborate sono disponibili con i concetti dei **pilastri della sicurezza**, ovvero:

- **Authentication** grazie all'utilizzo di credenziali univoche per ogni utente connesso alla rete.
- **Confidentiality** grazie a una serie di algoritmi di crittografia come l'MD5 o l'SHA.
- **Integrity check** che permette di firmare digitalmente un documento così che il criterio della paternità.
- **Access restriction** con l'utilizzo di VPN con protocolli di tunneling e incapsulamento dei dati.
- **Firewall** ovvero difese di tipo hardware e software presenti nei dispositivi che dovrebbero formare ina linea difensiva, come dice il nome muro taglia fuoco.

I Protocolli

Definizioni e concetti generali

Come già detto prima, i **protocolli** sono delle vere e proprie regole che definiscono il formato dei messaggi inviati e ricevuti tra le entità di un network. L'organizzazione dei protocolli è di tipo stratificato,

dove ogni protocollo lavora in un ben definita area della comunicazione.

Tipo	Descirzione
Applicazione	Supporta le applicazione network come il protocollo HTML o SMTP
Trasporto	Per il trasferimento dei dati da processo a processo grazie all'incapsulazione
Rete	Invio dei datagrammi incapsulati da sorgente a destinatario e invio dei dati tra due elementi vicini in una rete
Fisico	Livello base dove si lavora con l'hardware

Questa pila è detta **TCP/IP** ed è stata inventata da inforamtici Americani. Noi Europei invece abbiamo creato una pila simile chiamata **ISO/OSI** che però non ha avuta vita lunga dato che aggiungeva 3 layer alla già esistente pila Americana che in realtà sono stati unificati nella fase di trasporto. Questi layer sono **Sessione, Presentazione e Link**.

Livello Applicazione

Client server VS p2p

Architetture **clinet server** formate da un client che richiede un servizio. Il server invece è sempre in attesa di una richiesta per fornire un servizio.
Architetture **peer2peer**, ovvero formato da nodi uguali tra di loro che offrono le stesse possibilità. Abbiamo una ottima scalabilità, ma non sempre raggiungibile perchè non sempre accesi. Il p2p a livello ipotetico è molto più conveniente di un server, questo perchè non ha nessun tipo di complessità nell'infrastruttura.
Piccolo esercizio con un server che deve mandare un file a una rete. Possiamo dire che il tempo necessario per inviare il file è dato dalla banda trasmissiva del server e dalla capacità trasmissiva del client più lento. Il tempo di upload di un file da parte del server è direttamente proporzionale al numero di client presenti sulla rete.
Nella situazione p2p invece, avendo un numero N di host che possono fare da client e server possiamo dire che il tempo impiegato per madare un file è dato dalla capacità di upload del server, dalla capacità di download del più lento e dal numero di file da mandare diviso la capacità totale della rete (trasmissione di tutti i peer). Su carta possiamo dire che il p2p è quello che ha la meglio tramite un grafico.
Protocollo p2p principale ***BitTorrent***, inventato per garantire i file e di non essere identificati come server settando l'upload a 0. I file **.torrent** sono formati da una lista di peer (un file di testo) che indica da dove andare a prendere il file. In linea di massima ogni volta che scarichiamo **deve** esser rimandato agli altri peer, funzionando da client/server. In una rete p2p abbiamo i file divisi in tanti piccoli pezzi *chunks* di lunghezza fissa. Poi ci sono i tracker che controllano i peer partecipanti ad un torrent e l'insieme dei tracker forma il file torrent. Quando un nuovo client entra in una rete inizia con lo scaricare il pezzo meno presente di tutto il file che stai cercando di scaricare. Questo perchè la rete ha una grossa scalabilità e quindi i pc possono cambiare il loro stato in ogni momento. In generale il protocollo permette di mandare e ricevere da ogni host nella rete, ma se si dovesse trovare uno o più host

affidabili allora si possono formare dei gruppi che permette la condivisione in modo più veloce perchè la connessione degli altri host risulta molto più veloce di quella normale.

Streaming

La specificità di questo servizio è quella di avere già in un inventario i video o film richiesti dall'utente. Naturalmente lo spazio di archivizazione non è infinito e per questo dobbiamo capire come funzionano i video. I film sono una sequenza di immagini passate molto velocemente, dove ogni immagine è un insieme di pixel che hanno una grandezza e una palette di colori che ne indica il peso del singolo pixel. Un metodo per la compressione dei film è quella di controllare il frame attuale con il successivo e, mantenere i pixel che sono uguali tra le due in modo da non far pesare troppo l'immagine. L'encoding dei video in streamign può essere fatto tramite **CBR O VBR**. Il CBR ha un bit rate costante, mentre il VBR ha una frequenza variabile. Per lo streaming dobbiamo tenere in conto il problema della connessione con bit rate variabile e con una possibile perdita di pacchetti. Possiamo avere per esempio il **buffering* di un video (chiamato anche *jittering*).

Protocollo *DASH* (dynamic Adaptive Streaming over HTTP) serve per fare lo streaming di video pre registrati. Il server divide il file in chunks e, ogni chunk viene codificato con diverse caratteristiche (risoluzione). Per ogni risoluzione abbiamo un file su server. I *CDN* (Content distribution network) sono delle agenzie che portano i video fino alle case degli utenti. Una volta che sono arrivati tutti i chunk abbiamo la creazione di un *manifest file*, ovvero un file che descrive per ogni singolo chunk associa una serie di url da dove poter andare a prendere il film. Il client una volta che ha ottenuto il manifest file stima la risoluzione massima accettabile per la velocità della banda dell'utente. Questo protocollo è gestito interamente dal client che stima la grandezza del buffer e la risoluzione.

Tutto lo streaming può essere riassunto con tre fasi, ovvero l'encoding, il DASH e il playout buffer. Con encoding definisco la qualità trasmessa dal server, il DASH adatta il filamto al client e il buffer permette una visione decente del video.

Quando noi avviamo netflix da browser, abbiamo come primo approccio verso l'esterno una richiesta da parte del browser stesso al CDN. Una volta ottenuto il manifest il client scegli da quale server andare a prendere il film.

Livello applicativo

Un'applicazione di rete ha la necessità di avere due parti, un client e un server che parlano tra di loro attraverso la rete ai fini di raggiungere l'obbiettivo dell'utente. La parte server è sempre in ascolto per poter soddisfare le richieste dei client che, non sempre sono accesi.
Quando pensiamo allo sviluppo di un'app dobbiamo tenere in contro degli aspetti come: l'**integrità dei dati** che va sempre rispettata; il **tempo di attesa** massimo che l'app può far aspettare all'utente; la **banda** necessaria alla trasmissione e ovviamente la sicurezza.
Un protocollo di livello applicativo è un insieme di regole che definiscono una standardizzazione per la giusta esecuzione e creazione di protocolli. Definiscono tutte le regole dei messaggi, dal metodo di invio alla semantica.

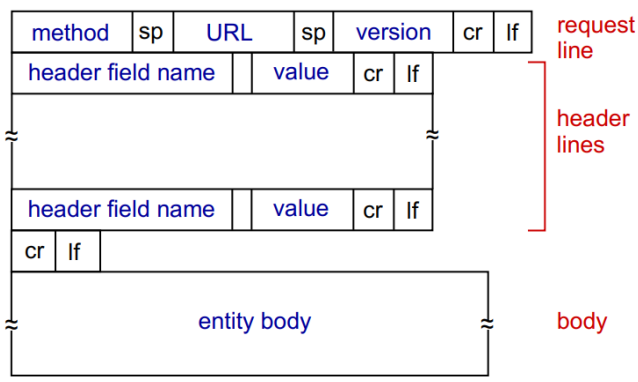
Abbiamo diversi modi di paralre in una rete e, questo può essere in base al tipo di connessione che i dispositivi hanno. Ce ne sono due di principali, il **TCP** e l'**UDP**.
L'*UDP* permette di la perdita di pacchetti a fronte di una maggiore velocità;
Il *TCP* invece, non permette perdita di dati, controllando e rispedendo i pacchetti persi, ma è molto più lento perchè ha molti meccanismi di controllo. Permette di utilizzare un canale non sicuro per inviare dati.
Gran parte della counicazione client server è data dai **socket**, ovvero delle interfaccia situate sul server che permettono la comunicazione e la connessione.

L'insieme di numero di porta e indirizzo ip forma un **processo**. Tale processo è gestito dalla rete.

HTTP

Una pagina web è una serie di oggetti conservati in un disco fisso che sono accessibili dal web. Il protocollo *HTTP* vuol dire *hyper text transfer protocol*. Questi protocollo funziona con il modello client server dove un client richiede al server un servizio tramite il paradigma *request e response*. Una connessione a un server HTTP avviene in pochi step, ovvero abbiamo la creazione della connessione TCP da parte del client sulla porta 80 (porta standard per navigare su internet) che, se accettata dal server HTTP, finalmente il client potrà fare la request della pagina e quindi il server gli manderà la response. Inoltre il protocollo HTTP è **stateless**, quindi non ha memoria di quello che fa l'utente; per questo sono nati i cookie per la profilazione utente.
Abbiamo due tipi di connessioni HTTP, ovvero quella **persistente** e **non persistente**. La non persistente permette la connessione di un solo oggetto alla volta volta, scarica tutta la pagina e chiude la connessione; mentre con la persistente invece la connessione permette l'accesso a più oggetti con una sola connessione TCP e, anche dopo aver scaricato i dati, la connessione rimane aperte tra client e server. Più nel dettagli la differenza tra le due connessione sorge nel momento in cui il client vuole più file di una stessa pagina. Con la non persistente vediamo che abbiamo bisogno di due richiesta (apertura e richiesta file) che hanno una durata di tempo chiamata **RTT** (Round trip time, ovvero tempo necessario inviare un pacchetto e ricevere una risposta) alla quale si aggiunge il tempo necessario a inviare il file (dipende dalla grandezza del file stesso). Questo procedimento avviene per ogni file necessario all'utente, dovendo arrivare ad avere tante connessioni quanti sono i file. Con la persistente invece andiamo ad eliminare tutte le connessione, permettendo al client di fare più richieste.

Le richieste HTTP sono dei messaggi codificati in binario formati da una **request line** nella quale abbiamo la richiesta con il suo metodo (PUT, POST, GET), una *header line* dove sono indicati i valori del mittente e destinatario e poi c'è il body, ovvero il corpo



In seguito alla richiesta abbiamo la risposta da parte del server che, come prima riga, invia una riga di stato. Questa riga di stato è essenzialmente un codice che identifica se la richiesta è andata a buon fine oppure no. Ad esempio con codice di stato 200 sappiamo che la richiesta è avvenuta con successo, ma con altri come 404 o 505 invece la richeista non è andata a buon fine.

Una parte portante dell'HTTP sono i cookie, ovvero una parte del browser che traccia tutti i percorsi dell'utente. Questi cookie sono identificati da un ID che viene generato la prima volta che entriamo su un sito per noi nuovo. Ad ogni richiesta che il client fa al server, questo indirizzo viene inviato insieme nell'header HTTP, continuando a controllare le nostre azioni. Questi cookie però sono i così detti **cookie di terze parti**, che si occupano di seguire noi utenti e di registrare le nostre azioni. Abbiamo anche dei cookie non dannosi che permettono la gestione delle autorizzazioni e sessioni.

Con **web cache** definiamo un server dislocato dal server principale che ha una copia di tutti i servizi disposnibili, così da renderli fruibili anche a client che sono molto fuori dalla rete principale. Però, pensara di fare un server copia di quello originale non ha senso e comporterebbe il doppio dello spreco di dati, per questo le web cache hanno immagazzinati solo i siti più richiesti e, in caso venisse richiesto un file non presente la richiesta andrebbe fatta fino al server di origine. In un'ottica generale le web cache sono molto utili e prestanti, in quanto diminuiscono il traffico dati e diminuiscono il tempo di risposta per i client. Con l'arrivo delle web cache abbiamo avuto bisogno di un nuovo metodo HTTP per evitare che le richieste fossero fatte direttamente al server di origine, per questo è nato il **Conditional GET** che permette di fare un controllo sulla data del file richiesto e, in caso il file presente nella cache ha stessa data di quello presente nel server di origine, allora si predilige la web cache.

Il protoccolo HTTP si è evoluto negli anni, cercando sempre di migliorare la propria velocità ed efficienza. Vediamo come l'**HTTP/1.1**, che è stata la prima evoluzione del protocollo, ha portate il parallellismo delle richieste e l'algoritmo FCFS come tipologia di risposta lato server. Come già visto però, questo algoritmo non performa in caso ci sia una richiesta molto grande. Per questo arriva l'**HTTP/2** che divide le richieste in tanti piccoli pezzi (frame) e, tramite round robin, esegue tutte le richieste senza congestione. Ma in caso ci fosse una perdita tutte le richieste verrebbero fermate in attesa che il TCP rimanda il frame perso. Anche questo problema viene superato con l'**HTTP/3** che, oltre ad avere una gestione della congestione migliore, è anche il primo a garantire la sicurezza in rete (seppur il termine sicurezza e rete non vanno bene nella stessa frase) e a sfuttare un protocollo molto simile all'UDP sviluppato da Google.

SMTP

L'**SMTP** è un protocollo che si occupa delle email e il suo acronimo è **simple mail transfer protocol**. Si basa su tre attori principali, ovvero un **user agent**(client), uno o più server e il protocollo stesso, quindi l'SMTP. Il client, tramite apposite applicazioni come outlook o gmail, può scrivere o leggere email; il server invece gestisce tutte le mail e ne dirige il flusso; il protocollo invece invia email tra client e server e tra i server.
Adesso che conosciamo le basi vediamo come avviene l'invio delle email. Come sempre avremo un client che
effettua la richiesta TCP al server sulla porta 25 (predefinita per la posta elettronica) e quindi avremo

già un *RTT*. Poi abbiamo tre fasi principali deifnite dal protocollo **SMTP RFC**, ovvero il primo dove i due dispositivi ***si conoscono*** tramite l'**SMTP Handshake**, poi possiamo mandare i nostri messaggi nella fase di trasporto e, una volta finita la conversazione avremo la chiusura della connessione. Come tutti sappiamo le email hanno un formato da rispettare, ovvero una header con le inforamzioni del destinatario, mittente e l'oggetto della mail seguite da un corpo. Una volta che noi inviamo una mail questa viene immagazzinata nel mail server e, per poterla leggere è necessario l'utilizzo di più protocolli come l'HTTP e l'**IMAP**. Il primo lo abbiamo già conosciuto, ma il secondo che vuol dire **Internet Mail Access Protocol** permette all'utente di utilizzare un mail **provider** come google o libero per richiedere al server tutte le mail a suo nome. Questo protocollo permette di archiviare ed eliminare mail.

Protocollo DNS

Il protocollo che abilita tutti gli altri servizi, il suo acronimo è *Domain Name System* ed è un database di indirizzi ip e nome simbolico. Questo protocollo mappa gli indirizzi IP dandogli dei nomi. Oltre a questo il protocollo può anche richiedere il mail server o gestire meglio il traffico delle richieste. Il server ha la necessità di scalare per le richieste, perchè deve essere un servizio veloce. Questo database distribuito e gerarchico è basato su un *root server* come radice che ha come *figil* i Top level domain che sono tutti quei server che gestiscono i DNS dei siti .net, .org, .uk .... Poi abbiamo al livello più basso gli Authoritative che gestiscono i domini come .edu.it gestiti da imprese e/o università. Per fare una richeista per accedere a un sito dobbiamo prima interrogare il server root chiedendo il dominio del sito che, nel caso del nostro esempio www.amazon.com, è .com. Una volta ottenuto il .com possiamo interrogare il server e cercare il server dove è presente amazon e poi andiamo a chiedere per la parte dell www, quindi la pagina web.
Quando andiamo a creare un DNS, questo viene registrato tramite un'organizzazione chamata *ICANN* (Internet Corporation for Assigned Names and Numbers).
Quando un host fa una richiesta a un server DNS, manda prima la richeista al server locale e , se non ci fosse risposta, il server locale interroga la root, poi i top level domain fino ad avere un vero indirizzo per interrogare il server finale e ottenere il sito richiesto dal client.
Abbiamo anche il modo *ricorsivo* che permette di far fare le richeiste tra i server. Per esempio in quella precedente abbiamo una centralizzazione sul server DNS locale, mentre nel ricorsivo ogni server chiede e aspetta la risposta dal successivo. Ogni server locale ha una cache che viene cancellata nell'arco di un giorno, per permettere possibiil cambiamenti.

Livello di trasporto

Questo livello, come dice il nome, permette di creare un collegamento fisico tra due interfacce applicative interpellate in un processo. La comunicazione avviene mandando tanti pacchetti che, una volta riassemblati formano il messaggio originale mandato dal mittente nel protocollo applicativo. La trasmissione è sviluppata su due protocolli, l'UDP e il TCP. La differenza principale tra i due, nonostante è già riportata al capitolo precedente, è che il TCP è più sicuro e gestisce il reinvio dei pacchetti persi in modo autonomo, mentre nell'UDP questo deve essere gestito dal programmatore dell'applicazione su cui si mandano i pacchetti, questo perchè ci possono essere perdite totali dei pacchetti. Quando un'applicazione ha la necessità di mandare pacchetti per terminare un processo,

vediamo al divisione dei paccehtti in pezzi più piccoli e poi, tramite il livello di trasporto gli viene assegnato una **header** (testa) nella quale sono presenti delle informazioni utili al destinatario per capire la provenienza come l'indirizzo IP, il protocollo utilizzato(UDP, TCP) e la grandezza. L'UDP invece invia sempre i pacchetti, ma non ha mai una notifica dell'arrivo, quindi possiamo anche inviare paccehtti a un socket inesistente e comunque non avremo errori.

Multiplexing e Demultiplexing

Il **Multiplexing** è quell'operazione che permette a un server di gestire più richieste. Questo nasce dal fatto che un server ha un solo socket, come gli host, ma riesce a gestire contemporaneamente più processi. In generale i server si splittano (dividono) creando una copia per ogni richiesta e, ad ogni processo abbiamo una **header** (intestazione) diversa. Nell'intestazione abbiamo un campo formato da due valori a 16 bit, ovvero la porta di destinazione e di origine del processo. Questo vuol dire che quando un host richiede un servizio lo fa su una porta e, quando il pacchetto deve essere spedito al mittente, viene inviato con la coppia indirizzo IP e numero di porta. Tutto questo ha valenza per i pacchetti mandati con protocollo UDP, perchè così possiamo rintracciare solo i pacchetti.
Per il TCP invece abbiamo bisogno di 4 informazioni nell'header del pacchetto per rintracciare sia il pacchetto che la connessione, e queste informazioni sono la porta e l'indirizzo IP del mittente e del destinatario. Più precisamente, queste informazioni servono nel caso particolare di aprire più tab nello stesso browser che siriferisc nono allo stesso siti web. Per questo avremo che gli indirizzi del mittente e destinatario uguali, con porta del destinatario uguale (80 per web service), ma la porta del mittente cambia ad ogni richiesta. Ogni tab del nostro browser è una connessione TCP e, per essere riconosciuta, dobbiamo avere queste 2 informazioni ben chiare. Quidni finalmente possiamo parlare di **demultiplexing** perchè abbiamo più processi provenienti da uno stesso host con socket differenti.

UDP (User Datagram Protocol)

Protocollo di livello trasporto che letteralmente non offre nessun servizio. In generale è un protocollo di tipo **best effort**, quindi prova a portare a termine un'operazione senza avvisare se effettivamente i pacchetti sono arrivati o meno. È un protocollo di tipo **connectionless**, questo perchè tra i due attori non esiste una connessione vera e proprio, ma si mandano solo dei pacchetti e, alcune volte, possono arrivare in ordine sparso. Questo permette di aprire il socket e , se conosco indirizzo IP e numero di porta del destinatrio, allora posso spedire i pacchetti. Analizzando la header dei pacchetti UDP possiamo trovare come prima due informazioni la porta del mittente e del destinatario, la lunghezza del messaggio, il checksum e il messaggio vero e proprio. Il **checksum** è una parte che non ha molta rilevanza, è un simil controllo che in realtà non può essere mai efficace per colpa del protocollo stesso. Essendo l'UDP non sicuro il checksum che trasporta un numero dato dalla somma dei bit può corrompersi e quindi essere totalmete inutile al controllo finale dei pacchetti. Anche perchè abbiamo detto che l'UDP non è minimamente intenzionato a rimandare il pacchetto perso. Nonostante ciò questo protocollo è molto usato oggi, grazie a delle migliorie nelle performance e al fatto che la perdita dei pacchetti è molto più bassa. La necessità di utilizzare un protocollo di trasporto come questo oggi è molto importate, data la quantità di pacchetti che viaggiano al secondo. Per questo vediamo che è uno dei protocolli principali dell'HTTP/3.

RDT (Reliable data transfer protocol)

Questo è un protocollo molto complesso che permette di rendere sicuro l'arrivo dei pacchetti che passano da un canale di comunicazione non affidabile. Questo protocollo funziona tramite chiamata di API detta **rdt_send** da parte dell'applicazione del mittente (socket), aggiungendo una header particolare per rendere sicuro il trasporto del pacchetto. Quando il pacchetto si trova davanti un canale non sicuro chiama una funzione detta **udt_send** (unreliable data transfer) che permette il passaggio del pacchetto tramite quel tipo di canale e poi dall'altra parte del canale abbiamo la **rdt_recieve** che permette la lettura del pacchetto. Adesso andaimo a descrivere il funzionamento di questo protocollo con dei casi limite, ovvero delle rappresentazioni prendendo delle ipotesi come sempre vere (cosa che in realtà non può esistere):

- **rdt1.0** dove il canale di passaggio è totalmente sicuro (in realtà il protocollo è inutile ma a noi non interessa) il protocollo semplicemente quando viene chiamato dall'applicazione inserisce l'header nei pacchetti e una volta usciti dal canale le header vengono rimosse e i pacchetti sono spediti al mittente.
- **rdt2.0** canale non sicuro al 100% perchè potrebbe perdere dei pacchetti. Con questo possiamo intrudurre la capacità del destinatario di inviare un messaggio di venuta recezione o meno chiamato **acknowledge (ack)** e **negative acknowledge(nack)**. Nel caso di una nack viene inviata una richiesta al mittente di reinvio del paccheto non arrivato. Questa trasmissione si bassa sullo **stop and wait**, ovvero che per ogni pacchetto si aspetta l'ack o la nack e poi uan volta che tutto è arrivato si può passare al pacchetto succesivo, questo però comporta un spreco di tempo enorme e per questo dobbiamo unire al protocollo le **pipeline (rdt 3.0)** per poter parallellizzare l'invio dei pacchetti. Il problema principale di questo protocollo è che può corrompere gli ack o nack, lasciando in un limbo una delle parti in attesa perenne.
- **rdt2.1** è un'evoluaizone del 2.0 ma con la gestione delle perdite dei pacchetti di ack o nack, questo perchè dobbiamo assumere che possono corrompersi nel tragito e non arrivare mai. Per fare questo abbiamo bisogno di un'altra informazione all'interno dei pacchetti inviati dal mittente, ovvero un **codice identificativo**. Questo codice permetterà in caso di perdita delle ack o nack di poter inviare una copia del pacchetto facendo capire al ricevente di quale pacchetto si parla. Essendo il protocollo di tipo **stop and wait** è necessario essere sicuri che il pacchetto sia arrivato alla fine per poter mandare il successivo.
- **rdt2.2** evoluzione del protocollo che alleggerisce il tutto andando a togliere le nack e assumendo che se non arriva una ack in un frangente di tempo allora i pacchetti non sono arrivati. Però, andando ad eliminare la nack abbiamo la necessità di aggiungere al pacchetto di ack il numero di serie del pacchetto ricevuto, così da poter notificare il mittente.
- **rdt3.0** ha una nuova assunzione, ovvero che il canale è molto più vicino a quello reale dove può perdere i pacchetti, corromperli o avere dei ritardi. Per questo necessita di un metodo di gestione della perdita dei pacchetti come il **checksum e un identificativo dei pacchetti**. Inoltre, per gestire la perdita di un pacchetto aggiungiamo un **timer** che, in caso di attesa di una ack per troppo tempo il mittende rimanda il pacchetto con il suo id, assumnedo che non sia arrivato. Per trovare in quanto tempo un pacchetto viene inviato abbiamo una piccola formula, ovvero **L/R / RTT + L/R**, dove L/R è

la genereica formula di lunghezza del messaggio diviso capacità trasmissiva della banda tutto fratto RTT (tempo necessario a un pacchetto per essere spedito e tornare indietro)

- **rdt3.0 con pipelining** permette al protocollo di inviare in successione i pacchetti, assumendo che ci sarà sempre, o quasi, un ritorno da parte del destinatario. Questo è necessario perchè le performance di tutti i protocolli precedenti sono bassissime per poter essere usati in una rete odierna. Con questo protocollo abbiamo l'introduzione di una formula per poter calcolare la velocità di trasmissione dei pacchetti su una banda. Questa formula è: **N*(L/R) / RTT + L/R** dove N è il numero di paccheti mandati in un istante e tutto il resto è uguale alla formula precedente.
  Con questo metodo nascono però altri problemi della gestione della congestione e ritrasmissione dei pacchetti. Abbiamo due famiglie che permettono questo, la prima che non è molto utilizzata si chiama **Go back N**. Questo metodo implementa una finestra di tempo, ovvero in tempo reale prende in considerazione un numero N di pacchetti da spedire. Essendo tanti i pacchetti presenti in una finestra utilizziamo le **cumulative ack**, ovvero delle ack di gruppo, queste vengono inviate al mittente solo quando un pacchetto si è perso. Ad esempio i primi quattro pacchetti sono arrivati, ma il quinto no, allora arriva una ack sul pacchetto 4. Questo vuol dire che anche se il protocollo era in attesa di N pacchetti dopo il 4 rimanderà tutti i paccehtti dal 4 in sù, 4 escluso. La seconda famiglia si chiama **Selective repeat** che, a differenza della precedente, ha due buffer (uno dal mittente e uno dal destinatario) che mantengono i pacchetti con ack. In caso l'ack non fosse arrivata il mittente ritrasmette solo i pacchetti persi, mandando avanti al finestra di tempo.

Modulo 2 Appunti

Mezzi trasmissivi

Per segnale definiamo la forma dell'informazione. Può essere di tipo analogico o digitale. Quello analogico varia in un certo range di valori.

Segnali analogici

Un segnale è caratterizzato da 3 parametri:

- **Ampiezza**, ovevro il picco di un segnale;
- **Frequenza**, quanto velocemente il picco torna a valle e vice versa;
- **Fase** movimento della curva sull'asse delle x;

Più sinusoidi unite tra di loro possono avere come risultato un segnale molto diverso dalle 2 originarie. Questo perchè sicuramente i primi 2 segnali (sinusoidi) erano diverse. L'unità di misura della frequnza è l'**Hertz**.
Sequenze significative si chiamano **larghezza di banda (bandwidth)**. Più è complesso un segnale più sarà grande la lunghezza di banda. Non tutte le frequenze sono trasmesse allo stesso modo, questo comporta un cambio del mezzo trasmissivo per diverse frequnze. Con l'arrivo dell'elettronica negli anni 70'/80' si può applicare la teoria del **sampling**, ovvero del campionamento nel tempo dell'onda sonora prendendo l'ampiezza ogni x secondi. È molto importante avere un criterio di campionamento dell'onda analogica.

Il **Teorema di Shannon** dice che il campionamento di un'onda analogica, affinchè non si perda sengale, deve essere fatto 2 volte la frequenza del sengale stesso. Con il quantizzatore possiamo prendere il singolo campione e digitalizzarlo in un segnale digitale trasformando il valore prima in decimale e poi in binario.
Le caratteristiche che comporta il passaggio al digitale sono:

- L'**integrazione** che permette di avere un solo sistema per la comunicazione dei dati, dato che ora le informazioni sono condivise tramite bit e non più pacchetti come prima che potevano essere più grandi rispetto al canale trasmissivo e quindi venivano troncati.
- La **computazione** è la trasformazione dell'informazione da analogico a digitale tramite i bit che sono processabili con computer.

Trasmissione

La **Multiplazione** è un network che permette la condivisione in un canale di più link. Ci sono 2 tipologie di multiplazione, ovvero quella a divisione di **frequenza** e quella a divisione di **tempo**. Le differenze principali, anche se già spiegate in precedenza, sono principalmente il tipo di sengla che viene trasmesso. Ovvero in quella di frequenza dividiamo il segnale, nella seconda dividiamo i bit.
La multiplazione a **divisione di codice** invece, tramite una elaborazione matematica permete di mandare le informazioni per tutto il tempo dividendole dalle altre che stanno comunicando.
Quando faccio la multiplazione a divisione di tempo dobbiamo definire delle *fasi*, ovvero dobbiamo tener presente degli istanti di tempo, o meglio **slot**, dove dividiamo il tempo in unità fondamentali che permettono la trasmissione solo ad intervallo di tempo predefinito. La versione **unslotted**, senza istanti di tempo, permette la trasmissione senza divisione temporale, dove tutto viene condiviso insieme allo stesso tempo. Nel caso della soluzione slotted, possiamo formare dei *gruppi* o **frame** dove le informazioni vengono organizzate con dei divisori.
Il canale telefonico digitale ha uno standard trasmissivo di 64000 bit al secondo.
L'Europa per trasmettere utilizza lo standard **frame E1**, questo perchè la trasmissione slotted framed permette un controllo accurato di tutta la trasmissione. Questo però porta un limite massimo di trasmissione. Il limite di trasmissione viene superato da un **TDM unslotted (Time divisione multiplexing)** dove ogni blocco di bit viene legato al mittente.
Se la nostra comunicazione ha bisogno di una grandezza ben specifica dobbiamo usare lo slotted framed, mentre se non abbiamo bisogno di una grandezza fissa (come il messaggio di whatsapp o skype) usiamo quello unslotted. In realtà la differenza è che con l'unslotted non siamo interessati al messaggio e a sapere se è arrivato o meno.
Due leggi importanti sull'evoluzione della tecnologia sono quella di **Moore** e quella di **Edholm**.
La prima dice che ogni 18 mesi il numero di transistor per processore raddoppia, e questa è vera sin dal primo processore; e poi quella di Edholm che dice che ogni 18 mesi la banda a disposizione dell'utente raddoppia a costo costante.

Mezzi trasmissivi

Come primo contetto trasmissivo abbiamo l'**attenuazione**, ovvero la misura di degrado del segnale (elettromagnetico), misurando la sua perdita di potenza e si misura in dB/km. Questa cresce in base

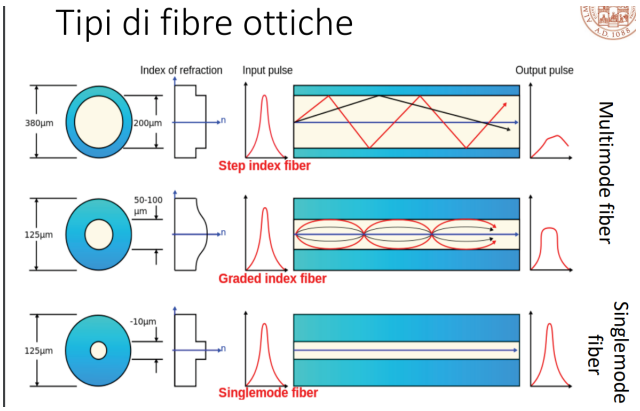
alla distanza e a segnali di alta frequenza.
I mezzi trasmissivi principali sono stati i **doppini telefonici** e cavi di rame. In realtà il doppino è formato da 2 cavi di rame intrecciati. Questo intreccio (chiamato **twisted pair**), se fatto bene, diminuisce l'attenuazione del segnale. Con lo sviluppo i mezzi trasmissivi avevano la necessità di migliorare la loro efficienza e qualità, per questo vediamo come il twisted pair si può dividere in due categorie, ovvero l'**STP (Shielded twisted pair) e UTP (Unshielded twisted pair)**. Con l'STP abbiamo un cavo formato da tante coppie avvolte da un conduttore che fa da schermo. Naturalmente i contro di questo cavo sono il maggior costo, la necessità di essere messo a massa e la grandezza. Con l'UTP invece abbiamo un cavo meno costoso, più semplice da spostare ma non schermato. I cavi vengono divisi in categorie, nello specifico dalla 1 alla 7 e vediamo come ogni categoria ha un bandwidth maggiore e una frequenza maggiore.
Il cavo STP più comune è il **cavo coassiale**. La sua sezione ha forma cilindrica, è formato da due conduttori, uno interno e uno esterno, entrambi con schermatura. Questi cavi possono differire per diamentro e quindi, possono diventare molto costosi e difficili da maneggiare. Il diametro infatti è fattore importante della resistenza all'attenuazione del nostro cavo, più è grande meno attenuazione ci sarà.

Radio comunicazioni

La radio comunicazione permette il broadcast (tramisssione a tutti coloro che hanno la stessa frequenza) e quindi si presenta come un ottimo mezzo diffusivo. Inoltre, non ha vincoli fisici e per questo è adatto per la mobilità. Ma come difetto ha il fatto che presenta un solo spettro radio. Le onde elettromagnetiche prodotte per la diffusione radio viaggiano in linea retta, oppure vengono trasmesse tramite la ionosfera, ovvero rimbalzando su questo strato per poi essere captate da altre antenne. La prima comunicazione radio venne effettuata nel 1895 da Marconi con il ponte radio. Tutti questi servizi radio funzionano tramite antenne e tralicci. Ogni singola antenna ha una propria area di copertura che permette a un certo numero di utenti di potersi collegare alla rete tramite frequenza. Con la venuta delle metropoli si è ideato il sistema **cellulare** dove venivano piazzate tante **celle** per permettere la copertura totale ai clienti. Ogni cella ha una frequenza diversa e, grazie alla loro forma a scacchiera la sovrapposizione delle celle non è un problema perchè hanno frequenze diverse.

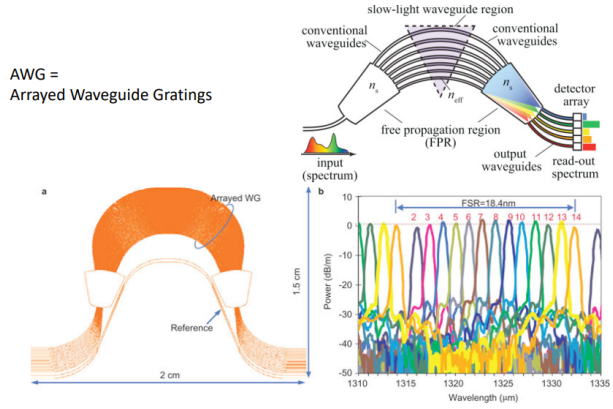
Fibra ottica

La fibra ottica è un filamento di vetro o plastica molto sottile.Facendo passare un fascio di luce in un filamento di vetro con il tempo perderà intensità perchè il vetro assorbe la luce. Per esempio un vetro di una finestra assorbe moltissima luce, questo perchè al suo interno ci sono molte impurità. Questo è dato dal fatto che il vetro è fatto di ossido di silicio, che può contenere impurità. La fibra ottica invece è fatta da vetro molto puro che non permette l'assorbimento della luce. La luce all'interno della fibra viaggia tramite riflessione grazie a un indice di rifrazione del **cladding**(mantello) più grande di quello del core.

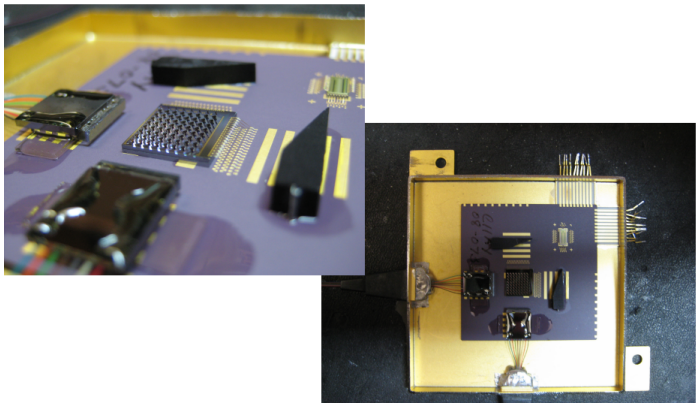


Divisione tra fibre monomodali e multimodali

La trasmissione in una fibra è iniziata da un diodo led o laser e poi letta in uscita da un foto-rilevatore. A livello storico la fibra ha fatto un paradosso nelle comunicazioni, ovvero che è molto più performante rispetto al rame e meno costosa del rame stesso. L'unica criticità delle fibre è la **giuntatura** (collegamento) tra due fibre. Per far si che questo avvenga è necessario un macchinario molto costoso che fonde i due cavi e il core. Con un **amplificatore ottico** possiamo far tornare l'intensità del fascio di luce al suo stadio originale tramite il **drogaggio** di una parte di cavo con il *germanio* che funzionerà da ripetitore.
Con l'evoluzione negli anni 90 si è sviluppata la possibilità di poter cambiare il colore dei laser che fino a poco prima erano solo in luce bianca. Con la presenza di laser *colorati* possiamo dividere la fibra ottica con diversi laser che però, per entrane nella fibra, devono passare per un prisma che permette a tutti i laser colorati di diventare luce binaca, ottenendo una capacità molto più elevata in base al numero di colori dei laser trasmissivi. Naturalmente alla fine della fibra ci sarà un altro prisma che da luce bianca restituisce i laser colorati. Questa divisione viene chiamata **WDM (Wavelength Division Multiplexing)** ed è un tipo di divisione di frequenza.



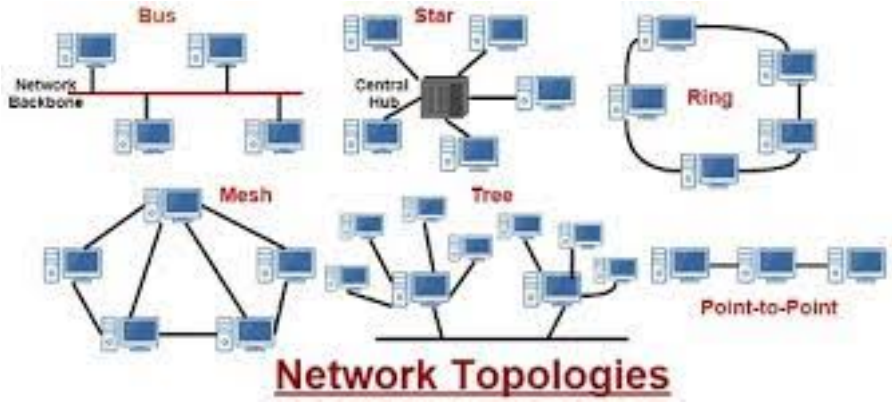
Questa immagine rappresenta una **AWG = Arrayed Waveguided Gratings** e descrive il funzionamento dei prismi in entrata e uscita della fibra ottica. Questa possibilità di spostare i fasci di luce è resa possibile dai **MEM**, ovvero dei piccoli specchi meccanici che, se angolati, permettono di direzionare i fasci di luce. Se invece non si ha più bisongo di uno specifico fascio di luce lo si fa sbattere contro un pezzo di plastica nero, questo perchè il nero assorbe la luce.



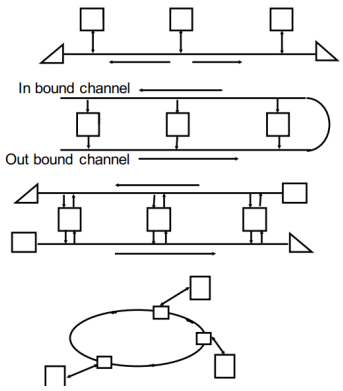
Nome	Descrizione
FTTCab	Fiber to the cabinet, la fibra arriva fno agli armadi e poi arrivano nelle case con i doppini
FTTC	Fiber to the curb, la fibra arriva fino al marciapiede
FTTB	Fiber to the building, dove la fibra arriva fino al
FTTH	Fiber to the home, dove la fibra raggiunge la singola abitazione

Reti LAN

Per **LAN** (Local Area Network) definiamo un tipo di infrastruttura di telecomunicazioni che consente agli host (dispositivi connessi alla rete) di comunicare in un'area *delimitata* attraverso un **canale fisico condiviso** con un tasso di errore basso e con un elevato bit rate (con velocità di trasmissione alta).
Le LAN sono molto utilizzate per piccole reti, dato il loro prezzo basso e conveniente. Inoltre prediligono il rame come mezzo trasmissivo e non la fibra ottica perchè essendo piccole non possono sopportare un prezzo così alto per poter ottenere delle fibre di tipo FTTH. La loro peculiarità però è la topologia (forma), ne possiamo avere di diverso tipo e questo perchè sono adattabili alle esigenze. Le più diffuse sono quelle a stella, a maglia e gerarchica. Quelle peer-to-peer (punto-punto) vengono utilizzate solo se i terminali sono pochi e quindi non serve un nodo di comunicazione.



- Bus bidirezionale
- Bus unidirezionale
- Doppio bus (dual bus)
- Anello



La seconda immagine invece rappresenta delle **topologie** punto-multipunto, queste sono particolari perchè sono in broadcast, ovvero che ogni messaggio inviato da un host viene visto da tutti gli host della rete ma, solo il destinatario effettivo lo può leggere o vedere il suo contenuto, questo grazie a un protocollo. Questi reti sono importanti perchè lavorano con un solo mezzo trasmissivo, il bus e permettono lo scambio di dati a tutti gli host in una LAN. Un problema di queste reti però rimane la collisione, perchè non esiste un mediatore tra host e bus e quindi, seppur piccola, c'è la possibilità che 2 host trasmettano nello stesso istante e quindi i pacchetti alla fine andranno persi.

Ogni scheda di rete è segnata da un indirizzo **MAC**, che sono dei codici univoci per ogni device così da poter essere riconoscibile all'intenro di una rete. Questi indirizzi hanno 48 bit (6 byte) e indicano il produttore della scheda e le schede presenti nel nostro device. La rappresetnazione del mac è in esadecimale e un esempio è: 00-60-b0-78-e8-fd

Cablaggio

Si basa sugli standard **EIA/TIA 568** (standard di mercato) e **ISO 11801**. In generale in un edificio vengono inseriti diversi cavetti con coppie diverse che forniscono la classica presa a muro con il connettore RJ45.
Vengono utilizzati degli **armadi di rete** per permettere la connessione in un palazzo dall'esterno verso l'interno. Con i **patch panel** abbiamo delle prese che permettono alla rete esterna di poter arrivare in un palazzo tramite dei collegamenti laterali all'armadio stesso. Tramite l'**hub** possiamo connettere il patch panel all'hub stesso tramite i **patch cord** (cavi RJ45 UTP) così che la rete possa essere fruibile in uscita.

Wireless LAN

Per la comunicazione Wireless ci basiamo su l'etere come canale tramisisivo e, soprattutto, su alcune frequenze radio specifiche chiamate **ISM** a 2.4 Ghz. Molti sono gli standard per la trasmissione wireless e appartengono tutti alla stessa famiglia, ovvero 802.11 e tutte le sue evoluzioni.
Con **BSS** definiamo l'infrastruttura che si viene a formare quando colleghiamo degli host a un **Access Point** . Quando si vanno a guardare gli indirizzi in una trasmissione di pacchetto vederemo che gli ultimi 2 indirizza sono il destinatario e il mittente per una ragione di controllo e poi, in cima a questi due, avremo tanti indirizzi quanti sono stati gli **Hop** del pacchetto, quindi tra quanti dispositivi intermiedi è passato il pacchetto per arrivare al destinatario. I pacchetti spediti con cavo hanno solo 2 indirizzi.

Appunti di laboratorio

La virtualizzazione

Esistono due tipi di virtualizzazione, di tipo 1 e 2. Semplicemnte quella di tipo 1 permette di far girare sistemi operatvi da *"console"* in bootstrap, mentre quelle di tipo 2 permette di avviare applicazione per la virtualizzazione già da un sistema operativo.

Tabella con comandi che sarà aggiornata con il tempo. Per aprire il cmd in linux **ctrl + alt + t**, per apreire una nuova finestra **ctrl + shift + t**, per bloccare l'esecuzione di un programma **ctrl + c** e per pulire tutta la cli basta fare **ctrl + l** oppure scrivere **clear**. Molto spesso verranno citate le tabelle **arp** che in soldoni sono delle tabelle dove possiamo vedere l'indirizzo ip e il MAC del dispositivo.

Comandi di base

Nome	Descrizione
cd	change directory, quindi cambia la cartella inserendo il path (assoluto o relativo)
cp nomefile	copia il file selezionato e, dopo il nome, scrivere il path
rm nomefile	rimuove il file o cartella
man	concatenato ad un altro comando permette di leggere il manuale del comando
history	permette di vedere tutti i comandi fatti in precedenza
pwd	vedere il percorso in cui ti trovi
ls	permette di vedere la lista dei file e cartelle nella cartella attuale
ping indirizzo IP	manda dei pacchetti all'indirizzo IP segnalato e aspetta una risposta
traceroute indirizzo IP	è un ping con molte più informazioni
alias comando='comando'	permette di mascherare il comando di destra con quello di sinitra. Per una migliore comprensione degli esercizi consiglio di fare alias ip='ip -c'. Per togliere questo alias basta scrivere unalias ip
sudo nome_applicazione &	avvio applicazione in background
sudo chmod +x nomefile	permette di dare l'accesso di eseguire il file. come negli sh

Nota bene l'appendice (o meglio flag) *-c* permette di colorare ed evidenziare le scritte in uscita, quindi è applicabili alla gran parte dei comandi che restituiscono un output.
Per lavorare da shell abbiamo bisogno dei **Namespace**, ovvero degli *oggetti* (anche se il termine è improprio) di tipo astratto creando uno stack di rete che si appoggia sul kernel di sistema isolato da

tutto. Questi namespace, anche se possono essere utilizzati in molti modi, nel nostro caso saranno i dispositivi di rete, ovvero host, router e switches tramite il comando **ip netns**.

Comandi per la rete

Nome	Descrizione
ip	Questo è il comando universale che permette di vedere le tabellle di routing, manipolare dispositivi e interfacce. Funziona da ipconfig
nano nomescript.sh	per aprire uno script e scrivere da cmd
touch	per creare dei file
ip –c neigh	interroghiamo la tabella di routing, neigh vuol dire neighbourhood quindi vicinato
ip -c link	interroga la routing table del nostro dispositivo per vedere le interfacce, questo è più completo rispetto al comando sovrastante. In realtà questo comando è la versione troncata di ip link, questo perchè ubuntu accetta questa sintassi
ip -c a	permette di vedere gli indirizzi ip delle interfacce e l'indirizzo mac del dispositivo
ip -c r	permette di vedere le routing table del dispositivo su cui lo avviamo. Per routing table definiamo le vie percorribili dai pacchetti
ip addr	mostra tutte le interfacce e il relativo indirizzo IP
ip addr add indirizzo/subnet mask dev nome_interfaccia	permette di aggiungere e indirizzi alle interfacce
ip addr del indirizzo/subnet mask dev nome_interfaccia	permette di rimuovere e indirizzi alle interfacce
sudo	**super user do** serve per fare comandi da amministratore (in questi esempi useremo sempre questo comando perchè non abbiamo il ruolo di amministratore profilo ubuntu)
sudo !!	permette di utilizzare il comando precedente aggiungendo sudo senza doverlo ricopiare
ip netns	comando per la gestione dei namespace (dispositivi)
sudo ip netns add nome_host	permette di creare un nuovo host (namespace)
sudo ip netns del nome_host	permette di rimuovere un host già esistente
sudo ip netns exec nome_host comando	esempio pratico che permette di eseguire dei comandi sull'host selezionato tramite il comando exec
sudo ip netns list	Permette di vedere in una lista tutti i dispositivi (namespace) esistenti

Nome	Descrizione
sudo ip -all netns delete	rimuoviamo tutti i namespace creati
ip link	è un comando per la configrazione dei dispositivi, è sempre concatenato con altre *particelle o **flag*** (è la stessa cosa di ip l)
sudo ip link add **nome** type **tipo**	permette di creare le interfacce (porte) inserendo un nome simbolico e un tipo, il nostro tipo predefinito per ora sarà veth ovvero virtual ethernet
sudo ip link add nome1 type veth peer name nome2	esempio reale della creazione di un cavo ethernet con due interfacce (nome1 e nome2 sono le estremità)
sudo ip link set nome_interfaccia	permette di modificare l'interfaccia selezionata grazie all'aggiunta di altre flag
sudo ip link del nome_interfaccia	permette di eliminare un'interfaccia
sudo ip link set nome_interfaccia netns nome_dispositivo	esempio reale che permette di collegare una delle estremità del cavo a un dispositvo
sudo ip link set nome_interfaccia netns bridge	esempio reale che permette di collegare la seconda estremità del cavo allo switch (bridge è un nome simbolico che indica la porta dello switch)
bridge	comando per gestire lindirizzamento degli switch
bridge link	imposta le proprietà delle interfacce ai collegamenti
bridge fbd	gestisce il database dei pacchetti inoltrati dal bridge (switch)
sudo ip link set nome_interfaccia master nome_switch	esempio pratico per collegare la seconda estremità del cavo allo switch
sudo ip link set nome_interfaccia nomaster	esempio pratico per scollegare la seconda estremità del cavo allo switch
sudo ip net exec nome_host ip l add default via indirizzo_ip_gateway	serve per inserire il gateway neglio host ai fini di poter paralre tra più reti
sudo ip net exec nome_gateway sysctl -w net.ipv4.ip_forward=1	permette di attivare il protocollo per l'invio dei pacchetti da un'interfaccia a un'altra

La prima cosa a cui devi pensare per svolgere un esercizio è: **Di cosa ho bisogno?** Naturalmente di host, switch e cavi.

Comandi specifi per l'esercizio

sudo ip addr add 10.0.2.16/24 dev eth0	sifica il loro indirizzo ip ask
sudo ip link add eth0 type dummy	creazione di una nuova rete di tipo dummy
sudo ip l add eth_1 type veth peer name veth1_1	creo una interfaccia ethernet
sudo ip link add LAN type bridge	creazione di una rete
sudo ip link set eth1 master LAN	connetto il mio host(eth1) alla rete LAN
bridge fbd	vedo le connessioni
sudo ip link set eth1 no master LAN	sconnetto l'host dalla rete
sudo ip n	
sudo ip netns exec H1_1 ip addr add 192.168.1.1/24 dev veth1_1	inserisco un indirizzo ip nel mio host