

Una RETE DI CALCOLATORI è un insieme di dispositivi autonomi e interconnessi tra loro da supporti fisici. Vengono utilizzate per fornire un supporto

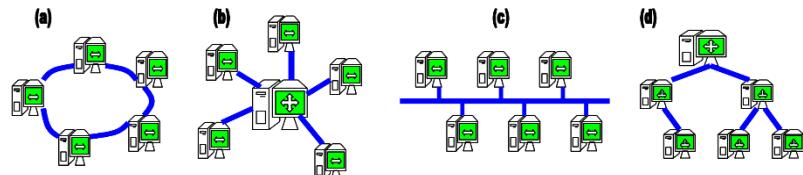
- alla comunicazione tra utenti, ovvero servizi di comunicazione come mail, www, IoT, wireless
- alla comunicazione tra dispositivi, per permettere
  - condivisione di informazioni, ovvero database e pagine web
  - condivisione di dispositivi e di risorse, come stampanti e supporti di memorizzazione
  - accesso a calcolatori remoti
  - calcolo distribuito e sistemi scalabili

#### CLASSIFICAZIONE PER ESTENSIONE GEOGRAFICA

- PAN(Personal Area Network), possono connettere dispositivi in una stanza o su una persona e sono finanziate e gestite dal singolo utente
- LAN(Local Area Network), possono connettere uffici, edifici o campus e sono gestite da organizzazioni, università, enti o aziende
- MAN(Metropolitan Area Network), possono connettere intere aree urbane e sono gestite da provider o gestori di servizi telefonici
- WAN(Wide Area Network), reti geografiche in grado di coprire distanze internazionali o planetarie e gestite da enti nazionali e internazionali o grossi gestori delle comunicazioni
- Internet è la rete globale composta dall'unione di reti di vari tipi, connesse tra loro e conformi a un determinato insieme di regole di comunicazione comuni: i protocolli di Internet.  
Ogni dispositivo connesso a Internet è dotato di un nome logico dato dai servizi DNS

#### CLASSIFICAZIONE PER TOPOLOGIA

- Anello: segnali trasmessi nella sequenza di connessioni in senso orario o antiorario.  
Ha grado di ridondanza pari a uno, se un collegamento si guasta la rete funziona comunque.
- Stella: componente centrale (hub o switch) direttamente connesso a tutti gli altri.  
Single point of failure: se si rompe il componente centrale gli host sono tutti partizionati.
- Bus: trasmissione broadcast attraverso regole di accesso al bus condiviso
- Albero: organizzazione gerarchica delle connessioni
- Grafo complesso o maglia: topografia formata da più strutture diverse collegate tra loro e utilizzata nelle reti più grandi



#### Le PRESTAZIONI DI UNA RETE sono misurate

- dalla capacità di trasmissione, ovvero la quantità di dati che è possibile trasmettere o ricevere in un secondo.  
Il throughput è la capacità effettiva del collegamento destinata alla sola trasmissione dei dati.
- dal ritardo del collegamento, ovvero il tempo richiesto ai dati per transitare dal mittente al destinatario e quindi dato dalla distanza fisica e dai tempi di gestione dovuti ai protocolli di comunicazione.  
Il jitter è la variazione nel tempo del ritardo della rete, se questo parametro è elevato il flusso di dati trasmessi non sarà costante. Il jitter è la misurazione empirica della variazione del ping che misura il round trip time dei dati trasmessi, ovvero il tempo che impiegano i dati trasmessi da un dispositivo ad arrivare a destinazione.

Le COMPONENTI di un host sono:

- dispositivo o scheda di rete: dispositivo hardware per memorizzare temporaneamente i dati, codificare e trasmettere, oppure ricevere e decodificare dati inviati dal calcolatore alla rete, o viceversa. È amministrata dai driver, ovvero componenti software del sistema operativo.  
Il tipo di scheda dipende dalla tecnologia di trasmissione utilizzata e prende il nome dai protocolli o dallo standard utilizzato per la trasmissione, ad esempio:
  - Ethernet, la scheda di rete ascolta il canale mentre trasmette il segnale (collision detection). Se viene rilevata una collisione interrompe la trasmissione e riprova dopo un tempo random.
  - IEEE 802.11 (Wi-Fi), la scheda di rete ascolta il canale radio (carrier sensing) e trasmette solo se nessuno sta già trasmettendo (collision avoidance), ma è diverso da ethernet perché non rimane in ascolto mentre trasmette il segnale.
  - Token ring, è usata se i dispositivi sono collocati su topologia ad anello cablato. Esiste un frame detto token che viene passato come un “testimone” tra i dispositivi, solo chi detiene il token ha diritto di trasmettere, poi, passato il token rotation time, si passa il token al dispositivo successivo. È l'unica tecnologia in grado di garantire la Quality Of Service.

Possiede un indirizzo fisico di livello MAC a 48 bit, ovvero un codice identificativo univoco e non modificabile assegnato dai costruttori di rete.

È dotata di:

- connettore di rete: interfaccia hardware standard presente sul dispositivo di rete per il collegamento del dispositivo di rete al mezzo di trasmissione della rete
- interfaccia di collegamento al calcolatore sul quale transitano i bit ricevuti o trasmessi
- mezzo di trasmissione: supporto fisico alla propagazione e trasmissione dei segnali che realizza l'infrastruttura fisica di rete. Esistono tre tipi:
  - Cavi conduttori, trasmettono segnali elettrici. È il metodo più utilizzato nelle LAN, affidabile, buon rapporto qualità/prezzo.
  - Fibre ottiche, trasmissione di segnali ottici. Offre migliori prestazioni di capacità della rete ma molto costosa l'infrastruttura di rete e il collegamento della fibra.
  - Wireless, radiazione elettromagnetica. Il collegamento non sempre è affidabile ma permettono la mobilità dei calcolatori entro una distanza limitata e riducono le infrastrutture di rete fisse
- protocolli di rete: regole e procedure semantiche di gestione dei processi di comunicazione e regole e formati sintattici per definire uno scambio di messaggi non ambigui. Permettono compatibilità dei dispositivi e dei sistemi conformi allo stesso standard.

Una INFRASTRUTTURA di rete è la struttura di connessione dei collegamenti della rete e può essere

- punto a punto, instaurata tra una coppia di calcolatori e semplice da gestire
- a connessioni multiple
  - completamente connesse, ci sono cammini ridondanti ed è costosa
  - parzialmente connesse, ho il minimo numero di connessioni per garantire un cammino per trasferire informazioni per ogni coppia di host quindi se si guasta una connessione potrebbe risultare una partizione della rete
  - partizioni di rete, dove un gruppo di host è isolato dagli altri

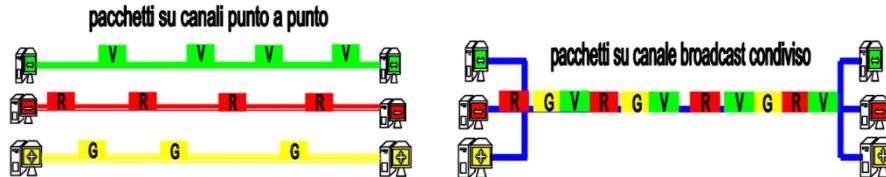
I cammini dei segnali sono i metodi alternativi che i segnali devono seguire per essere trasmessi in rete e possono essere diretti o indiretti attraverso connessioni in sequenza

Il CANALE DI COMUNICAZIONE è una visione astratta del mezzo di trasmissione, quindi può essere

- punto a punto: canale riservato a due soli dispositivi, regole di accesso molto semplici
- ad accesso multiplo, broadcast: tutti i dispositivi trasmettono e ricevono sullo stesso canale con regole di accesso specifiche per evitare collisioni. Questi canali richiedono l'indirizzamento esplicito di ogni pacchetto trasmesso (MAC address)

RETI A COMMUTAZIONE DI CIRCUITO o circuit switched (es. linea telefonica): creazione di un circuito riservato tra mittente e destinatario formato da una serie di canali logici punto. Ha un ritardo di comunicazione basso ma si paga il tempo di connessione, anche se non vengono scambiati dati sul canale.

RETI A COMMUTAZIONE DI PACCHETTO o packet switched (es. rete internet): i dati vengono suddivisi in pacchetti indipendenti che vengono spediti separatamente sul canale, ad ogni nodo intermedio i pacchetti vengono temporaneamente immagazzinati e inoltrati verso il destinatario finale. Se il canale è broadcast ogni pacchetto deve contenere l'indirizzo del destinatario e del mittente, si utilizza lo stesso canale per diversi flussi di pacchetti quindi sono richieste meno risorse (vantaggioso per il provider) e sono meglio utilizzate ma c'è un maggiore ritardo della rete. Si paga per la quantità dei dati trasmessi (vantaggioso per l'utente).



- Servizi orientati alla connessione o connection oriented: garantiscono la consegna ordinata e la ritrasmissione di eventuali pacchetti persi. Può essere fatto definendo un cammino riservato unico per i pacchetti o attraverso la numerazione dei pacchetti inviati
- Servizi non orientati alla connessione o connectionless: i pacchetti possono seguire strade diverse e arrivare in ordine diverso o non arrivare mai

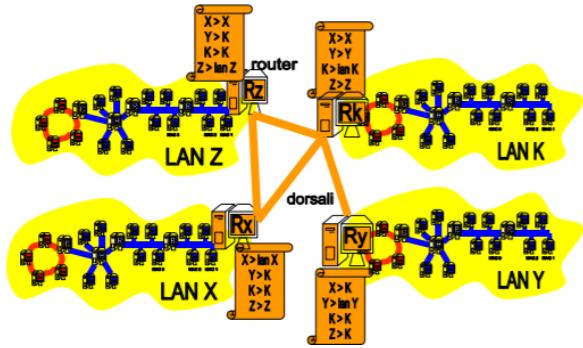
Livello Applicazione	7	ARCHITETTURA DEI PROTOCOLLI DI RETE: suddivisione in livelli, ognuno affronta e
Livello Presentazione	6	risolve un problema della comunicazione, effettua richieste di servizio al livello inferiore e fornisce un servizio al livello superiore. Le richieste e i servizi realizzano l'interfaccia di un protocollo verso altri livelli.
Livello Sessione	5	
Livello Trasporto	4	<u>Standard ISO/OSI RM</u> (Open System Interconnection Reference Model): costituita da sette livelli: il dialogo tra livelli paritari avviene astraendo l'architettura
Livello Rete	3	sottostante, mentre il dialogo tra livelli sovrapposti avviene attraverso l'interfaccia comune per tutti i protocolli.
Livello MAC/LLC	2	
Livello Fisico	1	Il livello 1 e 2 sono implementati nella scheda di rete, mentre gli altri nel sistema operativo.

Architettura dei livelli dei protocolli di Internet: utilizza solo 5 dei 7 livelli dello standard ISO/OSI RM. In trasmissione, ogni livello riceve dati dai livelli superiori e li inserisce (incapsula) in "buste" con dati aggiuntivi, utili ad istruire il corrispondente livello del dispositivo ricevente.

- 7 Livello applicazione: fornisce alle applicazioni in esecuzione sul calcolatore i servizi e le primitive di trasmissione e ricezione dei dati. Passa i dati che l'applicazione richiede al livello inferiore.
- In questi livelli la rete esiste, funziona ed è utilizzabile dalle applicazioni dell'utente.

- 6 Livello presentazione: risolve eventuali eterogeneità del formato dei dati (non presente nell'Internet stack)
- 5 Livello sessione: mantiene e gestisce lo stato attuale del collegamento (non presente nell'Internet stack)
- 4 Livello trasporto: garantisce una connessione affidabile (incapsula i dati aggiungendo informazioni utili all'ordinamento e al controllo della velocità di invio delle buste e controlla se il SEGMENTO arriva a destinazione, servizio connection oriented) e controlla la congestione della rete.  
In questo livello la rete è una collezione di reti gerarchica.
- 3 Livello rete: si occupa di frammentare i dati in pacchetti, scrivere gli indirizzi dei destinatari finali (IP) e instradare i PACCHETTI verso i destinatari intermedi del cammino (servizio connectionless).  
In questo livello la rete è una collezione di reti gerarchica (reti di reti, sottoreti).  
Utilizzo i router (rappresentanti delle reti locali) che smistano i pacchetti di dati tra le reti.
- 2 Livello MAC/LLC:
  - MAC (Medium Access Control): controllo e gestione dell'accesso per trasmettere il FRAME per evitare jamming signal e protocolli per indirizzamento MAC.
  - LLC (Logical Link Control): controlla la correttezza della trasmissione del frame al nodo successivo (il quale MAC address si trova nell'header del FRAME), ovvero rende il canale affidabile (senza errori di trasmissione dovuti a collisioni o interferenza): la trasmissione di un frame procede per tentativi, fino alla ricezione di un frame di conferma (acknowledgement) da parte del destinatario.
- In questo livello la rete è locale, può integrare mezzi trasmissivi e tecnologie diverse.  
Utilizzo bridge e switch per collegare due segmenti di rete locale diversi:
  - Bridge: traduce i frame ricevuti da un segmento nel formato frame dell'altro segmento e ritrasmette il frame tradotto usando il protocollo MAC opportuno.
  - Switch (commutatore): permette di connettere fino a 10-12 segmenti e ha la capacità di instradare il frame sul segmento giusto, leggendo l'indirizzo MAC del destinatario del frame (se non conosce la porta associata ad un indirizzo MAC, invia il frame broadcast con un opportuno protocollo MAC e aspetta il frame di conferma, così da memorizzare il numero di porta dal quale proviene attraverso un buffer). Evitano rallentamenti e accodamenti perché non uniscono i domini di collisione dei segmenti che collegano.
- 1 Livello fisico: si occupa di definire le tecniche di codifica dei dati digitali (bit) in segnali analogici e decodifica, la trasmissione e la ricezione dei dati sul mezzo fisico di trasmissione.  
In questo livello la rete è solo un segmento con la stessa tecnologia in comune a tutti i dispositivi, ovvero un canale ad accesso multiplo.  
Utilizzo i repeater o gli hub per collegare due segmenti di rete locale:
  - Repeater: i segnali trasmessi sul mezzo fisico degradano con la distanza, questo dispositivo amplifica e rigenera il segnale ricevuto collegando due o più segmenti di rete della stessa tecnologia, ma ne unisce il dominio di collisioni, alzando il rischio di collisioni
  - Hub o repeater multiporta: è il punto di contatto delle connessioni a stella. Come il repeater connette segmenti della stessa tecnologia e unisce il loro dominio di collisione.

**RETI DI RETI:** ogni rete locale ha un rappresentante, **router**, che si fa carico di recapitare a livello MAC/LLC i pacchetti dati destinati a un calcolatore della sua rete locale e di inoltrare i pacchetti uscenti dalla propria rete locale verso i router delle reti di destinazione a livello rete attraverso collegamenti dati veloci, dorsali o backbone. Ogni router deve ricordare in una tabella di instradamento, forwarding table, quale sia il primo router intermedio per raggiungere ogni altro router.



Il LIVELLO RETE per Internet si basa sul protocollo IP (Internet Protocol) che definisce un nuovo schema di indirizzamento globale e gerarchico che permette di identificare univocamente tutti i dispositivi di rete e allo stesso tempo la loro rete locale di appartenenza. Si utilizzano nuovi dispositivi amministratori di tale livello: i router

- in grado di nascondere, al livello rete, i dettagli della rete locale
- forniti di tabelle di instradamento (forwarding table) che illustrano la topologia della rete vista al livello dei router stessi e sono costantemente aggiornate attraverso protocolli di routing
- gestiscono la frammentazione dei dati da spedire nei pacchetti, e la creazione della busta di livello rete con gli indirizzi del router mittente e destinatario di ogni pacchetto inoltrato

Al protocollo IP si possono associare protocolli di instradamento dei pacchetti dal mittente al destinatario finale (forwarding) originando servizi di trasmissione a pacchetto di tipo connectionless.

#### IPv4 (protocollo attualmente usato):

Un indirizzo IPv4 viene associato a una sola scheda di rete, con una associazione univoca tra indirizzo MAC dell'interfaccia di rete e indirizzo IP.

Può essere statico, quando l'associazione tra MAC address e IP address non cambia (più soggetto a spoofing), o dinamico, se tale associazione cambia.

È formato da 32 bit (4 byte)= sequenza di quattro valori decimali separati da un punto, ogni valore può essere compreso tra 0 e 255.

È composto da due parti:

- network number: numero della rete IP alla quale appartiene la scheda di rete
- host number: numero dell'interfaccia di rete all'interno della rete

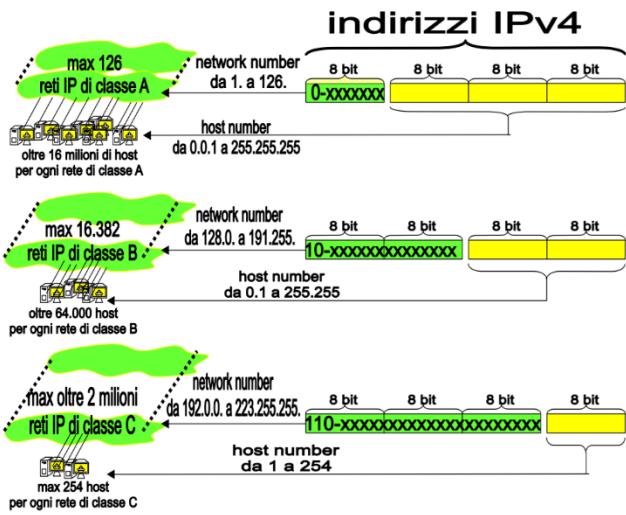
Sono definite tre classi di reti IP, che si differenziano sulla base del numero massimo di host supportabili.

Il valore dell'indirizzo IP determina la classe della rete: A,B,C

- classe A: Network number= primo byte più significativo (a sx), ha il primo bit pari a 0 e può assumere  $2^7$  valori da 1 a 126. Host number= tre byte rimanenti possono assumere oltre  $2^{24}=16$  milioni di combinazioni.

• classe B: Network number= primi due byte di indirizzo più significativi che hanno primi due bit pari a 10 e può assumere  $2^{14}$  valori da 128.0 a 191.255. Host number= due byte rimanenti possono assumere oltre  $2^{16}=64.000$  combinazioni.

• classe C: Network number= primi tre byte di indirizzo più significativi che hanno primi tre bit pari a 110 e può assumere  $2^{22}$  valori da 192.0.0 a 223.255.255. Host number= byte rimanente può assumere oltre 254 (su  $2^8=256$ ) combinazioni utili.



Nell'host number in verità ho due valori che non posso usare per identificare gli host:

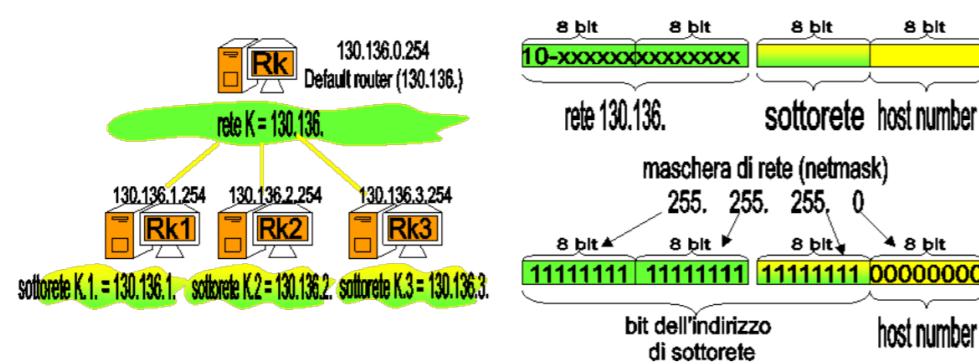
- .0.0 : indirizzo di rete (identificato dai byte precedenti)
- .255.255 : indirizzo broadcast

**SOTTORETI (SUBNETTING):** Reti IP e router sono organizzati secondo una gerarchia, basata sul concetto di rete e sottorete, ognuna amministrata da un router (default router).

Ogni indirizzo IP può essere spezzato in due componenti logiche con la maschera di rete (netmask):

- indirizzo di sottorete (subnetwork): bit con la stessa posizione dei bit uguali a uno nella netmask
- host number che identifica gli host appartenenti alla sottorete, e corrisponde ai bit corrispondenti ai bit uguali a zero nella netmask

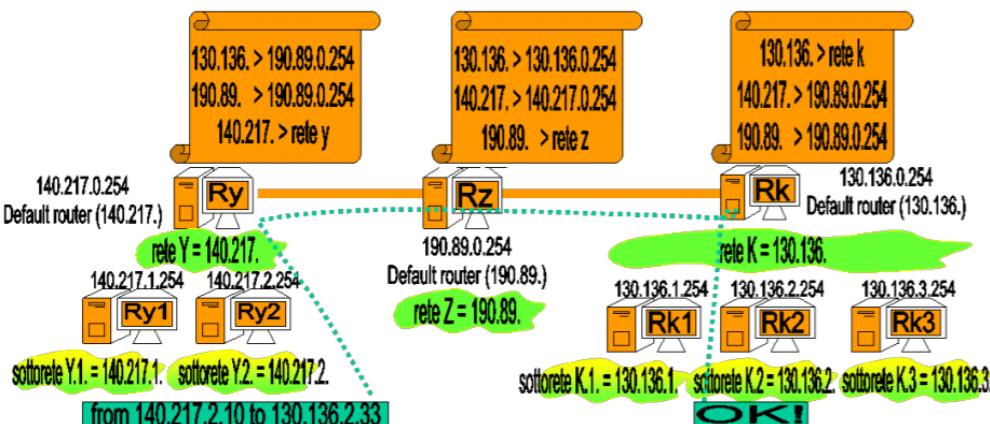
Per ogni dispositivo di rete o calcolatore connesso a una rete IP (es. Internet), la maschera di rete, l'indirizzo del default router e l'indirizzo IP, costituiscono i tre parametri fondamentali di configurazione del protocollo IP, e devono essere forniti, al momento della connessione, al livello IP.



Esempio:

Rete di classe B 130.136.0.0 e Netmask 255.255.255.0. Il numero della sottorete è quindi fornito dal terzo byte dell'indirizzo IP, es. 130.136.1.0 è la sottorete 1 con default router 130.136.1.254. Mentre ad esempio 130.136.1.22 è l'host 22 della sottorete 1, 130.136.3.48 è l'host 48 della sottorete 3, e così via.

#### INSTRADAMENTO DEI PACCHETTI (FORWARDING):



Esempio in figura:

Un pacchetto IP spedito dall'host 140.217.2.10 (rete y, sottorete 2, host 10) all'host 130.136.2.33 (rete k, sottorete 2, host 33) deve compiere il seguente tragitto: passa per il default router di sottorete 140.217.2.254 che, notando che il destinatario non appartiene alla sottorete, lo inoltra al default router del livello superiore di rete: 140.217.0.254. Il default router di rete controlla la propria tabella di forwarding e scopre che per raggiungere la destinazione il pacchetto deve essere

inoltrato al router intermedio 190.89.0.254. Il router intermedio riceve il pacchetto, verifica la propria tabella di forwarding, scopre che il prossimo destinatario intermedio è il router 130.136.0.254, al quale inoltra il pacchetto. Il router 130.136.0.254 riceve il pacchetto e verifica che appartiene alla propria rete, inoltrando quindi il pacchetto internamente. Il router di sottorete 130.136.2.254 riceve il pacchetto e scopre che appartiene alla propria sottorete, inoltrando il pacchetto internamente. Finalmente, l'host 130.136.2.33 riceve il pacchetto a lui destinato.

Le tabelle di instradamento sono limitate agli elementi che agiscono allo stesso livello, e quindi l'instradamento è gerarchico.

Algoritmi di ROUTING: utilizzati per l'aggiornamento delle tabelle di forwarding dei router della rete dovuto alle modifiche dei cammini per i dati nella rete causate dalla mobilità degli host in reti senza fili, guasti di mezzi trasmissivi, interruzione delle linee, guasti di router, nuove politiche e accordi per lo scambio dei dati tra gestori di dorsali e sistemi autonomi AS (ovvero collezioni di rete soggette alla stessa politica di amministrazione). Così si diminuisce la perdita di pacchetti o il loro arrivo disordinato (ecco quindi una causa del servizio connectionless ottenuto dal livello rete basato solo sul protocollo IP). I protocolli di routing hanno la funzione di richiedere e scambiare informazioni per trovare cammini alternativi (idealmente il cammino migliore tra le possibili alternative), tra mittenti e destinatari dei pacchetti, e consentire quindi l'aggiornamento delle tabelle di forwarding.

Esempio di algoritmi di routing su Internet: Routing Information Protocol (RIP), Open Shortest Path First (OSPF), Border Gateway protocol (BGP).

PROTOCOLLO ICMP (Internet Control Message Protocol): protocollo dei messaggi di controllo su Internet. È uno standard per definire la comunicazione di informazioni utili alla gestione di Internet. È usato da host, router e gateway (router speciali con ulteriori funzioni) per scambiare informazioni utili alla gestione del livello rete, trasferite sotto forma di pacchetti IP.

Esempi di possibili messaggi:

- situazioni di rete di destinazione irraggiungibile (possibile sintomo di problemi di routing, o di rottura di un router)
- rete di destinazione sconosciuta (possibile sintomo di indirizzo IP di rete male specificato)
- host di destinazione non raggiungibile (possibile sintomo che l'host sia spento o il cavo di connessione sia male collegato)
- host di destinazione sconosciuto (possibile sintomo di host number dell'indirizzo IP male specificato, malgrado la rete indicata esista e sia raggiungibile)
- protocollo richiesto non disponibile (sintomo di un tentativo di dialogo tra dispositivi male configurati, che non forniscono i servizi richiesti)
- ricerca di cammino alternativo (può essere usato per risolvere i problemi di routing).

Esempi di applicazioni basate su ICMP (per la verifica di eventuali problemi di rete):

- PING: test di verifica di connessione tra due host
- Traceroute: mostra la sequenza di router attraversati da un pacchetto per arrivare all'host destinazione (realizzato con messaggi ICMP spediti in sequenza, verso distanze via via maggiori)

ARP (Address Resolution Protocol) E RARP (Reverse-ARP):

Il protocollo ARP lega l'indirizzamento a livello MAC con l'indirizzamento a livello IP. Quando un router riceve un pacchetto a livello IP destinato alla sua sottorete, esso verifica se a tale IP risulti o meno associato un indirizzo MAC. In caso contrario, il router spedisce sui segmenti della rete locale un frame in broadcast contenente il codice di richiesta ARP e l'indirizzo IP del destinatario del pacchetto (chiedendo così "quale indirizzo MAC ha il dispositivo corrispondente al seguente indirizzo IP"). Se tale dispositivo esiste, risponde con un frame di livello MAC indirizzato al router, nel quale viene evidenziato l'indirizzo MAC richiesto. A questo punto il router può quindi preparare e spedire la busta di livello MAC/LLC, indirizzata al MAC del dispositivo destinatario del pacchetto IP, contenente il pacchetto IP encapsulato all'interno. Il destinatario riceve il frame e risponde con il frame di conferma per il sottolivello LLC. RARP, invece, risponde alla domanda: "quale indirizzo IP corrisponde al dispositivo con questo indirizzo MAC".

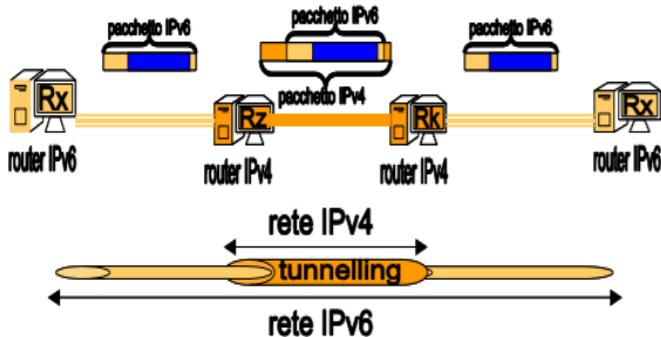
I numeri di rete delle classi A, B e C vengono assegnati da enti internazionali (RIPE, ICANN, ARIN, APNIC) a enti, aziende e imprese che ne fanno richiesta motivata.

Quando un nuovo dispositivo viene connesso a una rete esistente, bisogna associare al suo indirizzo MAC un indirizzo IP della rete stessa:

- Attraverso un amministratore di rete che assegna manualmente uno dei numeri di host disponibili al nuovo indirizzo MAC. In questo modo l'associazione indirizzo MAC e indirizzo IP può essere mantenuta per un tempo indeterminato, e quindi si considera l'indirizzo IP come statico.
- Attraverso l'utilizzo del protocollo **DHCP (Dynamic Host Configuration Protocol)**: Il server DHCP è dotato di una lista di numeri di host liberi per la sottorete amministrata che provvede ad associare su richiesta agli indirizzi MAC dei dispositivi che lo richiedono. Allo stesso indirizzo MAC possono essere associati di volta in volta indirizzi IP diversi, e si parla in questo caso di indirizzi IP dinamici.  
Si parla di rete "plug and play".

### IPv6 (nuova versione del protocollo di indirizzamento IP)

- Indirizzi composti da 128 bit (16 Byte) (anziché i 32 bit, 4 Byte di IPv4).
- Nuova struttura dei campi della busta dei pacchetti di livello rete (IP) aggiungendo parametri per la gestione di flussi di pacchetti IP con diversi livelli di priorità.



- La sperimentazione IPv6 avviene su reti separate (nuovi router IPv6), ma ci sono casi di integrazione tra IPv4 e IPv6 usando tecniche di tunnelling (il pacchetto IPv6 viene incapsulato da ogni router IPv4 in pacchetti IPv4, in modo da poter essere instradato lungo la rete di router IPv4 e uscito dal tunnel IPv4 il pacchetto IPv6 prosegue il suo inoltro fino alla destinazione IPv6 finale).
- IPv6 e IPv4 sono best effort (non garantiscono la qualità del servizio).

### LIVELLO TRASPORTO utilizza i protocolli TCP e UDP

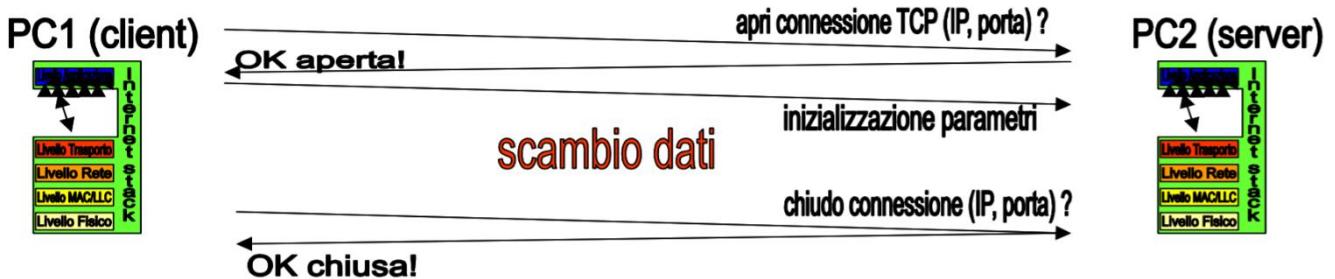
- UDP, User Data Protocol, fornisce un servizio connectionless, non affidabile.
- TCP, Transmission Control Protocol, fornisce un servizio di tipo connection-oriented, affidabile (usato di fatto su Internet).

Si occupa di numerare sequenzialmente i dati spediti, garantire il ripristino dell'ordinamento dei pacchetti ricevuti e la ri-trasmissione dei pacchetti perduti (nel caso non riceva l'acknowledgment, entro il timeout, di un pacchetto spedito).

- TCP consente lo smistamento dei pacchetti verso le rispettive applicazioni in ascolto su "porte":

Richiede l'attivazione della connessione punto a punto tra due socket, ovvero un punto di arrivo o partenza (virtuale) dei dati a livello trasporto, dal quale è in atto l'invio e la ricezione di pacchetti destinati a un'applicazione ed equivale a una coppia: (indirizzo IP, numero di porta dell'applicazione). La connessione viene instaurata con una richiesta dell'host client nei confronti dell'host server specificando l'indirizzo IP del server e il

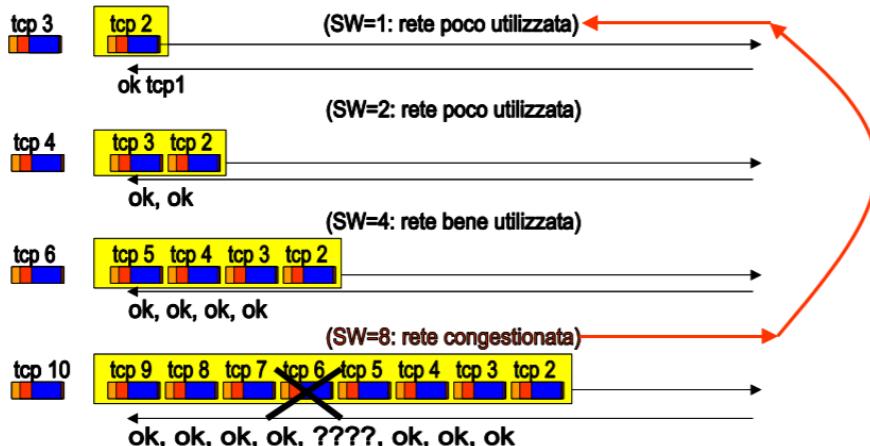
numero di porta sul quale è in attesa l'applicazione con la quale si intende dialogare. Il server verifica che il socket esista (verifica il numero di porta) e che non sia già occupato, e in caso affermativo risponde con un pacchetto di conferma. Se il pacchetto di conferma è ricevuto, il client invia un ulteriore pacchetto di conferma contenente i dati di configurazione, dopodiché la comunicazione procede attraverso lo scambio di pacchetti dati tra i livelli TCP, che verranno smistati verso le porte indicate. Lo scambio dati avviene attraverso pacchetti che saranno ordinati e ritrasmessi in caso di perdita a cura del protocollo TCP. Al termine del dialogo, la connessione TCP tra due applicazioni può essere rilasciata, liberando la porta occupata.



- TCP implementa tecniche di controllo di flusso e di congestione di rete con lo scopo di:
  - saturare il più possibile la rete di pacchetti, inviandoli a un ritmo elevato per favorire l'utilizzo delle risorse e le prestazioni della rete
  - evitare che un ritmo di invio troppo elevato possa causare congestioni nei router intermedi del cammino (se un router si trova a dover inoltrare troppi pacchetti provenienti da flussi TCP diversi, i pacchetti si accumulano fino ad andare perduti)

Quindi, per cercare il massimo ritmo di spedizione che possa garantire l'inoltro dei pacchetti da parte del router più lento del cammino e prevenire la saturazione del destinatario finale, controlla la velocità di invio dei flussi dei pacchetti mediante il meccanismo a finestra scorrevole, ovvero SLIDING WINDOWS (SW):

Una finestra scorrevole è un valore intero che rappresenta il numero massimo di pacchetti che un mittente può spedire di seguito, in attesa di ricevere la loro conferma. Ogni mittente TCP spedisce al massimo ritmo possibile un gruppo di SW pacchetti, se gli SW pacchetti sono tutti confermati, aumenta SW. Appena un pacchetto degli SW inviati non viene confermato (scade timeout) assume che ci sia una congestione su un router e rallenta il ritmo di invio ripartendo da SW minimo (e risedendo i pacchetti non confermati =controllo del flusso), per dare modo al router congestionato di smaltire i pacchetti accumulati (=controllo della congestione).



Gli utenti preferiscono usare nomi per le risorse in rete, anziché indirizzi IP: i nomi di dominio per le reti sono mnemonici e hanno una struttura gerarchica, sono assegnati da enti internazionali in modo univoco, come gli indirizzi IP.

Le risorse appartenenti a un dominio possono avere nomi scelti arbitrariamente (ad esempio nomi di host, indirizzi di e-mail) purchè non siano duplicati all'interno del dominio stesso.

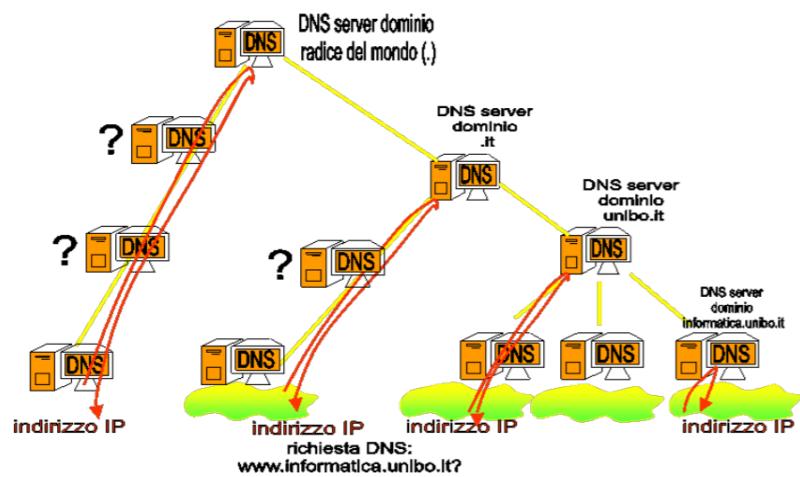
Struttura gerarchica dei nomi di risorsa: (nomerisorsa.sottodominio.sottodominio.dominioradice).

I protocolli di rete e i router pretendono, però, di usare gli indirizzi IP: da qui nasce il servizio DNS, ovvero DOMAIN NAME SYSTEM:

È basato su una catena di server DNS organizzati gerarchicamente. Ogni host in rete deve conoscere almeno un DNS server, ogni server DNS conosce almeno un DNS server superiore.

I server ricevono richieste (attraverso il protocollo DNS che permette di propagare le richieste verso la radice o verso le foglie) e forniscono indirizzi IP. Se un server non conosce la risposta inoltra la richiesta DNS a un server superiore.

I server di livello superiore conoscono un numero sempre maggiore di nomi e relativi indirizzi IP, ma sono sempre meno per motivi di costo. I server DNS radice (DNS root server) conoscono tutti i domini e i loro IP.



LIVELLO APPLICAZIONE: l'implementazione delle funzioni e dei servizi che permettono alle applicazioni di rete in esecuzione sull'host di spedire e ricevere i dati.

Si appoggia sul livello trasporto e molte applicazioni che richiedono servizi connection-oriented si basano sul protocollo TCP attraverso numeri di porta che col tempo sono diventati standard.

Esempi di famose applicazioni di rete e servizi del livello applicazione

- Posta elettronica (E-mail): basati su protocolli di livello applicazione SMTP, POP3, IMAP
  - Simple Mail Transfer Protocol (SMTP): per la spedizione e trasporto dei dati (porta 25).
  - Post Office Protocol 3 (POP3) e Internet Mail Access Protocol (IMAP): per la consegna dei dati all'utente.
- World Wide Web (WWW): basato su applicazione e protocollo HTTP
  - Hyper Text Transfer Protocol (HTTP): protocollo per trasferire pagine web (porta 80).
- DNS: Domain Name Service (protocollo DNS)

Le applicazioni e i servizi su Internet possono essere realizzati secondo due modalità architettoniche

- ARCHITETTURA CLIENT/SERVER: i Client sono host che spediscono richieste di servizio, i Server sono host sui quali sono in esecuzione i servizi che soddisfano le richieste (es. DNS, www, mail).
- ARCHITETTURA PEER TO PEER (P2P): tutti gli host sono contemporaneamente sia client che server, ogni host agisce da Server cercando di soddisfare, se possibile, le richieste ricevute, e ogni host agisce da Client quando spedisce ad altri host le richieste, o per se stesso, o per soddisfare richieste di terzi, es. Freenet, Gnutella, Kazaa (file-sharing).
- Servizi ibridi: esistono server che aiutano a trovare i giusti host P2P, es. Napster (file sharing).

## RETI DI CALCOLATORI E INTERNET

Internet è una rete di calcolatori. Tutti i dispositivi sono detti **host** o **sistemi periferici**.

I sistemi periferici sono connessi tra loro tramite una **rete di collegamenti** e **commutatori di pacchetti**. Questi collegamenti possono essere di diverso tipo.

Collegamenti diversi trasmettono a velocità diverse, e tale velocità viene chiamata **velocità di trasmissione** e si misura in bit al secondo (bps).

Quando un host deve mandare dati ad un altro host, le informazioni vengono trasmesse sottoforma di **pacchetti**. Un commutatore di pacchetto prende un pacchetto che arriva da un collegamento in ingresso e lo immette in uno di uscita Principali commutatori di pacchetto:

- Router
- Commutatore a livello di collegamento

Dal sistema di invio a quello di ricezione, la sequenza di collegamenti e di commutatori di pacchetto che il pacchetto attraversa, è nota come **percorso** (route o path).

I sistemi periferici accedono ad Internet tramite gli **Internet Service Provider (ISP)**. Un provider è un insieme di commutatori di pacchetto e di collegamenti. Gli ISP rendono disponibile l'accesso ad Internet ai fornitori di contenuti, connettendoli direttamente al sito web.

Sistemi periferici, commutatori di pacchetto a altre parti di Internet fanno uso di particolari protocolli che regolano l'invio e la ricezione di informazioni nella rete. Tra i protocolli di Internet abbiamo:

- TRANSMISSION CONTROL PROTOCOL TCP
- INTERNET PROTOCOL IP: che specifica il formato dei pacchetti scambiati tra router ed host.

Possiamo vedere Internet come un'infrastruttura che fornisce servizi alle applicazioni. Quest'ultime sono dette **applicazioni distribuite**, perché comprendono vari sistemi periferici che si scambiano dati. Le applicazioni per internet vengono eseguite su sistemi periferici e non su commutatori di pacchetti.

I sistemi periferici connessi ad Internet forniscono delle **API**, ossia un insieme di regole che il mittente deve rispettare in modo tale che i dati siano trasmessi correttamente.

Ogni attività su Internet deve seguire un determinato protocollo.

Un **protocollo** definisce il formato e l'ordine dei messaggi scambiati tra due o più entità in comunicazione, così come le azioni intraprese in fase di trasmissione e/o di ricezione di un messaggio o di un altro evento.

Gli host sono divisi in **client** e **server**. I client sono host che richiedono servizi, mentre i server sono host che forniscono i servizi.

Una rete di accesso è una rete che connette fisicamente un sistema al suo **edge router**, ossia il primo router che si incontra sul percorso mittente-destinatario.

Ad oggi, i due accessi residenziali a larga banda più diffusi sono i **Digital Subscriber Line (DSL)** e quello via cavo. Mentre la DSL usa la linea telefonica per fornire Internet, l'accesso alla rete via cavo utilizza le infrastrutture esistenti della televisione via cavo. Questo tipo di accesso richiede dei cable modem.

Aziende, università ma anche privati, per collegare i sistemi periferici ad Internet, usano una **rete locale LAN**. Esempio di LAN: Ethernet che usa un doppino di rame per collegare sistemi periferici e li connette ad uno switch. Quest'ultimo viene poi connesso ad Internet.

Per collegare un host ad Internet a volte è necessario disporre di un mezzo fisico. Tra questi ricordiamo il doppino di rame intrecciato, il cavo coassiale, la fibra ottica e lo spettro radio.

I mezzi fisici possono essere **vincolanti** (onde mantenute in un mezzo fisico) e **non vincolanti** (le onde si propagano nell'atmosfera).

- Doppino di rame intrecciato: è costituito da due fili di rame distinti disposti a spirale, ciò viene fatto per evitare l'interferenza generata dalle altre coppie di fili vicine.
- Cavo coassiale: è costituito da due conduttori di rame concentrici. E' un esempio di mezzo fisico condiviso vincolato.
- Fibra ottica: è un mezzo sottile e flessibile che conduce impulsi luce, ognuno dei quali rappresenta un bit. La fibra è immune all'interferenza elettromagnetica.
- Canali radio: trasportano segnali all'interno dello spettro elettromagnetico. Le caratteristiche di questo mezzo fisico dipendono dall'ambiente e dalla distanza, in quanto determinano la perdita di segnale a causa della distanza, dalla riflessione sulle superfici o dall'interferenza con altri canali.
- Canali radio satellitari: il satellite collega due o più trasmettitori terrestri a microonde. I tipi di satelliti più usati per le comunicazioni sono i satelliti a bassa quota e quelli geostazionari.

## COMMUTAZIONE DI PACCHETTO

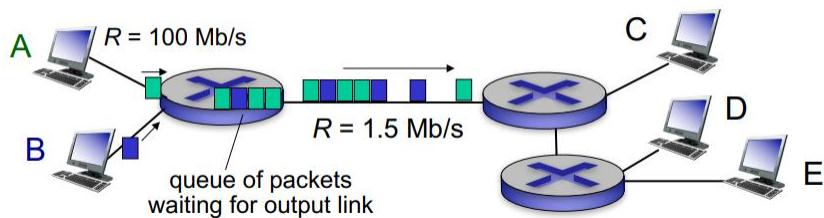
Come si mandano i messaggi?

1. La sorgente prende il messaggio da spedire
2. Il messaggio viene suddiviso in pacchetti di lunghezza L
3. Il messaggio viene spedito con una velocità R, quindi il tempo di trasmissione sarà  $L/R$

La maggior parte dei commutatori utilizza la tecnica dello **store-and-forward**, ossia il commutatore deve ricevere l'intero pacchetto prima di poterne spedire un altro.

Ogni commutatore ha un buffer di output (coda di output) che contiene i pacchetti da mandare. Se il pacchetto da mandare trova il canale occupato, si mette in coda nel buffer. Quindi, oltre al ritardo di store-and-forward avremo anche un **ritardo di accodamento**.

Dato che la dimensione del buffer è limitata, se all'arrivo del pacchetto il buffer è pieno, può succedere una **perdita del pacchetto**, ossia verrà eliminato il pacchetto in arrivo o uno di quelli in coda.



Come vedere dove mandare il pacchetto? Ogni pacchetto ha nell'intestazione l'indirizzo della sua destinazione. Quando il pacchetto arriva al router, questo ha una **tavella di instradamento (forwarding table)** che mette in relazione gli indirizzi di destinazione dei pacchetti con i collegamenti in uscita.

## COMMUTAZIONE DI CIRCUITO

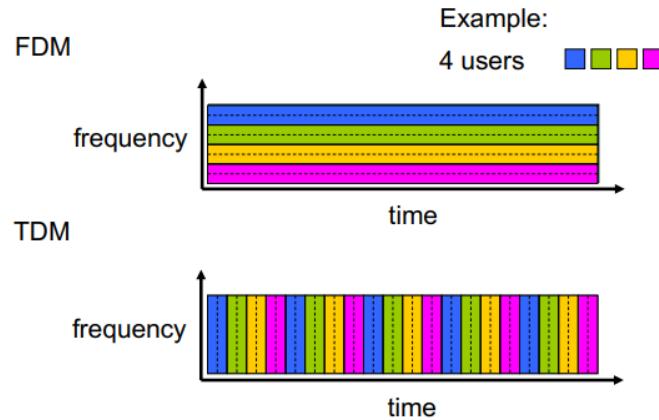
Per poter mandare i dati servono due procedure:

- Commutazione a pacchetto: le risorse non sono riservate, vengono utilizzate quando necessario, si rischia di attendere per un collegamento
- Commutazione a circuito: le risorse sono riservate per tutta la durata della comunicazione

Un circuito all'interno di un collegamento è implementato tramite **multiplexing a divisione di frequenza (FDM)** o **multiplexing a divisione di tempo (TDM)**.

Con FDM il collegamento dedica una banda di frequenza a ciascuna connessione per la durata della

connessione stessa. Con TDM, il tempo viene suddiviso in frame di durata fissa, a loro volta ripartiti in un numero fisso di slot temporali.



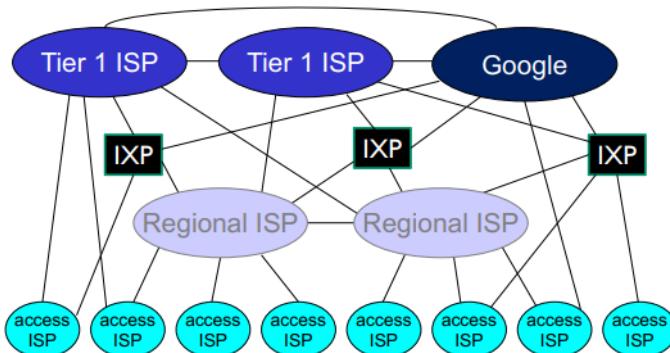
#### Commutazione di circuito vs Commutazione di pacchetto

Secondo alcuni, la commutazione di pacchetto non è adatta a servizi in tempo reale a causa dei suoi ritardi end-to-end variabili. Per altri, la commutazione di pacchetto non offre solo una migliore condivisione della larghezza di banda, ma è anche semplice, più efficiente e meno costosa.

### UNA RETE DI RETI

I sistemi periferici si connettono ad Internet tramite un ISP, che può collegarli attraverso una rete cablata o senza fili.

Però, per completare il quadro generale, gli ISP devono essere connessi tra loro.



Struttura di rete 1: interconnette tutti gli ISP di accesso con un unico ISP globale di transito.

Struttura di rete 2: gerarchia a due livelli nella quale i provider globali stanno in cima alla gerarchia e gli ISP di accesso alla base. In ogni regione è presente un **ISP regionale** che si connette all'**ISP di primo livello** (tier-1 ISP).

In questa gerarchia, ogni ISP di accesso paga l'ISP regionale a cui si connette, che a sua volta paga l'ISP di primo livello, che invece non deve pagare nessuno dato che è in cima alla gerarchia.

Struttura di rete 3: in alcune regioni ci può essere un ISP regionale più grande al quale gli ISP regionali più piccoli della regione si connettono, in questo caso l'ISP regionale più grande si connette all'ISP di primo livello.

Per arrivare a capire come è strutturato Internet, dobbiamo aggiungere alla struttura di rete 3 i **PoP**, ossia un gruppo di router vicini tra loro nella rete del provider.

Per avere una connessione globale, gli ISP clienti pagano i loro fornitori. Il costo dipende dalla quantità di traffico che l'ISP cliente scambia col fornitore. Per ridurre questi costi, una coppia di ISP vicini può fare uso di **peering**, cioè connette direttamente le loro reti in modo tale che tutto il traffico tra esse passi tramite una connessione diretta invece che avere qualcuno che fa da tramite. Formiamo così la struttura di rete 4.

Struttura di rete 5: è costituita dalla struttura di rete 4 aggiungendo i **content provider networks** (coloro che forniscono i contenuti).

## RITARDI E PERDITE

Un pacchetto, nel suo percorso da mittente a destinatario può subire diversi ritardi tra cui:

- **Ritardo di elaborazione:** tempo per esaminare l'intestazione del pacchetto e per determinare dove dirigerlo.
- **Ritardo di accodamento:** ritardo che si ha quando un pacchetto si accoda nel buffer aspettando la trasmissione.
- **Ritardo di trasmissione:** tempo per spedire tutti i bit, è dato da  $L/R$ .
- **Ritardo di propagazione:** tempo per propagarsi fino al router di destinazione, è dato da  $d/v$ , con  $d$  distanza e  $v$  velocità.

Siano ora  $d_{elab}$  il ritardo di elaborazione,  $d_{acc}$  il ritardo di accodamento,  $d_{trans}$  il ritardo di trasmissione e  $d_{prop}$  il ritardo di propagazione, allora il ritardo totale del nodo  $d_{nodo}$  sarà:

$$d_{nodo} = d_{elab} + d_{acc} + d_{trans} + d_{prop}$$

## RITARDO DI ACCODAMENTO E PERDITA DEI PACCHETTI

A differenza degli altri, il ritardo di accodamento può variare da pacchetto a pacchetto.

Supponiamo sia  $a$  la velocità media di arrivo dei pacchetti nella coda,  $L$  la lunghezza in bit del pacchetto e  $R$  la velocità di trasmissione. Quindi la velocità media di arrivo dei bit in coda è di  $La$ , mentre il rapporto  $La/R$  è detto **intensità di traffico**.

- $La/R > 1$ : ritardo di accodamento infinito
- $La/R \approx 0$ : poco ritardo di accodamento
- $La/R \leq 1$ : abbastanza ritardo di accodamento

Le code, come abbiamo visto, hanno capacità finita. Se un pacchetto arriva quando la coda è piena, data l'impossibilità di memorizzarlo, esso andrà perduto.

## TRACEROUTE

Per capire quanto ritardo ha fatto il nostro pacchetto usiamo traceroute. Quando l'utente immette l'indirizzo di destinazione, il modulo dell'utente invia un certo numero di pacchetti speciali verso tale destinazione.

Durante il loro percorso, questi pacchetti passano attraverso una serie di router. Quando un router riceve uno di questi pacchetti speciali, invia un messaggio che torna al mittente. Questo messaggio contiene il nome e l'indirizzo del router.

Così facendo, l'origine è in grado di ricostruire il percorso del pacchetto e determinare i ritardi di andata e ritorno per tutte le tratte.

## ESEMPIO

1 cs-gw (128.119.240.254) 1 ms 1 ms 2 ms  
2 border1-rt-fa5-1-0.gw.umass.edu (128.119.3.145) 1 ms 1 ms 2 ms  
3 cht-vbns.gw.umass.edu (128.119.3.130) 6 ms 5 ms 5 ms  
4 jn1-at1-0-0-19.wor.vbns.net (204.147.132.129) 16 ms 11 ms 13 ms  
5 jn1-so7-0-0-0.wae.vbns.net (204.147.136.136) 21 ms 18 ms 18 ms

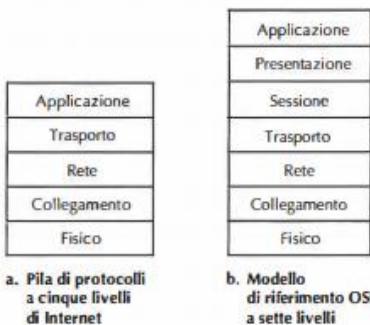
Numero router  
Nome router  
Indirizzo IP router  
Ritardo di andata e ritorno nelle ultime 3 prove

**Troughput:** velocità con cui i bit sono trasferiti dal mittente al destinatario. Può essere:

- Istantaneo: velocità in un dato punto nel tempo;
- Medio: velocità in un certo periodo di tempo. Sarà  $F/T$ , con F bit e T secondi.

## ARCHITETTURA A LIVELLI

I protocolli di rete si organizzano in livelli o strati. Ciascun protocollo appartiene ad un livello.



- **LIVELLO APPLICAZIONE:** sede delle applicazioni di rete. I protocolli presenti sono HTTP, SMTP, FTP. I pacchetti vengono visti come messaggi.
- **LIVELLO TRASPORTO:** sposta i messaggi del livello applicazione alle applicazioni. Vi sono due protocolli: TCP e UDP. I pacchetti vengono visti come segmenti
- **LIVELLO RETE:** trasferisce i pacchetti (o datagrammi) da un host all'altro. Protocollo presente: IP, che definisce i campi dei datagrammi. Vi sono poi altri protocolli utili

all'instradamento dei pacchetti.

- **LIVELLO COLLEGAMENTO:** trasferisce i pacchetti da un nodo all'altro. Diversi protocolli, ad esempio Ethernet o il WiFi. I pacchetti sono detti frame.
- **LIVELLO FISICO:** trasferisce bit del frame da un nodo al successivo. Protocolli dipendono dal livello collegamento + il mezzo fisico.

## Modello OSI

E' formato da sette livelli: i cinque che compongono Internet (vedi sopra) e

- **LIVELLO PRESENTAZIONE:** consente di interpretare il significato dei dati scambiati.
- **LIVELLO SESSIONE:** fornisce la delimitazione e la sincronizzazione dello scambio dei dati.

## RETI SOTTO ATTACCO

I **malware** sono dei contenuti che possono entrare nel nostro computer ed infettarlo. Si dividono in: **virus**, interagiscono con l'utente (es. arrivo mail dove bisogna cliccare su un link fasullo) e **worm**, che non interagiscono con l'utente (es. pagine web fasulle).

**Botnet:** sito web in cui è presente la lista di tutti i computer infettati.

Una classe di minacce alla sicurezza può essere classificata come attacchi di **negazione al servizio (DoS)**, ossia un attacco che rende inutilizzabile dagli utenti in rete, un host o una parte di infrastruttura.

Ad oggi molte persone si collegano in rete tramite il WiFi, ma bisogna stare attenti in quanto, se ci si collega in una rete pubblica, potremmo avere a che afre con i **packet sniffer**, ossia un ricevitore passivo che memorizza una copia di ogni pacchetto che transita su quella rete.

La capacità di immettere pacchetti in Internet con un indirizzo sorgente falso è nota come **IP spoofing**, ed è uno dei modi in cui un utente si spaccia per qualcun altro.

## RIASSUNTO LIVELLO APPLICAZIONE.

-ES applicazioni: e-mail, messaggistica di rete, accesso in remoto, scambio di file in modalità Peer to Peer, videogiochi multigiocatore, fruizione di video streaming, chiamate con modalità “voice over IP” (Skype), video-conferenze in real-time, social network ecc.

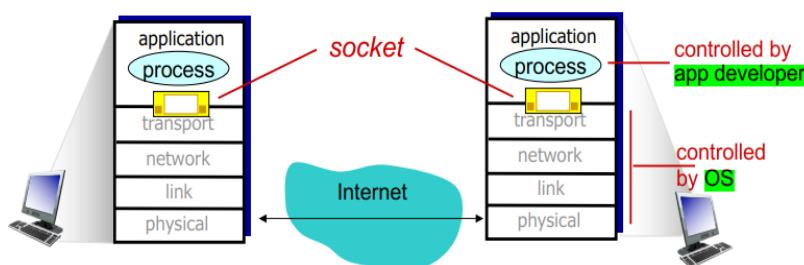
### INTRODUZIONE.

Il cuore dello sviluppo di un'**applicazione di rete** è costituito dalla compilazione di programmi che sono eseguiti da *sistemi periferici* (edge network) e che comunicano tra loro tramite la rete. Un'app deve poter essere eseguita su macchine diverse, ma non occorre predisporre programmi per i dispositivi della *rete core* (cioè l'insieme dei dispositivi quali router o commutatori di pacchetto, che lavorano ad un livello più basso dello stack Internet).

Alla base del livello applicazione, distinguiamo **2 architetture**:

- **client-server**: chiamiamo *server* un host “sempre attivo”, pronto ad eseguire le richieste dei client, dotato quindi di un indirizzo IP permanente e spesso allocato all'interno di data center per fini di “scalabilità del sistema”. Chiamiamo invece *client* un host che comunica con il server (e non direttamente con altri client), inviando richieste. Non è sempre connesso, di conseguenza il suo indirizzo IP può cambiare nel tempo;
- **P2P**: basata sul principio secondo cui ogni host può assumere sia il ruolo di client che quello di server. Di conseguenza, i server non sono sempre on-line, e coppie arbitrarie di dispositivi periferici possono comunicare tra loro in modo diretto. La necessità di avere indirizzi IP dinamici fa sì che questo tipo di architettura sia di più difficile gestione rispetto a quella client-server, ma allo stesso tempo si ottiene un grande vantaggio in termini di “auto-scalabilità” (ogni nuovo dispositivo aggiunge ulteriore “capacità di calcolo” al sistema complessivo, migliorandone l’efficienza) e di sicurezza (le app, essendo distribuite, sono più difficili da attaccare).

Chiamiamo “**processo**” un programma che lavora all'interno di un host. → Distinguiamo un processo client, che dà inizio alla comunicazione, e un processo server, che resta in attesa di essere contattato. Più processi comunicano tra loro, tramite scambio di *messaggi* se fanno parte di host diversi, e tramite la “*comunicazione inter-processo*” se sono in esecuzione sullo stesso host.



Questo scambio di messaggi avviene attraverso i **socket**, simili a porte che costituiscono un'interfaccia tra il livello applicazione e i sottostanti livelli dello stack (lo scambio non potrebbe avvenire senza appoggiarsi al livello di trasporto che implementa TCP e UDP). Ogni porta socket ha un suo identificatore, costituito da un IP a 32 bit e da un numero di porta.

Cosa definisce un **protocollo** del livello applicazione: il *tipo* di messaggio scambiato (richiesta o risposta), la *sintassi* del messaggio (i campi che costituiscono il messaggio), la sua *semantica* (cosa significa ciascun campo e quali informazioni trasporta), *regole* per coordinare lo scambio tra richieste e risposte.

Ogni applicazione ha bisogno di diversi **servizi** garantiti dal **livello di trasporto**, al fine di mantenere uno stato di “efficienza”. Tra questi distinguiamo:

- **Trasferimento dati affidabile**: mentre alcune app sono tolleranti alle perdite (riproduzione di audio e video a uso personale), altre necessitano di un trasferimento che sia al 100% affidabile (si pensi alle app che riguardano transizioni finanziarie);
- **Throughput**: riguarda il tasso al quale il processo mittente può mandare i bit al processo ricevente. Le app che hanno bisogno di un minimo tasso garantito (come quelle di servizi di telefonia su Internet) sono dette sensibili alla banda, le altre sono definite “elastiche” (posta elettronica);
- **Sicurezza**: alcune app necessitano di un trasporto sicuro con criptazione dei dati e protezione da attacchi esterni;
- **Timing**: alcune app necessitano che il “ritardo” di ricezione sia basso (come nel caso dei giochi multiutente o delle conferenze online).

Al momento di implementare un'app, occorre quindi tener conto di questi fattori e scegliere il protocollo di trasporto più adatto, considerando che:

- **TCP**: garantisce un trasporto affidabile, il controllo del flusso e della congestione, un servizio connection oriented, ma non dà garanzie sul timing, sul minimo throughput garantito e sulla sicurezza.
- **UDP**: implementa un servizio di trasferimento non affidabile e non dà nessuna forma di garanzia.

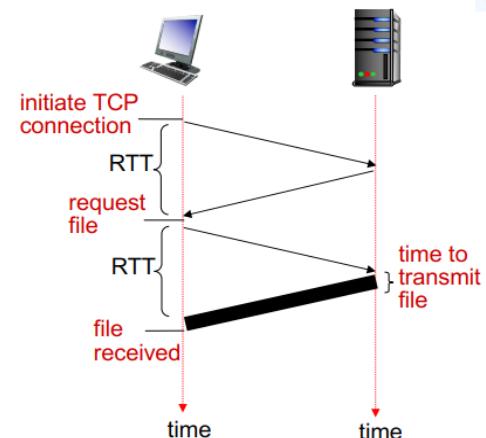
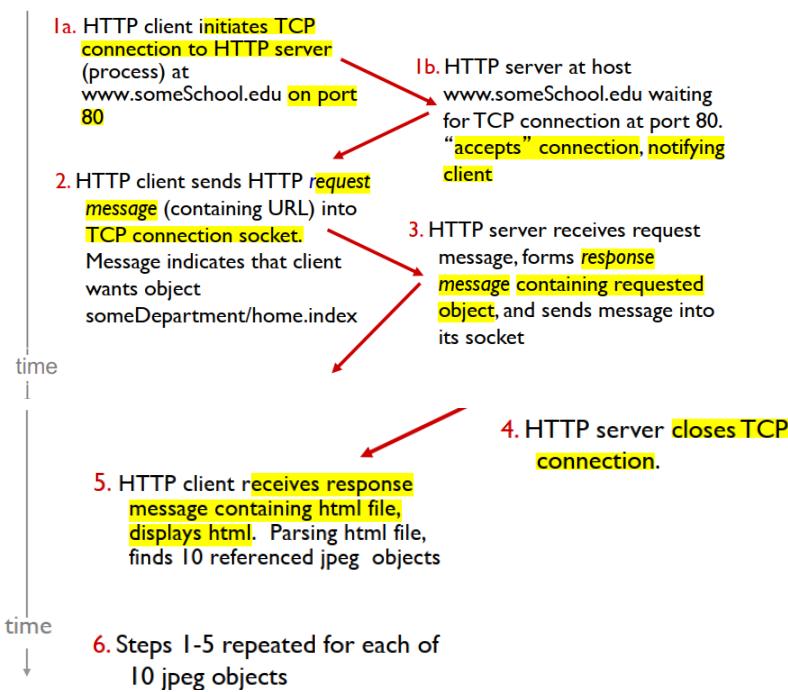
[Parentesi: esiste un protocollo di livello applicazione che rende sicuro il protocollo TCP: stiamo parlando del **SSL**, che oltre a fornire criptazione dei dati, ne garantisce l'integrità e l'autenticazione].

## WEB E HTTP.

Il Web è uno dei maggiori servizi di livello applicazione. Una pagina web è formata da una "base", costituita da un file HTML, che contiene riferimenti a diversi **oggetti**. Questi possono essere file HTML, immagini JPEG, applicazioni JAVA, file audio e video... Ognuno di essi è indirizzabile con un indirizzo **URL**, contenente il nome dell'host e il nome del percorso → (ES: www.someschool.edu/someDept/pic.gif).

Il servizio Web è implementato con **protocollo HTTP**, basato su architettura *client-server*. In particolare, il client è il browser che richiede, riceve oggetti e li dispone nella corretta posizione nella pagina web al fine di mostrarli all'utente. Il server è un Web server che invia gli oggetti richiesti dal client con protocollo HTTP. Questo protocollo è *stateless*, quindi non mantiene alcuna informazione riguardo le richieste passate, ed usa esclusivamente TCP.

Distinguiamo due tipi di connessioni HTTP: **non persistente** e **persistente** → nella prima, per ogni connessione instaurata è inviato un solo oggetto, poi la connessione è chiusa. Di conseguenza, per ricevere N oggetti, si avrà bisogno di N connessioni consecutive. Nel secondo caso, invece, con una singola connessione possono essere inviati molti oggetti. Vediamo la prima delle due connessioni nel dettaglio, con 2 immagini:

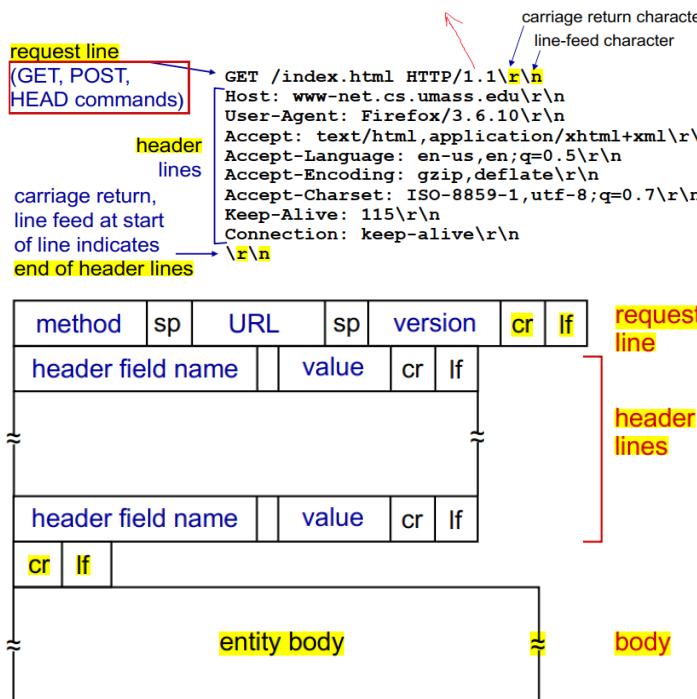


Analizziamo ora il **tempo di risposta (RTT)**, ossia il tempo impiegato da un pacchetto per viaggiare da client a server e viceversa. Spendiamo un RTT per instaurare la connessione, un RTT per effettuare la richiesta

HTTP, attendiamo infine il tempo necessario al trasferimento dei file. Otteniamo un totale di **2RTT + tempo trasmissione file**. Se si pensa di ripetere questa procedura per ogni oggetto, il sistema operativo è sovraccaricato di lavoro per ogni richiesta. L'unica alternativa è quindi quella di aprire connessioni parallele per ottenere tutti gli oggetti in minor tempo (*pipeling*).

Nel caso della connessione HTTP **persistente**, il server lascia aperta la connessione per il tempo necessario affinché il client invii le sue N richieste e riceva gli N oggetti. Nel caso ottimo si avrà un RTT = 1 per ottenere tutti gli N oggetti. Il tempo di invio di un file non è influente, perché il client può inviare una richiesta nell'esatto momento in cui trova il riferimento corrispondente nella pagina web, senza aspettare che la richiesta precedente sia stata completamente soddisfatta.

Vediamo ora la **sintassi di un messaggio di richiesta HTTP** (tutto in codice ASCII).



**NOTE:** nella linea di richiesta, il campo *method* dovrebbe indicare quale comando sceglieremo tra GET, POST, HEAD, URL, PUT, DELETE. Oltre al campo URL, abbiamo un campo *versione* dell'HTTP (spiegato dopo). Ogni riga è interrotta da \r e \n (ritorno a capo e nuova linea).

La riga successiva a quella di richiesta apre l'area di intestazione (header line), che è chiusa da una riga contenente i soli caratteri \r \n.

Successivamente inizia l'**entity body**, utilizzato in alcuni casi particolari.

-**metodo POST:** si usa di solito quando il client compila un form (ad esempio quello del motore di ricerca) → l'utente sta ancora richiedendo una pagina web al server, ma i contenuti specifici della pagina dipendono dall'input che l'utente ha messo nel form. Questo è caricato sul server proprio nell'"entity body".

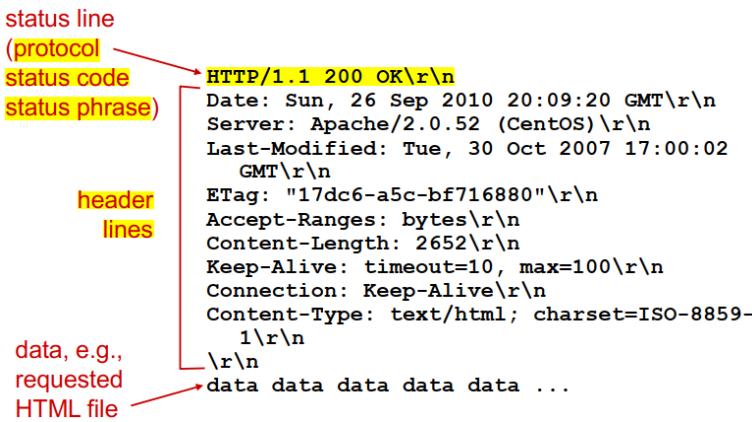
-**metodo URL:** usa a sua volta il metodo GET: l'input è caricato nel campo URL della prima riga (quella di richiesta).

Per quanto riguarda le *versioni*, distinguiamo:

-**HTTP/1.0:** include i metodi GET, POST, HEAD (chiede al server di lasciare l'oggetto richiesto senza risposta);

-**HTTP/1.1:** include i metodi GET, POST, HEAD, PUT (carica i file dell'entity body in base al percorso specificato nel campo URL), DELETE (cancella i files specificati nel campo URL);

Facciamo ora un esempio di **messaggio di risposta**:



Analizziamo lo **status code**, che compare nella prima riga (status line).

-**200 OK** → successo, l'oggetto richiesto si trova in fondo a questa risposta;

-**301 Moved Permanently** → l'oggetto è stato spostato, in fondo è indicata la nuova posizione;

-**400 Bad Request** → il server non ha compreso la richiesta;

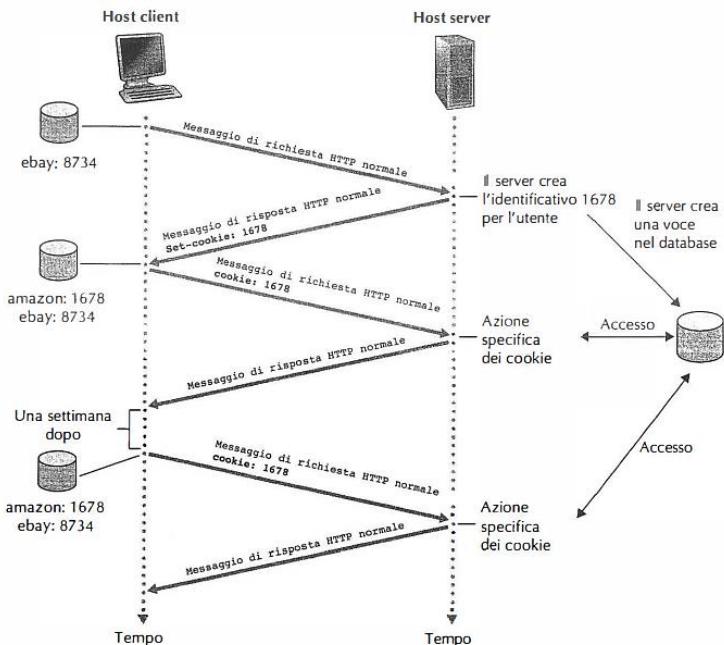
-**404 Not Found** → l'oggetto cercato non è su questo server;

-**505 HTTP Version Not Supported**;

## I COOKIES.

Sappiamo che i server HTTP sono *stateless* → la progettazione è più semplice e possiamo raggiungere prestazioni molto alte. Spesso però alcuni server necessitano di tenere traccia degli utenti per offrire contenuti in base alla loro identità → si usano i cookies [definiti in RFC 6265]. Questa tecnologia presenta **4 componenti**:

- Una riga di intestazione nel messaggio di risposta HTTP;
- Una riga di intestazione nel messaggio di richiesta HTTP;
- Un file mantenuto sul sistema dell'utente e gestito dal browser (una lista di cookie);
- Un database sul sito;



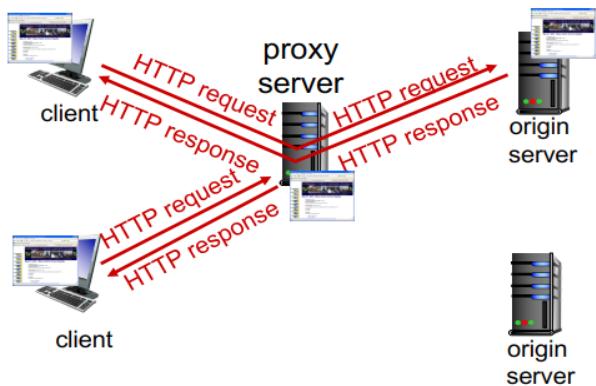
I cookie sono usati per concedere autorizzazioni, per gestire carrelli della spesa, per ottenere informazioni sulle preferenze di navigazione, per suggerire "proposte consigliate" agli utenti in base alle loro scelte precedenti. Vediamo ora questo esempio: Susan è già registrata su Ebay (ha un cookie per l'accesso), ma questa volta accede su Amazon per la prima volta. Il server di Amazon crea un nuovo ID per Susan (1678), e crea un accesso nel server. Il nuovo ID è comunicato a Susan con l'istruzione `set-cookie: 1678`, nell'intestazione del messaggio di risposta.

Il browser di Susan che lo riceve aggiunge una riga al file dei cookie che gestisce, contenente il nome del server di Amazon e il numero comunicato con `set-cookie`. Ogni nuova richiesta alla pagina di Amazon conterrà nell'intestazione del messaggio di richiesta

**Cookie: 1678** → in questo modo Amazon può tenere traccia dell'attività di Susan sul sito. Se Susan fornisce dati personali, Amazon aggiorna il suo database mettendoli in corrispondenza del numero 1678.

Tramite i cookie Amazon può tenere una lista dei prodotti messi nel carrello affinché un utente possa pagare tutto insieme alla fine, può fornire l'"one-click shopping" grazie ai dati registrati nel database, può usare le informazioni raccolte per indagini di mercato e per collezionare "feedback" sui suoi servizi.

## WEB CACHES O PROXY SERVER.

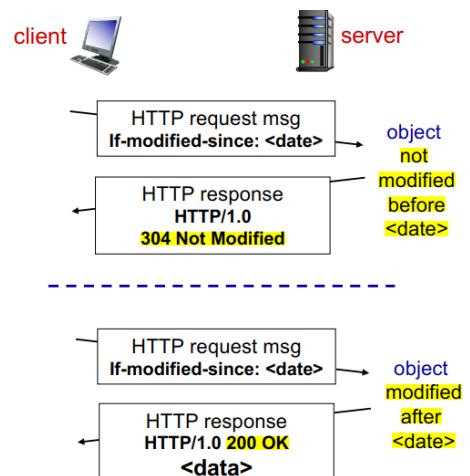


Obiettivo: soddisfare le richieste dei client senza consultare il server vero e proprio.

L'utente setta il browser in modo che invii tutte le richieste alla **cache** → se l'oggetto è in cache, viene ritornato al client. In caso contrario, la cache richiede l'oggetto al server, lo ottiene e lo ritorna al client (il proxy server agisce quindi sia da client che da server!). In genere, questo tipo di caches sono installate da ISP quali compagnie, università e organizzazioni. Quali sono i vantaggi? Si riduce il tempo di risposta (RTT), si riduce il traffico su un link di accesso istituzionale e, come nel caso di P2P, anche content provider "deboli" (non aventi cioè server enormi a propria disposizione) possono distribuire efficacemente i loro contenuti.

-Caso particolare: il **GET condizionale** → il server non invia l'oggetto al proxy se questo ha una versione più recente dell'oggetto (si eliminano i ritardi di trasmissione e diminuisce il traffico sul link).

- **Cache:** specifica la data dell'ultima modifica dell'oggetto che contiene, compilando il messaggio di richiesta HTTP in questo modo: `If-modified-since:<date>`;
- **Server:** la risposta contiene `304 Not Modified` se prima di `<date>` non ci sono state altre modifiche. Altrimenti ritorna `200 OK <data>`, con allegato l'oggetto più recente.

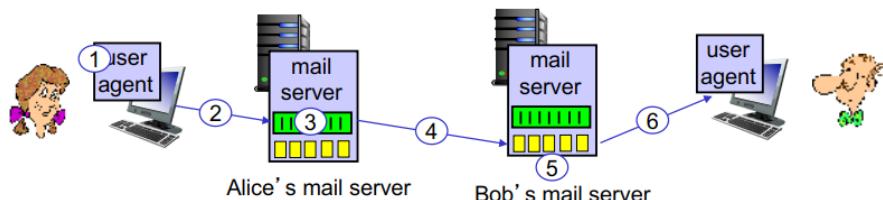


## SERVIZIO E-MAIL.

Tre componenti principali: uno “**user agent**”, un **mail server**, e il protocollo **SMTP** (Simple Name Transfer Protocol, RFC 2821). Per user agent intendiamo un lettore di e-mail, che l’utente usa per scrivere, editare e leggere e-mail.

Il mail server, invece, memorizza le mail in entrata e in uscita. Anch’esso è costituito da 3 parti: una *mailbox* (che contiene i messaggi in arrivo per l’utente), una *coda dei messaggi in uscita* e l’*SMTP* che mette in comunicazione diversi mail server. Anche la posta elettronica è un servizio di *tipo client-server*.

L’*SMTP* usa solamente TCP per trasferire messaggi in modo affidabile, nella porta 25, dopo il three-way- handshake. [N.d.A: Vedi l’esempio di conversazione tra client e server in fondo al pdf].



Nella foto, Alice usa il suo user agent per scrivere un’e-mail indirizzata a Bob, e la invia sul suo mail server. Il messaggio entra nella coda dei messaggi in uscita, e raggiunge dopo un certo tempo il mail server di Bob,

depositandosi nella sua mailbox. Tramite il suo user agent, Bob scarica la mail e la legge, usando i protocolli POP e IMAP, spiegati dopo.

I **messaggi SMTP** sono scritti in codifica 7-bit-ASCII → L’HTTP è un protocollo *pull*: qualcuno carica informazioni sul web server, e gli utenti usano HTTP per attirarli a sé. Ogni oggetto è incapsulato nel suo messaggio di risposta. Il *SMTP* è invece un protocollo *push*: è il client che spinge i file verso il server, non il contrario. Inoltre, oggetti multipli sono inviati in un singolo messaggio.

Vediamo il formato del messaggio di posta elettronica [RFC 882] → abbiamo una parte **header** che contiene i campi To:, From:, Subject:, e una parte **corpo**, in cui è scritto il messaggio vero e proprio in codifica ASCII.

Oltre al *SMTP* abbiamo altri 2 protocolli fondamentali: il *POP* (Post Office Protocol, [RFC 1939]) e l’*IMAP* (Internet Mail Acces Protocol, [RFC 1730]).

Il **POP3** è un protocollo *stateless* che riguarda la gestione delle autorizzazioni e del download. Nella fase di autorizzazione, il client dichiara user e password, il server risponde con +OK, -ERR. Nella fase di transazione, il client può scegliere tra le seguenti operazioni: **list** (si ottiene una lista ordinata degli identificativi dei messaggi nel server), **retr X** (ritorna il messaggio che nella lista occupa la posizione X), **dele X** (cancella il messaggio in posizione X), **quit**.

Se il *POP* prevede il comando delete, è di tipo *download and delete* (una volta eliminata, l’e-mail è irrecuperabile). Se DELETE non è previsto, è di tipo *download and keep*.

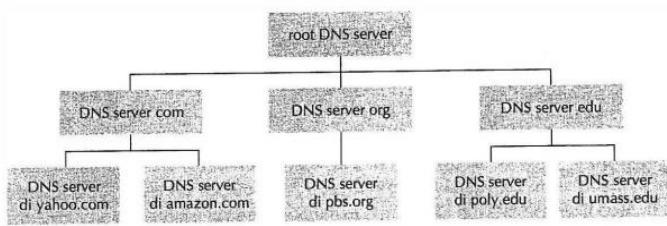
Il protocollo **IMAP** fornisce strumenti di manipolazione dei messaggi memorizzati nei server, di organizzazione in cartelle. È un protocollo che mantiene lo stato, in particolare mantiene la corrispondenza tra l’ID dei messaggi e le cartelle in cui sono conservati.

## DNS: DOMAIN NAME SYSTEM.

Un host può essere identificato con il suo hostname (ES: www.amazon.com) o con il suo IP. Le persone ricordano meglio il primo, mentre i router prediligono gli IP a lunghezza fissa e strutturati gerarchicamente. Al fine di conciliare i due approcci, il DNS traduce i nomi degli host nei loro indirizzi IP. Si tratta di (1) un **database distribuito** e implementato in una gerarchia di server DNS, e (2) di un **protocollo** di livello applicazione basato su *UDP* (porta 53), che consente agli host di interrogare il database. Vediamo i servizi principali:

- **Host aliasing:** un host dal nome complicato (nome canonico) può avere uno o più sinonimi, più facili da ricordare;
- **Mail server aliasing:** ad indirizzo e-mail canonico può essere associato un sinonimo più semplice da ricordare.
- **Distribuzione del carico di rete:** è possibile distribuire il carico tra server replicati.

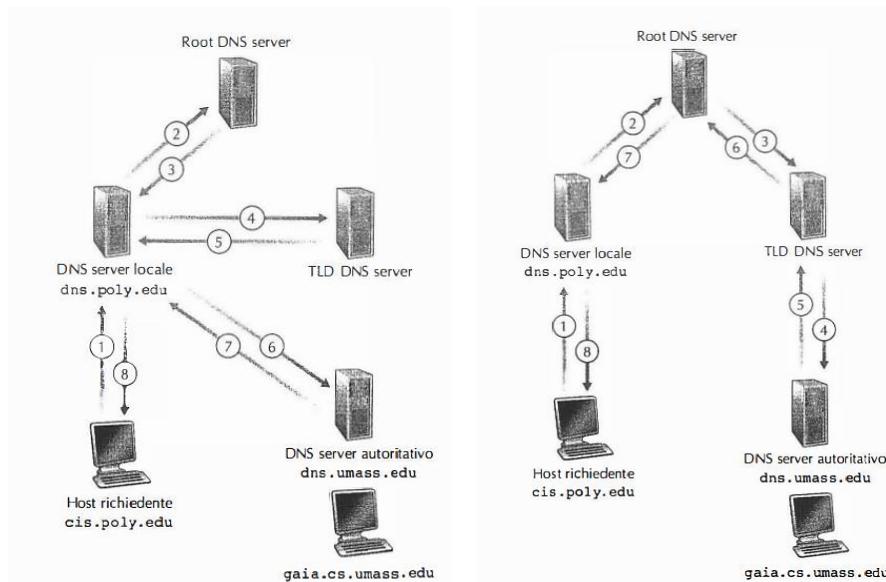
Perché non usare un unico server DNS centralizzato? Perché rappresenterebbe il *single point of failure*, dovrebbe sopportare un traffico troppo elevato, la manutenzione sarebbe difficile e costosa, e gli host "di periferia" sarebbero raggiungibili solo con ritardi molto lunghi. In altre parole, non sarebbe scalabile. Proprio per questioni di scalabilità, invece, i DNS server sono organizzati in modo gerarchico.



Distinguiamo tre classi di DNS:

- **Root server**: in cima alla gerarchia, ne esistono 13 in tutto l'Internet, sono i server contattati per primi dai local name server che necessitano di risolvere un nome. A sua volta il root server, se non conosce la traduzione di quel nome, contatta gli autoritativi, ottiene il nome, e lo restituisce al local name server.
- **top-level domain (TLD)**: si occupano dei domini di primo livello (.com,.org,.net,.edu,.gov) e dei domini dei vari paesi (.it,.fr,.uk...);
- **server autoritativi**: ogni organizzazione dotata di host pubblicamente accessibili tramite internet fornisce anche DNS server che permettano agli utenti di accedere pubblicamente a questi host, fornendo loro il corretto IP. L'organizzazione stessa può scegliere di mantenere da sola il proprio server autoritativo, oppure può pagare un fornitore i servizi per ospitare questi record su un suo server.
- **Local DNS**: non fa idealmente parte della gerarchia, ogni ISP ne ha uno (si tratta del "default name server"), possiede una cache, che funziona come un proxy server.

Due esempi di **risoluzione** di un nome: per iterazione e per ricorsione.



Il DNS sfrutta il caching per migliorare le prestazioni generali: dopo aver ricevuto una risposta contenente un mapping "hostname → IP", la mette in cache. In una richiesta per quell'hostname, il server con cache può fornire direttamente l'IP corrispondente anche se non è il server autoritativo per tale indirizzo. Dal momento che le associazioni hostname-IP non sono permanenti, i DNS invalidano gli indirizzi tenuti in cache per più di due giorni (*out-of-date*). Questi meccanismi sono definiti nel [RFC 2163].

I server che implementano il database distribuito dei DNS memorizzano i RR (record delle risorse o resource records).

Un RR ha il seguente formato: (name, value, type, TTL).

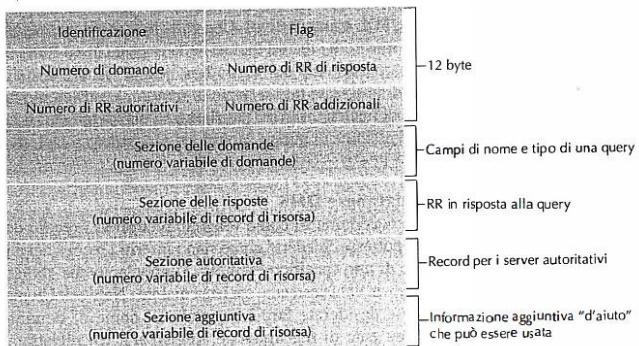
- TTL è il time to leave;
- Name e Value assumono diversi significati in funzione del tipo;
- Se Type= A: name è il nome dell'host, value il suo IP → fornisce la corrispondenza tra hostname e IP.  
ES: (foo.com, 145.37.93.126, A);

- Se Type=**NS**: *name* è un dominio, *value* è l'hostname del DNS autoritativo che sa come ottenere gli IP del dominio → viene usato per instradare le richieste DNS successive alla prima nella concatenazione delle query. ES: (foo.com, dns.foo.com. NS). → “per conoscere gli IP di foo.com chiedi al server autoritativo dns.foo.com”;
- Se Type=**CNAME**: *value* rappresenta il nome canonico dell'host per il sinonimo *name* (servizio di Host aliasing).  
ES: (foo.com, relay1.bar.foo.com, CNAME);
- Se Type=**MX**: *value* rappresenta il nome canonico del mail server che ha il sinonimo *name* (servizio di Mail server aliasing). ES: (foo.com, mail.bar.foo.com, MX).

**NOTE:** 1) Un DNS autoritativo per un certo hostname conterrà un record di tipo A per quell' host name. 2) Se invece il DNS NON è quello autoritativo per l'hostname: potrebbe contenere un record A in cache, OPPURE contiene un record NS per il dominio che include l'hostname (di conseguenza conterrà anche il record A, che fornisce l'IP del DNS inserito nel campo value del RR NS).

ES: host → gaia.cs.umass.edu. Un TLD non autoritativo conterrà un RR per un dominio che include l'host, quindi cs.umass.edu. Il record RR sarà: (umass.edu, dns.umass.edu, NS). Il TLD ha inoltre un RR di type A che associa l'IP al DNS server specificato nel RR NS, quindi (dns.umass.edu, 128.119.40.111, A).

Vediamo ora **il formato dei messaggi DNS** (uguale per query e risposte).



- *L'identificazione* è un numero a 16 bit, una sorta di “ID della transazione” → ad una query con un certo identificatore corrisponderà univocamente una risposta con lo stesso identificatore;
- *Flags*: si tratta di una query o di una risposta? Si desidera la ricorsione o l'iterazione? La ricorsione è disponibile? La risposta viene da un DNS autoritativo?
- *L'intestazione* termina con 4 campi che iniziano con “numero di”, che indicano il numero di occorrenze delle 4 sezioni di tipo dati che seguono l'intestazione;
- *La sezione delle domande* contiene informazioni sulle richieste che stanno per essere effettuate. Include un campo “nome richiesto” e il type della domanda.

- *La sezione delle risposte* contiene i RR inviati dal DNS server contattato. Una singola risposta può avere più RR, dato che ad un hostname possono corrispondere più IP;
- *La sezione autoritativa* contiene i record dei server autoritativi;
- Infine, abbiamo una serie di *informazioni* aggiuntive.

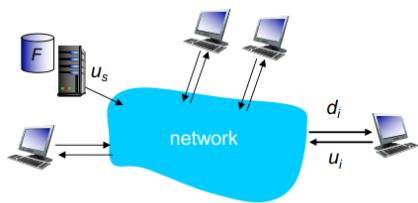
### Attaccare un DNS:

- Attacchi di tipo **DDoS** → bombardare i root server di traffico (poco dannoso, perché la gerarchia e il caching permettono di distribuire il traffico in modo efficiente), oppure bombardare i TLD server (decisamente più pericoloso);
- Man in the middle → intercettazione delle query “in viaggio”;
- DNS poisoning → l'attaccante invia risposte fasulle, che sono inserite in cache → difficile da realizzare;
- Sfruttare l'infrastruttura del DNS per provocare DDoS → inviare molte richieste a DNS diversi, a nome di un unico bersaglio fasullo. Quando i DNS risponderanno in massa, se le risposte sono state amplificate (avranno cioè un numero di byte superiore a quello della query), il bersaglio sarà sommerso. Il successo di questi attacchi è comunque limitato.
- Conclusioni → il DNS è molto **robusto** e nessuno è stato ancora in grado di interromperne il servizio.

### CLIENT-SERVER VS. P2P.

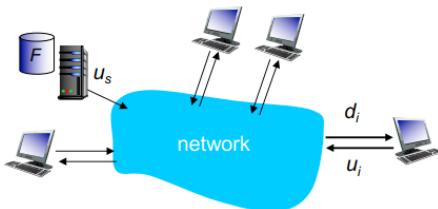
Quanto tempo impieghiamo per distribuire un file di dimensione F da un server a N peers, sapendo che la capacità di download, upload è limitata?

- Caso 1: client-server:



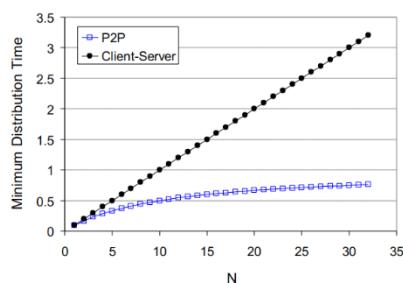
- Lato server: tempo per l'invio di una copia:  $F/u_s$ ; Tempo per inviare  $N$  copie:  $N \cdot F/u_s$ ;
- Lato client: ogni client deve fare il download del file  $\rightarrow d_{\min}$ : minimo tasso di download del client; il tempo minimo di download del client è  $F/d_{\min}$ ;
- Il tempo per distribuire il file  $F$  a  $N$  client è  $D \geq \max \{N \cdot F/u_s, F/d_{\min}\}$ . Notiamo che la prima componente cresce linearmente al numero  $N$  di host.

- Caso 2: P2P:



- Lato server: bisogna trasmettere come minimo una copia:  $F/u_s$ ;
- Lato client: ogni client deve fare il download di una copia: il tempo minimo di download del client è ancora  $F/d_{\min}$ ;
- Insieme dei client: devono fare il download di  $N \cdot F$  bits in totale  $\rightarrow$  il massimo tasso di upload (cioè l'upload che può essere effettuato da tutti i client che hanno già scaricato la loro copia di  $F$ ) è  $u_s + \sum u_i$ .

- Otteniamo un tempo totale di  $D \geq \max \{F/u_s, F/d_{\min}, N \cdot F/u_s + \sum u_i\}$ .

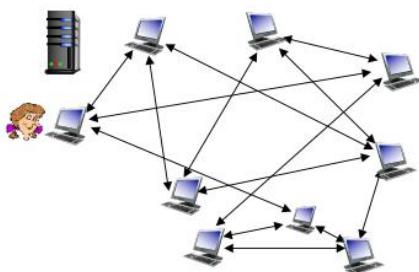


Anche in questo caso abbiamo una componente (la terza) che cresce linearmente in  $N$ . Ma lo stesso fa il suo denominatore, dal momento che ogni peer aumenta la capacità complessiva del server (mette a disposizione le sue risorse di calcolo e viene usato come un server). Di conseguenza, come si vede dal grafico, il minimo tempo di distribuzione di  $F$  tra gli  $N$  peers diminuisce con l'aumentare di  $N$ , a differenza della retta client-server che cresce linearmente in  $N$ . **La distribuzione file è dunque più efficiente con architettura P2P.**

## SERVIZI CON ARCHITETURA P2P.

(Distribuzione di file, Streaming video, VoIP. Noi vedremo solo i primi due).

- Distribuzione dei file tramite BitTorrent.



Definiamo Torrent un gruppo di peers che si scambiano *chunks* (ognuno di dimensione 256Kb), in una rete P2P. Un sistema Torrent è dotato di un tracker, che mantiene e aggiorna una lista dei peers che stanno prendendo parte al torrent. Un client che si aggiunge al torrent (Alice) sarà inizialmente sprovvisto di chunks; ottiene la lista dal tracker, entra in connessione con un sottoinsieme di peers della lista (con quelli a lei "vicini") e inizia a scambiare file. Mentre effettua il download, mette a disposizione in upload i chunks appena ottenuti. Dal momento che alcuni peer potrebbero abbandonare il torrent una volta ottenuto il file completo, i peers da cui un client ottiene chunks variano nel tempo.

In un dato istante, peer diversi hanno chunk diversi dello stesso file: periodicamente, Alice ottiene da ogni peer la lista dei chunk che possiede, e richiede i pezzi che le mancano partendo dal più raro al più comune.

- invio dei chunk *tit-for-tat*  $\rightarrow$  Alice è collegata con un sottoinsieme dei peer del torrent: con chi effettua lo scambio? Sceglie i 4 che le stanno mandando chunks alla velocità maggiore, e "strozza" tutti gli altri. Ogni 10 secondi questa top 4 è aggiornata, e ogni 30 sec è scelto randomicamente un altro peer, che si unisce alla top 4.

- Video Streaming e CDN

Il traffico video è il maggior consumatore di banda. Come rendiamo scalabile un sistema streaming? E soprattutto, come possiamo gestire le differenti prestazioni/capacità degli host?  $\rightarrow$  ci serve un'infrastruttura di livello applicazione distribuita e non centralizzata.

**Def:** Un video è una sequenza di immagini mostrate su schermo ad un tasso costante (es. 24 immagini al secondo);

**Def:** Un' immagine digitale è un array di pixels, ognuno rappresentato da bit.

**Def:** Codifica con *ridondanza* -> con la ridondanza interna (spaziale), anziché inviare N bit che hanno lo stesso valore (quindi lo stesso colore), inviamo solamente 2 valori: quello del colore e il numero di volte in cui il colore è ripetuto. Nella ridondanza tra immagini (temporale), dato il frame i, anziché inviare il frame i+1 completo inviamo solo le differenze tra i+1 e i (che normalmente sono poche tra due frame consecutivi).

-**CBR:** constant bit rate → il tasso di codifica è fisso; **VBR:** variable bit rate → il tasso di codifica cambia al variare della quantità di dati necessari per la codifica.

*ES: MPEG 1 (CD-ROM): 1.5 Mbps, MPEG2 (DVD): 3-6 Mbps, MPEG4 (usato Internet, < 1 MBPS).*

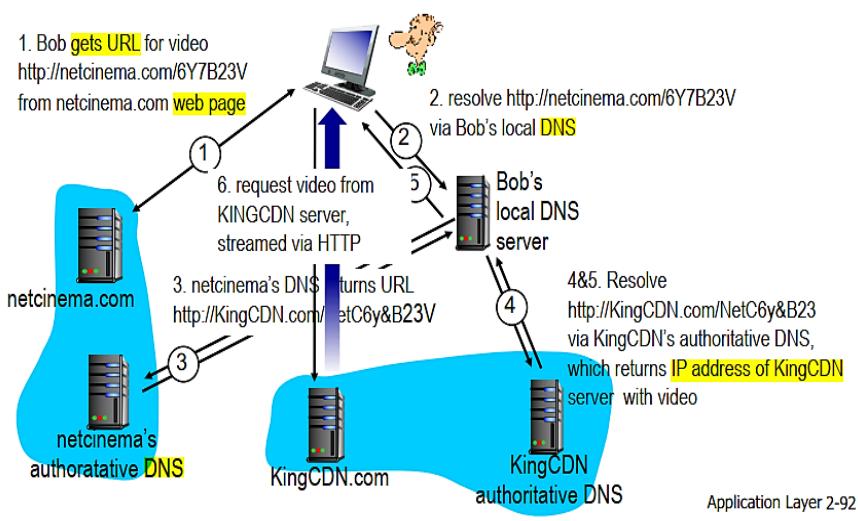
### DASH: Dynamic Adaptive Streaming over HTTP.

- *Server:* divide il file in tanti chunks (ognuno dei quali è memorizzato, codificato a velocità differenti), costruisce un *manifest file* (che assegna un URL per ogni chunk);
- *Client:* misura la banda disponibile tra server client a intervalli costanti, consulta il manifesto, richiede un chunk la volta → in base alla banda disponibile sceglie il massimo tasso di codifica possibile. Può scegliere tassi di codifica diversi in diversi intervalli di tempo. Si evince subito che in questa forma di comunicazione è il client che detiene l’”intelligenza”, dal momento che è lui a decidere: quando richiedere il chunk per evitare *overflow* o *starvation*, a che tasso di codifica richiederlo, a che server richiedere quel chunk (tenendo conto della vicinanza del server e della banda che ha a disposizione).

Come nel caso del Web server, un ipotetico mega-server unico e centralizzato non scala. L'unica strategia possibile è quella di distribuire più copie dello stesso video in siti distanziati geograficamente (che insieme formano la Content Distribution Network, o CDN). Possiamo farlo in 2 modalità:

- **Enter deep:** proposta da Akamai, si entra profondamente nelle reti di accesso degli ISP installando gruppi di server (cluster) negli ISP di accesso sparsi in tutto il mondo. L'obiettivo è quello di essere vicini agli utenti in modo da diminuire il ritardo percepito dall'utente e il throughput, diminuendo il numero di collegamenti tra utente finale e cluster CDN da cui riceve i contenuti. La manutenzione e gestione dei cluster è molto complicata.
- **Bring home:** si porta a casa l'ISP costruendo grandi cluster in pochi punti chiave e interconnetterli usando una rete privata ad alta velocità. Invece di entrare negli ISP di accesso queste CDN pongono ogni cluster in un luogo vicino ai PoP. Questo approccio è più facile da gestire, ma garantisce una qualità di servizio inferiore.

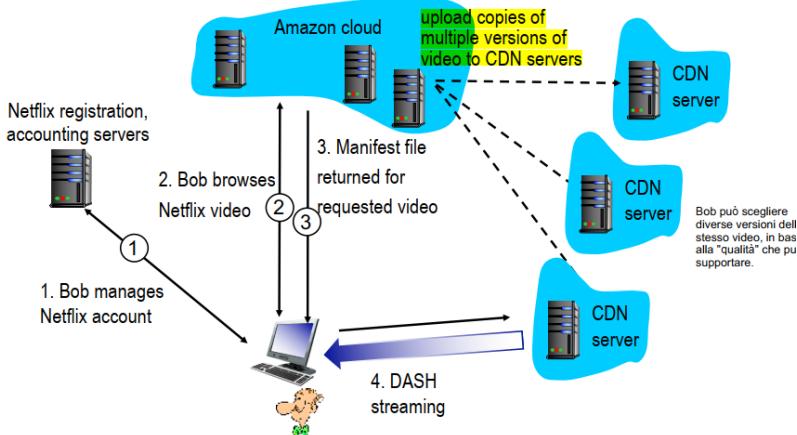
Il CDN memorizza copie del contenuto ai suoi nodi; il client che necessita di una copia, la richiede al CDN più vicino. Può scegliere diverse copie se la rete è congestionata.



Questo è un esempio di **accesso ai contenuti** di un CDN.

-il video richiesto da Bob è <http://netcinema.com/6Y7B23V>, che è memorizzato in <http://KingCDN.com/NetC6y&B23V>.

-Il DNS autoritativo di netcinema decide su quale server della CDN verrà scaricata la copia (in base alla provenienza della richiesta, il DNS consulta un mapping delle reti di accesso locali e dei CDN loro vicini), e lo comunica al local server di Bob.



### Caso particolare: Netflix.

Dopo che Bob ha effettuato l'accesso, cerca il video tramite browser. Il cloud Amazon carica tante copie dello stesso video in diversi CDN, e fornisce a Bob il manifest dei server CDN che contengono la copia di quel video. A questo punto Bob può effettuare uno streaming DASH, scegliendo la versione che preferisce in base alle capacità dei suoi dispositivi.

### ESEMPIO DI SCAMBIO DI POSTA ELETTRONICA TRA CLIENT E SERVER CON PROTOCOLLO SMTP:

Il client comunica con le seguenti parole chiave: **HELO**, **MAIL FROM**, **RCPT TO**, **DATA**, **QUIT**, il server risponde con il **numero di protocollo** utilizzato per la risposta.

```
flora:~> telnet sharpless.cs.unibo.it 25
Trying 2001:760:2e00:f001::25...
Connected to sharpless.cs.unibo.it.
Escape character is '^>'.
220 sharpless.cs.unibo.it ESMTP Postfix (Debian/GNU)
HELO flora.cs.unibo.it
250 sharpless.cs.unibo.it
MAIL FROM: <leonardo.davinci@pippo.it>
250 2.1.0 Ok
RCPT TO: <luciano.bononi@unibo.it>
250 2.1.5 Ok
```

- HELO: dichiara l'IP del mail client che sta facendo richiesta di invio di un'e-mail (in questo caso è la macchina flora). → la risposta 250 è un numero di protocollo che indica "risposta affermativa";
- MAIL FROM: digita chi è che spedisce l'e-mail. In questo caso sceglio volutamente un indirizzo fake; come prima, si ha risposta affermativa del server (250 2.1.0 Ok);
- RCPT TO: digita chi deve ricevere il messaggio; risposta affermativa da parte del server.

```
DATA
354 End data with <CR><LF>.<CR><LF>
From: Leonardo Da Vinci <leonardo.davinci@pippo.it>
To: Luciano Bononi <luciano.bononi@unibo.it>
Subject: Che cavolo avete inventato? email?????

Ciao Luciano

qui nell'adilà fa un po' freddo.....
mi spieghi che cosa è questa email?
Come mai non ci avevo pensato io prima?

Ciao
Leo

.
250 2.0.0 from MTA(smtp:[127.0.0.1]:10025): 250 2.0.0 Ok: queued as 53101541A
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
```

- DATA: tutto quello che viene scritto dopo DATA sarà il vero testo del messaggio. Includere anche dati di intestazione (from-to-subject); Il server ci ricorda che per chiudere il campo messaggio basterà mettere un punto in una riga vuota e premere invio.

```
From: [...]
To: [...]
Subject: [...]

[--testo del messaggio--]
```

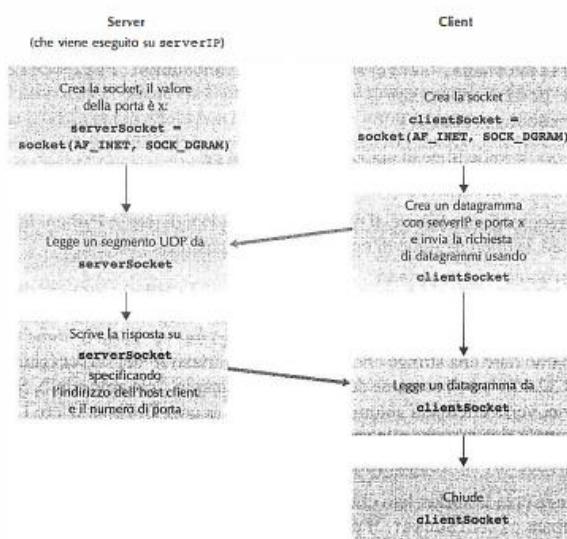
- Il server risponde che il messaggio è inserito in coda, con ID di accodamento 53101541A. Il messaggio sta viaggiando in Internet, e si andrà a depositare nella mailbox del mail server di Buciano.

- QUIT: chiudiamo la connessione, il server risponde con il protocollo di chiusura (221 2.0.0 Bye), e la connessione viene abbattuta.



## PROGRAMMAZIONE SOCKET.

### 1) Caso con connessione UDP:



### Python Udp client:

```

from sokcket import *          //importiamo il modulo socket;
serverName = 'hostname'       //assegniamo alla variabile di tipo stringa serverName il nome dell'host (indirizzo IP);
serverPort = 12000             //assegna alla variabile intera serverPort il numero 12000;
clientSocket = Socket(socket.AF_INET, socket.SOCK_DGRAM) //crea il socket lato client, memorizzandone il riferimento al clientSocket; il primo parametro indica la famiglia di indirizzi (AF_INETI = IPv4), il secondo che il socket è di tipo SOCK_DGRAM (UDP e non TCP). Il numero di porta non è specificato (lo aggiungerà da solo il s.o.); 
message = raw_input ('Frase in minuscolo:') //L'utente può digitare da tastiera una stringa, assegnata a message;
clientSocket.sendto(message, (serverName, serverPort)) //sendto() attacca l'indirizzo di destinazione (serverName, serverPort) al messaggio e invia il pacchetto nella clientSocket;
modifiedMessage, serverAddress = clientSocket, recvfrom (2048) //quando un pkt arriva da internet al socket client, i dati contenuti vengono assegnati a modifiedMessage, mentre l'indirizzo sorgente del pkt è messo in serverAddress (che contiene sia l'IP che il numero di porta del server). recvfrom prende in input la grandezza del buffer, che è 2048;
print modifiedMessage      //mostra modified message sul display dell'utente;
clientSocket.close()       //chiude il socket, il processo termina.
  
```

### Python UDP server:

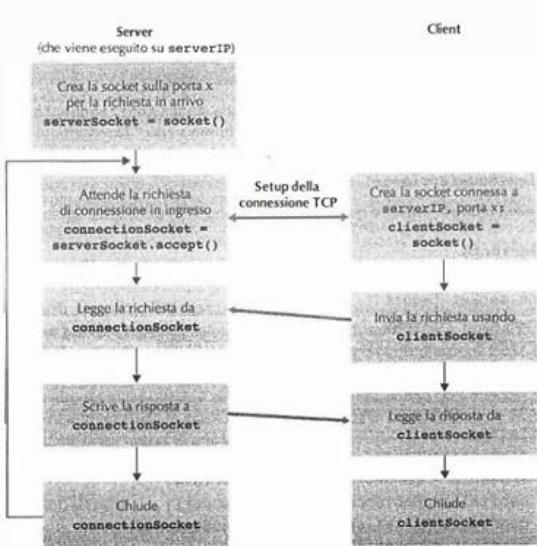
```

from socket import *
serverPort = 12000
serverSocket = socket (AF_INET, SOCK_DGRAM)
serverSocket.bind((" ", serverPort)) //assegna al socket lato server il numero di porta 12000 → quando qualcuno spedirà un pacchetto a questo numero di porta, lo manderà in questo socket;
print "il server è pronto a ricevere"
  
```

**while 1:**

```
message, clientAddress = serverSocket.recvfrom(2048) //quando un pacchetto arriva al socket lato server i dati sono messi in message, e l'indirizzo sorgente a clientAddress (IP e porta del client);
modifiedMessage = message.upper() //uppercase del messaggio inviato dal client;
serverSocket.sendto(modifiedMessage, clientAddress) //attacca l'indirizzo del client (IP e porta) al messaggio in maiuscolo e invia il pkt al socket lato server.
```

## 2) Caso con connessione TCP:



### Python TCP Client:

```
From socket import *
serverName = 'servername'
serverPort = 12000
clientSocket = socket (AF_INET, SOCK_STREAM) //creazione socket lato client; SOCK_STREAM indica connessione TCP. Il numero di porta lato client è aggiunto dal s.o;
clientSocket.connect ((serverName, serverPort)) //connect inizializza una connessione TCP tra client e server (handshake a 3 vie);
sentence = raw_input ('Frase in minuscolo: ')
clientSocket.send(sentence) //invia la stringa sentence attraverso il socket del client alla connessione TCP. Non viene creato un pacchetto in modo esplicito che contiene l'indirizzo di destinazione, come in UDP. Il client "lascia cadere" i suoi dati in TCP e attende la risposta del server;
modifiedSentence = clientSocket.recv(1024) //i caratteri arrivati dal server sono assegnati alla stringa modifiedSentence;
print 'Dal server: ', modifiedSentence //mostra sullo schermo la frase modificata;
clientSocket.close()
```

### Python TCP server:

```
from socket import *
serverPort = 12000
serverSocket = socket (AF_INET, SOCK_DGRAM)
serverSocket.bind((" ", serverPort)) //in TCP, serverSocket è una "porta di benvenuto";
serverSocket.listen(1) //il server si mette in ascolto e attende richieste. Il parametro indica il max num. di connessioni da tenere in coda;
print 'Il server è pronto a ricevere'
while 1:
    connectionSocket, addr = serverSocket.accept() //accept() blocca temporaneamente l'esecuzione del programma finchè il client bussa alla porta di benvenuto. Quando l'esecuzione riprende, viene creato un nuovo socket nel server (connectionSocket), dedicato allo specifico client che ha bussato. A questo punto l'handshake è completo;
    sentence = connectionSocket.recv(1024) //non resta che scambiarsi byte;
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence)
    connectionSocket.close()
```

# LIVELLO TRASPORTO

A livello applicazione sono presenti due protocolli: **UDP**, che fornisce alle applicazioni un servizio non affidabile e non orientato alla connessione, l'altro è **TCP**, che offre un servizio affidabile e orientato alla connessione.

A livello trasporto il pacchetto verrà chiamato segmento.

Il protocollo a livello rete è IP, che fornisce comunicazione logica tra gli host. Il suo modello prende il nome di best-effort, in quanto IP fa di tutto per consegnare i segmenti tra host comunicanti, ma non offre garanzie, poiché non assicura l'integrità dei pacchetti o la loro consegna in ordine sequenziale. Per questo IP è un servizio non affidabile.

I protocolli UDP e TCP lavorano in modo tale da estendere il servizio di consegna host-to-host a process-to-process. Questo passaggio prende il nome di **multiplexing e demultiplexing a livello trasporto**.

## MULTIPLEXING E DEMULTIPLEXING

Il livello trasporto ha il compito di consegnare i dati dei segmenti al processo applicativo appropriato in esecuzione nell'host. Dato che un processo può gestire più socket, il livello trasporto nell'host di ricezione non manda i dati direttamente al processo ma ad una socket intermedia. Siccome ci possono essere più socket nell'host di ricezione, ognuna di esse avrà un identificatore univoco che ci farà capire se si tratti di UDP o TCP.

Immaginiamo ora che l'host mittente indirizzi verso la giusta socket il segmento da mandare. Ogni segmento al suo interno presenta diversi campi. Nel lato destinatario, il livello trasporto esamina questi campi per trovare la socket corrispondente e quindi vi manda il segmento. Il demultiplexing si occupa di trasportare i dati dei segmenti a livello trasporto verso la giusta socket. Mentre il compito di radunare frammenti di dati da diverse socket sull'host di origine e incapsulare ognuno con intestazioni a livello trasporto spetta al multiplexing.

Affinchè il multiplexing funzioni abbiamo bisogno che:

- Le socket abbiano identificatori unici;
- Ogni segmento abbia i campi che indichino la socket a cui va consegnato il segmento.

I campi sopra citati sono:

- Il campo **numero di porta di origine**;
- Il campo **numero di porta di destinazione**.

Questi numeri di porta hanno dimensione pari a 16 bit, quelli che vanno da 0 a 1023 sono detti numeri di porta noti (utilizzati per HTTP e FTP).

Quando una socket UDP viene definita come:

```
clientSocket = socket(socket.AF_INET, socket.SOCK_DGRAM)
```

il livello trasporto le assegna un numero di porta casuale e non ancora utilizzato tra 1024 e 65535. Altrimenti se si vuole assegnare un numero di porta ben preciso, bisogna introdurre il comando:

```
clientSocket.bind (('',19157)) .
```

In generale, il lato client assegna un numero casuale, mentre il lato server uno specifico.

Possiamo quindi dire che una socket UDP viene identificata tramite una coppia formata da indirizzo IP e numero di porta di destinazione.

La differenza tra socket UDP e socket TCP è che quest'ultima viene identificata da 4 parametri:

1. Indirizzo IP di origine;
2. Numero di porta di origine;
3. Indirizzo IP di destinazione;
4. Numero di porta di destinazione.

Con l'utilizzo del demultiplexing, il ricevente utilizza tutti e 4 i valori per dirigere il segmento verso la giusta socket. L'host server può ospitare più socket TCP contemporaneamente collegate a diversi processi, ognuna delle quali sempre identificata da 4 valori. Contrariamente, i web server hanno socket diverse per ogni client.

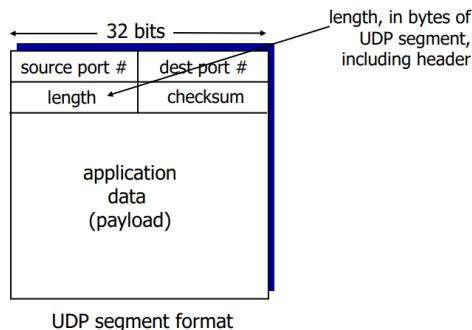
### TRASPORTO NON ORIENTATO ALLA CONNESSIONE: UDP

Il protocollo UDP non è orientato alla connessione, ossia non effettua nessun handshaking.

Perché si sceglie UDP invece che TCP?

- Controllo più fine a livello applicazione su quali dati sono inviati e quando;
- Nessuna connessione stabilita;
- Nessuno stato di connessione;
- Minor spazio usato per l'intestazione del pacchetto.

Nei segmenti UDP i dati dell'applicazione occupano l'ultimo campo. L'intestazione UDP ha solamente quattro campi da 2 byte ciascuno.



Il numero di porta consente all'host di destinazione di mandare i dati al processo corrispondente. Il campo lunghezza indica la dimensione del segmento.

L'host ricevente fa utilizzo del checksum per vedere se si sono verificati degli errori all'interno del segmento.

### CHECKSUM

Il checksum è un meccanismo che fa la somma a complemento ad 1.

In ricezione, si ricalcola la somma normale, si somma ad essa il checksum e se il risultato sarà 1111111111111111, allora non ci sono errori, altrimenti se ci fosse un bit pari a zero, allora il segmento potrebbe presentare errori.

UDP mette a disposizione il checksum in quanto non ci sono garanzie che tutti i collegamenti tra origine e destinazione controllino gli errori. In più, sappiamo che il protocollo UDP non è affidabile e non rileva errori in memoria, per questo motivo mette a disposizione questo meccanismo end-to-end per rilevare eventuali errori.

Nonostante questo però, UDP non fa nulla per correggere gli errori trovati, anzi, scarta il pacchetto alterato.

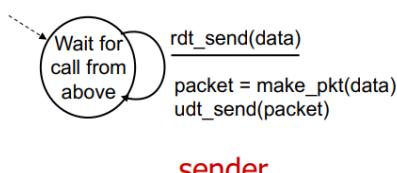
### TRASFERIMENTO AFFIDABILE

Invocando un protocollo che offre servizio affidabile, come TCP, esso sarà affidabile a livello applicazione e a livello collegamento, ma non c'è garanzia che lo sia anche a livello rete.

Vediamo ora come costruire un protocollo di trasferimento dati affidabile.

- Canale perfettamente affidabile: rdt1.0

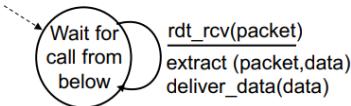
Introduciamo la **macchina a stati finiti (FSM)**. Esistono due diverse FSM, una per il lato mittente, l'altra per il destinatario.



Con `rdt_send(data)` si accettano i dati.

Il comando `packet=make_pkt(data)` crea un pacchetto di dati.

Infine `udt_send(packet)` invia il pacchetto sul canale.



Con `rdt_rcv(packet)` si raccolgono i pacchetti.  
Il comando `extract (packet,data)` rimuove i dati dai pacchetti.  
In ultimo, `deliver_data(data)` passa i pacchetti al livello superiore.

## receiver

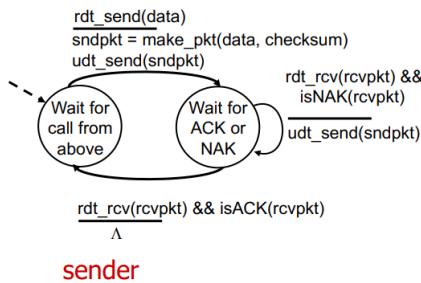
- Canale con errori sui bit: rdt2.0

Consideriamo ora il caso in cui i bit di un pacchetto possono essere corrotti. Assumiamo che i pacchetti vengono ricevuti in ordine sequenziale.

Per risolvere questo problema, si utilizza un protocollo che usa delle **notifiche (acknowledgement) positive** o **notifiche negative**. Questi messaggi consentono al destinatario di far sapere al mittente se il pacchetto è corretto o meno, richiedendo, in questo caso, di rimandarlo. I protocolli di trasferimento dati affidabili basati sulla ritrasmissione vengono chiamati **protocolli ARQ**.

Tali protocolli hanno 3 funzioni aggiunte:

1. Rilevamento degli errori: è necessario un meccanismo in grado di trovare eventuali errori. Ad esempio, UDP fa uso del checksum;
2. Feedback del destinatario: dato che mittente e destinatario operano su sistemi periferici diversi, l'unico modo che ha il mittente di capire se il pacchetto mandato è corretto o meno è ricevere una notifica, positiva (**ACK**) o negativa(**NAK**), dal destinatario;
3. Ritrasmissione: se un pacchetto presenta errori sarà ritrasmesso dal mittente.



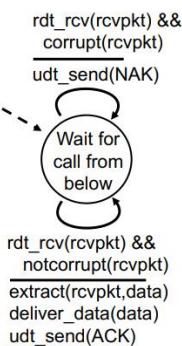
A livello mittente sono presenti due stati. Nella parte a sinistra si sta aspettando di ricevere i dati.

Quando si verifica `rdt_send(data)`, si crea un pacchetto con `snpkt` contenente i dati e il campo per il checksum.  
Infine si spedisce il pacchetto con `udt_send(snpkt)`.

Nello stato di destra, il mittente sta aspettando un ACK o un NAK.  
Se riceve un ACK (`rdt_rcv(rcvpkt)&& isACK(rcvpkt)`) allora il pacchetto non presenta errori, altrimenti se riceve un NAK (`rdt_rcv(rcvpkt) )&& isNAK(rcvpkt)`) allora il pacchetto presenta errori e dovrà rimandarlo.

Se il mittente è in attesa di un ACK o un NAK non potrà mandare dati, perciò si dice che è un protocollo **stop-and-wait**.

## receiver



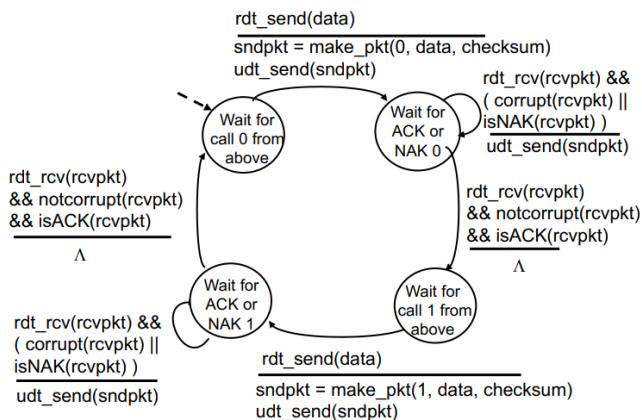
Il lato ricevente ha solo uno stato. All'arrivo del pacchetto il destinatario risponde con un ACK o con un NAK.

Con `rdt_rcv(rcvpkt)&& corrupt(rcvpkt)` indichiamo un pacchetto alterato.

E' possibile che l'ACK o il NAK possono essere corrotti a loro volta. Per risolvere questo problema, il mittente rimanda il pacchetto dopo aver ricevuto una notifica alterata, ma facendo così si introducono pacchetti duplicati nel canale. Facendo questo, però, il destinatario non sa se l'ultimo ACK o NAK mandato sia stato ricevuto correttamente.

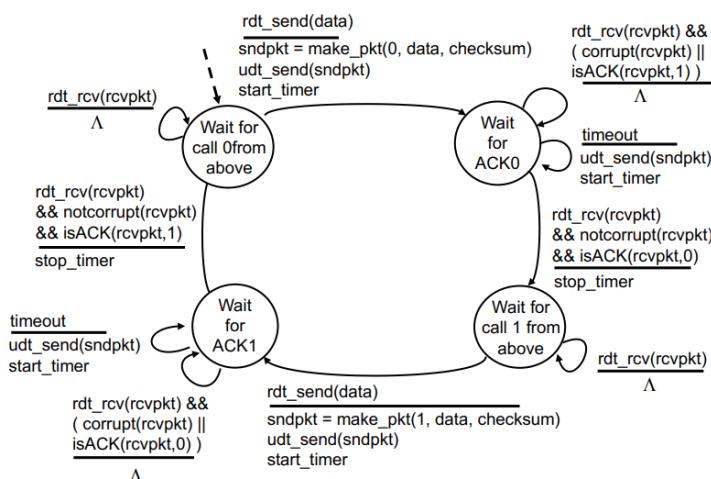
Soluzione: aggiungere un campo al pacchetto dati, numerando i pacchetti con un **numero di sequenza**.

- Canale con errori sui bit: rdt2.1 (versione corretta di rdt2.0)



pacchetto confermato dall'ACK.

- Canale con perdite ed errori sui bit: rdt3.0



Consideriamo il caso in cui il mittente manda un pacchetto al destinatario ma l'ACK di questo vada smarrito, così il mittente non saprà mai se il pacchetto sia arrivato o meno. Se il mittente è disposto ad aspettare un tempo sufficiente per essere certo della perdita, può rimandare il pacchetto. Si sceglie quindi un valore di tempo per il quale lo smarrimento risulti probabile, ma non sicuro. Se non si riceve l'ACK in questo lasso, il mittente rimanda il pacchetto. Può succedere, però, che il pacchetto presenti ritardo ma non sia andato perso, quindi il destinatario riceverà un duplicato (risolto con l'utilizzo dei numeri di sequenza). La ritrasmissione sembra l'unica soluzione.

Il mittente deve fare uso di un contatore che segnali la scadenza di un dato periodo di tempo:

1. Si inizializza il contatore quando si manda il pacchetto (anche se si tratta di una ritrasmissione)
2. Si risponde all'interrupt del timer
3. Si ferma il contatore

Il protocollo rdt3.0 viene anche chiamato **protocollo ad alternanza di bit**.

### TRASFERIMENTO AFFIDABILE CON PIPELINE

Il protocollo rdt3.0 è funzionale ma le sue prestazioni sono poco apprezzate.

L'utilizzo del mittente (o del canale) sarà la frazione di tempo in cui il mittente è stato veramente occupato nell'invio di bit sul canale.

Il protocollo stop-and-wait ha poco utilizzo del mittente, pari a:

Ora bisogna gestire anche il numero di sequenza. Il protocollo rdt2.1 usa acknowledgement positivi e negativi verso il mittente: positivo quando arriva un pacchetto fuori sequenza, negativo se è alterato. Se il mittente riceve due ACK per lo stesso pacchetto (ACK duplicati), capisce che il destinatario non ha ricevuto correttamente il pacchetto che viene dopo quello confermato due volte.

Come abbiamo visto prima, una differenza tra rdt2.1 e rdt2.2 è che ora si ha a che fare anche con i numeri di sequenza, per questo motivo il mittente deve controllare il numero di sequenza del

Vediamo ora un canale che, oltre a danneggiare i bit, possa anche perdere i pacchetti.

Il protocollo deve ora risolvere due problemi: come rilevare la perdita dei pacchetti e cosa fare quando succede. Per risolvere quest'ultimo problema, abbiamo già a disposizione il checksum, i numeri di sequenza, gli acknowledgement e la ritrasmissione.

Per quanto riguarda il primo problema, spetterà al mittente rilevare e risolvere la perdita dei pacchetti.

$$U_{\text{mittente}} = \frac{L/R}{RTT+L/R}$$

Questo ci fa capire che i protocolli di rete limitano il rendimento dell'hardware. In più, non abbiamo considerato i vari ritardi e l'accodamento che potrebbero verificarsi, che avrebbero solamente accentuato le prestazioni scadenti.

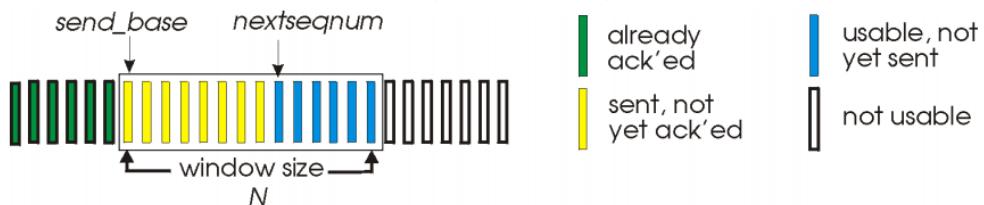
Per risolvere questo problema, si passa ad una tecnica detta **pipeling**, che permette di mandare più pacchetti senza dover aspettare gli ACK.

Esempi di pipeling sono: **Go-Back-N (GBN)** e **ripetizione selettiva**.

- **GO-BACK-N (GBN)**

Con questo protocollo il mittente può mandare più pacchetti senza aspettare l'ACK ma non può avere più di N pacchetti in attesa dell'acknowledgement nel pipeline.

Definiamo come base il numero di sequenza del pacchetto più vecchio senza ACK, e nextseqnum come il numero di sequenza del prossimo pacchetto da spedire. Abbiamo quindi 4 intervalli:

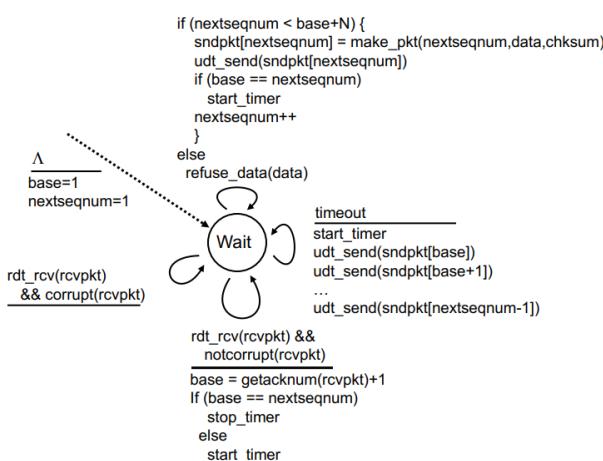


L'intervallo di numeri di sequenza ammissibili per i pacchetti trasmessi, ma che non hanno ricevuto ancora ACK, è una finestra di dimensione N.

Quando il protocollo è attivo, questa finestra scorre lungo lo spazio dei numeri in sequenza.

Per questo motivo, N viene chiamata **ampiezza della finestra** e il protocollo viene detto **finestra scorrevole (SW)**.

### MITTENTE



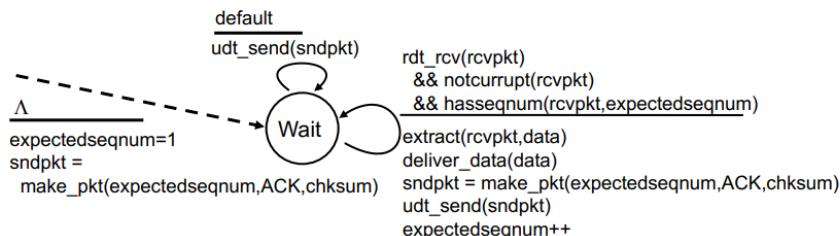
Il mittente GBN deve rispondere a 3 tipi di evento:

1. Invocazione dall'alto: quando si chiama rdt\_send, inizialmente si controlla se la finestra è piena. Se la finestra non è piena, si crea e si invia il pacchetto e le variabili vengono aggiornate. Se la finestra è piena, i dati vengono restituiti a livello superiore.

2. Ricezione di un ACK: l'ACK del pacchetto con il numero di sequenza n verrà visto come un ACK cumulativo, e ci fa capire che tutti i pacchetti con numero di sequenza minore o uguale a n sono stati ricevuti correttamente dal destinatario.

3. Evento di timeout: anche qui viene usato un contatore per risolvere il problema dei pacchetti persi. Quando si verifica il timeout, il mittente rispedisce tutti i pacchetti ancora senza ACK.

### DESTINATARIO



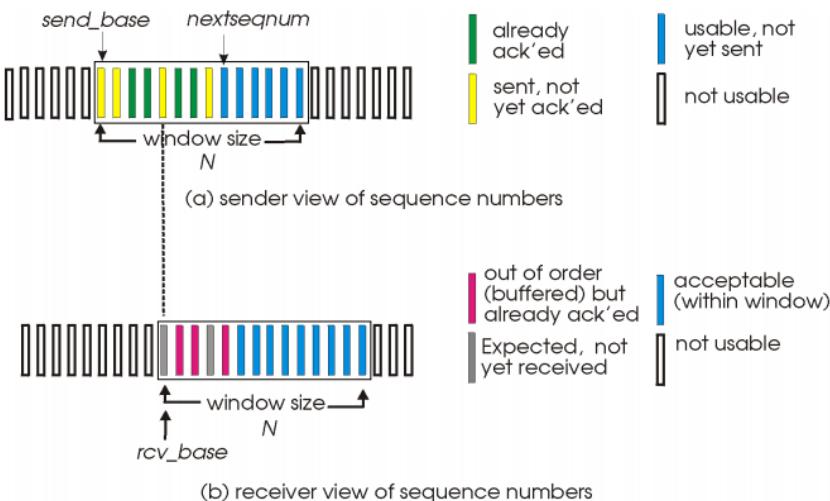
Se un pacchetto con numero di sequenza  $n$  arriva in sequenza e correttamente, il destinatario manda ACK per quel pacchetto e consegna i dati al livello superiore. Negli altri casi, il destinatario scarta i pacchetti e manda ACK per il pacchetto in ordine e senza errori.

- **RIPETIZIONE SELETTIVA (SR)**

Un altro esempio di pipeline è la ripetizione selettiva, dove si ritrasmettono al mittente solo i pacchetti dove c'è il sospetto di errore, evitando così delle ritrasmissioni inutili.

Il destinatario manderà ACK specifici per i pacchetti corretti.

Il meccanismo della ripetizione selettiva è simile a GBN, una differenza è che il mittente avrà già ricevuto qualche ACK per qualche pacchetto presente nella finestra.



Inoltre il destinatario manderà gli ACK solo per i pacchetti che non presentano errori, anche se non sono in sequenza. Questi ultimi (pacchetti non in sequenza), verranno mantenuti in un buffer finchè non saranno arrivati i pacchetti per ristabilire l'ordine.

Se il destinatario non invia un ACK per il pacchetto, la finestra del mittente non può avanzare.

Questo ci fa capire che non sempre mittente e destinatario hanno le finestre che coincidono. Ciò porta a delle conseguenze quando si ha a che fare con un intervallo finito di numeri di sequenza.

**Esempio:** intervallo di numeri di sequenza per i pacchetti 0,1,2,3 e ampiezza finestra= 3. Supponiamo che i pacchetti 0,1,2 sono inviati e ricevuti correttamente e che abbiano i loro ACK. Così la finestra del destinatario si sposta sul quarto,quinto e sesto pacchetto, quindi quelli con numero di sequenza 3,0,1. Si verificano due scenari: 1) gli ACK dei primi tre pacchetti vanno persi e il mittente li rispedisce, si verificano quindi dei duplicati. 2) gli ACK dei primi tre pacchetti arrivano correttamente, il destinatario sposta in avanti la sua finestra, verso il quarto, quinto e sesto pacchetto, con numero di sequenza 3,0,1. Il pacchetto 3 va perso ma arriva il pacchetto 0 con nuovi dati.

Quello che vede il destinatario è la sequenza di messaggi ricevuti dal canale e che lui stesso manda sul canale. Non c'è modo,quindi, per distinguere la ritrasmissione del primo pacchetto alla trasmissione originaria del quinto.

## TRASPORTO ORIENTATO ALLA CONNESSIONE: TCP

Principali proprietà di TCP:

- **Orientato alla connessione:**

Due host prima di iniziare a scambiarsi dati devono effettuare un handshake.

- **Servizio full-duplex:**

Flusso bidirezionale dei dati nella stessa connessione.

- **Punto a punto:**

Un solo mittente per un solo destinatario.

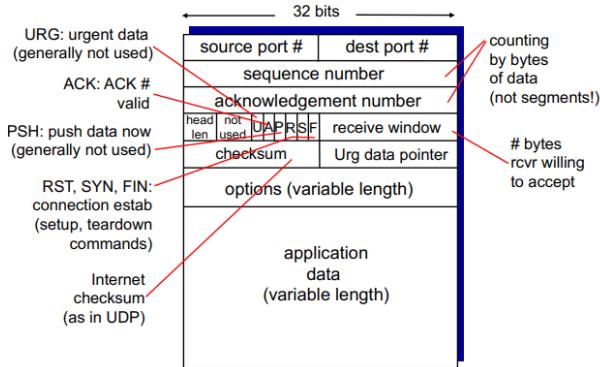
Immaginiamo che un host voglia iniziare a scambiare dati con un altro host. Per far sì che questo avvenga, l'host mittente deve inviare un segmento (senza dati a livello applicativo) all'host destinatario. Quest'ultimo, quando riceve il segmento, deve a sua volta mandare un segmento (anch'esso senza dati) al mittente. Infine,

il mittente, ricevuto il segmento dal destinatario, invia un ulteriore segmento (che potrebbe contenere dati) al destinatario. Dopo ciò lo scambio di dati tra gli host può cominciare.

Dato che i segmenti scambiati per instaurare una connessione sono tre, questo meccanismo prende il nome di **handshake a tre vie**.

Consideriamo l'invio dei dati tra client e server: il primo manda i dati verso la socket. Quando questi l'hanno attraversata, sono in mano di TCP, che li dirige verso un **buffer di invio** della connessione.

La massima quantità di dati all'interno di un segmento viene limitata dalla **dimensione massima di segmento (MSS)**: questo valore viene creato determinando prima la lunghezza del frame più grande che può essere mandato sul canale (detto **unità trasmissiva massima MTU**), poi si sceglie un MSS tale che il segmento TCP stia all'interno di un frame.



#### Struttura dei segmenti TCP:

Il segmento TCP consiste di campi di intestazione e un campo contenente i dati.

L'intestazione include **numeri di porta d'origine** e **numeri di porta di destinazione** e un campo per il **checksum**.

L'intestazione prevede poi:

- **Il campo per il numero di sequenza e il campo numero di ACK:** entrambi di 32 bit, utilizzati per il trasferimento affidabile;
- **Il campo finestra di ricezione:** 16 bit, usato per il controllo di flusso;
- **Il campo lunghezza dell'intestazione:** 4 bit, specifica la lunghezza dell'intestazione in multipli di 32 bit.
- **Il campo opzioni:** facoltativo con lunghezza variabile, utilizzato quando mittente e destinatario decidono la lunghezza massima;
- **Il campo flag:** 6 bit. Il bit **ACK** indica che l'ACK è valido, i bit **RST,SYN,FIN** usati per iniziare e chiudere la connessione, il bit **PSH** che ha valore 1 se il destinatario deve immediatamente mandare i dati a livello superiore, il bit **URG** se i dati sono urgenti.

TCP vede i dati come un flusso di byte non strutturati ma ordinati. Il **numero di sequenza per un segmento** è quindi il numero nel flusso di byte del primo byte del segmento. Dato che TCP fa l'ACK solo dei byte fino al primo byte mancante del flusso, si dice che il protocollo offre **ACK cumulativi**.

TCP usa un meccanismo di timeout e ritrasmissione per recuperare i segmenti persi. L'RTT misurato, chiamato SampleRTT, è la quantità di tempo che passa tra invio del segmento e quello di ricezione dell'ACK. Il SampleRTT non si misura ogni volta, ma viene calcolato per uno solo dei segmenti mandati che ancora non hanno l'ACK. Inoltre, il SampleRTT non si calcola per i segmenti ritrasmessi, ma solo per quelli trasmessi per la prima volta.

Per fare una stima, bisogna calcolare una media dei valori di SampleRTT, che TCP chiama EstimateRTT:

$$\text{EstimateRTT} = (1 - \alpha) * \text{EstimateRTT} + \alpha * \text{SampleRTT} \quad \text{con } \alpha = 0,125$$

Oltre ad avere una stima di RTT, è importante avere anche una misura della sua variabilità, che chiameremo DevRTT:

$$\text{DevRTT} = (1 - \beta) * \text{DevRTT} + \beta * |\text{SampleRTT} - \text{EstimateRTT}| \quad \text{con } \beta = 0,25$$

Dati questi valori, possiamo calcolare l'intervallo di timeout di TCP come:

$$\text{TmeoutInterval} = \text{EstimateRTT} + 4 * \text{DevRTT}$$

Il servizio di Internet a livello Internet (IP) non è affidabile, ossia non ci garantisce che i pacchetti arrivino privi di errori ed in sequenza. Per questo TCP crea un servizio di **trasporto di dati affidabile** al di sopra del best-effort di IP.

Esistono tre eventi principali che riguardano la trasmissione e la ricezione dei dati: dati provenienti dall'applicazione, timeout e ricezione di un ACK. Il primo evento che ha luogo incapsula i dati che gli arrivano dal livello applicazione e li passa a IP, sottoforma di segmento. In seguito, con il timeout, succede che il TCP risponde ritrasmettendo il segmento che lo ha causato e quindi riavvia il timer. In ultimo, si ha l'arrivo dell'acknowledgement con un valore valido nel campo ACK.

Ci sono delle varianti utilizzate nella maggior parte dei TCP. La prima interessa la lunghezza dell'intervallo di timeout dopo la scadenza del timer. Apportando la modifica, si ha che TCP ritrasmette il segmento con il numero di sequenza più basso tra tutti quelli confermati dall'ACK. Ma ogni volta che questo si verifica, TCP imposta il successivo intervallo di timeout al doppio del valore precedente. Quindi, questa modifica porta ad una limitazione nel controllo della congestione.

Uno dei problemi legati alla ritrasmissione è che il periodo di timeout può essere molto lungo. Quando si perde un segmento, il periodo di timeout, obbliga il mittente ad aspettare prima di rimandare il pacchetto, aumentando il ritardo. Per fortuna, il mittente, può rilevare la perdita dei pacchetti prima che si verifichi il timeout, grazie agli ACK duplicati.

Questo succede quando il destinatario riceve un pacchetto con numero di sequenza maggiore rispetto a quello che aspettava, e quindi capisce che c'è un buco nel flusso di dati, ossia un segmento mancante. Questo buco può essere dovuto alla perdita di segmenti o di un riordine all'interno della rete. Il destinatario non può inviare un ACK negativo, dato che TCP non li prevede, e quindi manda un ACK per l'ultimo pacchetto che ha ricevuto in ordine. Dato che il mittente spedisce svariati pacchetti, se uno di questi pacchetti viene smarrito ci saranno probabilmente più ACK duplicati. Quando questo succede, il mittente esegue una ritrasmissione rapida, rispedendo il segmento mancante prima della fine del timer.

## CONTROLLO DI FLUSSO

TCP offre un servizio di controllo di flusso per evitare che il mittente saturi il buffer del destinatario. Per fare ciò, TCP fa mantenere al mittente una variabile, chiamata **finestra di ricezione (RW)**, che fornisce al mittente un'indicazione dello spazio libero nel buffer del destinatario.

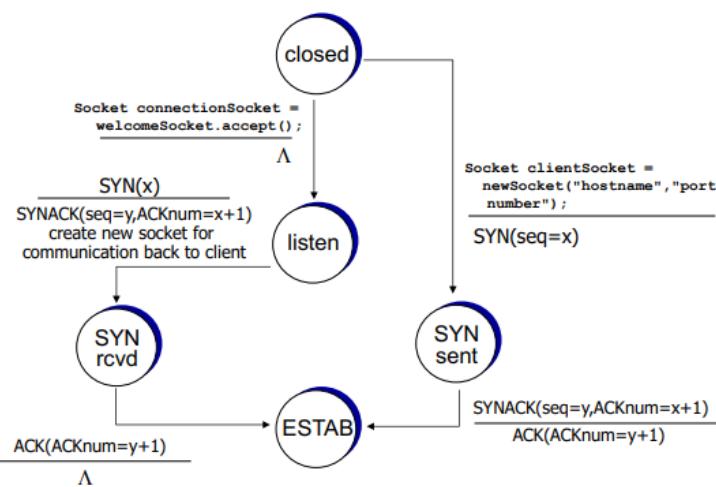
## GESTIONE DELLA CONNESSIONE

Vediamo ora nel dettaglio come si stabilisce una connessione tramite TCP:

1. TCP lato client manda un segmento a TCP lato server. Questo segmento, **SYN**, non contiene dati a livello applicativo.
2. Quando il datagramma SYN arriva a TCP al server, questo estrae il segmento dal datagramma, alloca le variabili e il buffer ed invia un segmento (senza dati), chiamato **SYNACK**, di connessione approvata al client.
3. Quando al client arriva SYNACK, anche lui alloca buffer e variabili ed infine invia un segmento (che può contenere dati) al server per dire che la connessione è stata approvata.

Questo procedimento viene detto **handshake a tre vie**.

Durante una connessione TCP, i protocolli in esecuzione attraversano vari stati TCP.



Se si vuole chiudere la connessione:

- Client e server chiudono la loro parte. Viene mandato un segmento con il bit FIN=1;
- Si risponde a FIN con un ACK;
- Si possono scambiare più FIN contemporaneamente.

## CONTROLLO DELLA CONGESTIONE

In poche parole, si ha la congestione quando troppe fonti inviano pacchetti troppo velocemente per essere gestiti dalla rete. La congestione si manifesta con perdita dei pacchetti e lunghi ritardi.

Andiamo a considerare tre scenari diversi:

- Scenario 1: due mittenti e un router con buffer illimitato

Due host A e B con una connessione che condivide un router. Immaginiamo che un'applicazione in A stia inviando dati sulla connessione con una frequenza  $\lambda_{in}$ . I dati vengono incapsulati e mandati, senza tener conto di errori, congestione o controllo di flusso. Il tasso al quale A presenta traffico al è proprio  $\lambda_{in}$ .

B si comporta analogamente. I pacchetti da A e da B passano da un router e da un collegamento uscente di capacità R. Ipotizziamo che il buffer sia illimitato.



A sinistra possiamo notare il throughput per connessione in funzione del tasso d'invio. Finché non supera  $R/2$  tutto quello mandato dal mittente viene ricevuto con ritardo finito. Il collegamento non riesce a mandare con un tasso superiore a  $R/2$ .

- Scenario 2: due mittenti e un router con buffer limitato

Ora il buffer ha memoria limitata. Conseguenza: se i pacchetti quando arrivano trovano il buffer pieno vengono scartati.

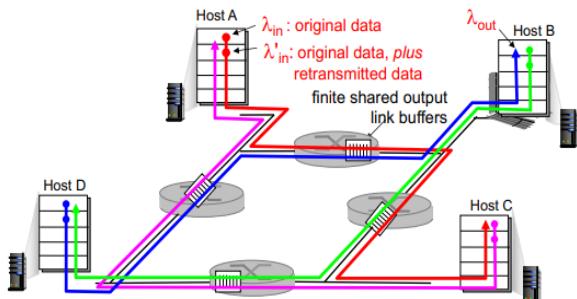
Supponiamo ora che le connessioni siano affidabili, quindi se il pacchetto viene scartato, il mittente lo rimanda. Indichiamo con  $\lambda'_{in}$  il tasso con cui il livello trasporto manda i segmenti.

Le prestazioni dipendono dal comportamento della rete con le ritrasmissioni. Consideriamo il caso in cui A sia capace di capire quanto spazio libero vi è nel buffer di B e quindi manda il pacchetto solo quando il buffer è libero. In questo caso avremo che  $\lambda'_{in} = \lambda_{in}$  e il throughput sarà  $\lambda_{in}$ .

Prendiamo ora in considerazione il caso in cui il mittente trasmette solo quando il pacchetto è perso: il carico offerto  $\lambda'_{in}$  vale  $R/2$ , con questo carico di rete il tasso vale  $R/3$ .

Ultimo caso: mittente che va in timeout prematuramente e ritrasmette un pacchetto che ha subito ritardi, ma che non è perduto. Il destinatario quindi si tiene una copia e scarta le altre. Questo rappresenta un altro costo legato alla congestione di rete: ritrasmissioni inutili. Il throughput così avrà valore  $R/4$ .

- Scenario 2: quattro mittenti, un router con buffer limitato e percorsi formati da vari collegamenti



All' aumentare della  $\lambda'$  in rossa, tutti i pacchetti blu in arrivo nella coda superiore vanno persi ridotta, il throughput blu andrà a zero.

## CONTROLLO DELLA CONGESTIONE

TCP impone a ciascun mittente un limite sulla velocità di invio sulla propria connessione relativamente alla congestione della rete. Se il mittente si accorge che vi è poco traffico, allora incrementa il proprio tasso trasmissivo; analogamente, se vi è molto traffico, lo diminuisce.

Per controllare la congestione, TCP mette a disposizione una variabile, **finestra di congestione (cwnd)**, che impone un vincolo alla velocità di immissione di traffico sulla rete da parte del mittente.

La quantità di dati che ancora non ha ACK non può superare la soglia:

$\text{LastByteSent} - \text{LastByteAcked} \leq \min\{\text{cwnd}, \text{rwnd}\}$ .

La velocità con cui il mittente spedisce dati è  $\text{cwnd}/\text{RTT}$ .

Se dovesse esserci una congestione eccessiva, uno o più buffer dei router vanno in overflow, causando la perdita di un datagramma. Questo evento fa capire al mittente che vi è della congestione sulla rete.

Dato che TCP utilizza gli ACK per scatenare (o temporizzare) gli incrementi della finestra, si dice che TCP è **auto-temporizzato**.

**Algoritmo di congestione TCP:**

1. **Slow start (obbligatoria):** il valore di cwnd parte da 1 MSS e si incrementa di 1 MSS ogni volta che un pacchetto trasmesso riceve un ACK. In TCP la velocità di trasmissione sale in modo esponenziale.
2. **Congestion avoidance (obbligatoria):** quando entra in questa fase il valore di cwnd è circa la metà di quello che aveva l'ultima volta in cui era stata rilevata la congestione. Si incrementa cwnd di 1 MSS ogni RTT.
3. **Fast recovery (facoltativa):** il valore di cwnd è aumentato di 1 MSS ogni ACK duplicato ricevuto relativamente al segmento perso che ha portato l'entrata di TCP in fast recovery. Quando arriva l'ACK per il segmento perso, TCP entra nello stato di congestion avoidance e riduce il valore di cwnd. Se si verifica un timeout, TCP entra nella fase slow start.

## FAIRNESS

Su alcune reti vige la **fairness**, ossia se ci sono K connessioni TCP aperte su un mezzo di capacità R, ogni connessione dovrebbe avere  $R/K$  rate medio.

Bisogna ora fare in modo che  $\lambda'_{in} = \lambda_{in}$

Se conoscessimo buffer e quantità di perdita del router più lento potremmo limitare il sender ma non sappiamo nulla dei router intermedi.

(Capitolo 4, Computer Networking: a top-down approach)

### INTRODUZIONE LIVELLO RETE

Il livello rete verrà descritto attraverso due punti di vista: il data plane e il control plane, uno incentrato sulla gestione dei dati e uno sul controllo.

A livello rete, lato destinatario, il segmento che viene passato dall'alto viene encapsulato in una busta arancione chiamata correttamente pacchetto. Lato ricevente quel pacchetto viene aperto e si passa al livello trasporto solo il segmento, quindi la busta rossa.

Il livello rete ha dei protocolli che sono implementati su tutti gli host: il mittente e il destinatario, come nel caso del livello trasporto, ma anche tutti gli host intermedi che si trovano a dover gestire i pacchetti in viaggio. Il livello rete, quindi, unisce concretamente tutta la rete internet dando a tutti gli host un protocollo in comune, il protocollo IP.

I router sono gli elementi intermediari che fanno andare avanti i pacchetti e hanno il compito di esaminare l'header della busta arancione e decidere in che direzione inoltrare i pacchetti. I router sono degli smistatori di pacchetti verso le destinazioni finali.

Riprendiamo anche il concetto di routing e forwarding:

- Forwarding: instradare i pacchetti nella direzione giusta grazie alle tabelle di instradamento
- Routing: aggiornamento delle tabelle di instradamento

### DATA PLANE VS CONTROL PLANE

Il data plane è quella serie di azioni realizzate e implementate localmente in ogni router. Si determina attraverso il data plane e tutte le sue funzioni la destinazione di inoltro di ogni pacchetto. In pratica tutto ciò che arriva dalle porte di input deve essere analizzato e inoltrato dalla porta di output corretta il più velocemente possibile (trasferimento da buffer di ingresso a buffer di uscita) (forwarding).

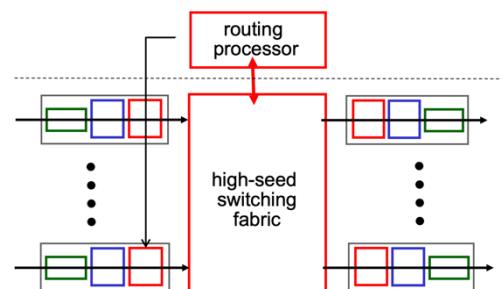
Il control plane è una logica di livello rete. Non è relativo al comportamento di un solo router, bensì di tutta la rete. Grazie al control plane si può determinare come i pacchetti andranno inoltrati tra i router in modo da farli giungere a destinazione nel minor tempo possibile (routing).

### MODELLO DI SERVIZIO A LIVELLO TRE

Come sappiamo a livello tre ogni pacchetto è gestito in modo individuale e il tipo di servizio è inaffidabile. A livello individuale non mi garantisce l'arrivo entro un certo tempo, ma in realtà non mi garantisce in generale proprio l'arrivo. A livello di flusso di pacchetti non garantisce né l'ordine, né una banda minima, né restrizioni sulle possibili modifiche dei pacchetti durante la trasmissione. Non abbiamo neanche feedback sulla congestione, se non grazie ai timer.

### STRUTTURA DI UN ROUTER

Con la linea tratteggiata distinguiamo il control plane, sopra, che si occupa del routing ed è implementato a livello software. Sotto abbiamo il data plane che si occupa del forwarding ed è implementato a livello hardware. Il forwarding opera in nanosecondi, risultando molto più veloce del routing che invece impiega millisecondi.



In generale quando arriva un pacchetto da sinistra, dalle porte di input, viene analizzato l'header e a seconda delle informazioni contenute nel routing processor, si inoltra nella porta di output corretta.

Scendiamo ora nel dettaglio di questa struttura.

## STRUTTURA DELLE PORTE DI INPUT

Come possiamo notare, nelle porte a sinistra sono contenuti tre quadrati di colore diverso, che rappresentano i vari livello che un pacchetto attraversa quando arriva a un router.

- Il verde è il livello fisico dove i segnali ricevuti vengono convertiti in bit.
- Il blu è il livello MAC/LLC relativo alla tecnologia implementata in quel router e a cui il livello fisico passa i bit appena ricevuti.
- Il rosso rappresenta il livello rete che riceve i dati dal livello sottostante e possiede un buffer di ricezione che serve per mantenere i pacchetti che aspettano di essere analizzati e inoltrati.

In particolare nell'ultima fase (rosso) viene analizzato l'header del pacchetto. Si fa il lookup (controllo del valore della rete destinazione del pacchetto) e si controlla nella tabella di forwarding per decidere in che direzione l'high-speed switching fabric deve inoltrare il pacchetto. L'high-speed switching fabric andrà quindi configurato ogni volta in base alle informazioni contenute nel routing processor.

Si parla di match & action: si confronta il valore della rete destinazione del pacchetto con tutte le reti presenti nella tabella (matching avviene nel routing processor) e quando si trova il corrispondente si agisce di conseguenza, inoltrando il pacchetto nella porta corrispondente presente nella tabella.

Tutto questo meccanismo può richiedere numerosi cicli di memoria e riuscire a farlo in millisecondi è un ottimo risultato. L'obiettivo è realizzarlo nel tempo che quei bit impiegherebbero per attraversare la linea senza nessuna analisi. Ovviamente è impossibile e un ritardo ci sarà sempre ma bisogna tendere a quel livello di perfezione.

Se i pacchetti arrivano più velocemente del ritmo di forwarding si creerà un accodamento che può portare anche alla perdita dei pacchetti.

Il forwarding può essere di due tipi:

- Forwarding basato sull'IP destinazione: è la versione tradizionale e il forwarding si basa esclusivamente sull'IP destinazione del pacchetto
- Forwarding generalizzato: in questo caso il forwarding si basa su anche altri campi contenuti nell'header, come l'urgenza, la necessità di QoS e ovviamente anche l'IP destinazione.

La prima versione è migliore dal punto di vista della velocità, mentre il secondo meno.

## FORWARDING BASATO SULL'IP DESTINAZIONE

il forwarding basato sull'IP destinazione dipende dal matching che abbiamo già introdotto: avviene nel routing processor e si confronta il valore della rete destinazione del pacchetto con tutte le reti presenti nella tabella e quando si trova il corrispondente si agisce inoltrando il pacchetto nella porta corrispondente.

Dobbiamo immaginarcici il routing processor come una CPU con dei registri dove viene caricato l'indirizzo IP destinazione e a seconda dei risultati si comunicato allo switching fabric la porta di uscita da predisporre.

## TABELLA DI FORWARDING

Le righe che compongono la tabella rappresentano la complessità di ricerca, meno ne ho meglio è.

Il range di indirizzi corrispondenti ad una porta viene espresso come si vede in figura.

Destination Address Range	Link interface
11001000 00010111 00010*** *****	0
11001000 00010111 00011000 *****	1
11001000 00010111 00011*** *****	2
otherwise	3

Confronto l'indirizzo IP destinazione con i bit espliciti di tutte le righe e scelgo quella che mi permette di avere la corrispondenza più lunga. Il confronto avviene con un and logico, l'operazione che a livello computazionale richiede un solo ciclo di clock.

Inoltre rendersi conto che non c'è nessun abbinamento può essere fatto in modo anticipato, saltando subito all'otherwise, perché le righe sono in ordine ed è immediato accorgersi di aver superato il punto in cui avrei già dovuto fare la mia scelta. Otteniamo un meccanismo molto veloce.

Inoltre possiamo implementare il controllo in parallelo: come se avessi un registro per ogni riga, inserisco l'IP destinazione in questi registri, in ciclo di clock faccio tutti gli and logici e scelgo il risultato più lungo. Quindi non viene fatto nessun ciclo for per confrontare tutte le righe con l'IP destinazione.

#### ESEMPIO DI MATCHING E DI UTILIZZO DELLA FROWARDING TABLE

Il primo indirizzo avrà come interfaccia corrispondente la numero 0 perché facendo il matching con tutte le righe la seconda e la terza mi danno un matching di 20, mentre la prima di 21.

Per quanto riguarda il secondo indirizzo l'interfaccia corretta è la numero 1, perché la prima mi da un matching di 20, la terza di 21 ma la seconda di 24, dato che abbiamo quei 000 esplicativi che migliorano il matching. Ovviamente di scelta giusta ce n'è solo una.

Se nessuna riga mi da un matching, anche piccolo, si sceglie il caso otherwise.

Ovviamente gli asterischi posso riempirli come voglio ma essendo bit impliciti non mi creano un matching. Il matching vale solo se fatto con bit esplicativi.

Un router di rete locale ha solitamente solo due righe: la prima identifica la rete locale (i cui bit in chiaro saranno definiti come l'indirizzo IP del router and la maschera di rete) e l'altra (otherwise) verso internet, o meglio verso il default gateway.

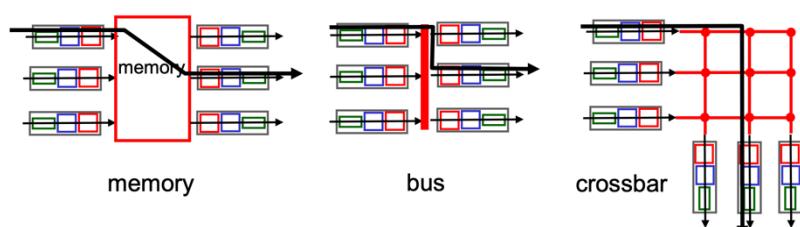
Questo è un bene perché i router di una rete locale sono piccoli e spesso non specializzati alla funzione di router e non possono gestire grandi quantità di dati, mentre i tritacarne presenti nella core di Internet sono specializzati a produrre risposte ad una velocità incredibile, anche ogni ciclo di CPU.

#### HIGH SPEED SWITCH FABRIC

La switching fabric ha il compito di trasferire i pacchetti dai buffer delle porte d'ingresso alle corrette porte di output e definiamo come ritmo di switching il ritmo con cui i pacchetti vengono trasferiti. Idealmente se abbiamo N input dovremmo avere uno switching rate pari a N per il rate delle linee in ingresso (su tre linee abbiamo 1000 pacchetti in arrivo al secondo, allora lo switching rate dovrebbe essere di 3000). Quindi la switching fabric, per riprogrammarsi per trasferire il prossimo pacchetto da ingresso a uscita dovrebbe essere in grado di tenere il ritmo di tutti i pacchetti possibili in arrivo su tutte le linee. Solitamente la switching fabric è il collo di bottiglia perché non riesce a stare dietro al ritmo dei pacchetti in arrivo.

Esistono tre tipi di switching fabric:

- Memoria: questo sistema è molto lento, in particolare per la velocità di lettura in memoria del sistema e per il fatto che ci sono due accessi in memoria, uno per scrivere in input e uno per leggere in output. Quindi abbiamo due flussi ma essendo gestito da un bus condiviso esso può essere occupato da un solo flusso di dati alla volta e un pacchetto occupa per due volte il bus (input-memoria e memoria-output).
- Bus condiviso: anche in questo caso abbiamo una contesa sull'utilizzo del bus che può essere utilizzato da una sola porta alla volta. La velocità è inoltre limitata dalla banda del bus e costituire un collo di bottiglia notevole se le porte di entrata e uscita sono molto potenti.
- Crossbar switch (contatti elettrici programmabili): risolve quei problemi che avevamo con il bus condiviso, infatti è possibile il parallelismo finché i segmenti non si incrociano (tre percorsi possibili contemporaneamente).



### CODA SULLE PORTE DI INPUT E BLOCCO HOL

Se su due porte in input arrivano due pacchetti diretti nella stessa porta uno dei due dovrà aspettare il transito dell'altro. Parliamo di head of the line blocking (blocco HOL).

Ma questo problema potrebbe portare ad un altro problema forse più grave: dietro al pacchetto in attesa potrebbe esserci un pacchetto la cui porta di output è libera ma non può proseguire perché bloccato dal pacchetto che lo precede.

### CODA SULLE PORTE DI OUTPUT

Possiamo avere un accodamento sulle porte di uscita quando la fabric switching lavora più velocemente di quello che i buffer di uscita riescono a gestire. A causa dell'overflow dei pacchetti possiamo avere ritardo ed eventuale perdita dei pacchetti.

### SCHEDULING

Lo scheduling sarebbe la scelta del prossimo pacchetto da inviare sul canale trasmisivo. Chiaramente la tecnica più intuitiva è la FIFO , quindi inviare i pacchetti in ordine di arrivo (come la coda alle poste).

Un'altra scheduling policy è la priority: quando arriva un pacchetto se ne controlla l'urgenza e si smista nella coda di urgenza corrispondente (abbiamo delle ulteriori code).

Consideriamo il caso di due code, una ad alta priorità e l'altra a bassa priorità: si controlla se ci sono pacchetti nella prima coda e nel caso vengono spediti per primi, se non ci sono si passa alla seconda coda. Un pacchetto urgente ha comunque la precedenza su un pacchetto non urgente anche se è arrivato dopo.

Così facendo si rischia che i pacchetti non urgenti non vengano mai inviati ma statisticamente non succede. Comunque nel caso succedesse si parla di starvation, scatta il timer del pacchetto in questione che viene rispedito inutilmente creando traffico inutile. Inoltre aumenta il RTT dei pacchetti. Questo porta a un rallentamento nell'invio non necessario.

Altra scheduling policy è il round robin (RR): in questo caso prendiamo un pacchetto da una coda e un pacchetto dall'altra in modo alternato. Evitiamo così il problema della starvation dei pacchetti ma perdiamo il concetto di priorità.

Abbiamo anche una generalizzazione del RR, ovvero la weighted fair queuing: ad ogni ciclo viene mandato un certo numero di pacchetti (più alto se sono più urgenti, più basso se sono meno urgenti) per ogni coda.

Nel caso di riempimento della coda quale pacchetto andrebbe scartato per liberare la coda? Ci sono tre politiche diverse per scartare i pacchetti (discard policy):

- Tail drop: si scarta quello in coda
- Priorità: si scarta in base alla priorità dei pacchetti
- Random: si scarta un pacchetto random (quando hanno tutti la stessa priorità o per evitare che una connessione TCP domini sulle altre. In questo modo facciamo chiudere la finestra di trasmissione di una connessione ma aiutiamo il buffer a liberarsi. Infatti scegliendo in modo random è statisticamente più probabile che scegliamo un pacchetto facente parte di una connessione molto presente nel buffer, che così si da una calmata).

## HEADER PACCHETTO IP

Quello che andremo ad analizzare è un pacchetto IPv4, un pacchetto IPv6 avrebbe una struttura diversa.

Abbiamo a disposizioni righe da 32 bit in cui distinguiamo i seguenti campi:

- Ver: versione del pacchetto IP, in questo caso 4.
- Header length: lunghezza in byte dell'header senza campo data ma che va specificato per la presenza del campo options, la cui lunghezza può variare.
- Type of service: usato per valutare l'urgenza ma poco usato ultimamente
- Length: lunghezza in byte totale del pacchetto (al massimo  $2^{16}$ )
- Identifier: numero identificatore del pacchetto a 16 bit
- Flags: per la frammentazione
- Fragment offset: per la frammentazione

In particolare un pacchetto IPv4 da 64000 byte non potrà mai essere trasmesso su una ethernet come un solo pacchetto da 64000 byte, andrà necessariamente frammentato in frammenti da 15000 byte.

L'identifier serve per capire a quel pacchetto originale un certo frammento fa parte. Esso avrà il bit che indica la frammentazione settato a 1 e nell'ultimo campo, fragment offset, ci sarà indicato quale pezzo del pacchetto originale rappresenta, così che possano essere riordinati (in particolare troviamo un'indicazione al primo byte, o meglio al primo gruppo di 8 byte, quindi dividiamo il numero del primo byte per otto).

Infatti frammentando un pacchetto IPv4 otteniamo tanti pacchetti IPv4 che avranno vita propria su Internet e potrebbero arrivare disordinati o non arrivare mai. Addirittura se perdo un frammento tutto il pacchetto va ritrasmesso, dato che quello che ritrasmette è TCP e la frammentazione non la fa lui.

L'unico che si occupa della ricostruzione dei pacchetti è il destinatario e tale lavoro risulta piuttosto oneroso, richiedendo un buffer adatto.

Con flag = 0 e offset != 0 indichiamo l'ultimo pacchetto

Con flag = 0 e offset = 0 indichiamo un pacchetto unico

- TTL: numero di volte che un pacchetto può essere gestito da un router prima di essere ucciso
- Upper layer: si specifica a che protocollo di livello trasporto va passato il pacchetto in questione (a un'istanza di TCP 6 o a un'istanza di UDP 20).
- Header checksum: checksum solo dell'header, vogliamo solo assicurarci che l'header sia corretto per poterlo mandare avanti correttamente. Non ci dice se ci sono errori sui dati.
- Indirizzo IPv4 sorgente (32 bit)
- Indirizzo IPv4 destinazione (32 bit)
- Options: tempo di partenza, tracciatura dei router attraversati, lista di router da cui passare
- Dati (segmento TCP o UDP)

In totale tra TCP (20 byte) e IP (20 byte) abbiamo 40 byte di overhead. Alcuni vanno sommati eventuali byte di overhead del livello applicazione.

## PRECISAZIONI SU INDIRIZZI IP

Come sappiamo un router è quel dispositivo che mette in comunicazione una rete locale con l'esterno. Il router viene visto dagli host della rete locale come un vero e proprio componente di quella rete.

Se più reti e sottoreti si appoggiano allo stesso router esso dovrà avere un indirizzo IP diverso per ogni interfaccia. Ci sono sempre almeno due interfacce: una verso la rete locale di cui è router e una verso fuori, verso Internet.

A meno che non ci troviamo nella rete locale corrispondente, fare il ping dell'indirizzo IP dell'interfaccia interna del router non ci porta nessun risultato.

## DHCP: DYNAMIC HOST CONFIGURATION PROTOCOL

La configurazione di un host all'interno della rete locale può essere fatta manualmente o attraverso il DHCP, un meccanismo attraverso il quale un host può connettersi ad una rete in modo 'plug and play'.

Questa connessione avviene attraverso quattro messaggi:

- DHCP discover: messaggio inviato da un host per capire se ci sono reti con DHCP server disponibili a cui collegarsi. È un messaggio in broadcast (opzionale).
- DHCP offer: se a un DHCP server arriva la richiesta, viene inviato un messaggio di offerta (opzionale)
- DHCP request: l'host allora manda un messaggio di richiesta di un indirizzo IP per entrare a far parte di quella rete
- DHCP ack: viene attribuito grazie a questo messaggio da parte del router un indirizzo IP a quell'host

In particolare il DHCP server offre oltre all'indirizzo IP: l'indirizzo del primo router più vicino a cui l'host deve far riferimento (il default gateway), nome e indirizzo IP del DNS server e la maschera di rete.

La richiesta DHCP viene incapsulata in un pacchetto UDP perché non richiede l'apertura di connessioni come TCP (come potrebbe essere possibile senza indirizzi). Viene poi incapsulato in un pacchetto IP e infine in un pacchetto ethernet (busta gialla) e inviato in broadcast.

Essendo un lavoro che avviene con un processo a livello applicazione abbiamo anche i numeri di porta, in particolare abbiamo la porta 67 per il client e 68 per il server.

## NAT: NETWORK ADDRESS TRASLATION

Sottoscrivendo un contratto con il nostro ISP otteniamo un indirizzo IP che ci permettere di connettere un solo host a Internet. Nelle nostre case però noi abbiamo decine di dispositivi da connettere a Internet e allora come possiamo fare avendo a disposizione un solo indirizzo IP? Utilizzano un processo chiamato NAT.

L'indirizzo IP che otteniamo dal nostro ISP è quello associato al nostro router e si trova all'ultimo livello, quello più esterno della struttura di Internet. Da una parte, quindi, è connesso al mondo di Internet, dall'altra è connesso a una rete locale, quella di casa nostra, dove avviene una magia: possiamo convincere il router che dalla parte della rete locale si trova una fittizia rete anche di classe A (dove possiamo creare una struttura ad albero anche molto complessa con reti e sottoreti).

Come faccio però a sfruttare questi indirizzi fittizi e permettere agli host della mia rete locale di comunicare su Internet? Fino a quando resto nella rete locale non ho nessun problema. Il router avrà come interfaccia interna verso la rete locale, un indirizzo IP coerente con la rete fittizia e tutti i dati scambiati all'interno verranno riconosciuti dal router come interni alla rete e non li farà uscire su Internet.

Il problema di comunicare su internet con questo sistema non è tanto l'invio della richiesta, perché basta inserire l'indirizzo destinatario e a prescindere da dove viene può essere recapitato al ricevente giusto. Il problema è nella risposta dato che l'indirizzo associato al mittente p un indirizzo fittizio che non esiste su Internet. È quindi necessario utilizzare l'unico indirizzo IP esistente su Internet in nostro possesso e poi occuparsi dello smistamento di tutto quello che arriva a quell'indirizzo ai vari host nella rete locale.

Di questo meccanismo se ne occupa il NAT. In particolare vengono inventati dei socket utilizzando delle "porte" finte che servono per tener traccia di chi ha fatto richiesta e vuole risposta. Facciamo un esempio:

L'host con indirizzo fittizio nella rete locale 10.0.0.1 apre un browser e si vuole mettere in contatto con il web server che ha indirizzo reale 128.119.40.186 sulla porta 80. Il NAT router che vede arrivare un pacchetto dalla sua rete locale, controlla la destinazione e se è interno alla rete locale non viene fatto uscire, altrimenti va inoltrato all'esterno, verso internet.

Fare questa operazione significa che il NAT router deve convertire l'indirizzo di rete locale in indirizzo pubblico e compilare la tabella di conversione NAT, presente su ogni router. Il router, quindi, rimpiazza l'indirizzo del mittente nella LAN con il suo indirizzo, unico esistente su Internet e per ogni pacchetto proveniente dalla LAN e diretto su Internet c'è una riga in cui troviamo la corrispondenza tra [indirizzo IP mittente + porta vera nell'host] e [indirizzo IP router + porta finta assegnata dal router a quell'host].

NAT translation table	
WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345

Nella parte 'WAN side address' che indica ciò che viene visto dall'esterno in corrispondenza ad ogni indirizzo nella LAN, non potrà che esserci sempre l'indirizzo del router. Ciò che poi aiuta il router a distinguere i vari host nella LAN sono i numeri di porta finti.

3345 è la porta vera da cui è partita la richiesta del mittente.

5001 è un "numero di porta" inventato dal router.

Moltiplico gli indirizzi utilizzabili su internet, utilizzandone uno solo e tanti numeri di porta.

Il destinatario ricevuto il pacchetto genererà una risposta indirizzata a [indirizzo IP router + porta finta assegnata dal router a quell'host] che arriverà al router e poi lui avrà premura di leggere il numero di porta e inoltrare il pacchetto al mittente originale nella LAN che aveva fatto richiesta.

Il pacchetto arrivato al router risalirà lo stack di rete fino al livello trasporto e dietro alla porta artificiale 5001 troverà il processo NAT che controllerà nella tabella la riga corrispondente alla porta 5001 e inoltrerà al destinatario legittimo nella LAN il pacchetto (la provenienza rimarrà sempre il server su internet).

Socket diversi dallo stesso host nella LAN avranno righe diverse nella tabella e allo stesso modo due richieste indirizzate allo stesso host su Internet avranno due righe diverse e l'host si vedrà arrivare due richieste dallo stesso IP ma da due porte diverse e genererà due risposte distinte.

L'unico limite che ho è il numero di porte artificiali che il NAT router può creare (64.000) ma allora creo due NAT router nella mia rete locale e ottengo il doppio delle porte possibili.

Le connessioni TCP vengono instaurate normalmente tra i due host con l'ausilio trasparente del NAT router che traduce o sostituisce al volo destinatario/mittente quando è necessario.

#### VIOLAZIONE STACK ISO-OSI

C'è una controversia legata al NAT che realizza, sfruttando i numeri di porta, un connubio con il livello trasporto, violando così il paradigma per cui ogni livello deve essere separato dagli altri livelli.

Questo implica che i vari livelli seguano gli stessi protocolli (non posso sostituire un IPv4 con un IPv6).

## IPv6

Le motivazioni che portano ad un passaggio ad IPv6 sono:

- Gli indirizzi a 32 bit sono già esauriti
- Overhead elevato soprattutto a causa della frammentazione
- Mancanza di QoS

IPv6 utilizza un header da 40 byte che è il doppio rispetto a IPv4 ma non consente la frammentazione.

Inoltre vengono introdotti dei nuovi campi:

- Priority: che aiuta a definire la priorità di un pacchetto in un flusso o di un flusso rispetto agli altri.
- Flow label: che serve per indicare che quel pacchetto appartiene ad uno specifico flusso.
- Payload length: dimensione del campo dati
- Next header: è un puntatore all'inizio dell'header relativo al livello trasporto nel campo data oppure punta ad una zona del campo data che è estensione dell'header: possiamo aggiungere campi addizionali senza dover modificare la dimensione dell'header (compatto e scalabile)
- Hop limiti: equivalente di TTL

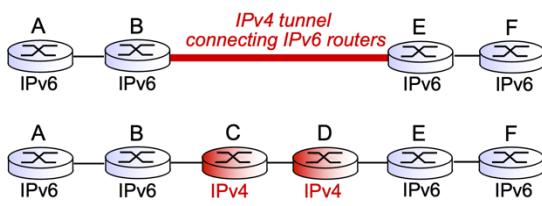
Ovviamente abbiamo anche

- Indirizzo IPv6 sorgente (128 bit)
- Indirizzo IPv6 destinazione (128 bit)

Non abbiamo più il campo checksum. Considerando la dimensione dell'header in rapporto alla dimensione degli indirizzi, l'header di IPv6 è molto più compatto.

Con all'evoluzione ICMPv6 del protocollo ICMP è possibile evitare la frammentazione grazie alla notifica di errore "pacchetto troppo grande": quando un pacchetto risulta troppo grande viene inviata questa notifica alla sorgente che si mette all'opera per creare dei pacchetti più piccoli. Viene quindi notificata la massima grandezza che possono assumere i pacchetti in relazione al percorso che devono fare. È un meccanismo che agisce alla sorgente e con un po' più di lavoro all'inizio possiamo evitare la frammentazione.

Tunneling: visto che non possiamo passare da una rete basata su IPv4 a una basata su IPv6 in modo netto, per la transizione da uno all'altro si ricorre al tunneling. Il pacchetto IPv6 viene incapsulato in un pacchetto IPv4 (aggiungiamo esternamente l'header IPv4). In questo modo possiamo far viaggiare dei pacchetti IPv6 su una rete che presenta anche router IPv4.



L'ultimo router IPv6 (B) incapsula il pacchetto in modo che esso possa viaggiare attraverso i router IPv4 (che altrimenti non riuscirebbero a interpretarlo) fino a quando si raggiunge un altro router IPv6 (E) che decapsula il pacchetto e lo fa proseguire come pacchetto IPv6. È come se i router IPv4 vengono saltati.

## FORWARDING GENERALIZZATO E APPROCCIO SDN

SDN sta per Software Defined Network.

L'idea da cui partire è quella di avere un controller generalizzato a livello di control plane: immaginiamo quindi un processo centrale dove vengono fatti dei calcoli o eseguiti degli algoritmi che implementano delle funzioni di routing oppure possiamo avere un sistemista che scrive fisicamente le regole di funzionamento del routing all'interno della rete. Quindi il routing e il trasferimento dei dati sono programmabili.

In un approccio SDN il networking è controllato da software e noi possiamo scrivere questo software, programmando le funzioni di rete concernenti al forwarding di pacchetti, o meglio di flussi di pacchetti.

Un algoritmo o un sistemista possono definire a livello centrale, in modo molto compatto, delle regole generali di funzionamento che vengono poi trasmesse a livello locale ad ogni singolo router e inserite nelle tabelle locali di controllo dei flussi, diventando l'equivalente delle tabelle di instradamento.

SDN permette la gestione di switch, router, firewall (filtraggio, diagnostica, monitoraggio, ...).

Vediamo un esempio per capire il funzionamento di SDN in maniera più approfondita: immaginiamo che nella nostra rete ci sia un router che riceve un pacchetto; confrontando i dati nell'header con i dati nelle local flow table può capire se quel pacchetto può proseguire o meno e se può in che direzione.

Questo è l'operazione primaria di instradamento ma non è l'unica operazione che può essere fatta, infatti nelle tabelle ci sono tre colonne:

- Headers: che serve per l'instradamento
- Actions: indica che azioni fare per ogni pacchetto
- Counters: serve per tener traccia di svariati valori (numero di pacchetti transitati per un certo flusso, con determinate caratteristiche, provenienti o destinati a certi indirizzi, ...)

## OPEN FLOW

OpenFlow è una metologia di definizione delle local flow tables basato su SDN.

Partiamo con la definizione del flusso che avviene nell'header del pacchetto (come sappiamo in IPv6 c'è proprio un campo dedicato). Le regole di forwarding sono generalizzate e standard:

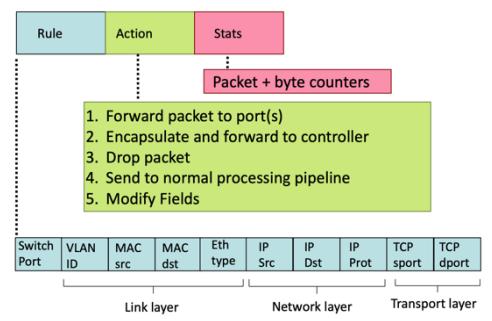
- Pattern: prima questione è riconoscere un pattern, quindi riconoscere il flusso e ricondurre i valori contenuti nell'header ad una delle righe della tabella di flusso.
- Actions: a seconda del match si fanno delle determinate azioni da fare su quel pacchetto (drop, forward, modify (come nel caso di NAT), inoltro del pacchetto al controller nel control plane (nel caso magari si presenti una situazione nuova che richiede un'implementazione specifica)).
- Counters: conta byte e pacchetti scambiati
- Priority: distingue quei flussi che hanno priorità diverse e necessitano gestioni diverse.

La flow table in un router definisce quindi la coppia match+actions per un router.

Come possiamo vedere dall'immagine

I campi dell'header che vengono analizzati sono sia relativi al livello due, che a livello tre, che a livello quattro e a seconda del matching vengono prese delle determinate decisioni (actions). Possiamo agire e discriminare i pacchetti con un'unica formula rispetto a moltissimi campi

Esempio:  $\text{src}=1.2.*.*$ ,  $\text{dest}=3.4.5.* \rightarrow \text{drop}$   
 $\text{src}=*.*.*.*$ ,  $\text{dest}=3.4.5.* \rightarrow \text{forward}(2)$



Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	*	51.6.0.8	*	*	port6

Possiamo così implementare grazie a questa astrazione quattro diversi servizi match+actions che abbiamo già visto o che vedremo:

- Router:
  - Match: longest destination IP prefix
  - Actions: forward su una delle porte di uscita
- Switch:
  - Match: indirizzo MAC di destinazione
  - Actions: forward del pacchetto
- Firewall:
  - Match: controllo dell'indirizzo IP e della porta UDP e TCP
  - Actions: permesso o blocco del pacchetto
- NAT:
  - Match: indirizzo IP e porta
  - Action: riscrittura di questi due valori

Viene così generalizzato il forwarding che non si limita più ad essere solo destination-based, ma diventa generalizzato. Chi si occupa quindi della computazione delle tabelle di forwarding e delle local flow tables? Il control plane trattato nel prossimo capitolo.

## LIVELLO RETE: CONTROL PLANE

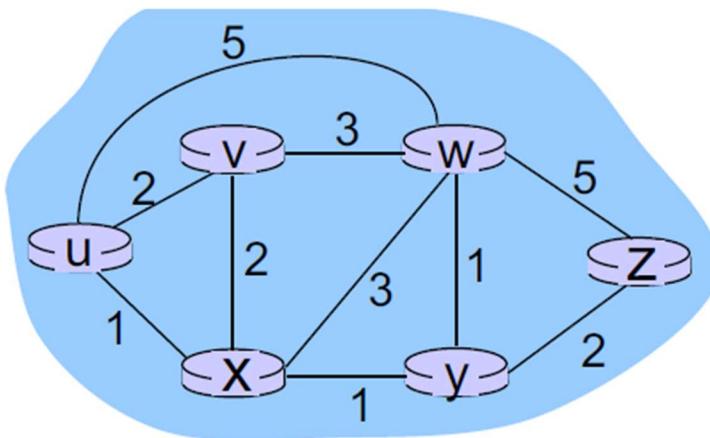
Mentre il data plane implementa le funzioni di forwarding, control plane scrive le tabelle (routing) che poi vengono usate dal data plane per il forwarding. Ci sono due approcci per scrivere le tabelle di routing:

- **Per-router** (metodo tradizionale): ogni router scrive la sua tabella di routing;
- **Controllo centralizzato logicamente** (SDN): le tabelle di routing sono gestite da un sistema centralizzato da cui tutti i router attingono.

Approfondimenti nelle slide sul data plane.

### PROTOCOLLI DI ROUTING

I protocolli di routing cercano i cammini migliori da un router mittente a un router destinatario passando dai router della rete. Il concetto di “**cammino migliore**” non è fisso: si può parlare di costo, di velocità o di congestione della rete. Il problema del routing è tra i più complessi nelle Reti, specialmente per quanto riguarda le reti senza fili, all’interno delle quali gli host possono spostarsi, cambiando i possibili cammini.



Per analizzare i problemi di routing si può astrarre una rete a un **grafo** non orientato (o orientato se necessario) con dei costi, che sono appunto relativi al problema da risolvere: posso rappresentare la larghezza di banda di un collegamento o un indicatore di congestione. Si implementa quindi un algoritmo che risolva un problema di cammini di costo minimo e salvi nelle **tabelle di routing** i cammini trovati.

Implementare un simile algoritmo è complicato seguendo l’approccio distribuito (per-router): un router non vede necessariamente tutto il grafo (tutta la rete) quindi potrebbe scegliere un percorso che non è ottimale, risulterebbe necessario confrontare le tabelle di routing di ogni router, rendendo questo approccio poco funzionale.

Gli **algoritmi di routing** sono classificati come:

- Algoritmi link state, se tutti i router vedono interamente la rete;
- Algoritmi distance vector, se i router conoscono solo le connessioni (e i relativi costi) con i router vicini (a distanza 1), rendendo quindi necessario un approccio iterativo e basato sullo scambio di informazioni con i vicini.

Inoltre, in base al “refresh” delle tabelle di routing gli algoritmi possono essere:

- Statici, se cambiano molto raramente (es. connessioni cablate);
- Dinamici, se cambiano molto velocemente (es. connessioni wireless).

## Algoritmi Link-State

Il più famoso algoritmo link-state è l'algoritmo di **Dijkstra**, che computa i percorsi di costo minore da un host (sorgente) a tutti gli altri host (destinazioni), riuscendo quindi a costruire una forwarding table completa per l'host sorgente.

### NOTAZIONE:

- $c(x, y)$  = costo collegamento tra host  $x$  e host  $y$  (settato a  $+\infty$  se non sono direttamente collegati);
- $D(v)$  = valore corrente del costo del cammino dalla sorgente alla destinazione  $v$ ;
- $P(v)$  = nodo predecessore di  $v$  nel cammino dalla sorgente;
- $N'$  = nodi raggiunti e il cui costo è stato calcolato definitivamente;

## Dijkstra's algorithm

### 1 Initialization:

```

2   N' = {u}
3   for all nodes v
4     if v adjacent to u
5       then D(v) = c(u,v)
6     else D(v) = ∞
7

```

### 8 Loop

```

9   find w not in N' such that D(w) is a minimum
10  add w to N'
11  update D(v) for all v adjacent to w and not in N' :
12    D(v) = min( D(v), D(w) + c(w,v) )
13  /* new cost to v is either old cost to v or known
14  shortest path cost to w plus cost from w to v */
15 until all nodes in N'
```

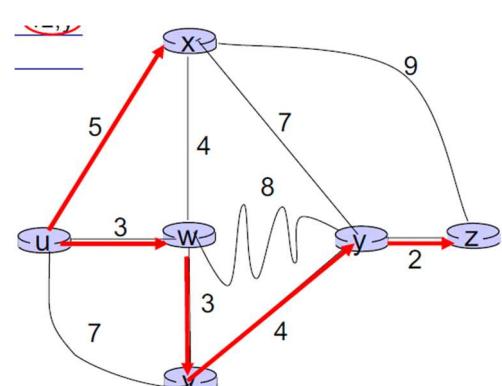
La parte di inizializzazione mette tra i nodi già raggiunti solo 'u' (sorgente) e poi setta i costi dei nodi adiacenti 'v' come  $c(u, v)$ , altrimenti setta il costo a  $+\infty$ .

Nel loop invece sceglie il nodo 'w' in  $N'$  con costo minimo, lo aggiunge a  $N'$  e analizza tutti i 'v' adiacenti a 'w' che non siano salvati in  $N'$  (ancora non definitivi, quindi): controllo se sia più conveniente lasciare il loro costo di raggiungimento come prima, oppure se aggiornarlo con il costo di 'w' +  $c(w, v)$  (costo dell'arco  $w-v$ ), ovvero sceglie se passare per 'w' per raggiungere 'v'. Ripete questo finché tutti i nodi non sono in  $N'$ .

E' omessa la parte relativa ai predecessori, che però è fondamentale per salvare quali host si sono attraversati per raggiungerne uno.

### TABELLA DI ESECUZIONE DI DIJKSTRA

Step	$N'$	$D(v)$ $p(v)$	$D(w)$ $p(w)$	$D(x)$ $p(x)$	$D(y)$ $p(y)$	$D(z)$ $p(z)$
0	u	7,u	3,u	5,u	$\infty$	$\infty$
1	uw	6,w	5,u	11,w	$\infty$	
2	uwx	6,w		11,w	14,x	
3	uwxv			10,v	14,x	
4	uwxvy				12,y	
5	uwxvyz					

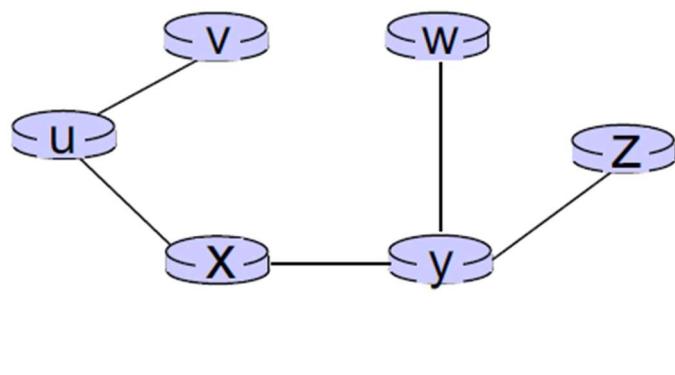


I costi e predecessori cerchiati sono i vari nodi scelti con costo minimo, quindi si lavora sui relativi adiacenti, ad ogni iterazione. E' così costruito un albero grazie ai vari  $p(v)$  che costituiscono un percorso.

**Problema:** mandare pacchetti lungo la strada migliore non rischia di congestionarla? Certo! Ma è comunque meglio che mandarli lungo uno strada che non sia la migliore. La soluzione è il fatto che le tabelle di routing vengono costruite con refresh rate di pochi secondi/millisecondi, quindi se si pone come parametro la congestione dei collegamenti, il problema verrà risolto subito, cambiando rotta repentinamente.

### ESEMPIO DI RISULTATO FINALE

destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)



**Complessità Dijkstra:**  $O(n^2)$  (esistono implementazioni migliori con costi  $O(n \log n)$ )

**Oscillazioni possibili:** quelle citate sopra, ovvero spedendo pacchetti sul cammino migliore, la congestione delle reti migliora, quindi dopo si potrebbe scegliere un altro cammino meno congestionato, che potrebbe congestionarsi a sua volta ecc.

### ALGORITMI DISTANCE VECTOR

L'algoritmo più famoso è quello di Bellman-Ford, che si basa sulla programmazione dinamica (ovvero, salvare i risultati già ottenuti e usarli per restituire il valore richiesto).

#### NOTAZIONE

- $d_{x(y)}$  = costo del percorso di costo minimo da  $x$  a  $y$ , ovvero:
- $d_{x(y)} = \min(v) \{c(x, v) + d_{v(y)}\}$  (si somma il costo di un collegamento a un nodo adiacente con il costo del cammino da quel nodo al nodo destinazione desiderato: prendo quindi il nodo adiacente che fornisce la somma minima).

L'algoritmo è quindi un'iterazione asincrona (detto algoritmo Gossip, ovvero un algoritmo che si basa sul cambiamento di eventuali parametri che porta quindi a un ricalcolo) che avviene per ogni cambiamento di costi nei collegamenti. Si tratta inoltre di un sistema distribuito poiché un host avverte i vicini solo se il loro vettore ' $d$ ' cambia, i vicini a loro volta iterano bellman-ford e avvisano i vicini **solo se** il loro ' $d$ ' cambia.

Bellman-Ford quindi non raggiunge mai l'ottimale vero e controlla continuamente, tuttavia in caso di reti basate su distance vector per forza, come le reti wireless, l'unica scelta è usare Bellman-Ford.

### DIFFERENZE TRA LINK STATE E DISTANCE VECTOR

Velocità:

- LS:  $O(n^2)$ ;
- DV: il tempo è variabile, potrebbero esserci dei routing loops, inoltre l'algoritmo è sempre pronto a reiterare.

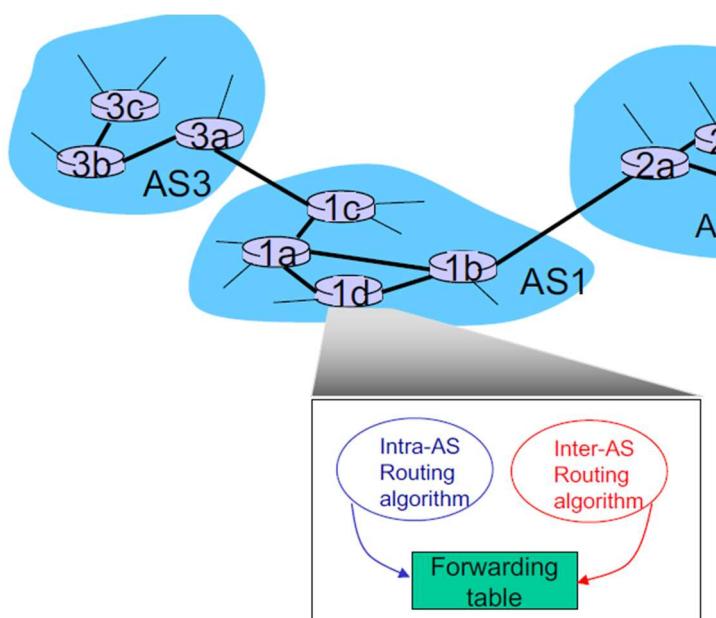
Resistenza (ai malfunzionamenti):

- LS: se un router segnala un costo di collegamento errato a un altro router-> quel router avrà una routing table sbagliata;
- DV: se un router segnala un costo di collegamento errato a un altro router-> tutti i router della rete avranno una routing table sbagliata, dato che i vari 'd' si basano sulla comunicazione dagli altri router (l'errore si propaga quindi nella rete intera).

### **INTRA-AS (Autonomous System)**

Le reti non sono "piatte" ma gerarchiche, è quindi molto difficile creare un algoritmo che non lavori su router "allo stesso livello", si creano quindi distinzioni gerarchiche, detti **AUTONOMOUS SYSTEMS**" (noti come "domini").

Una famiglia di router che usano la stessa routing table sono quindi definiti come AS e per loro vengono definiti algoritmi detti Intra-AS, che collegano router che hanno soprattutto stessi protocolli di routing. Per connettere poi i router di un AS a quelli di un altro AS si usano gli algoritmi detti Inter-AS.



Si spezza quindi il problema di forwarding in due parti: muoversi nel proprio AS e muoversi tra AS.

Es. "Voglio andare da 3b a 1a" -> per muovermi da 3b a 3a uso un algoritmo Intra-AS, per poi andare da 3a a 1c uso un algoritmo Inter-AS, per poi andare da 1c a 1a un Intra-AS.

E' quindi facile per un algoritmo come Dijkstra capire come spostarsi da un AS a un altro: usa gli unici collegamenti esistenti, riuscendo quindi a stabilire direttamente i cammini Inter-AS.

Nel caso Inter-AS quindi non si tratterà per forza di congestione o di larghezza di banda parlando di "costo del collegamento", ma anche di prezzi di collegamento tra ISP diversi, ad esempio, oppure di logiche di sicurezza tra nazioni diverse, ecc.

I protocolli Intra-AS sono anche detti **IGP (Interior Gateway Protocol)** e i più comuni sono:

- RIP: Routing Information Protocol;
- OSPF: Open Shortest Path First (essenzialmente uguale a IS-IS);
- IGRP: Interior Gateway Routing Protocol (proprietà di Cisco fino al 2016).

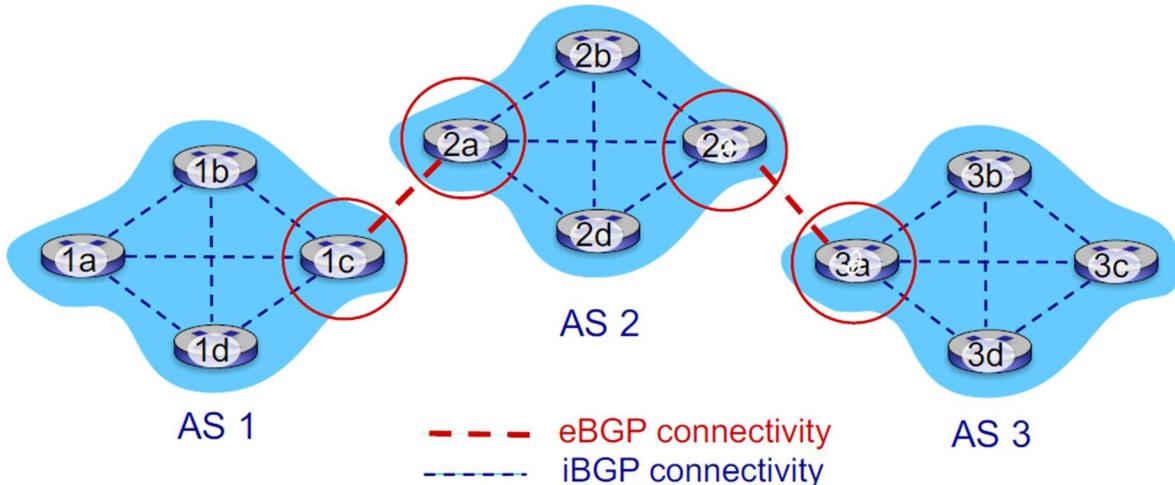
OSPF fornisce anche una certa sicurezza, garantendo che tutti i messaggi siano autenticati, evitando messaggi da routing "intromessi" nella rete. Inoltre, questo protocollo gerarchizza ulteriormente i router Intra-AS, definendo, ad esempio, boundary routers i router che si connettono ad altri AS, che necessitano quindi sia un algoritmo Intra-AS che uno Inter-AS.

### **INTER-AS**

Il protocollo principale è il **BGP (Border Gateway Protocol)**, esso è definito come "la colla che tiene insieme tutto internet", poiché permette appunto che ogni AS sia connesso agli altri AS, garantendo la globalità di internet. Esso fornisce a ogni AS due componenti informative:

- eBGP (external BGP): ottiene le informazioni sulla raggiungibilità dei router di altri AS;
- iBGP (internal BGP): propaga queste informazioni a tutti i router interni all'AS.

Esso determina quindi i router migliori degli altri AS da raggiungere a partire dall'AS in questione. Questo funzionamento permette quindi a tutte le sottoreti di essere riconosciute dalla rete globale come raggiungibili e, soprattutto, permette di sapere come raggiungerle.



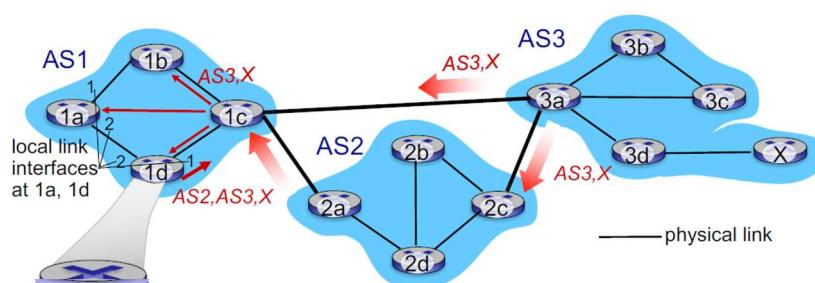
gateway routers run both eBGP and iBGP protocols

Esempio di gestione dell'informazione da ottenere e propagare da parte di BGP

Per scambiarsi informazioni si crea una sessione BGP, ovvero uno scambio di messaggi attraverso una connessione TCP semi-permanente, tra i router di "confine". Se ad esempio si unisse una nuova rete X all'AS-3, 3a avviserebbe 2c (via eBGP) che lui è il router migliore da cui passare per mandare messaggi a X. 2c quindi, via iBGP, avviserebbe tutti i router di AS-2. A quel punto 2a, avviserebbe 1c (via eBGP) che lui è il router migliore da cui passare per arrivare a X. Infine, 1c, via iBGP, avvisa tutto l'AS-1 di queste informazioni.

#### MESSAGGI PROTOCOLLO BGP:

- OPEN: apre una connessione TCP con un BGP peer remoto, autenticando il BGP peer mittente;
- UPDATE: avvisa di un nuovo cammino;
- KEEPALIVE: tiene la connessione aperta anche se non vengono inviati update (funge anche da ACK per l'OPEN);
- NOTIFICATION: avvisa di errori nel messaggio precedente (serve anche a chiudere la connessione).



Esempio di aggiornamento di una routing table in seguito all'aggiunta di un router X.

dest	interface
...	...
X	1
...	...

Se la tabella fosse quella relativa a un OpenFlow ovviamente sarebbe più complessa.

## **SDN CONTROL PLANE**

SDN si basa sull'esistenza di una struttura che calcola tutte le tabelle di instradamento necessarie ai router della rete, favorendo un sistema scalabile che non si preoccupa della specificità dei singoli router, ma elabora i percorsi in maniera generalizzata, permettendo quindi una manutenzione più efficiente e una maggiore semplicità di aggiornamento dei protocolli.

**Due problemi:**

- Single Point Of Failure: un solo sistema che deve gestire l'intera rete, se fallisce tutta la rete crolla;
- Enorme congestione possibile: tutti i dati delle reti devono arrivare al sistema centrale e da quest'ultimo devono partire dati per tutti i router delle reti.

Ci si chiede: se un sistemista volesse far passare i dati da una sorgente a una destinazione **non** dal cammino minimo, come può fare?

Risposta possibile: cambiare i costi dei collegamenti per portare l'algoritmo a restituire il risultato sperato.

**Errato:** i costi non sono dei semplici pulsanti da poter premere o meno, ma spesso sono frutto di valutazioni specifiche relative, ad esempio, al traffico del collegamento.

Risposta **giusta**: si utilizzano sistemi come quelli OpenFlow permessi da SDN per dare comandi specifici ai singoli router (es. "se arriva pacchetto da u, mandalo in v" senza curarsi di un possibile nodo w collegato con costo minore).

Il problema di questa "potenza" di modulazione di cammini è l'eventuale errore umano: se vengono impostati manualmente percorsi impossibili o fortemente tendenti a congestione, c'è la forte possibilità di rendere la rete completamente inutilizzabile. Sarebbe inoltre difficile scoprire l'errore data la complessità e la grande mole di codice relativo ai sistemi OpenFlow.

## **TRAFFIC ENGINEERING**

Letteralmente "ingegneria del traffico (di rete)", è lo scopo finale del sistema SDN, ovvero poter scegliere arbitrariamente quali percorsi utilizzare per un determinato traffico, in base a QoS, priorità dei pacchetti, ecc.

## **SNMP (Simple Network Management Protocol)**

E' un protocollo che sfrutta degli "agenti" definiti per ogni rete che comunicano tra di loro fornendosi dettaglio riguardo alle funzionalità e i problemi della rete.

## **ICMP (Internet Control Message Protocol)**

E' un protocollo che fornisce una serie di messaggi (principalmente di errore) necessario per funzioni come Ping o Traceroute.

Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

Traceroute, ad esempio, dato un host destinazione, restituisce il RTT relativo a ogni router attraversato per raggiungere la destinazione.  
Vengono inviati delle serie di pacchetti UDP con TTL crescente (il primo ha 1, il secondo 2 ecc.) e quando la serie n-esima arriva all' n-esimo router viene restituito un messaggio ICMP (tipo 11 codice 0) al mittente, che così sa quale sia il router n-esimo e ne ha calcolato il RTT grazie al messaggio ICMP. Traceroute si ferma una volta raggiunta la destinazione, oppure se la porta indicata è irraggiungibile, oltre che, ovviamente, se il mittente cessa di inviare pacchetti.

# CRITTOGRAFIA

La sicurezza nelle reti ha come obiettivo principale, la garanzia di:

1. **Riservatezza:** solo mittente e destinatario devono essere in grado di leggere il contenuto del messaggio
2. **Integrità del messaggio:** il messaggio non deve subire alterazioni durante il trasporto
3. **Autenticazione:** mittente e destinatario devono autenticarsi per avere maggior sicurezza nella comunicazione
4. **Accessibilità e disponibilità:** i servizi devono essere accessibili e disponibili agli utenti finali

I principali attacchi alla sicurezza sono:

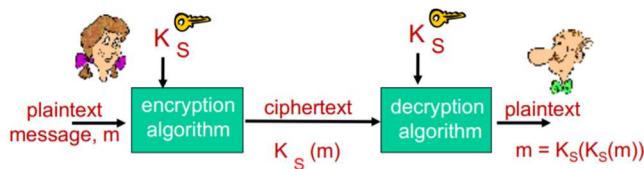
- **Eavesdrop:** Origliare (attuato attraverso un packet sniffer)
- **Insert:** inserire messaggi all'interno di una conversazione
- **IP spoofing:** falsificare il proprio IP, per fingere un altro
- **Hijacking:** dirottamento dei pacchetti, cambiando IP sorgente, destinazione
- **Denial of Service:** attacco Dos, impedire che il servizio venga usato

Nel linguaggio della crittografia, il testo in chiaro prende il nome di PLAINTEXT, mentre il testo cifrato CHIPERTEXT. Gli algoritmi di cifratura si basano sull'uso di chiavi. Esistono due diversi tipi di cifratura: a chiave simmetrica o a chiave pubblica.

Per forzare uno schema di crittografia, si possono effettuare diversi attacchi:

- **Attacco al testo cifrato:** Trudy ha accesso al chipertext ed effettua un'analisi statistica
- **Attacco con testo in chiaro noto:** Trudy possiede alcuni accoppiamenti tra chipertext e plaintext
- **Attacco con testo in chiaro scelto:** Trudy ottiene la forma cifrata di un messaggio a lui noto.

## CRITTOGRAFIA A CHIAVE SIMMETRICA



Mittente e destinatario condividono la stessa chiave  $K_s$ , con cui, il mittente cifrerà il messaggio, mentre il destinatario lo decifrerà

**Esempi base di cifrari simmetrici sono:**

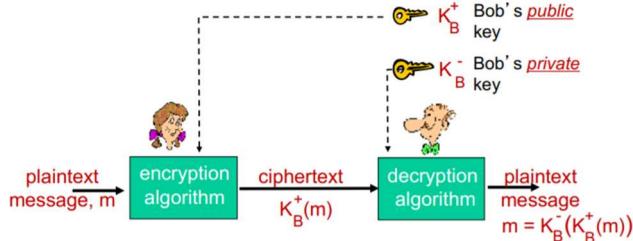
1. **Cifrario di Cesare:** scelto un  $k$ , si shiftano tutte le lettere di  $k$  posizioni (con  $k=3$ , a->d, b->e ecc...)
2. **Sostituzione di caratteri:** si crea una permutazione dell'ordine delle lettere, e si cifra il messaggio. Si possono creare più pattern e usarli ciclicamente, ad esempio con  $n$  pattern ( $M_1, \dots, M_n$ ) e fissato un  $k=3$  con:  $M_2, M_3, M_7$ , potremmo cifrare una qualsiasi parola di qualsiasi lunghezza seguendo quei  $k$  pattern ciclicamente. Ad esempio, dogs-> d da  $M_2$ , o da  $M_3$ , g da  $M_7$ , s nuovamente da  $M_2$
3. **DES:** (Data encryption standard) Usa una chiave simmetrica di 64 bit (di cui 8 di controllo e 56 usati effettivamente per la cifratura). L'algoritmo è noto. Sono le chiavi utilizzate ad essere segrete. Durante la cifratura il testo in chiaro viene diviso in blocchi da 8byte (64 bit) e vengono applicate 16 operazioni di trasposizione, usando due porzioni della chiave di 28 bit ciascuna delle quali viene ruotata a sinistra di un certo numero di bit che dipende dal numero di iterazione. Infine, si esegue una trasposizione inversa alla iniziale. Durante la decifratura, vengono svolte le stesse operazioni, con la stessa chiave ma nell'ordine inverso. DES è stato violato nel '98, e si è passati a 3DES o AES.
4. **3DES:** applica DES per 3 volte. Cifra il testo con la prima chiave, lo decifra con la seconda e lo cifra nuovamente con la terza. Si possono usare 3 chiavi diverse, 2 uguali e 1 diversa, o una sola chiave per 3 volte, ma equivale ad applicare DES
5. **AES:** processa dati a blocchi da 128 bit, usando chiavi da 128, 192 o 256 bit. Con il metodo forza bruta, se per provare tutte le chiavi su DES ci mettessimo 1 secondo, per AES impiegheremmo 149 trilioni di anni.

NB. Il principale problema nella crittografia simmetrica è lo scambio della chiave, che ovviamente non può avvenire in chiaro. Si sviluppa così la crittografia a chiave pubblica.

## CRITTOGRAFIA A CHIAVE PUBBLICA

La crittografia a chiave pubblica si basa su una coppia di chiavi per ogni utente, una pubblica (conosciuta da tutti) e una privata (conosciuta solo dal possessore). Sfrutta funzioni matematiche calcolabili ma non invertibili. Per ogni utente, la coppia di chiavi

(pubblica, privata) è interscambiabile, vale a dire che se un utente cifra un dato messaggio con la sua chiave privata, poi può decifrarlo con la sua pubblica e viceversa.



Esempio di funzionamento crittografia a chiave pubblica.

Alice cifra il testo con la chiave pubblica di Bob, così che solo Bob con la sua chiave privata può decifrare il messaggio.

Il principale algoritmo a chiave pubblica è RSA

#### Funzionamento RSA

1. Generazione di 2 numeri primi p, q (da 1024 bits)
2.  $n = p \cdot q$  (circa 2048 bits)
3.  $z = (p-1) \cdot (q-1)$
4. e = tra tutti i coprimi di z, cioè i numeri che non hanno divisori in comune con z, scegliamo uno casuale
5. d = scegliamo d tale che  $(e \cdot d - 1) \bmod z = 0$ , cioè che  $e \cdot d - 1$  sia divisibile per z
6. la coppia  $(n, e)$  sarà la nostra chiave pubblica, mentre la coppia  $(n, d)$  sarà la nostra chiave privata
7. Dato m, definiamo  $c = m^e \bmod n$ , il testo cifrato (chipertext). Dato c, definiamo  $m = c^d \bmod n$ , il testo decifrato(plaintext)

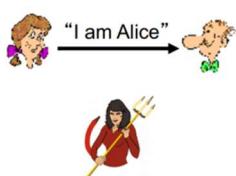
NB. La dimensione di  $m < n$ , altrimenti dovremmo “spezzettare” il messaggio m in più parti e applicare l’algoritmo di cifratura a tutte le sotto parti.

$K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$  Dati  $K_B^+$  e  $K_B^-$ , rispettivamente chiave pubblica e privata di B, allora vale la seguente equazione. È una delle principali proprietà delle cifrature a chiave pubblica.

#### Autenticazione nelle reti

Il protocollo ap (authentication protocol) è un protocollo usato per autenticarsi in rete. Ha varie versioni

##### AP 1.0



Alice non fa altro che autenticarsi a Bob mandando un messaggio in cui conferma di essere Alice.

Errore: Trudy potrebbe mandare lo stesso messaggio a Bob fingendosi Alice

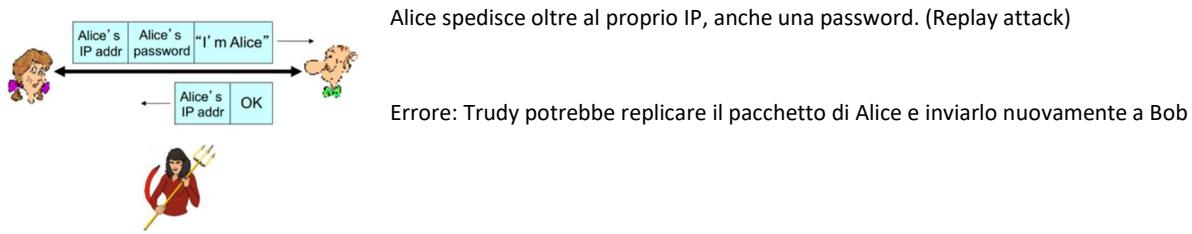
##### AP 2.0



Alice manda un messaggio a Bob contenente la sua “autenticazione” e il suo IP.

Errore: Trudy può generare il suo pacchetto inserendo nell’header l’indirizzo IP di Alice, facendo IP spoofing

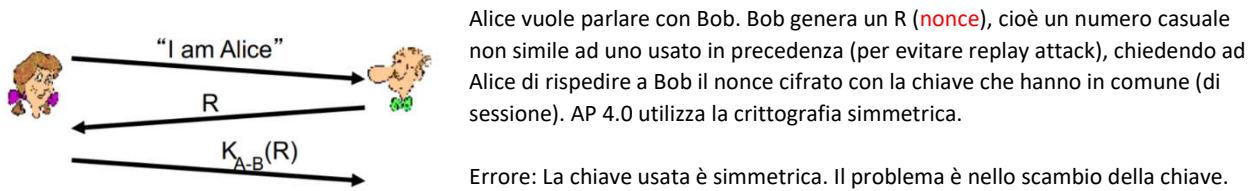
### AP 3.0



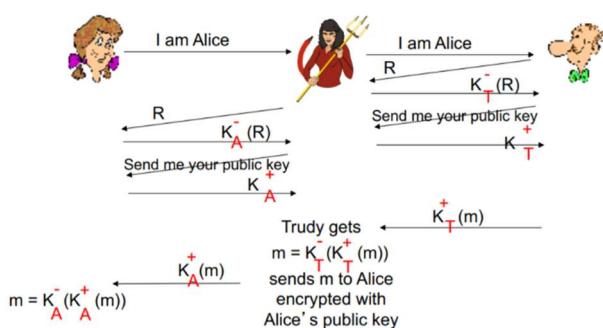
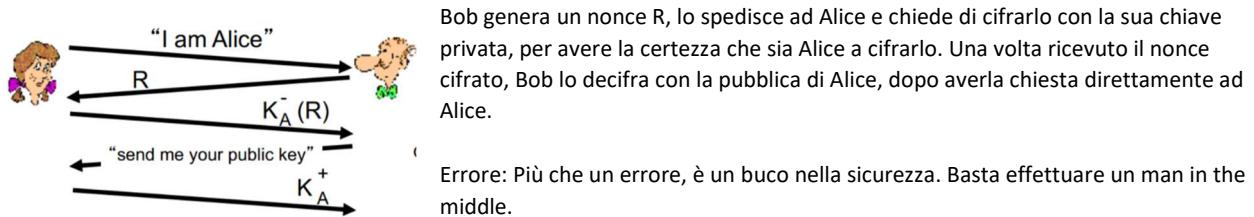
### AP 3.1

Uguale al 3.0, ma la password viene cifrata. Anche qui il Replay attack (o playback attack) funziona, dato che i bit che funzionavano nella comunicazione tra Alice e Bob, vengono accettati nuovamente nella comunicazione tra Trudy (che invia il pacchetto replicato) e Bob.

### AP 4.0



### AP 5.0



In questo caso, Trudy è il man in the middle, e come si può vedere dallo schema, Trudy si interpone tra Alice e Bob cambiando IP e chiavi. In questo caso Bob è convinto di comunicare con Alice.

Trudy cifra il nonce con la sua chiave privata, e invia la sua chiave pubblica a Bob.

Erre: la chiave pubblica non è poi così pubblica. Il sistema di reperimento della chiave pubblica è errato.

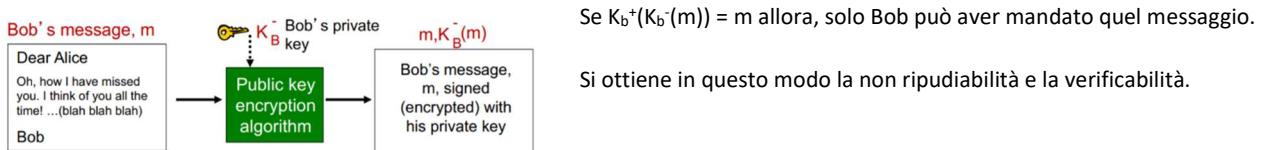
### Come risolviamo questo problema?

Esistono degli enti chiamati Certification authority. Quando una persona E vuole registrare la sua chiave pubblica alla CA, deve identificarsi. La CA crea un certificato che associa la persona E alla sua chiave pubblica. Nel momento in cui un altro utente vuole comunicare con E, chiede la chiave pubblica direttamente alla CA, la quale provvederà ad inviarla cifrandola con la chiave privata della CA ( $K_{ca}^-$ ), in modo tale da garantire l'autenticazione della CA.

## Firma digitale

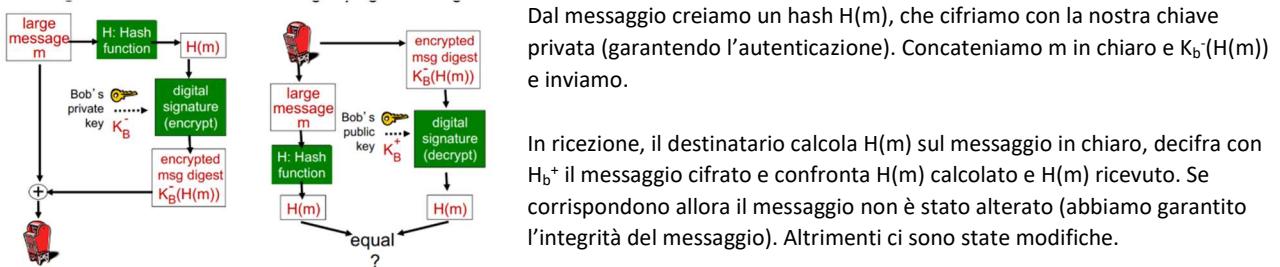
È una tecnica crittografica simile alla firma cartacea. Ma come facciamo a garantire che un messaggio sia stato mandato esattamente da qualcuno? Vogliamo garantire i principi di verificabilità e di non ripudiabilità.

Per ottenere una firma digitale, basta cifrare il messaggio con la propria chiave privata. In questo caso, il messaggio viene spedito sia in chiaro, che in forma cifrata. Una volta raggiunta la destinazione, il messaggio viene decifrato con la pubblica del mittente, e viene confrontato con il messaggio in chiaro.



## Message digest (hashing di un messaggio)

Una problematica della firma digitale è la grandezza del messaggio, mandare 2 volte il messaggio, e cifrare il messaggio per intero quando ha dimensioni grandi, risulta un'operazione pesante. Si usa in questo caso una funzione di hash. Sono funzioni che prendono in input un messaggio grande e danno in output una specie di "impronta digitale" del messaggio. La caratteristica di queste funzioni è che per due messaggi diversi  $m, m'$  avremo  $\text{hash}(m) \neq \text{hash}(m')$ . Per garantire integrità del messaggio e autenticazione del mittente, possiamo quindi attuare il seguente schema



I principali algoritmi di message digest sono: MD5 (message digest 5, genera un hash a 128 bit) e SHA-1 (Secure Hash algorithm, genera un hash a 160 bit).

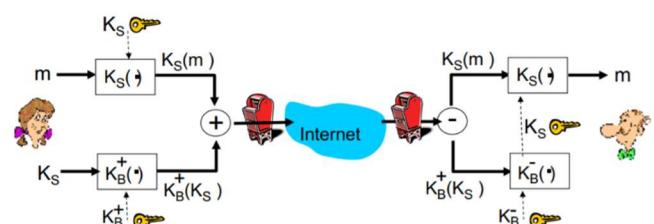
Unendo funzioni Hash a crittografia, possiamo garantire insieme: Integrità del messaggio, Autenticazione e Riservatezza.

## Cifratura ibrida (chiavi di sessione)

Il problema della cifratura a chiave pubblica è la pesantezza degli algoritmi. Per ovviare a questo problema, si può usare la cifratura ibrida. Consiste nel generare una chiave di sessione simmetrica (usata per la cifratura a chiave simmetrica) e di trasmetterla al destinatario cifrandola con la sua chiave pubblica, per garantirne la riservatezza. In questo modo, se Alice vuole comunicare con Bob, non deve fare altro che accordarsi su quale algoritmo a chiave simmetrica usare (DES, 3DES, AES ecc), generare una chiave di sessione simmetrica, cifrarla con la pubblica di Bob ( $K_B^+$ ) e inviarla a Bob. Solo Bob potrà decifrare la chiave simmetrica con la sua chiave privata ( $K_B^-$ ) e successivamente potranno usare la cifratura simmetrica con la chiave di sessione, che sarà cambiata ad ogni sessione.

## E-mail sicura

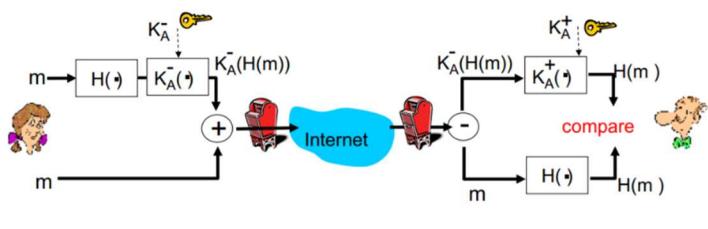
Se Alice volesse mandare una mail a Bob garantendo la confidenzialità della mail:



Alice genera la chiave di sessione  $K_S$ , cifra il messaggio con quella, e cifra la chiave di sessione con la pubblica di Bob, e manda tutto a Bob.

Bob decifra con la sua chiave privata la chiave di sessione, e con la chiave di sessione ottenuta, decifra il messaggio.

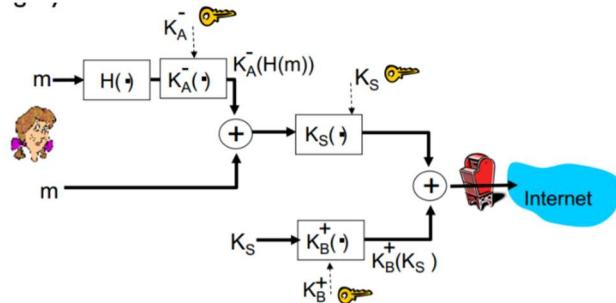
Se Alice volesse inviare un messaggio a Bob garantendo Autenticazione e Integrità del messaggio:



Alice calcola l'hash del messaggio e lo cifra con la sua chiave privata (garantiamo autenticazione). Concatena hash cifrato e messaggio in chiaro e invia a Bob.

Bob decifra l'hash con la pubblica di Alice, calcola l'hash sul messaggio ricevuto in chiaro, se coincidono non ci sono state modifiche (garantiamo integrità) ed è stata per forza Alice a mandarlo.

Per garantire contemporaneamente autenticazione, integrità e segretezza:



Alice calcola  $H(m)$  e lo cifra con la sua privata (autenticazione), concatena  $m$  e cifra tutto con la chiave di sessione (riservatezza), cifra la chiave di sessione con la pubblica di Bob e concatena tutto e invia in rete.  
In ricezione, Bob decifra con la sua privata la chiave di sessione, con la chiave di sessione decifra il messaggio contenente messaggio in chiaro e hash cifrato. Con la pubblica di Alice decifra hash, calcola hash sul messaggio in chiaro e se corrispondono abbiamo integrità dei dati.

## SSL-TLS

SSL (Secure Sockets Layer) è un protocollo usato per rendere sicura la comunicazione tra due socket in TCP. Si interpone tra il livello di trasporto (TCP) e il livello applicazione, fornendo delle API al livello applicazione.

TLS è una variazione del protocollo SSL. Entrambi servono a fornire un livello trasporto sicuro.

SSL/TLS fornisce segretezza, autenticazione e integrità dei dati. Viene utilizzato lo schema precedente delle e-mail sicure. Utilizza un set di chiavi segrete che durano per tutta la connessione. Durante la fase di handshake vi è lo scambio di certificati.

### Fase di handshake

Il client manda un "Hello" al server. Il server in risposta manda un "Hello", il suo Certificato e un "ServerHelloDone". Il client successivamente invia il suo certificato, e scambia le informazioni per generare la chiave simmetrica. Inoltre invia una specifica degli algoritmi di cifratura (simmetrici e pubblici) supportati e dichiara chiuso il suo handshake. Il server risponde inviando le specifiche di cifratura scelte e si chiude la fase di handshake. Si possono inviare i dati cifrati. Una volta finita la trasmissione, si chiude la connessione. In SSL viene generato un MS (Master secret) dal client che viene inviato cifrato al server con la sua chiave pubblica.

### Key derivation

In SSL, vengono usate chiavi diverse per la cifratura dei messaggi e per la cifratura del MAC (da non confondere con il MAC protocol di livello 2), che corrisponde ad un hash del messaggio. È consigliato usare una stessa coppia di chiavi per sessioni differenti. Vengono generati dal client dei nonce, dai quali server e client si ricaveranno la propria coppia di chiavi ( $K_c M_c$  usate dal client per cifrare il messaggio e il MAC, e  $K_s M_s$  usate dal server).

### Sequence number

Per evitare il replay attack, ogni pacchetto contiene un nonce, se arriva un pacchetto con un nonce già usato, vuol dire che è stato mandato nuovamente attraverso un replay attack

### Control information

Per evitare truncation attack (chiusura anticipata della connessione da parte di Trudy), si inserisce allora un record di tipo (record type), settato ad 1 per indicare la chiusura, e a 0 per indicare la trasmissione di dati.

### SSL chiper suit

Si scelgono algoritmo a chiave simmetrica, a chiave pubblica e un algoritmo MAC per la comunicazione.

Solitamente vengono scelti AES per la simmetrica, RSA per la pubblica e RC2/RC4 per il MAC.

### SSL record

Contiene un byte che indica il tipo di contenuto, 2 byte per la versione di SSL, 3 bytes per la lunghezza del campo data, e i campi data, MAC cifrati con le chiavi di sessione.

### IP Sec

IP Sec è un protocollo di livello rete (livello 3) per rendere sicure le comunicazioni a quel livello. IPsec è alla base delle VPN (virtuale private network) che sono reti nelle quali possiamo accedere da remoto comportandoci come se fossimo fisicamente connessi alla rete. Dobbiamo poter spedire pacchetti a livello IP in modo tale che non si capisca da dove stiamo spedendo e cosa stiamo spedendo, quindi dobbiamo garantire integrità, segretezza e autenticazione a livello IP, costruendo un tunnel cifrato con la VPN.

### Firewall

Il compito principale di un firewall è quello di filtrare i pacchetti. Solitamente è implementato su un router.

Esistono 3 tipi di firewall:

#### Stateless firewall

Filtrano tutti i pacchetti e sono firewall senza stato (non tengono conto dello stato delle connessioni). Ad esempio possiamo bloccare tutte le connessioni telnet (sulla porta 23) dirette ad un determinato host.

Policy	Firewall Setting
No outside Web access.	Drop all outgoing packets to any IP address, port 80
No incoming TCP connections, except those for institution's public Web server only.	Drop all incoming TCP SYN packets to any IP except 130.207.244.203, port 80
Prevent Web-radios from eating up the available bandwidth.	Drop all incoming UDP packets - except DNS and router broadcasts.
Prevent your network from being used for a smurf DoS attack.	Drop all ICMP packets going to a "broadcast" address (e.g. 130.207.255.255).
Prevent your network from being tracerouted	Drop all outgoing ICMP TTL expired traffic

Droppiamo tutti i pacchetti diretti a qualsiasi IP, sulla porta 80 (servizio HTTP)

Droppiamo tutti i pacchetti TCP SYN diretti ad ogni IP tranne al 130.207.244.203 sulla porta 80

Droppiamo tutti i pacchetti UDP in entrata eccetto quelli DNS e quelli broadcast dei router

Droppiamo tutti i pacchetti ICMP in uscita ad un indirizzo broadcast

Droppiamo tutti i pacchetti ICMP con TTL scaduto in uscita

Possiamo così definire delle ACL (Access Control List) per il filtraggio dei pacchetti.

action	source address	dest address	protocol	source port	dest port	flag bit
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----
deny	all	all	all	all	all	all

Nella prima riga, un pacchetto è inviato da un host della rete 222.22/16 ed è destinato ad un host fuori dalla rete 222.22/16 con protocollo TCP, con un numero di porta del mittente >1023 e un numero di porta del destinatario = 80, viene consentito (allow)

#### Stateful firewall

Tiene traccia di ogni connessione TCP attiva. Questi firewall monitorano e tracciano le connessioni TCP, tenendo conto delle connessioni che sono state aperte. Se vede arrivare una richiesta FIN di una connessione che non è mai stata aperta la blocca. Nella tabella ACL degli stateful firewall, viene aggiunta una colonna "Check Connection", che se settata ad 1, obbliga il firewall a controllare la tabella delle connessioni TCP di cui tiene traccia, prima di applicare la regola corrispondente indicata dalla ACL.

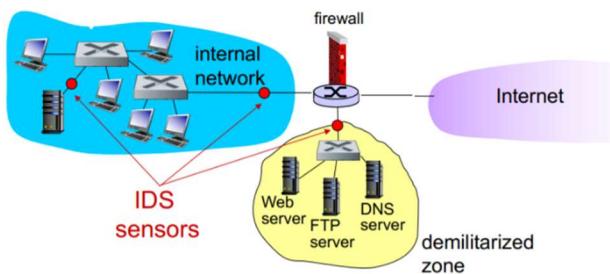
#### Application gateway firewall

Possiamo inserire nella nostra rete una macchina che farà da application gateway e utilizzarla per istaurare delle connessioni. Ad esempio, se un host volesse instaurare una connessione telnet, prima creerebbe una connessione ssh con l'application gateway nella rete e poi incaricherebbe quest'ultimo di aprire la connessione telnet con un host già identificato.

### DMZ (Zona demilitarizzata)

Area all'interno della rete in cui ci sono host che danno servizi ad utenti esterni alla rete. I Server vengono messi in una DMZ

la rete privata è isolata da un firewall, solo i server nella DMZ sono raggiungibili da Internet.



## Wireless Livello Fisico

## Proprietà delle Radio Frequenze

Come vengono generate le radiofrequenze?

L'**energia elettromagnetica** è generata da corrente alternata ad alta frequenza nelle antenne. Le antenne, convertono quindi la corrente elettrica in segnali a Radiofrequenza e viceversa

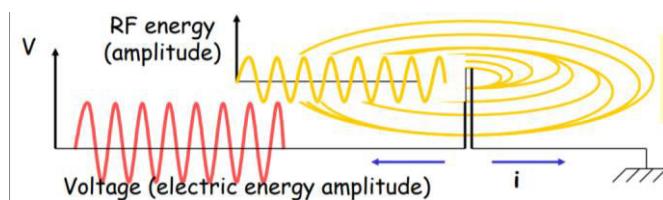
Il **Voltaggio (V)** varia sinusoidalmente ai due estremi di un'antenna creando uno sbilanciamento di carica, tanto maggiore è lo sbilanciamento di carica più elettroni “corrono da sinistra a destra e da destra a sinistra”

(Le **onde elettromagnetiche** si generano quando gli elettroni di un corpo si muovono, e tipicamente ogni oggetto materiale genera **onde** costantemente in condizioni ordinarie.

Fonte: <https://vivalascuola.studenti.it/come-generare-onde-elettromagnetiche-156993.html#:~:text=Le%20onde%20elettromagnetiche%20si%20generano,onde%20costantemente%20in%20condizioni%20ordinarie.>)

Una carica che si sposta nello spazio induce un'emissione di onde elettromagnetiche (il modo in cui varia il segnale rosso fa variare il segnale giallo)

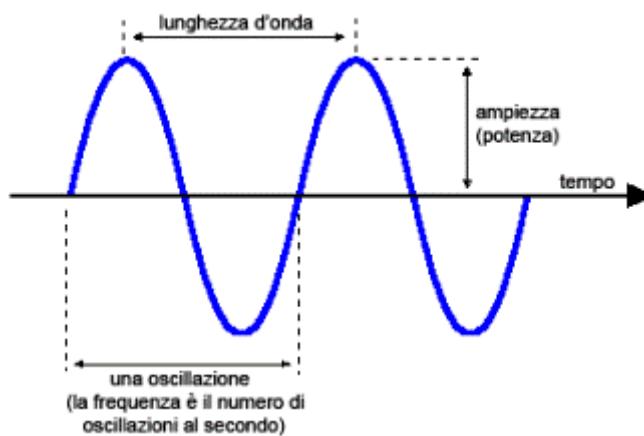
Il mezzo di propagazione delle onde radio è il vuoto.



**Aampiezza:** Graficamente, è la misura della distanza tra la base e la cima (la cresta). L'ampiezza rappresenta l'energia del segnale, e quindi la quantità di potenza energetica utilizzata nella trasmissione.

Per generare un segnale con ampiezza maggiore, abbiamo quindi bisogno di più energia.

**Transmission Power (Watts) = Energy / Time**



**Frequenza:** indica quante volte in un'unità di tempo la funzione si ripete (“Numero di cicli completi della sinusoide del segnale per unità di tempo”)

La **lunghezza d'onda** radio è pari alla velocità della luce diviso la frequenza (Praticamente sarebbe uguale alla lunghezza dell'onda che si propaga nello spazio).

# Wavelength = c / frequency

I WiFi ha lunghezza d'onda 12,5 cm.

$$\text{Wave Length} = 300.000.000(\text{m/s}) / 2.400.000.000 \text{ Hz} = 12.5 \text{ cm}$$

Nella pratica, esiste una proprietà fisica che dice che le antenne, per essere più efficienti, devono avere una lunghezza pari a 1,  $\frac{1}{2}$ ,  $\frac{1}{4}$  (o comunque, deve essere un sottomultiplo binario) della lunghezza d'onda del segnale che stiamo utilizzando. Infatti, le antenne dei router WiFi ad esempio, hanno le antenne di lunghezza pari a 12,5 cm e anche le antenne integrate negli Smartphone hanno lunghezza par ad un sottomultiplo binario della lunghezza d'onda.

## Propagazione RadioFrequenze

Una problematica nel mondo delle radiofrequenze ha a che fare con la loro propagazione; non è possibile ottenere una copertura generale, ovvero non è possibile ottenere un segnale che una volta trasmesso, arrivi ovunque.

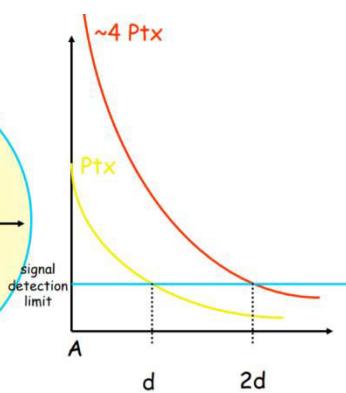
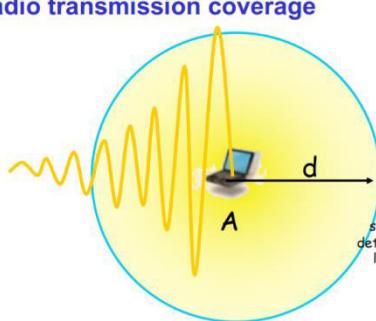
Il segnale parte con ampiezza molto alta ma all'aumentare della distanza l'ampiezza tende a diventare quasi nulla, cioè l'energia del segnale radio è molto bassa ed è come non riceverla.

L'energia radio però è capace di trasmettersi a grandi distanze. L'energia radio della sonda Voyager 1 ad esempio, che è ormai fuori il sistema solare, è potenzialmente ancora ricevibile, e la potenza utilizzata dal trasmettitore della sonda è pari a 50W (potenza pari a quella di una lampadina ad uso comune, da comodino).

Il segnale decade però in modo esponenziale. La Signal Detection Limit (linea azzurra nel grafico giù) è la minima energia necessaria al quale ricevere il segnale per capire informazioni (bit) dall'onda radio. Esiste una distanza  $d$  oltre la quale il segnale arriva con un'energia quasi pari (al limite) alla Signal Detection Limit. La circonferenza formata da questa distanza  $d$  (grafico 1 in basso) è pari quindi alla **Radio Trasmission Coverage**, ovvero alla zona in cui abbiamo una copertura necessaria per poter trasmettere.

Fuori da questa circonferenza, non è possibile instaurare un link di comunicazione (e quindi, non avrei un segmento di rete locale). Ci sono due modi per risolvere questo problema: o avvicinare il client o aumentare la potenza trasmittiva del trasmettitore.

### ▪ Radio transmission coverage



← Un'altra cosa molto importante da notare è che, per poter raddoppiare la distanza **1.** alle trasmettere, dobbiamo quasi quadruplicare la potenza trasmittiva del segnale in partenza (e quindi la potenza con il quale il segnale viene generato).

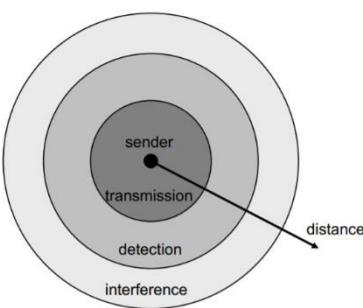
Formule molto importanti:

In 3D sphere:  $V=(4 \pi r^3/3)$   $S=(4 \pi r^2)$

V rappresenta il voltaggio, e quindi la potenza necessaria per emettere il segnale.

## Range di propagazione del segnale Wireless

- **Transmission range**
  - communication possible
  - low error rate
- **Detection range**
  - detection of the signal possible
  - no communication possible
- **Interference range**
  - signal may not be detected
  - signal adds to the background noise

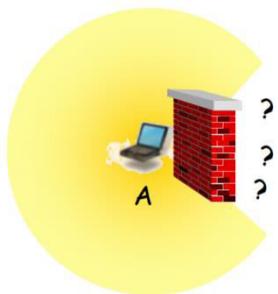


Ranges depend on receiver's sensitivity!

Queste 3 differenti situazioni creano ambiguità per il mac protocol (che deve agire e compiere delle operazioni differenti in base alla zona in cui si trova)



### ▪ Radio transmission coverage



Rules of thumb:

- high frequencies are good for short distances and are affected by obstacles
- low frequencies are good for long distances and are less affected by obstacles

obstacles can reflect or absorb waves depending on materials and wave frequencies

Gli ostacoli possono assorbire o riflettere le onde (dipende dal materiale e dalla frequenza delle onde).

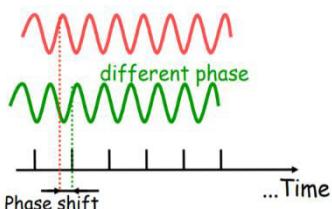
Gli ostacoli bloccano segnali con frequenze più alte (ad esempio, la luce del sole che ha frequenza altissima, è possibile bloccarla anche con un foglio)

## Fase di un segnale

La **Fase** è lo spostamento sull'asse del tempo della sinusoide dell'onda radio rispetto ad un segnale di riferimento.

"La fase varia in base al punto in cui mi colloco, il segnale impiega un certo tempo  $t$  per raggiungermi. Più mi allontano e più il segnale risulta essere in ritardo"

La fase è positiva se rispetto al segnale di riferimento ha uno spostamento a sinistra (segna in anticipo) e negativa altrimenti.



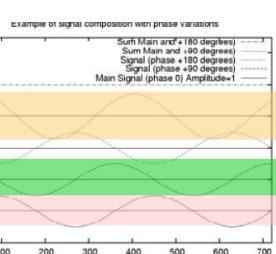
Due segnali con stessa ampiezza e frequenza, ma fase differente. Anticipo e ritardo corrispondono a quantificare gli slittamenti dell'onda.

Problema: il segnale che ricevo ad una certa distanza arriva sia per via diretta che facendo una strada più lunga (rimbalzano su degli ostacoli, ad esempio), ed i segnali che percorrono una strada più lunga hanno uno sfasamento maggiore.

Il segnale che parte dal trasmettitore è lo stesso; il segnale che il ricevente riceve è pari alla **somma vettoriale** tra segnale diretto e segnale che ha fatto una strada più lunga (con sfasamento maggiore).

In ogni istante di tempo, l'energia che l'antenna del ricevente cattura è sia quella del segnale diretto che del segnale che ha fatto la strada più lunga, e se i segnali hanno dei valori differenti il segnale risultante è la somma vettoriale dei segnali catturato dall'antenna.

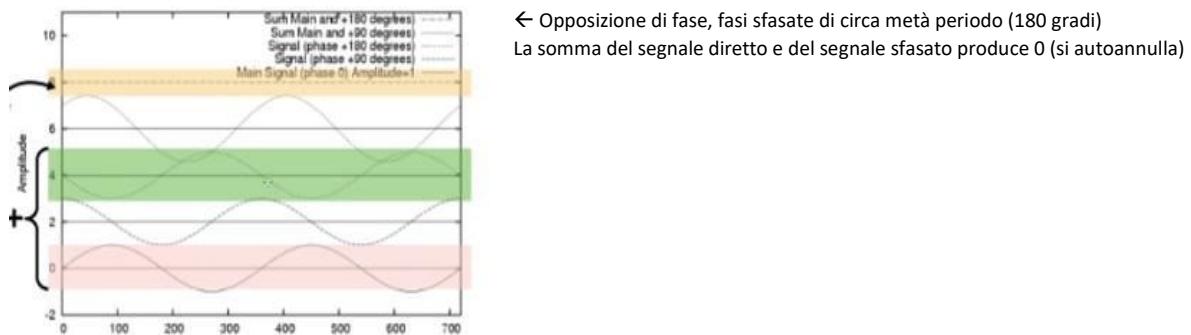
- Positive phase (left-shift), early wavefront
- Negative phase (right-shift), late wavefront



Sommiamo punto a punto (vettorialmente) il valore della sinusoide verde e della sinusoide rosa otteniamo la sinusoide arancione. In questo caso, avrei un guadagno di ampiezza.

Lato ricevente, ovviamente è preferibile ricevere segnali risultanti con ampiezze maggiori. In alcuni casi però, la somma vettoriale (punto per punto) di più segnali ha come risultante 0 (anche a pochi cm dal trasmittente, è possibile avere una somma di segnali che si autoannulla). Questo fenomeno viene chiamato **Opposizione di fase**.

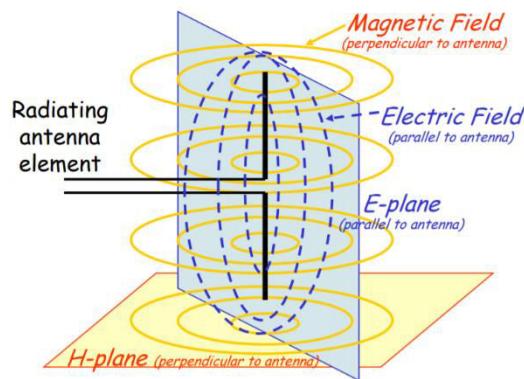
Possiamo immaginare questo anche attraverso i suoni, immaginando due segnali acustici opposti emessi da una sorgente, che sovrapposti si annullano.



## Polarizzazione

### Polarization: (physical orientation of antenna)

- RF waves are made by two perpendicular fields:
  - Electric field and Magnetic field



Horizontal Polarization  
(electric field is parallel to ground)

Vertical Polarization  
(electric field is perpendicular to ground)



La polarizzazione è, fisicamente, l'orientamento dell'antenna.

I piani su cui si trasmettono le onde radio sono 2, Magnetico (perpendicolare all'antenna) ed Elettrico (parallelo all'antenna).

“Questi due piani si autosostengono e viaggiano ruotati di 90 gradi”.

Se l'antenna viene posizionata in modo orizzontale la polarizzazione è Orizzontale, ed il campo elettrico è orizzontale come il pavimento.

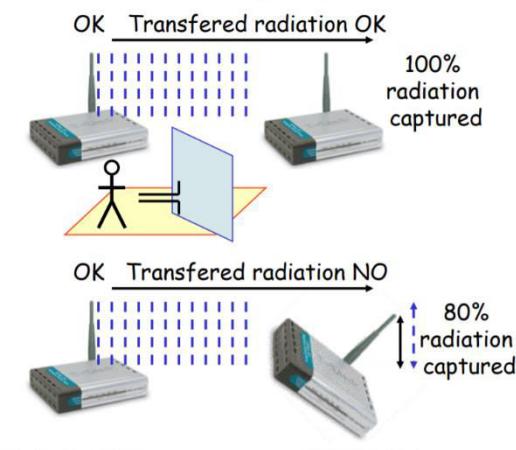
La polarizzazione Verticale (antenne poste in verticale) è tipicamente utilizzata nelle wlan.

Il massimo grado di trasferimento si ha quando l'antenna del ricevente è polarizzata come l'antenna del trasmittitore , altrimenti avremmo una perdita di energia del segnale trasferito, e potremmo rischiare di scendere (proprio a causa di questa perdita dovuta alla polarizzazione) sotto la Sygnal Detection Limit (linea blu grafico precedente).

In ambiente chiuso (indoor) la polarizzazione delle antenne è poco importante, poiché l'onda radio rimbalza su tutti i muri e ci sarà sempre un'onda che rimbalzando arriva alla stessa polarizzazione dell'antenna del ricevente.

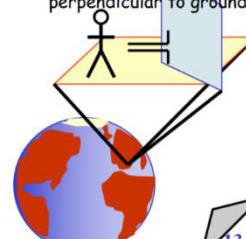
In ambiente Outdoor invece, la polarizzazione è molto importante; Soprattutto in un ambiente come lo spazio cosmico ad esempio, in cui il segnale deve percorrere grandi distanze in assenza di ostacoli, la polarizzazione è importantissima.

### Vertical Polarization: typically used in WLANs



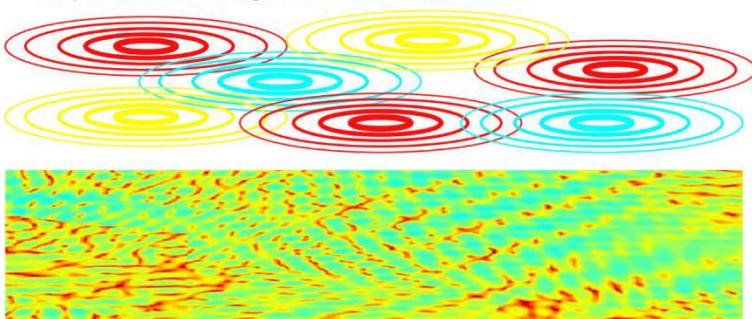
← Rappresentazioni intuitive

Vertical Polarization  
(electric field is perpendicular to ground)

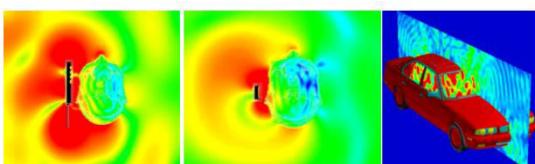


## Comportamento RadioFrequenze

- **Radio transmission interference**  
<http://www.met.rdg.ac.uk/clouds/maxwell/>



Lato ricevente, quello che è possibile captare è dinamicamente la composizione di fasi e frequenze di tanti segnali che si sommano vettorialmente generando interferenze.  
Sono i protocolli che riescono a portare ordine in questo sistema (e quindi, utilizzando l'immagine come esempio, riesco a capire quale sia la frequenza gialla, quale quella rossa e quale quella azzurra).



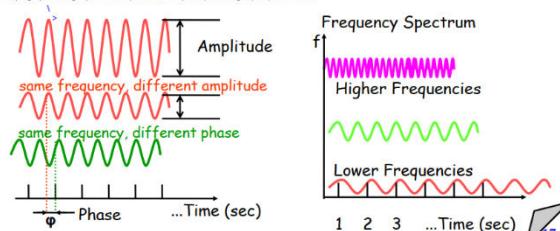
**Effetto radiofrequenze in auto:** simile a gabbia di Faraday, annulla il valore del campo elettromagnetico al di fuori della gabbia stessa. Un parallelo intuitivo è immaginare l'auto come una barriera fonoassorbente, lo smartphone che cerca di collegarsi ‘urla’ energia, che viene abbattuta dall'auto.  
Il cellulare si scarica prima perché utilizza più energia per cercare di connettersi al trasmettitore. Effetto che tempesta di Microonde chi sta in auto.

## Wireless transmission: Electromagnetic waves

### Different parameters of electromagnetic waves:

- amplitude  $M$  proportional to transmission energy (loudness)
- frequency  $f$  (tone) measured in Hertz (Cycle/sec)
- phase  $\phi$  (peak shift with respect to reference signal) (rad)

impiega più energia per essere prodotta, trasporta più energia (dati) al ricevente



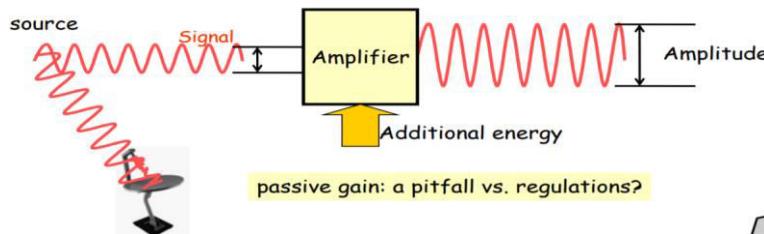
Le **onde elettromagnetiche** con ampiezza maggiore impiegano più energia per essere prodotte, ma trasporta più energia (dati) al ricevente

## Trasmissione Wireless

-Guadagno del segnale: (**Misurato in Db**, Decibels)

Aumenta in ampiezza

Guadagno **attivo** di energia: energia guadagnata ad esempio attraverso un amplificatore, che prende in input la sinusoide restituisce in output la stessa sinusoide variando solo l'ampiezza (attraverso una sorgente di energia, ad esempio una batteria)



Guadagno **passivo** di energia: è un altro modo di guadagnare energia, ad esempio attraverso una parabola. La parabola tende a catturare energia in quantità direttamente proporzionale alla grandezza del disco. L'energia viene catturata e focalizzata in un punto.

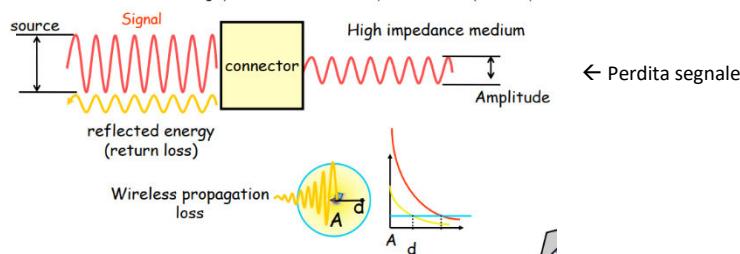
Questo è un esempio di guadagno passivo, poiché nessuno aggiunge una quantità di energia additiva come nel caso precedente, ma viene solo sfruttata al meglio l'energia trasmessa dal trasmettitore.

-Perdita del segnale: (Db)

Diminuisce ampiezza segnale. Esistono due tipi di perdite:

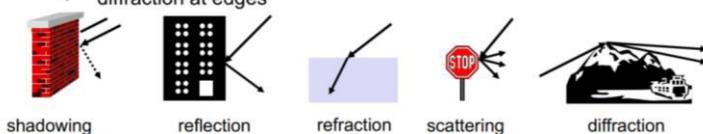
Perdita **Intenzionale**, ad esempio attraverso delle resistenze, l'ampiezza del segnale viene ridotta trasformando l'energia in calore.

Perdita attraverso **Ostacoli**: perdita del segnale 'non voluta', dovuta ad ostacoli, ad esempio in caso di nebbia, che rappresenta un ostacolo per le microonde che scaldando le particelle d'acqua di cui è composta la nebbia, facendo ridurre l'energia del segnale (e quindi l'ampiezza), riducendo la distanza che esso può percorrere.



## Effetti propagazione del segnale

- Propagation in free space always like light (straight line)
- Receiving power proportional to  $1/d^2$  ( $d$  = distance between sender and receiver)
- Receiving power additionally influenced by
  - fading (frequency dependent)
  - shadowing
  - reflection at large obstacles
  - refraction depending on the density of a medium
  - scattering at small obstacles
  - diffraction at edges



La propagazione del segnale è influenzata da questi ↑ effetti.

**Shadowing**: impedisce che l'onda prosegua

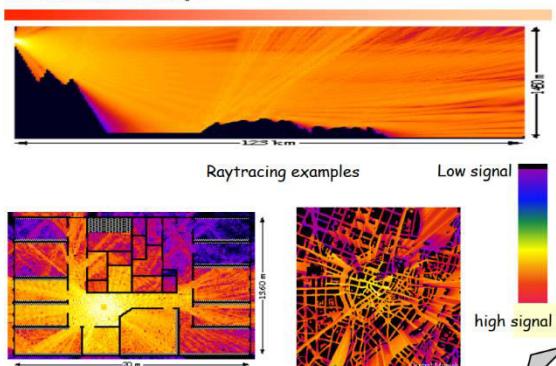
**Riflessione**: l'onda viene riflessa

**Refrazione**: causata dal cambio di dielettrico (ad esempio aria -> acqua) cambia leggermente la direzione del segnale.

**Difrazione ai bordi**: una deviazione, dovuta a bordi a curvatura molto alta (ad esempio i bordi di una montagna)

**Scattering**: quasi come un rimbalzo, quando l'onda colpisce uno spigolo vivo

#### Real world example



Tutti questi effetti di propagazione del segnale si traducono ad avere i problemi raffigurati nelle seguenti ← immagini.

A seconda di dove ci poniamo, il segnale ha energia diversa.

Prima immagine: raffigura la propagazione del segnale in una zona montuosa.

E' probabile che si formino dei coni d'ombra, come ad esempio ai piedi della montagna, che corrispondono a dei punti in cui abbiamo assenza di segnale.

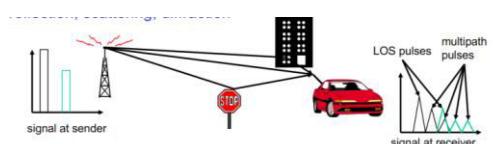
Le striature in foto rappresentano i rimbalzi dell'onda verso l'alto.

Seconda immagine: rappresenta la propagazione del segnale in un edificio, in alcune stanze ad esempio (quelle viola) il segnale non arriva.

Terza immagine: Rappresenta la propagazione del segnale in una città (ad esempio il segnale di un ripetitore sulla cima di un edificio). Abbiamo una buona propagazione nelle strade intorno, tranne due coni d'ombra dovuti molto probabilmente alla presenza di alcuni grattacieli che ombreggiano la propagazione del segnale radio.

Queste problematiche sono da considerare in fase di progettazione del sistema.

Altrimenti ci generano problemi nella fase di funzionamento.

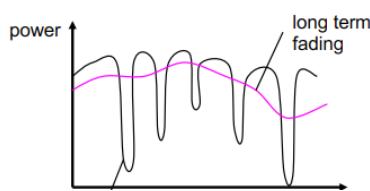


Il segnale può percorrere differenti percorsi tra trasmettitore e ricevente, e può subire vari effetti tipo riflessione, diffrazione.

Due segnali che viaggiano verso un auto (uno che arriva in modo diretto e l'altro effetto di una riflessione su un grattacielo); arrivano entrambe all'auto, ma uno (quello che ha rimbalzato sul grattacielo) arriva con un evidente ritardo di fase. (-**Dispersione temporale**: il segnale viene disperso nel tempo)

Se questo ritardo diventa importante c'è il rischio che due segnali consecutivi arrivino sovrapposti grazie all'effetto di sfasamento (l'eco del segnale che fa la strada più lunga arriva a destinazione insieme al segnale diretto, emesso successivamente). Questo effetto di sovrapposizione genera ulteriore confusione.

(come avere due persone che parlano, ed ascolto entrambe sentendo solo della confusione dovuta alla sovrapposizione delle due voci)



←Variazione di segnale che è possibile avere spostandosi in una città. Il segnale può variare a seconda della distribuzione del segnale nell'ambiente rappresentata da questa curva.

Questo effetto si chiama Fading (nascondersi del segnale). Il segnale scompare e poi riappare.

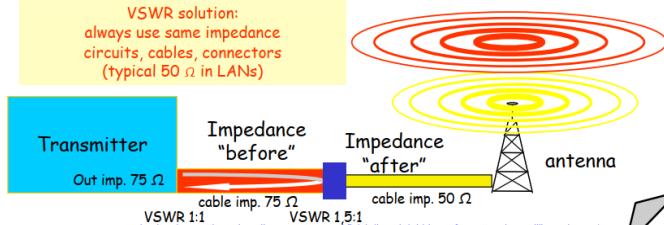
Dobbiamo cercare di superare questo effetto quando utilizziamo la comunicazione attraverso onde radio.

---

## Voltage Standing Wave Ratio (VSWR)

- **VSWR occurs with different impedance (Ohm) = resistance to AC current flow between transmitter and antenna**
  - VSWR is the cause of "return loss" energy towards the transmitter
  - Measured as ratio between impedance (before and after)
    - E.g. 1,5:1 (impedance ratio before/after is 1,5 times the ideal value)
    - 1 = normalized ideal impedance (1:1 means perfect VSWR)
  - VSWR Causes burnout of transmitter circuits, and unstable output levels

VSWR solution:  
always use same impedance  
circuits, cables, connectors  
(typical 50 Ω in LANs)



Progettando un sistema di comunicazione ogni volta che mettiamo in collegamento un dispositivo con un altro dispositivo successivo, dobbiamo fare attenzione al valore di **impedenza nominale** (ovvero alla resistenza della corrente) perché, se i due componenti hanno delle impedanze diverse, si verifica l'effetto chiamato **Voltage Standing Wave Ratio (VSWR)**; per evitare ciò dobbiamo fare attenzione che il rapporto di impedenza sia sempre 1 a 1, se non è 1 a 1 abbiamo un effetto di dissipazione di energia (e ciò non è mai positivo poiché abbiamo consumato dell'energia per generare quell'energia persa, senza poi avere degli effetti

positivi dal punto di vista della comunicazione).

In un sistema di comunicazione, tutto il blocco (esclusa l'antenna) viene chiamato **Intentional Radiator** (Radiatore intenzionale, IR). Misurando l'energia di un sistema di trasmissione radio, l'energia a cui facciamo riferimento è l'energia che arriva all'antenna. Quest'energia viene calcolata nel sistema come **Intentional Radiator Power Output** (ovvero la potenza di uscita dell'Intentional Radiator), che non è l'energia del trasmettitore, ma l'energia dello stesso meno tutte le perdite causate da collegamenti, connettori, etc..., fino all'ultimo elemento prima dell'antenna.

Un'antenna le cui caratteristiche sono quelle di concentrare le energie in un punto dello spazio (una parabola, ad esempio), equivale a dare dell'energia a questa antenna, ma essa non viene diffusa in tutte le direzioni omogeneamente (come una lampadina che illumina in tutte le direzioni).

La lampadina è l'equivalente di un **radiatore isotropico**, ovvero un'antenna che emette energia omogeneamente nello spazio. **Se un'antenna concentra l'energia almeno in una direzione preferenziale, avremmo una concentrazione di energia in quella direzione maggiore rispetto ad un Radiatore Isotropico.** Il valore **EIRP** (Equivalent Isotropically Radiated Power) è il valore dell'energia di **Intentional Radiator Power Output** che viene consegnata all'antenna equivalente a quell'energia che servirebbe a un radiatore isotropico per generare lo stesso effetto di onda elettromagnetica nella direzione preferenziale. (Energia che dovremmo dare ad un'antenna isotropica affinchè generi la stessa energia della parabola)

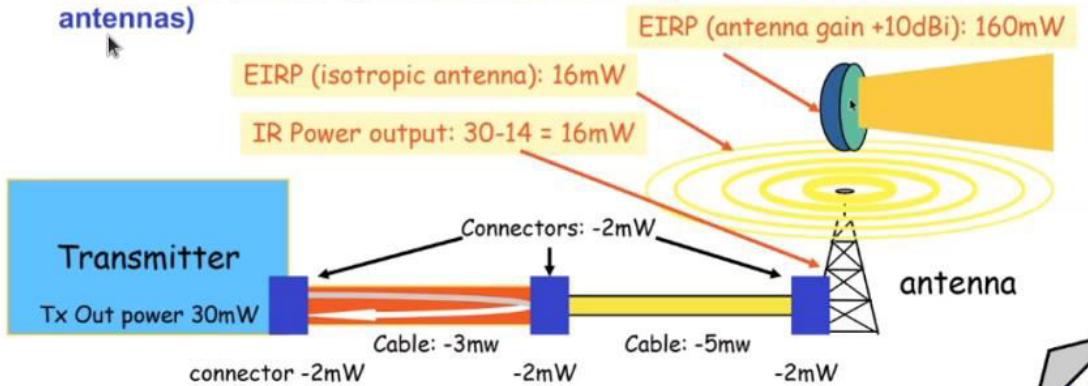
Quindi:

**IR** = sistema di comunicazione (esclusa l'antenna).

**IR Power Output** = energia del trasmettitore meno tutte le perdite fino all'ultimo elemento prima dell'antenna (che sarebbe proprio l'energia che arriva all'antenna).

**Equivalent Isotropically Radiated Power** = valore dell'energia IR Power Output che viene consegnata all'antenna equivalente a quella energia che servirebbe a un Radiatore Isotropico per generare lo stesso effetto di onda elettromagnetica nella direzione preferenziale.

- **Equivalent Isotropically Radiated Power (EIRP): the power radiated by the antenna (including the passive antenna gain effect of directional antennas)**



### Esempio EIPO:

Se diamo 16mW direttamente all'antenna isotropica, l'energia irradiata in tutte le direzioni è 16mW. Se invece diamo 16mW ad un'antenna che focalizza 10 volte l'energia in una direzione preferenziale (sottraendola alle altre direzioni ovviamente) il valore EIRP di questa nuova antenna è il valore che dovremmo dare ad un isotropico poiché esso possa irradiare 160mW, e quindi in questo caso il valore EIRP è pari a 160mW.

Quando l'antenna non è isotropica quindi EIRP mi dice quale sarebbe l'energia dell'isotropica equivalente.

### Rappresentazione schematica sistema radio trasmissivo

Un sistema di trasmissione (IR) è quindi composto da un trasmettitore, da cavi e vari dispositivi di connessione che connettono il trasmettitore all'antenna (tra le quali possiamo avere delle perdite, **VSWR**), ai bordi dell' IR abbiamo un IR Power Output destinato all'antenna che genera energia che viene irradiata in base al tipo di antenna utilizzato.

Una volta generata, quest'energia decade velocemente con la distanza, con la legge di propagazione, e quest'energia sarà sufficiente a ricevere il segnale radio fino a quando è superiore alla Receiver Sensitivity Limit.

**WATT:** Unità di misura potenza elettrica

Nella progettazione di sistemi a radiofrequenza l'unità di misura utilizzata è il **Db**.

Il **Decibel** (Db) è un'unità di misura per esprimere in modo semplice le perdite di potenza (ma anche i guadagni)

Il Db è nato soprattutto per esprimere le perdite che, possono arrivare a 11, 12, 13 cifre dopo la virgola, diventando difficili da gestire per eseguire dei calcoli (Il Db è una Scala logaritmica e non lineare)

Il Db mappa i mW ma su scala Logaritmica.

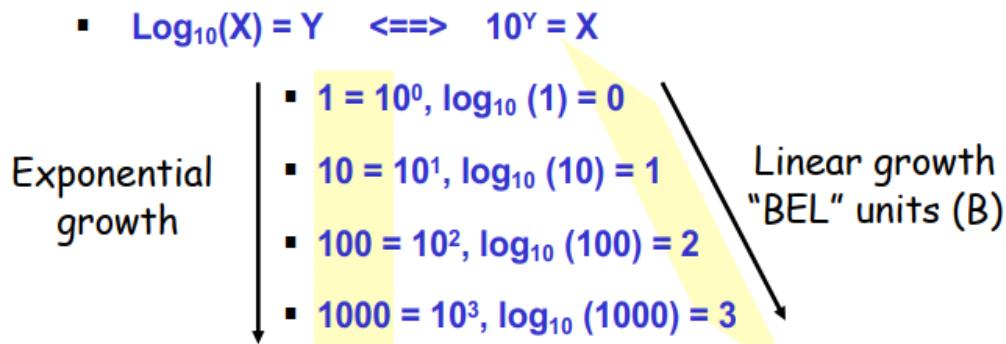
Il Db misura, in scala logaritmica, la forza relativa tra due segnali

**Importante:** Un valore in Db non è mai una quantità di energia pura, ma rappresenta sempre una differenza di potenza (energia) tra due riferimenti. Serve a marcare le differenze tra due livelli di potenza, in

scala logaritmica.

“A quanto corrisponde un segnale forte 10Db” ← Non vuol dire nulla, non ha senso X

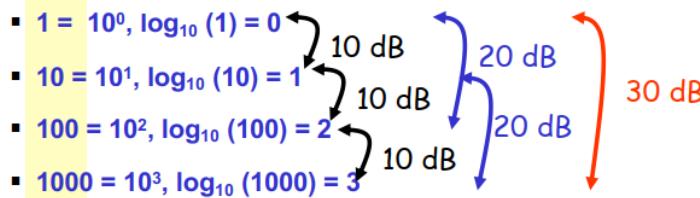
“Quanto è forte un segnale a 10Db rispetto ad un valore iniziale X” ← OK



La scala a destra è utile a rappresentare una differenza di potenza tra due livelli.

- Decibel (dB): 1/10 of a Bel
- E.g. 1000 is one Bel greater than 100 => 1000 is 10 dB greater than 100

Linear  
signal  
difference  
(factor)



$$1 \text{ Db} = 1/10 \text{ Bel}$$

Passare da 0 Bel a 1 Bel, corrisponde a fare un salto di 10Db  
Ogni volta che sommiamo 10Db stiamo passando da un valore (Bel) ad un valore successivo.

Sommare 10Db su scala logaritmica equivale a moltiplicare per 10 i valori su scala lineare.

Se ho 1mW ed aggiungo 10Db vuol dire che amplifico quel 1mW trasformandolo in 10mW.

Se ho invece 10mW ed applico 10Db di guadagno, amplifico i 10mW trasformandolo in 100mW.

Un valore in Db positivo rappresenta un guadagno di potenza, un valore in Db negativo rappresenta una perdita di potenza.

**Power Difference (dB) =  $10 * \log(\text{Power Rx(Watt)} / \text{Power Tx (Watt)})$**

|

Rapporto Minore di 1 per natura poiché di solito quello che trasmettiamo si attenua con la distanza.

Il logaritmico produce un valore negativo tra 0 e -Inf

Questa funzione trasforma in Bell il rapporto tra segnale ricevuto in segnale trasmesso

Moltiplicando questo valore per 10, ottengo un valore in Db.

Se Power Difference (Db) negativo rappresenta una perdita, altrimenti se positivo vuol dire che è stato collocato un amplificatore nel sistema.

<b>-3 dB</b>	<b>½ power in mW (/ 2)</b>
<b>+3 dB</b>	<b>2x power in mW (* 2)</b>
<b>-10 dB</b>	<b>1/10 power in mW (/ 10)</b>
<b>+10 dB</b>	<b>10x power in mW (* 10)</b>

Approximated table (values defined for ease of calculations)

- N.B. **dBs** are additive measures of gain (loss): e.g.  $6\text{dB} = +3+3\text{ dB}$ ,  $7\text{dB} = 10-3\text{ dB}$ 
  - E.g.  $100\text{ mW} -6\text{ dB} = 100\text{ mW} -3 -3\text{ dB} = 100 / 2 / 2 = 25\text{ mW}$
  - E.g.  $100\text{ mW} +7\text{ dB} = 100\text{ mW} +10 -3\text{ dB} = 100 *10 / 2 = 500\text{ mW}$
  - E.g.  $10\text{ mW} + 5\text{ dB} = 10\text{ mW} (+10+10-3-3-3-3)\text{dB} = 1000/32 = 31.25\text{ mW}$
  - E.g.  $10\text{ mW} + 11\text{ dB} = ?$
  - E.g.  $50\text{ mW} - 8\text{ dB} = ?$

N.B. Approximated values (values defined for ease of calculations)

**dBm** = dB-milliWatt, unità di misura della potenza del segnale

Assumiamo che  $1\text{mW} = 0\text{dBm}$

Attenzione:

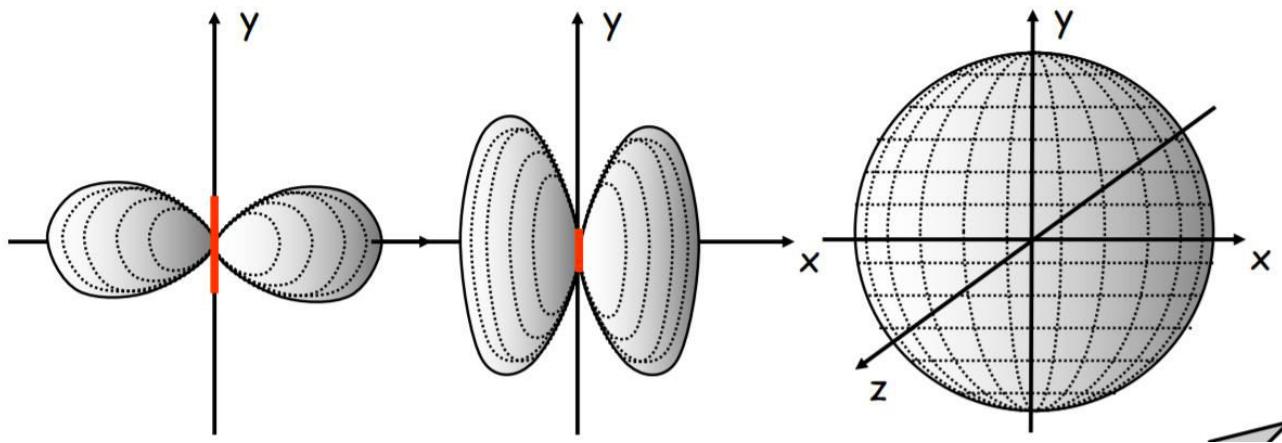
“A quanto corrisponde un segnale forte  $10\text{dBm}$ ” ← Ha senso solo se ci sta la m (e quindi in dBm e non in Db)

Tabella conversione da mW a dBm

<b>-40</b> dBm	<b>-30</b> dBm	<b>-20</b> dBm	<b>-10</b> dBm	<b>0</b> dBm	<b>+10</b> dBm	<b>+20</b> dBm	<b>+30</b> dBm	<b>+40</b> dBm
100 nW	1 μW	10 μW	100 μW	1 mW	10 mW	100 mW	1 W	10 W

<b>-12</b> dBm	<b>-9</b> dBm	<b>-7</b> dBm	<b>-6</b> dBm	<b>-3</b> dBm	<b>0</b> dBm	<b>+3</b> dBm	<b>+6</b> dBm	<b>+7</b> dBm	<b>+9</b> dBm	<b>+12</b> dBm
62,5 μW	125 μW	200 μW	250 μW	500 μW	1 mW	2 mW	4 mW	5 mW	8 mW	16 mW

**dB<sub>i</sub>** = dB-Isotropic viene utilizzato per rappresentare guadagni o perdite da parte delle antenne (ha come riferimento normativo l'antenna isotropica, che spalma il segnale in tutte le direzioni, a forma sferica)



La forma più circolare e più sferica possibile è ottenuta con la più semplice delle antenne, il [Dipolo](#), che è un segmento conduttore con due poli + e - che oscillano in modo sinusoidale a seconda della variazione di carica applicata. Produce una corrente sinusoidale, che corrisponde in qualche modo alla variazione di potenziale applicata ai capi. La corrente sinusoidale genera l'onda elettromagnetica che si irradia nell'ambiente con una forma che assomiglia a quella di una ciambella (irradia pochissime onde radio in verticale).

Alcuni Dipoli tendono ad irradiare un po' verso l'alto ma un po' meno a destra e sinistra (Dipolo a basso guadagno, figura centrale) e altri molto schiacciati che irradiano poco verso la coordinata Y ma irradiano molto sull'asse delle X (Dipolo alto guadagno, figura sinistra).

Si dicono ad alto guadagno, poiché se ci poniamo sull'asse delle X, veniamo colpiti da più energia.

La direzione nel quale un'antenna spara la massima componente energetica si chiama **direzione preferenziale**.

I dBi possono essere utilizzati per calcolare le perdite o il guadagno di energia tramite le antenne.

### Antenne Omnidirezionali

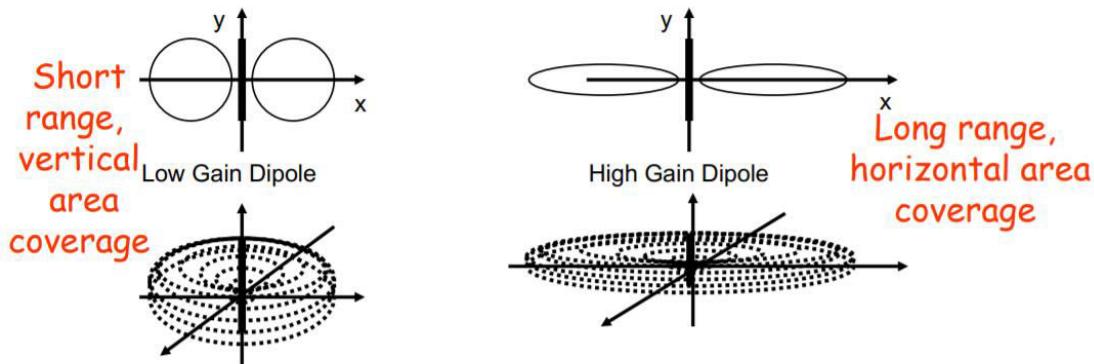
Le antenne Omnidirezionali sono le antenne che assomigliano di più all'isotropico (e quindi, che trasmettono il segnale in tutte le direzioni possibili).

L'antenna che utilizziamo di più (che è anche un esempio di antenna Omnidirezionale) nella vita di tutti i giorni è il dipolo.

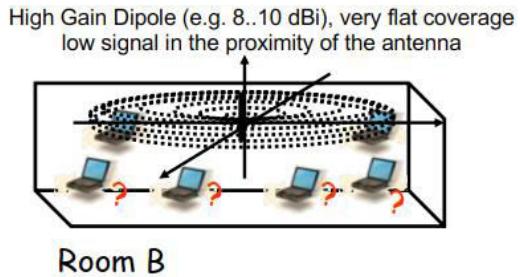
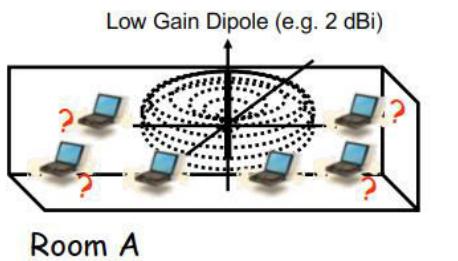
La direzione preferenziale è il piano X ed il piano Z (Emissione a ciambella)

E' il tipo di antenna più utilizzata: più semplice, omnidirezionale e più facile da realizzare (per alcuni aspetti).

### Rappresentazione grafica dipolo ad alto e a basso guadagno



I due tipi di dipolo hanno delle applicazioni differenti in base a ciò che abbiamo bisogno di realizzare

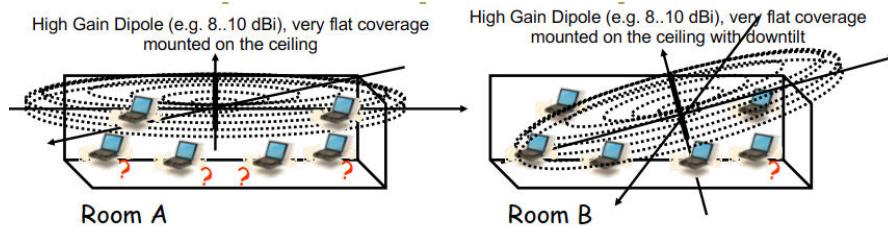


Inclinando l'antenna, otteniamo un'**antenna tilt**, ovvero un'inclinazione da dare all'antenna in modo da avere una migliore copertura.

Inclinando l'antenna verso il basso, otteniamo un **Down Tilt**.

Quando dobbiamo coprire un'area con il WiFi, dobbiamo porre attenzione al posizionamento delle antenne.

### Esempio Tilt



Anche se comunque ormai i router attuali hanno più di un'antenna; nei router di ultima generazione è utilizzata una tecnologia chiamata **MIMO**, quest'ultima fa in modo che ogni antenna generi un segnale che ottimizzi l'emissione verso il client di destinazione. Questa tecnologia quindi 'segue' il client se si sposta da una stanza all'altra come se le antenne si muovessero per offrire il "segnale a ciambella più ottimo", **spostando le fasi** e ottenendo la massima componente risultante nella direzione giusta.

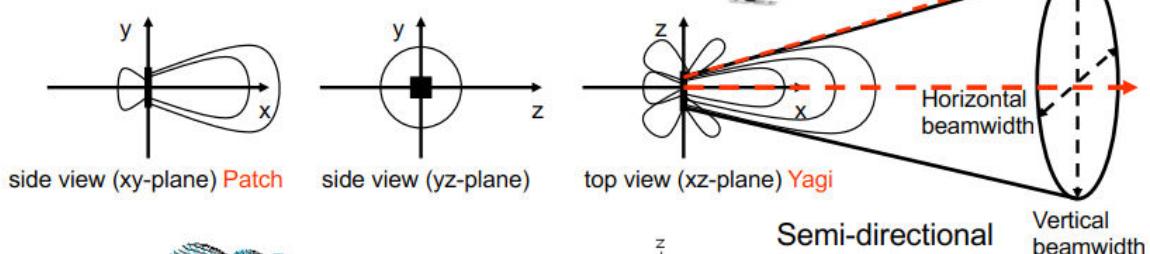
### Antenne Semi-direzionali

Le antenne Semi-direzionali trasmettono di più in una direzione (Utilizzate se non ci importa di trasmettere il segnale da tutte le parti in modo omogeneo, e da una parte non vogliamo trasmettere).

Assomigliano a delle torce elettriche (Direzione della luce = Direzione delle onde radio, a seconda da dove la punto)

Alcune antenne semi-direzionali(e le rappresentazioni grafiche della propagazione dei loro segnali) sono:

- **Patch (flat antennas mounted on walls)**
- **Panel (flat antennas mounted on walls)**
- **Yagi (rod with tines sticking out)**



Le antenne Yagi hanno un cono di ampiezza maggiore.

## Antenne Molto-Direzionali (Highly-directional, direzionali)

Sono antenne che tutta l'energia radio emessa la concentrano su un cono con ampiezza in gradi molto piccola.

Esempi:

-Parabola



-Grid

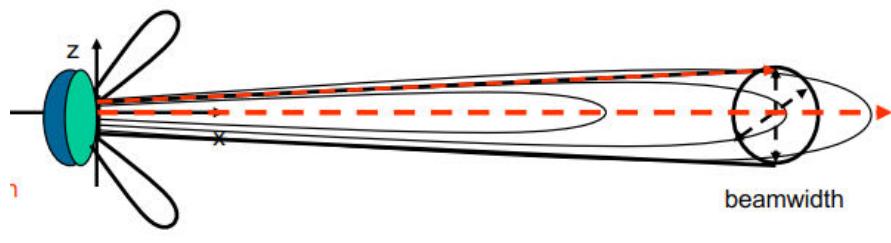
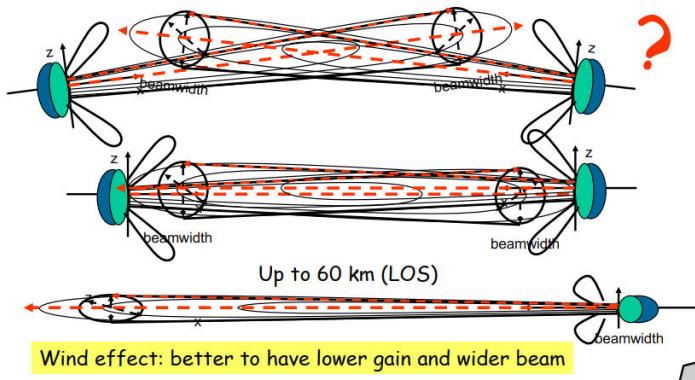


Tabella confronti direzione e coni di emissione

Antenna type	H beamwidth	V beamwidth
Omni-dir.	360°	7°.. 80°
Patch/panel	30° .. 180°	6° .. 90°
Yagi	30° .. 78°	14° .. 64°
Parabolic dish	4° .. 25°	4° .. 21°

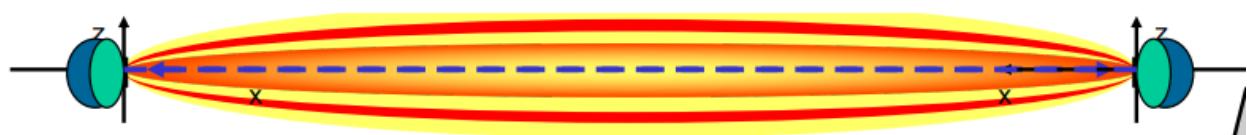
### Uso comune: Canale Point-to-Point

E' molto importante centrare correttamente le antenne (vincolarle bene in modo che non si muovano e che il vento non faccia oscillare la parabola), altrimenti i raggi possono andare fuori bersaglio e il segnale non arriva.



### Line Of Sight = Linea di vista

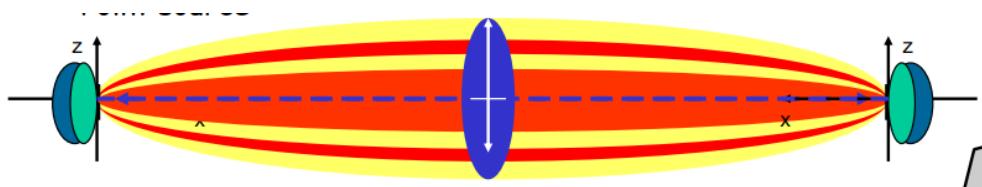
(Linea Blu nel grafico seguente, tra le due antenne)



Immaginiamo di avere due antenne. Queste due funzionano meglio nel trasferire energia quando la Line Of Sight è libera da ostacoli (mettendo in mezzo un ostacolo, si attenua l'energia che passa, disturbando la comunicazione).

La **Fresnel Zone (FZ)** è la zona centrale intorno alla [Line Of Sight](#) nella quale passano la maggior parte dei segnali di tipo additivo sul ricevente.

(Disco blu nel seguente grafico)



La Fresnel Zone deve quindi essere lasciata libera da ostacoli, altrimenti viene persa energia inutilmente. La regola che devono seguire i progettisti di sistemi radio è calcolare il raggio del disco blu con le appropriate formule e fare in modo che rimanga libero.

Il raggio di questo disco blu (Fresnel Zone) dipende da alcune costanti, ma anche dalla distanza tra le due antenne e dalla frequenza utilizzata dalla comunicazione radio.

Non incide il tipo di antenna (Due parabole ad esempio, non hanno una Fresnel Zone più piccola rispetto a due Dipoli).

Questa FZ non è rilevante in scenari InDoor.

#### [Formule per calcolare la Fresnel Zone:](#)

$$- R_{60\%} = 43.3 \times (d/4f)$$

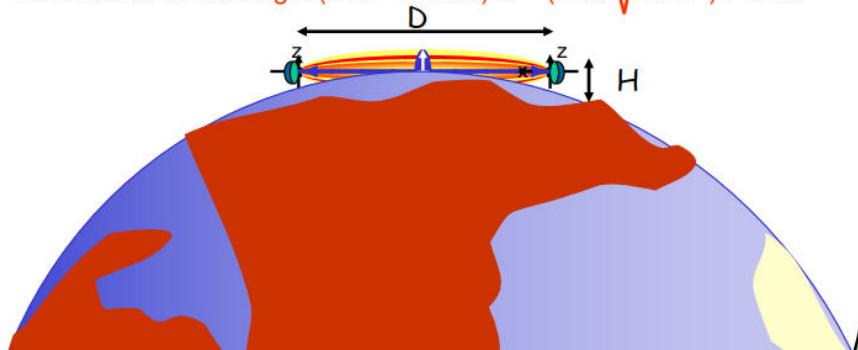
$$- R_{100\%} = 72.2 \times (d/4f)$$

Molto spesso la Fresnel Zone potrebbe essere ostruita dalla curvatura terrestre, in caso di comunicazione a grandi distanze.

La soluzione è semplice, basta spostare (verso l'alto) le parabole in modo che abbiano il disco blu completamente libero.

Per questo, nella pratica, vengono utilizzati dei tralicci (Ad esempio per i ripetitori tv/telefonia) Importante nell'immagine successiva, la formula per calcolare H, ovvero l'altezza minima a cui posizionare l'antenna.

- Very long point-to-point connections may have more than 40% FZ obstructed by Earth surface! **Earth Bulge height = h (feet) = D<sup>2</sup>/8**
- Minimum antenna height (link > 7 miles)  $H = (43.3 \sqrt{D/4F}) + D^2/8$



#### [Possibili domande orale:](#)

-Cos'è la Fresnel Zone?

-Perché va lasciata libera?

-Come la si calcola?



### Antenne Settorizzate (Sectorized, A Settore)

Utilizzando delle antenne direzionali o semi-direzionali è possibile coprire tutti i 360 gradi dell' ambiente intorno dividendo i raggi di emissione radio in 6 diversi raggi separati (ognuno di essi è chiamato Settore). Ogni stecca nella foto è un'antenna patch, emette segnale su un angolo aperto 30 gradi, questi tralicci sono utilizzati nella copertura della rete cellulare.

Con il cellulare quindi, quando utilizziamo la rete, usiamo solo una delle 6 antenne e se qualcuno utilizza un'antenna affianco, può comunicare e ricevere i suoi dati su un'emissione radio che anche se ha la stessa frequenza nelle due emissioni non fa conflitto perché sono due direzioni diverse (e quindi due settori diversi, con due antenne diverse).

Le onde radio in una direzione non si sovrappongono con le onde radio in un'altra (le zone di gestione delle antenne non si sovrappongono).

La settorizzazione delle antenne permette di utilizzare più volte la stessa frequenza verso più client a seconda di dove sono distribuiti nello spazio.

Questo è importante soprattutto per la rete cellulare, perché chi la gestisce ha pagato lo stato per poter utilizzare quelle determinate frequenze, e con quelle poche e costosissime frequenze deve cercare di far trasmettere il massimo grado di comunicazione ai propri clienti, perché facendoli comunicare il gestore (Provider di telefonia) ha un guadagno. In questo modo moltiplicano la possibilità di comunicare sulla stessa frequenza.

Non accade mai che due settori adiacenti siano lo stesso canale (per evitare dei problemi nel caso in cui un Host si pone in mezzo a due settori). Queste sequenze di emissione sono fatte 'colorando' i segnali in modo diverso (Problema del [Frequency Planning](#), pianificazione delle frequenze ("Stabilire i colori") all'interno del sistema cellulare, opportunità che abbiamo utilizzando questi array di antenne).

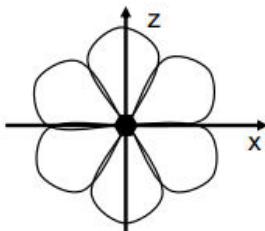
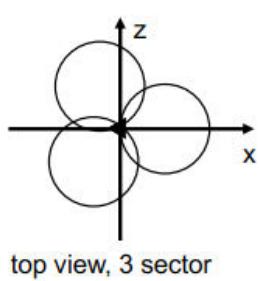
Come fa il cellulare a capire quale frequenza utilizzare?

Il telefono comunica con questa infrastruttura utilizzando una sequenza pilota che serve per fare tutta la parte amministrativa (frequenza unica che utilizzano tutti i telefoni), ascoltano quella frequenza e capiscono quali di queste sono a disposizione e in quali punti.

A questo punto, o ricevono dal ripetitore che fa lo scheduling la frequenza da utilizzare a quel telefono, o può scegliere il telefono quale frequenza utilizzare in base alle frequenze che vengono comunicate dal ripetitore come libere. Una volta fatta la scelta, i Gb di dati trasmessi viaggiano su una frequenza verso una di queste antenne in modo univoco (utilizza solo un cellulare quel canale).

Se un host si sposta, prima cambia antenna mentre si sposta (o alla stessa frequenza, o può anche cambiare). Se un host si sposta molto dal ripetitore, si aggancia al successivo ripetitore più vicino, questa operazione si chiama Hand Off o Hand Over (Passaggio di mano) .

#### ▪ **Arrays of sectorized directional antennas**



top view, 6 sector

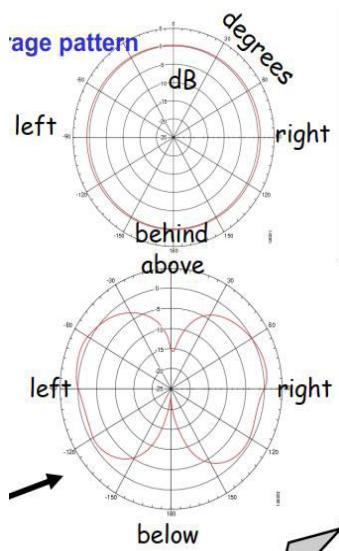


sectorized antenna

#### LETTURA PERSONALE (Non obbligatoria)

**Azimuth chart:** Modo di misurare le caratteristiche di emissione di un'antenna "girandole attorno su un piano" (pattern seen from front/right/behind/left)).

**Elevation chart:** modo di misurare le caratteristiche di emissione di un'antenna “ruotandole sopra e sotto”.  
(pattern seen front/below/behind/above)



← Rappresentazione Azimuth chart e Elevation chart di un Dipolo.

Servono non solo a capire il tipo di antenna, ma anche la forma dell'emissione radio (Shape).

Li possiamo trovare nel foglio tecnico delle antenne, e consente al progettista di sapere prevedere prima del setup del sistema quale sarà la forza/energia del segnale in qualsiasi direzione, per capire se un'antenna ha una copertura radio a cui è interessato.

### Diversity (wireless diversity schemes)

I router moderni, hanno gruppi di due o più antenne (Dipoli). Quest'antenne, sono posizionate tutte a distanza Lamda/2. Questa distanza non è casuale, ma è stabilita in modo che a quella distanza non è possibile avere opposizione di fase su tutte le antenne (avremo sempre un segnale su un'antenna che non sia in opposizione di fase).

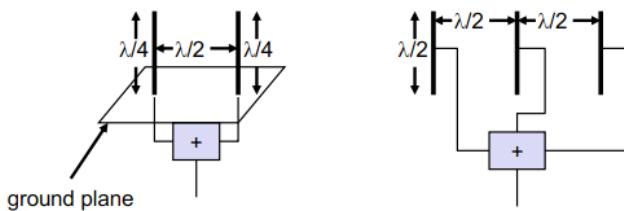
Sempre per composizione di fase, dalla diversity, estrarremo il massimo della componente additiva del segnale (con un processore) in modo da capire sempre meglio l'energia che arriva anche in condizioni non ideali.

In caso di rumore, ognuna delle antenne sente un rumore differente con un segnale comune (il messaggio); Estraiamo quindi da questo segnale la parte comune a tutte le antenne, e questo sarà il messaggio senza rumore.

#### ▪ Antenna diversity

- switched diversity, selection diversity
  - receiver chooses antenna with largest output
- diversity combining
  - combine output power to produce gain
  - cophasing needed to avoid cancellation (phased antenna array... Requires processor)

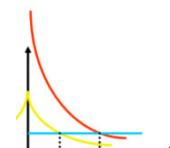
Anche lo smartphone ha più di un'antenna



### Path Loss (Perdita di segnali in funzione della distanza)

- Free space: Loss (in dB) =  $36.6 + (20 \log_{10}(F)) + (20 \log_{10}(D))$

(In questa formula non abbiamo la potenza di trasmissione perché il risultato è in Db)



Questa formula ci dice in Db quanto segnale perdiamo in funzione della distanza percorsa da quel segnale, che dipende dalla frequenza e dalla distanza.

F è in Mhz, D è in miles.

La costante 36,6 è un valore di perdita in ambiente terrestre, nel caso di valutazioni nello Spazio aperto (tra Marte e Giove) la costante moltiplicativa è pari a 32,4 (valore di perdita inferiore).

Questa formula caratterizza la forma che ha la curva rossa immagine precedente (legge di propagazione) se otteniamo il risultato in Db, qualunque potenza parta dalla trasmissione, dalla forma della curva che dipende da frequenza e distanza capire di quanto decade il segnale.

100 meters	- 80.23 dB
200 meters	- 86.25 dB
500 meters	- 94.21 dB
1000 meters	- 100.23 dB
2000 meters	- 106.25 dB
5000 meters	- 114.21 dB
10000 meters	- 120.23 dB

-6Db rules: quando raddoppiamo la distanza dal trasmittitore, il segnale che riceviamo si riduce di un fattore pari a circa  $\frac{1}{4}$ ; Questo è dovuto alla legge di propagazione, che dice che per raddoppiare la distanza percorsa dal segnale dobbiamo quadruplicare l'energia del segnale. (Fattore decadimento energia radio nell'ambiente è quadratico rispetto alla distanza)

### Link Budget

Il Link Budget è il segnale in eccesso ricevuto dal ricevente oltre a quello necessario della Signal Detection Limit (in modo da comprendere l'informazione).

Sarà quindi pari al segnale ricevuto (curva rossa →) meno il valore della Signal Detection Limit (Linea azzurra nel grafico →).

- Calculation:
  - Receiver sensitivity RS (weakest detectable signal)
    - The lower the better: e.g. IEEE 802.11 card (see device manual), -95 dBm (1Mbps), -93 dBm (2 Mbps), -90 dBm (5.5 Mbps), -87 dBm (11 Mbps)
  - Link Budget: received power (in dBm) - RS (in dBm)



Viene misurato in Db, dBm o mW.

Dobbiamo progettare il sistema in modo che questa quantità sia positiva (ovvero il segnale ricevuto sia maggiore della linea blu →).

Se il link Budget è positivo vuol dire che il canale funziona (ho comunicazione).

Dobbiamo fare in modo che anche questa quantità sia abbastanza grande (in modo da avere un margine abbastanza ampio) poiché con un margine troppo piccolo (1Db ad esempio) in caso di interferenza rischiamo di avere un segnale minore della Signal Detection Limit.

Quindi bisogna avere un Link Budget almeno pari a quelli che definiamo essere i **System Operating Margin** cioè i margini operativi del sistema.

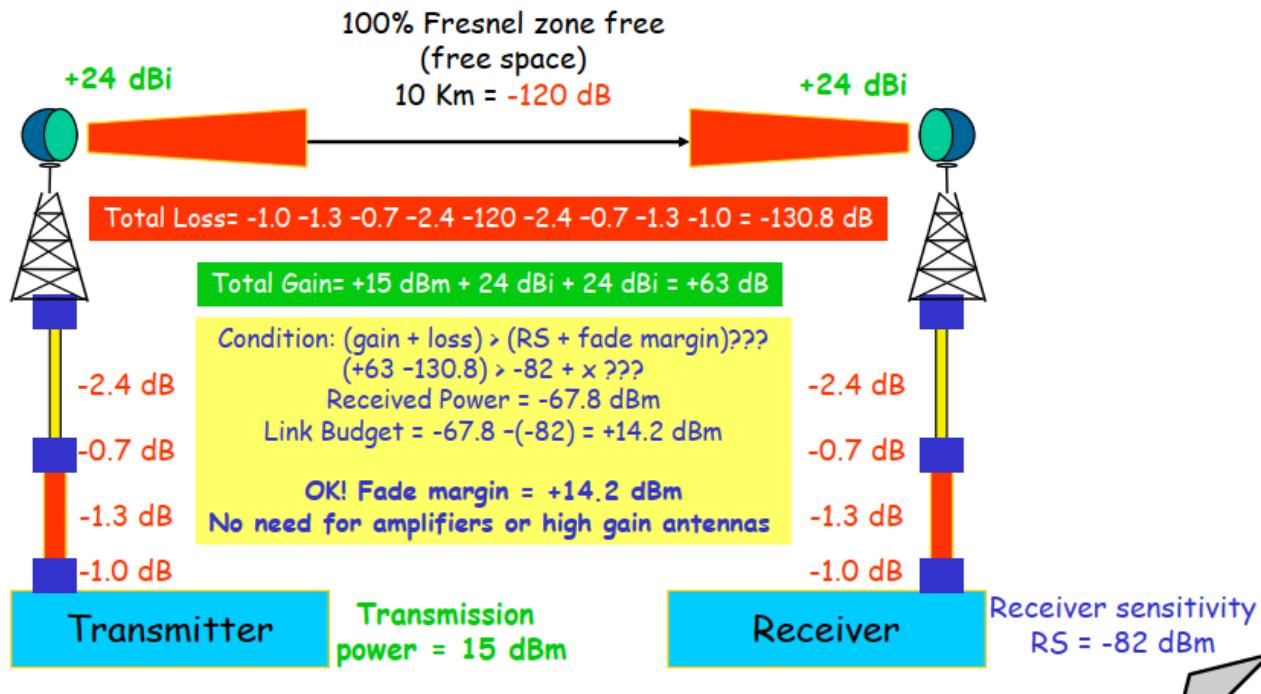
Graficamente, l'energia in eccesso (Valore tra curva rossa e linea blu) deve essere almeno pari a un valore chiamato **System Operating Margin** che deve garantire la funzionalità del sistema (ad esempio, in qualsiasi condizione atmosferica).

Il valore del System Operating Margin è almeno pari a [10,20] Db.

Il System Operating Margin può essere pari a 10 in un sistema in cui c'è poca interferenza. Altrimenti, conviene lasciare un margine maggiore o uguale a 20 Db.

## Esempio esercizio di applicazione dei vari argomenti di questo capitolo

- Example: design of transmission system, needs amplifier?



Abbiamo un ricevente e un trasmittente, che devono comunicare tra di loro.

Trasmettitore: Il trasmettitore è una componente attiva. Possiamo decidere quanta energia far trasmettere.

Più aumentiamo l'energia e più aumentiamo il consumo di energia.

Più energia → più potenza di trasmissione.

L'energia del trasmettitore viaggia su tutti i cavi e i connettori, fino ad arrivare all'antenna, perdendo dell'energia.

L'energia che arriva all'antenna (IR Power Output) sarà pari a 15dB meno tutte le perdite.

Nell'antenna, abbiamo un guadagno, dovuto al tipo di antenna avente una direzione preferenziale (e che quindi comporta un guadagno di energia). Il guadagno di quell'antenna è 24dBi. Segnali 24DB più forti rispetto ad un isotropico.

Il guadagno dell'antenna è un guadagno passivo e non attivo, poiché è dovuto al modo in cui è strutturata l'antenna (che concentra l'energia in un punto).

L'antenna a destra cattura il segnale che arriva da Sinistra, guadagnando altri 24dBi (più di 200 volte in più rispetto a quello che catturerrebbe un isotropico). In questo modo si utilizza il guadagno passivo di queste due antenne per guadagnare dell'energia.

L'energia arrivata all'antenna scende verso il ricevente, ed i cavi e connettori intermedi comportano delle perdite.

Il ricevente ha una sensibilità pari a -82dBm. Cioè, se il segnale ricevuto è pari a -82dBm, qualcosa il ricevitore riesce a capirla; in alternativa se quell'antenna dovesse ricevere un segnale pari a -83dBm, sarebbe sotto la linea blu, e quindi il ricevente non sarebbe in grado di capire l'informazione dal segnale.

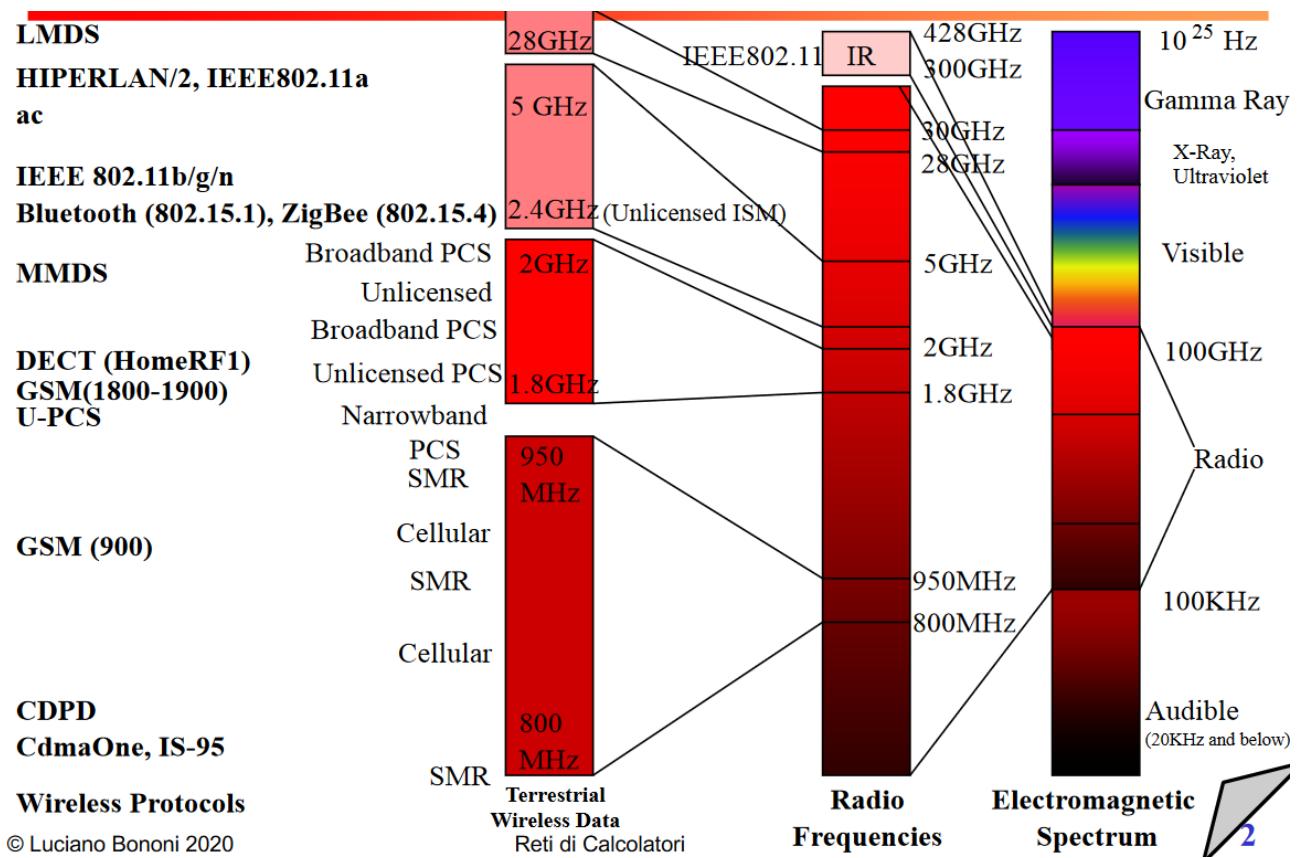
### Possibile domanda esame:

-Se l'energia del segnale ricevuto dal ricevente fosse minore della RS (Sensibilità del ricevente), cosa possiamo fare per risolvere il problema in modo che il ricevente possa avere un'energia sufficiente a capire l'informazione del segnale trasmesso?

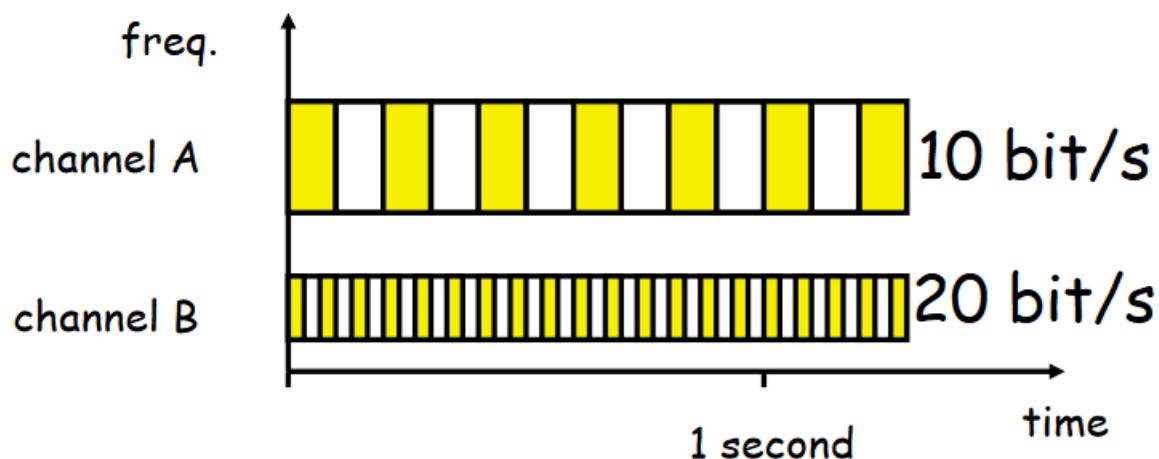
Vari modi, potremmo aumentare l'energia fornita al trasmettitore (in modo da avere un guadagno attivo).

Utilizzare delle antenne che hanno un guadagno passivo maggiore e cercare di guadagnare dell'energia utilizzando le antenne, oppure potremmo cercare dei cavi e connettori che comportino delle minori perdite di energia.

# Wireless



Intorno ai 100KHz inizia lo spettro radio, oltre i 100GHz siamo negli infrarossi, salendo si incontrano luce ultravioletta, raggi X e raggi gamma (svariati miliardi di cicli/s). Le frequenze radio utili si trovano tutte tra i 100KHz e i 100GHz. Esempi: tra 1.8 e 2.4 GHz, abbiamo il dualband. Tra 2.4 e 5 abbiamo Wifi e Bluetooth. Molte porzioni di frequenza hanno più utilizzi in parallelo, l'assegnazione dello spettro alle varie tecnologie è infatti una vera problematica.

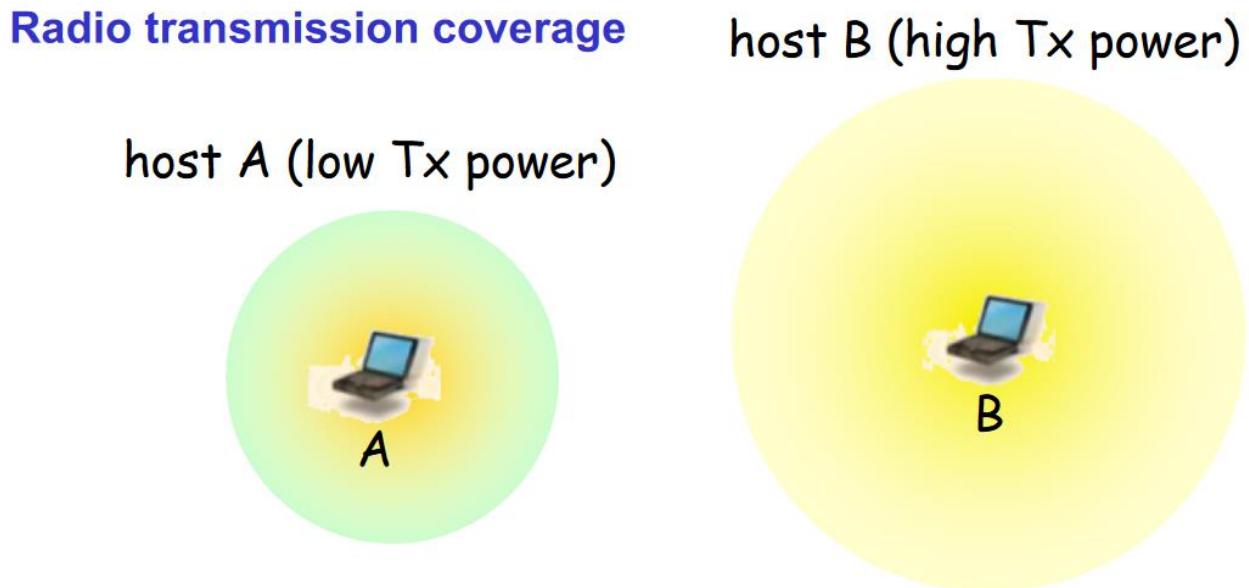


Se voglio trasmettere dei bit usando le onde radio per rappresentarne i valori, cosa fa differenza prestazionale nelle diverse tecnologie? Perché ho dei canali radio che funzionano a 100Mb/s ed altri che funzionano ad 1Mbs/s? Sicuramente non la velocità del segnale (le onde viaggiano alla stessa velocità, cioè a quella della luce, quindi circa 300.000km/s); potrebbe essere, in alcuni casi, la quantità di spettro che uso

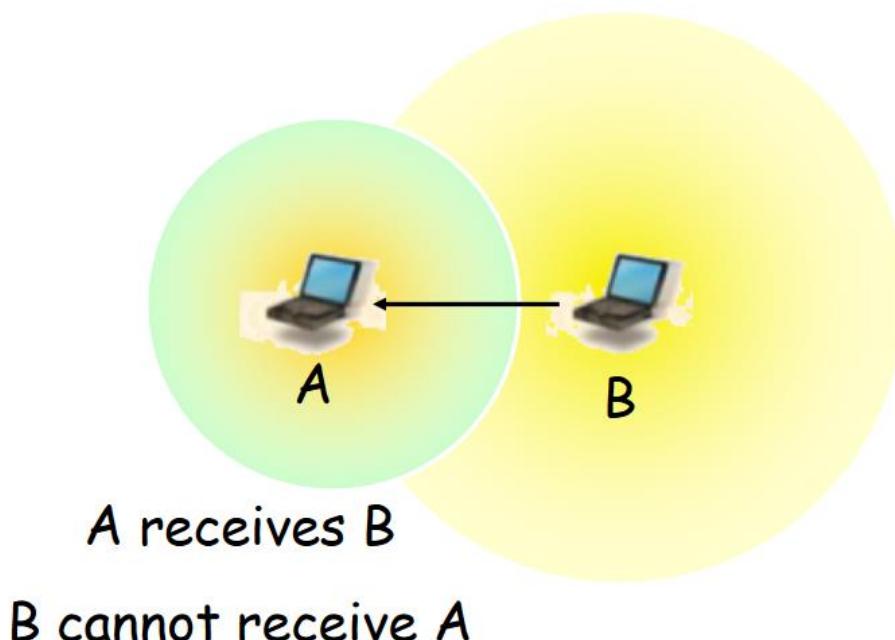
per comunicare; il tempo durante il quale descrivo la sequenza di bit è il parametro cercato, cioè quello che fa differenza. In particolare, la differenza si trova nella durata di tempo durante la quale le onde radio (in quel canale= rappresentano il valore di un bit rispetto al bit precedente/successivo, ovvero la durata di tempo durante la quale le onde radio rappresentano simbolicamente il valore di un bit. Se rappresento i bit a velocità sostenuta ne trasmetto di più, ma è difficile riceverli (viste anche le altre interferenze del mondo radio). Le tecniche di codifica di bit realizzano un compromesso tra ciò si vuole trasmettere bene e ciò che si vuole trasmettere velocemente. Ci possono essere diverse situazioni, dipendentemente dai range (detti dalla potenza trasmittiva).

### Link possibili durante la creazione di canali radio

Caso 1, sia A che B isolati:



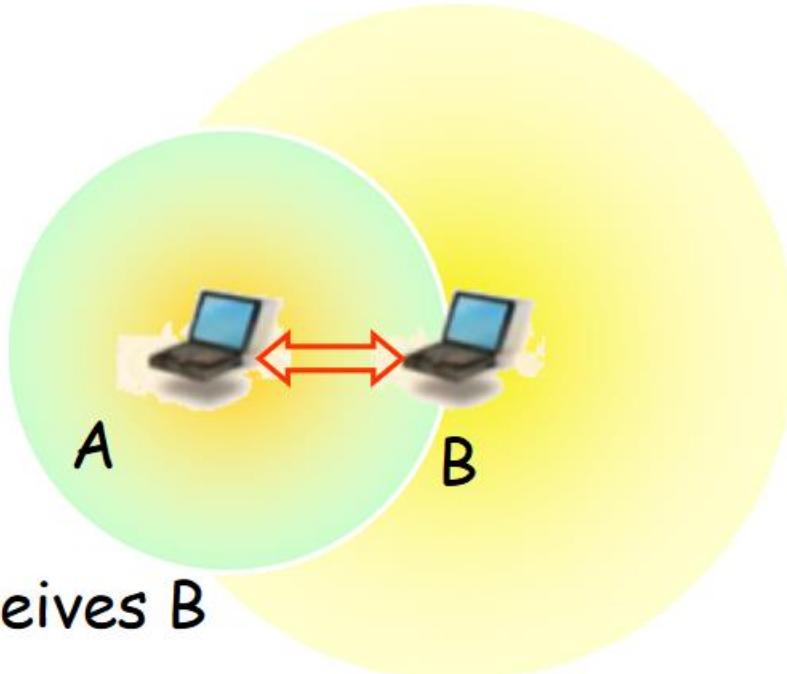
Caso 2, A riceve da B ma B non può ricevere da A, link unidirezionale.



Casi 3 e 4, A e B riescono a ricevere l'uno dall'altro (link simmetrico), ma possono esistere due tipi di link.

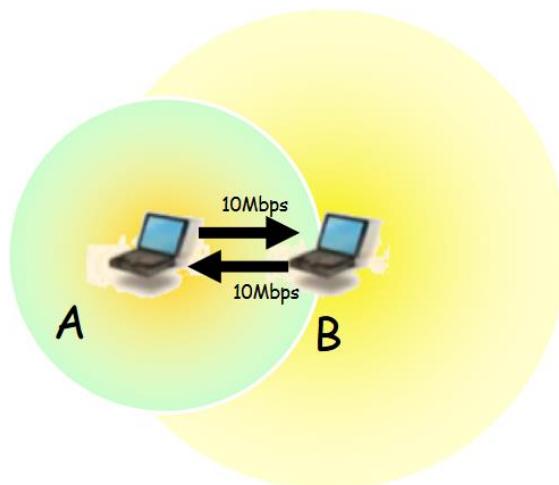
- Se i bitrate nominali  $A \rightarrow B$  e  $B \rightarrow A$  sono uguali, abbiamo un link bidirezionale simmetrico
- I bitrate nominali sono diversi ed avrò quindi un link bidirezionale asimmetrico.

Link bidirezionale (generico)

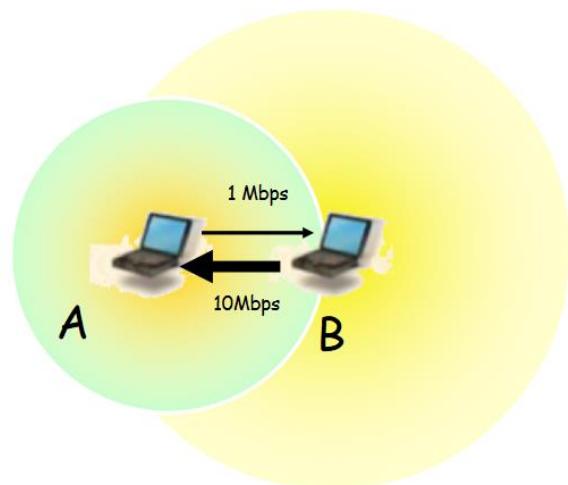


A receives B

B receives A



bidirectional symmetric link



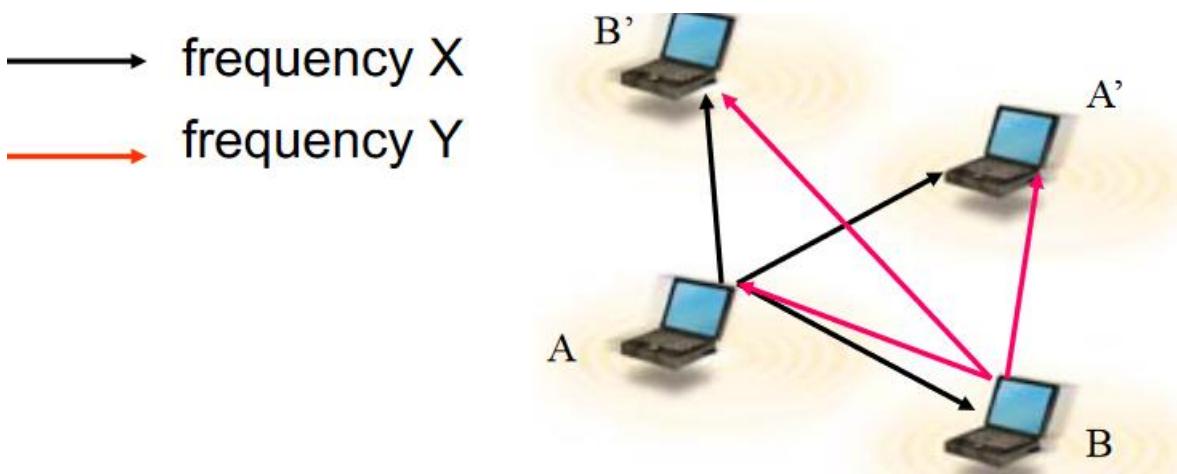
bidirectional asymmetric link

## Creazione dei canali radio

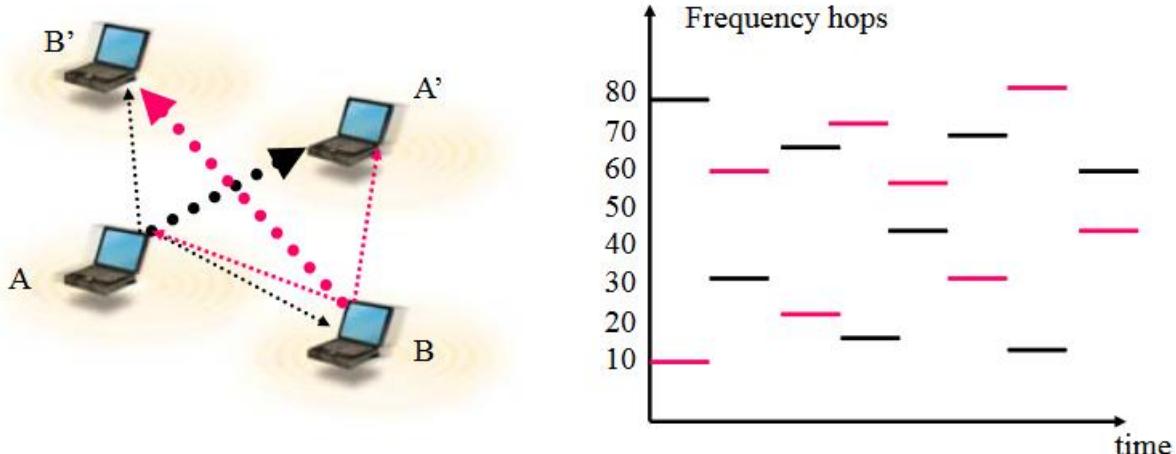
La creazione di diversi canali radio può essere di 3 tipi:

1. Narrow band, banda stretta
2. Frequency hopping spread spectrum
3. Direct sequence spread spectrum

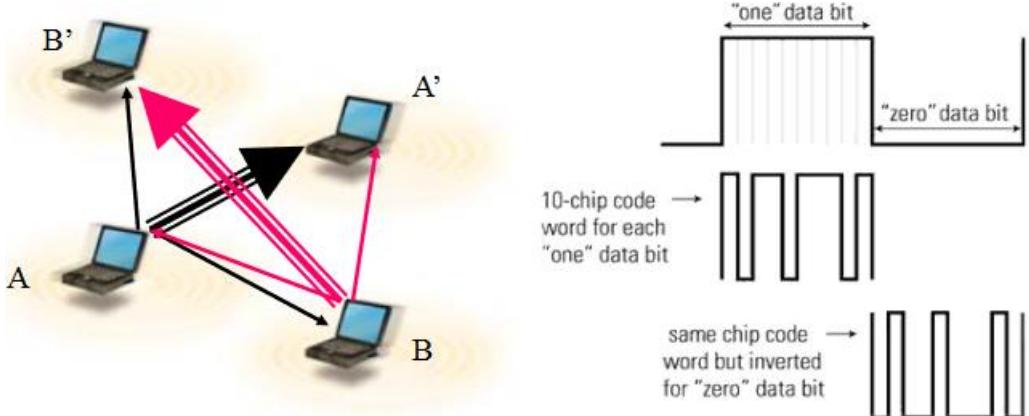
Nel narrow band, diversi canali radio potranno coesistere nello stesso spazio/tempo perché hanno una frequenza radio diversa. La differenza tra la frequenza X e Y rende possibile, lato ricezione, di creare un oggetto denominato filtro, capace di isolare l'energia che arriva su una certa frequenza radio. In particolare, l'energia catturata da una certa antenna è la somma delle energie entranti, ma il filtro (passabanda) riesce ad isolare ed estrarre solamente l'energia dal segnale con la frequenza voluta. Narrow band si chiama così perché utilizza una sola frequenza. Le collisioni radio si hanno quando si vuole usare una frequenza per comunicare tra più di 2 interlocutori.



Nelle altre 2 tecnologie troviamo la dicitura spread spectrum, essa sta ad indicare che il canale radio non è rappresentato da una frequenza singola e fissa ma da uno spettro di frequenze. Nel frequency hopping abbiamo una sequenza di salti sincroni, generati tramite dei pattern pseudocasuali. I canali utilizzano tante frequenze narrow band saltando tra i pattern. Qual è il vantaggio rispetto ad usare 2 narrow band? Col frequency hopping è più difficile intercettare la comunicazione, e se c'è interferenza sul canale, basta saltare quella frequenza (oppure usarla poco tempo dopo l'hop). E' difficile da implementare perché bisogna fare salti perfettamente sincronizzati tra chi trasmette e chi riceve. Il pattern di salti viene generato con una



funzione di hashing dell'indirizzo/chiave/parola deciso da chi trasmette. Pattern deriva da un algoritmo, la chiave va precondivisa. A deve comunicare funzione + seme ad A', stessa cosa B a B'. Direct sequence: ogni bit che voglio comunicare è rappresentato da un pattern di chip (valori binari). Per rappresentare 0 oppure 1 si usa un pattern di chip. La codifica di un bit a uno corrisponde alla trasmissione di chip, quando si vuole trasmettere 0 si trasmette lo stesso chip ma capovolto. Ogni chip viene codificato e trasmesso su tutto lo spettro radio della banda a disposizione. Il bit è, cioè, su tutta la banda. L'interferenza che si crea viene annullata grazie alla presenza di pattern/sequenze di bit che rappresentano ogni bit, questa tecnica viene detta a divisione di codice.

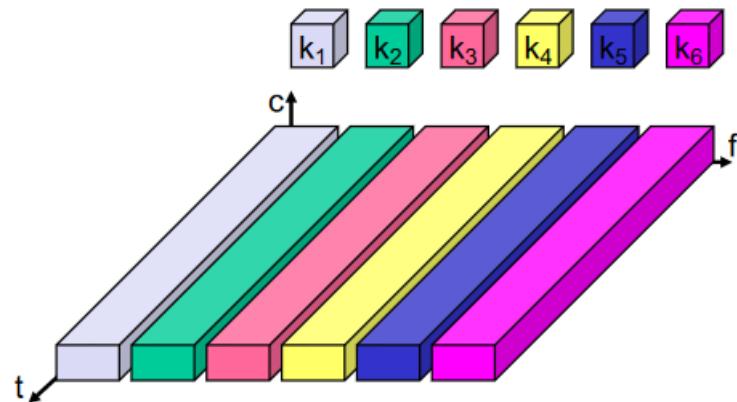


La scelta del codice (sequenza di bit usata per rappresentare i bit) da A ad A' è diversa rispetto al codice da B a B', i due diversi codici fanno in modo che chi riceve i segnali, anche se vede sovrapposizione/interferenza distruttiva, riconosca (grazie al codice), il valore del bit che a lui era dedicato. L'interferenza, pur distruggendo i segnali, non è comunque abbastanza potente da permettermi di sbagliare codifica. Il codice deve differire per ogni coppia di interlocutori. Ogni codice diverso dà luogo ad un canale diverso. Tutti questi canali vengono trasmessi sulla stessa finestra di frequenza. L'infrastruttura wireless creata attraverso l'utilizzo di queste tecnologie consiste in tanti AP (access point), cioè punti di accesso che permettono agli MT (mobile terminal) di entrare nel range di copertura dell'AP e di poter comunicare in rete. MT usano frequenze/tecniche diverse ogni volta che si connettono ad un AP. Le reti possono anche non avere AP, in questo caso si parla di reti ad hoc, cioè reti in cui i nodi si trovano in grado di comunicare tra loro senza aver predisposto la creazione di quella rete, cioè senza averne pianificato l'infrastruttura. Sono reti quindi senza infrastruttura. Esempio di rete ad hoc è il bluetooth. Le reti ad hoc sono "antagoniste" delle reti ad infrastruttura, cioè quelle che richiedono componenti cablate (backbone) e AP. Per fondere il mondo mobile/wireless col mondo cablato bisogna realizzare il bridging. Per fare questo, l'AP ha 2 stack, stack wireless e stack cablato (quindi anche 2 interfacce di rete). Per bridging si intende quando i dati nel mondo internet vengono tradotti nel formato/protocolli wireless e viceversa. Grazie al bridging è possibile avere protocolli wireless differenti da quelli del mondo internet (tecniche, principi di funzionamento e metodologie sono diverse). Dialogando wireless, se ho bisogno di dialogare con nodi in Internet, ho da qualche parte una bridging function, realizzata da un nodo intermedio (AP, wireless router o simili). Integrare i due mondi richiede un adattamento dei protocolli e dei nodi che facciano da collante architettonico. Tornando alle modalità di creazione del canale: voglio avere più comunicazioni possibili tra coppie/gruppi di interlocutori (reti, quindi, coesistenti). Questo è possibile tramite il multiplexing, cioè più comunicazioni di più host utilizzano le stesse risorse. (Multiplexing: utilizzi multipli di un mezzo condiviso). Il multiplexing può essere fatto in 4 dimensioni:

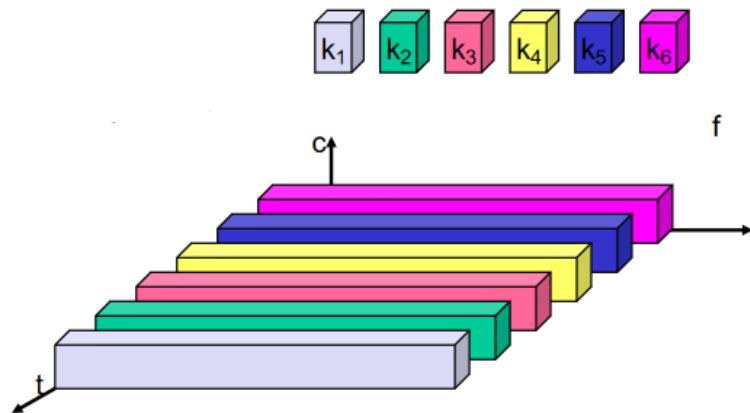
1. Spazio
2. Tempo
3. Frequenza
4. Codice

## Descrizione del multiplexing

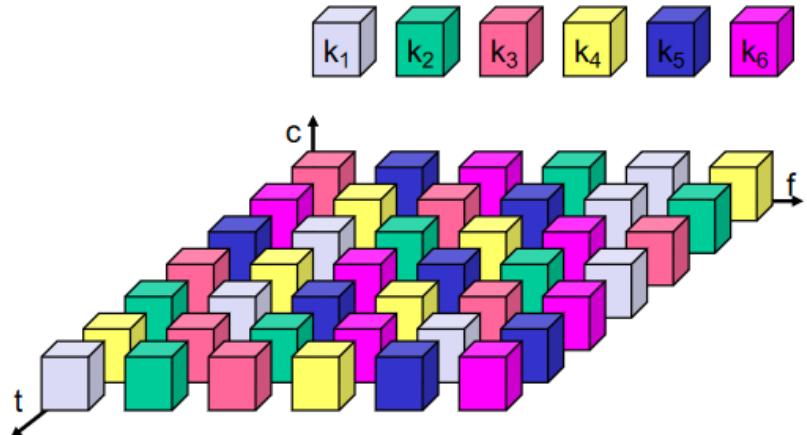
Per frequenza, ogni canale ha una fetta di frequenze diverse, il canale viene scelto scegliendo le frequenze.



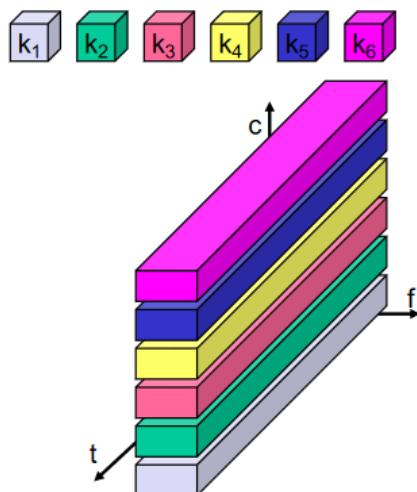
Per tempo, i canali usano tutte le frequenze ma solo per un intervallo delta  $t$  limitato. Dopo che un canale ha finito, c'è un punto di vuoto e subito dopo inizia il successivo. Ogni canale di rete che usa un canale diverso avrà il proprio intervallo di tempo nella quale può trasmettere, e quando può trasmettere usa tutto lo spettro radio.



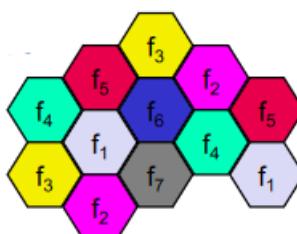
Tempo + frequenza: abbiamo una certa frequenza disponibile per un certo intervallo di tempo. Un canale (di un certo colore) è inteso come sequenza combinata di salti nelle frequenze a intervalli di tempo costante. Salti di frequenza (o intervalli di frequenza), nel tempo, sincroni e coordinati.



Multiplexing per codice: ogni canale ha un codice (chipping sequence, usata per rappresentare il valore di un bit) diverso, i canali usano lo stesso spettro sovrapposto (parlano tutti assieme nella stessa frequenza) e allo stesso tempo. Consente di estrarre i bit trasmessi malgrado il canale possa sembra un'unica grande interferenza.

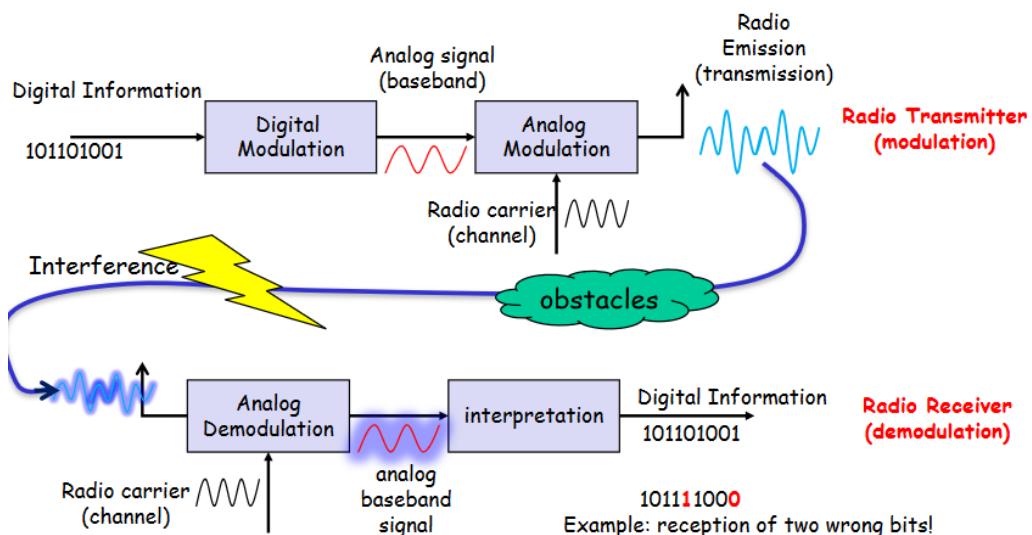


Multiplexing nello spazio: possiamo usare la stessa frequenza  $f$ , basta che dove la utilizziamo la prima volta sia abbastanza (spazialmente) separato da dove la riutilizziamo. Se lo spazio che separa le colorazioni è abbastanza elevato, posso utilizzare più volte lo stesso colore. Questo è il problema del frequency planning. Qualsiasi sia il modo in cui creo i canali di diverso colore, li possiamo usare identici a condizione che ci sia in mezzo dello spazio. Lo spazio che separa è proprio lo space multiplexing (riusare gli stessi colori/canali nello spazio più volte).



## Funzionamento di modulazione e demodulazione

Modulazione: chi vuole parlare/trasmettere, deve scegliere un colore/canale comune a tutti. In qualche modo ci si sta coordinando per capire esattamente cosa si sta trasmettendo e filtrando ciò che si sta trasmettendo rispetto a chi sta trasmettendo su altri canali. Quindi ogni canale diventa un dominio di trasmissione a sé. Siamo su un canale, vogliamo usare le onde radio per trasmettere dei bit, abbiamo bisogno di modulare l'onda radio, fornire all'antenna quest'onda e a questo punto il segnale viaggia e viene riconvertito in bit, dopo essere stato catturato dall'antenna del ricevitore. Modulazione digitale: trasmettere usando le onde radio (onda analogica, varia continuamente nel tempo) e codificare dei valori digitali (0 oppure 1). Questo si fa tramite ASK, FSK, o PSK. Una sequenza di bit viene data in pasto, a livello fisico, ad un modulatore digitale, che prende il valore dei bit e cerca di rappresentarlo usando una sinusoide. Generata la sinusoide, essa varierà rappresentando i valori della sequenza che vogliamo trasmettere. Dobbiamo prendere la sinusoide e copiarne le caratteristiche su un'altra sinusoide (che rappresenta il canale). In pratica il modulatore analogico prende un segnale analogico in ingresso e ne copia le caratteristiche su quello che è definito essere il nostro canale radio (il colore definito precedentemente, diviso in time, frequency, time + frequency o codice), che viene deciso sotto (nello schema, dove c'è scritto "Radio Carrier (channel)"), una volta stabilita però la frequenza usata per trasmettere i bit (che viene per l'appunto presa in input da sotto nel modulatore analogico). Il modulatore analogico difatti genera un segnale a quella frequenza che copia le caratteristiche della sinusoide che gli è arrivata in ingresso (da sinistra, cioè dal modulatore digitale). In output il modulatore analogico genera una frequenza del canale che incorpora le caratteristiche in input (che, a sua volta, incorpora le caratteristiche dei bit). Ed è proprio questo ciò che viene mandato all'antenna (traduce un'onda radio che ha le stesse caratteristiche nell'ambiente). Se nel canale ho il canale "blu", la frequenza è quella di "colore blu", che però varia in modo analogico ma le modifiche del segnale in realtà rappresentano i valori dei bit che vogliamo trasmettere. Tutto ciò viaggia nell'ambiente ed arriva ad un certo punto al ricevitore. Lo schema di ricezione è speculare a quello di trasmissione. Abbiamo un'antenna, che cattura l'effetto radio (così come arriva, magari alterato) dopodiché la corrente elettrica indotta nell'antenna viene passata come input al demodulatore analogico. Prende il canale radio (sempre da sotto, così come succedeva nel modulatore analogico del trasmittitore) ed estrae l'onda radio filtrando solo sulla frequenza richiesta (cioè quella arrivata da sotto). Il demodulatore controlla quale energia dell'onda radio estrarre proprio sulla base del radio carrier (la definizione comune di canale tra chi trasmette e chi riceve). Una volta che il demodulatore sa qualche frequenza ascoltare in quale istante, il suo output è "equivalente" alla sinusoide in input al modulatore analogico nel mittente, ("equivalente" perché ha subito tutte le modifiche date dall'ambiente e non saranno, quindi, mai uguali). Quello che riceviamo è quindi poco simile a ciò che è partito, ma la sinusoide che è partita è in un certo senso contenuta in quella che siamo riusciti ad estrapolare dal canale. Più precisamente, la sinusoide che è partita è proprio la spina dorsale della nuvola che ci è arrivata. Quindi anche se il segnale è disturbato, è correlato fortemente al segnale che è partito. L'ultimo blocco è quello dell'interpretazione, l'interprete deve capire se la "nuvola" assomiglia a bit 1 oppure a 0. A questo punto copia bit per bit e cerca di dare la corretta interpretazione. Se ci sono dei bit sbagliati, bisogna stare attenti a riuscire a trovare una soluzione.

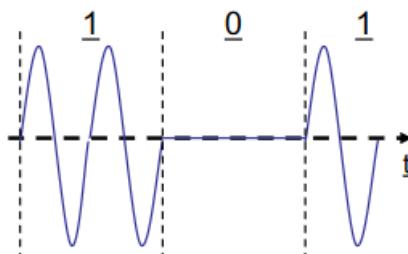


Per esempio, a livello MAC, se troviamo dei bit errati, non mandiamo l'ack, e quindi il trasmettitore ad un certo punto ritrasmette. Se si potessero, però, evitare le ritrasmissioni, sarebbe meglio. Idealmente vorremmo essere in grado di capire che in una certa ricezione c'è qualcosa che non va. Se riesco solo a capirlo, butto il pacchetto e aspetto la ritrasmissione. Se riuscissimo però, a capire quali sono i bit sbagliati, potremmo evitare la ritrasmissione (possiamo correggere autonomamente gli errori).

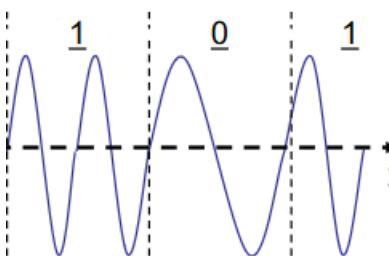
### Principali codifiche dei segnali

ASK, FSK e PSK (Amplitude, Frequency e Phase shift keying) sono 3 modi per rappresentare 0 e 1 tramite una sinusoida.

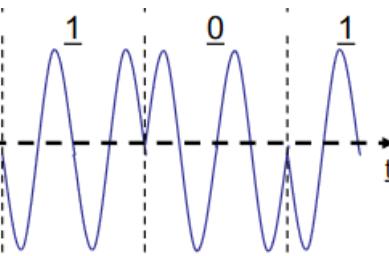
ASK: bit 1, rappresentato da un'ampiezza elevata (ampiezza = differenza di potenziale tra picco positivo e negativo), bit 0 trasmetto una sinusode piatta, cioè 0 energia. Questa implementazione è molto sensibile alle interferenze ma usa una sola frequenza (narrow band).



FSK, sinusode ad una certa frequenza  $f_1$  può essere variata in un'altra frequenza  $f_2$  inferiore. Possiamo fare in modo che  $f_1$  rappresenti il bit 1,  $f_2$  il bit 0. Lo svantaggio di questa soluzione è che usa più spettro (ha bisogno di un intervallo alta - bassa), non basta la singola frequenza. La durata della rappresentazione del bit può essere decisa da chi implementa (ma deve essere costante).

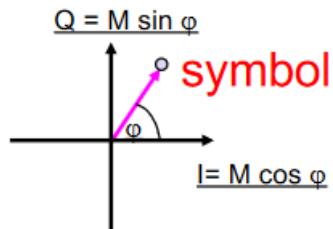


PSK, variamo la fase. L'ampiezza è la stessa, la frequenza è la stessa, però varia la fase, infatti il segnale parte con la sinusode positiva per lo 0 e con quella negativa per l'1.

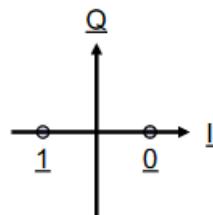


## Codifica tramite simboli

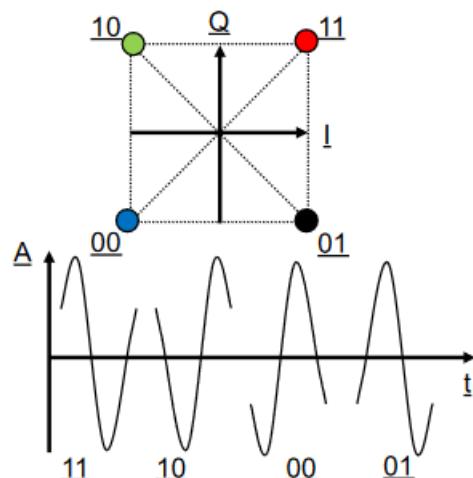
Un altro modo per rappresentare segnali di modo che rappresentino bit 0 o bit 1 è quello di utilizzare i simboli. Un simbolo (nell'immagine è il pallino) è uno dei possibili stati in cui si può trovare l'onda radio che viene



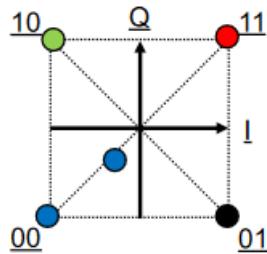
trasmessa. Variando ampiezza, fase e frequenza rappresento simbolicamente lo stato dell'onda. Con un simbolo so l'ampiezza e la fase, ma non la frequenza. La frequenza non mi interessa però, perché mi viene data come input da sotto, nella precedente spiegazione di modulatore/demodulatore analogico, di conseguenza non ne ho bisogno. Se ricevo o trasmetto un segnale corrispondente ad un simbolo, l'ampiezza è la lunghezza del vettore, cioè la distanza del simbolo dall'origine degli assi; la fase, invece, è l'angolo che la freccia forma con l'asse orizzontale. Un esempio di PSK rappresentata tramite simboli è la BPSK (Binary Phase Shift Keying). Ho 2 simboli, hanno la stessa frequenza (che non mi interessa), stessa ampiezza, costante, l'unica cosa che cambia è difatti l'angolo. Che può essere di 0 o di 180 gradi. In particolare rappresento  $\sin(t)$  per trasmettere il simbolo che identifica il bit a 0, con un angolo di 0 gradi. Se trasmetto invece il simbolo con l'angolo di 180 gradi, vuol dire che sto trasmettendo il simbolo corrispondente al bit 1. Cioè trasmetto 0 con  $\sin(t)$  e 1 con  $-\sin(t)$ . Difatti si chiama binary proprio perché abbiamo 2 simboli.



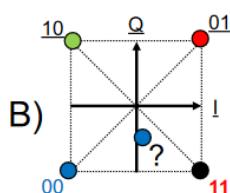
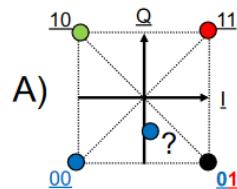
Possiamo fare anche di meglio però con la PSK, aggiungendo altri 2 simboli, avremo una QPSK (quadrature). Avrò, in questo modo, 4 simboli. Stessa ampiezza ma fase diversa. Difatti 11 ha fase 45, 10 ha fase 135, 00 ha fase 225, 01 ha fase 315 (tutti in anticipo, la fase può essere sia in anticipo che in ritardo). Ho 4 simboli anziché 2, posso etichettare con 2 bit ogni simbolo. Aumento il numero di simboli e di conseguenza cambia il numero di bit che trasmetto con ogni simbolo. Ovviamente posso rappresentare il tutto anche su un grafico con ampiezza e tempo (posso vedere, di conseguenza, anche la frequenza). Se costruisco l'hardware di un chip



ricevente/trasmittente, molto spesso il limite tecnologico di quel chip è dato dalla capacità che la tecnologia elettronica ha di variare lo stato della sinusoide tante volte al secondo. Per esempio, come faccio a generare una sinusoide che varia così velocemente al secondo? Supponiamo che una tecnologia consenta di avere 250.000 variazioni di stato della sinusoide al secondo, quindi, di conseguenza, 250.000 simboli al secondo. Se abbiamo 250.000 simboli trasmessi al secondo e uso una BPSK, trasferisco 250.000 simboli ognuno dei quali corrisponde ad un bit e quindi ho 250.000bit/s. Usando QPSK, la stessa tecnologia fa sempre 250.000 simboli/s, solo che i simboli sono 4, e dato che ogni simbolo traporta 2 bit, avrò un bit rate di 500kb/s (il doppio di prima). Tanti simboli equivalgono ad un più elevato numero di bit trasportati per simbolo. Durante la ricezione, ovviamente, il segnale subirà delle modifiche, e magari si potrebbe incorrere in una situazione di questo tipo:



Il simbolo blu vicino al centro è ciò che arriva (il fatto che sia blu indica che il mittente voleva mandare 00), quello blu nell'angolo in basso a sinistra è la sua rappresentazione "ideale". Ogni simbolo viene interpretato con l'etichetta del simbolo più vicino. In questo caso, il mittente voleva mandare un simbolo blu, il ricevente lo interpreta come tale, la comunicazione è andata bene. Ma cosa succede quando il simbolo blu, va, per esempio, nel quadrato vicino al simbolo nero? Lo interpreta male. L'errore si ha quando ciò che è partito è abbastanza diverso da essere confuso con un altro simbolo di quelli possibili. Aumentando il numero di simboli, questa tolleranza è sempre più sensibile, difatti la tolleranza della QPSK è metà di quella della BPSK. Se le condizioni fossero ideali ovviamente sceglieremmo la QPSK (ha un bitrate doppio). Mi accorgo che alcuni bit, usando la quadrature, sono sbagliati e per avere meno errori decido di passare alla binary (ho meno possibilità di errore). Posso scegliere una delle due adattandomi alla qualità del canale radio, inizio con la binary, vedo che non ho bit sbagliati, posso scegliere di passare alla quadrature, se va tutto bene continuo, altrimenti, se ho dei bit sbagliati, torno alla binary. Un altro aspetto importante di questo tipo di problematica è il modo in cui etichetto i simboli, perché questo può influire sugli errori (posso avere degli errori più grandi se etichetto male i simboli). Ciò quindi ci dice che le codifiche non sono equivalenti. Per ogni simbolo, la coppia dei valori adiacenti ha distanza di un bit. Nell'etichettatura le etichette di bit di ogni simbolo adiacente hanno la minima distanza di Hamming. La distanza di Hamming rappresenta il numero di bit differenti nelle varie posizioni, in questo caso la distanza di Hamming minima è 1. Ciò significa che ogni volta che sbagliamo il simbolo sbagliamo, al massimo, un bit. In altre combinazioni (con Hamming > 1) abbiamo la possibilità di sbagliare più bit alla volta.

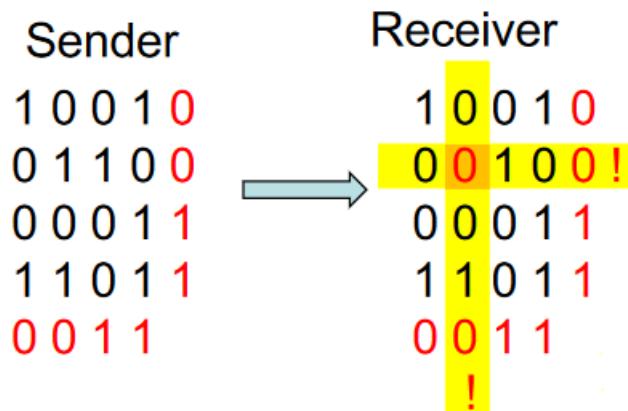


## Rilevazione/correzione errori

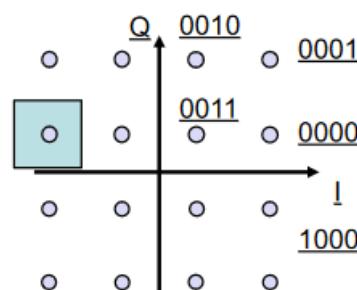
Come si può capire, però, se in una certa trasmissione c'è un errore o meno? In questo caso usiamo il bit di parità, cioè un bit di overhead (messo in coda) che rende pari il numero di bit ad 1 di quel pacchetto. La coppia pacchetto + bit di parità ha sempre un numero di bit ad 1 pari. Ovviamente, a livello MAC, se mi accorgo dell'errore, non mando l'ACK. Il problema del bit di parità si ha quando i bit sbagliati sono 2, perché il bit di parità sarebbe ancora corretto. In sostanza, il bit di parità mi consente di trovare l'errore se ho un bit sbagliato, non mi consente di trovarlo se i bit sbagliati sono 2.



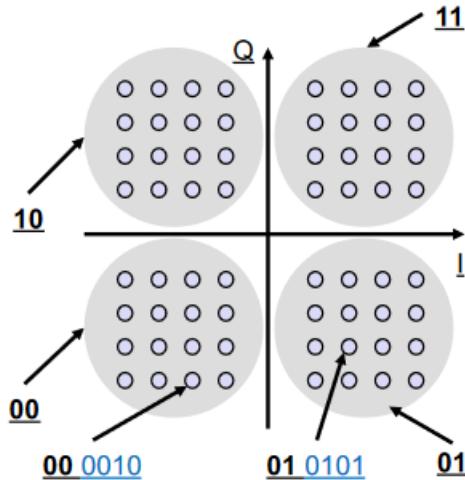
In questo caso, un qualsiasi bit diverso avrebbe reso dispari il numero di bit ad 1, consentendoci così di rilevare l'errore su singolo bit (ma, ricordiamo, non possiamo accorgercene quando l'errore è su 2 bit). È possibile creare una struttura (matrice di bit di parità) che permette anche la correzione di errori su singolo bit. La struttura aggiunge un bit di parità ad ogni riga ed uno ad ogni colonna. Se un bit è sbagliato, cosa succede in questo caso? Posso incrociare la riga e la colonna, trovando così l'errore. Di conseguenza sono anche capaci di correggerlo, cambiando il valore del bit. Se ci sono 2 bit sbagliati, so che ci sono, ma non posso correggerli, ovviamente. Si accenderebbero 2 righe e 2 colonne/2 righe ed una colonna/2 colonne ed una riga. Quindi ho tanti potenziali candidati come bit sbagliati, rendendo così impossibile la correzione. Questa è l'importanza di non sbagliare 2 bit in un solo colpo (derivanti da una pessima etichettatura), perché se ne trovo uno solo su tutto il pacchetto, con le matrici di bit di parità posso anche correggerlo.



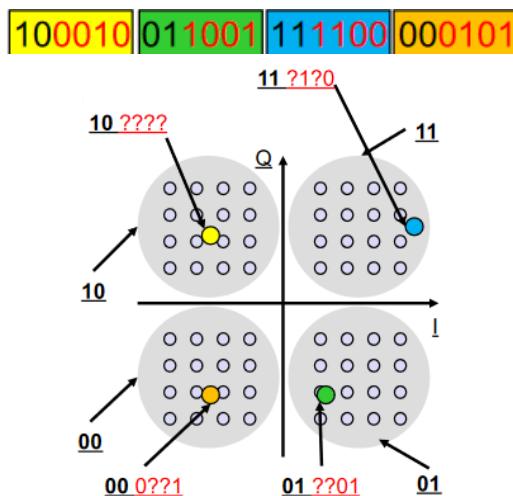
Fino ad ora abbiamo considerato codifiche a 1 oppure a 2 bit, ma la codifica può essere ben più articolata. Immaginiamo quindi di avere una 16-QAM, cioè 16 simboli (quindi 4 bit per ogni simbolo). La QAM (Quadrature Amplitude Modulation) è particolare perché per ogni simbolo da codificare è necessaria la modulazione sia dell'ampiezza che della fase (nella QPSK avevamo esclusivamente la fase).



Posso avere anche una codifica estremamente più complessa, come la 64-QAM, introducendo così una modulazione di tipo gerarchico.



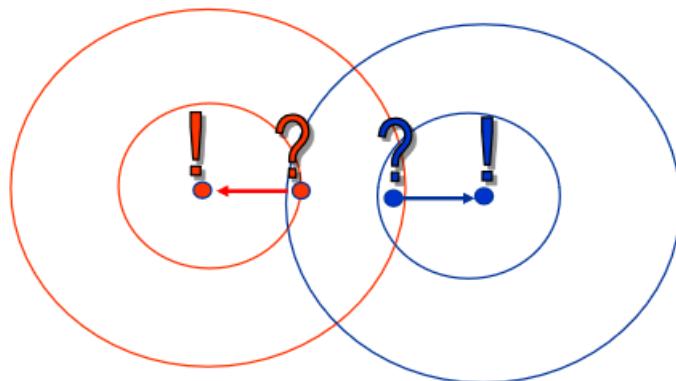
Viene definita gerarchica perché ogni nuvola grigia contiene 16 simboli (usate per codificare le sequenze di bit con priorità bassa), e ogni nuvola grigia è etichettata con una combinazione di 2 bit. I 2 bit ad alta priorità potrebbero essere, per esempio, quelli della voce durante una videochiamata, gli altri 4 a bassa priorità sono quelli del video. In particolare prendo una porzione (in un certo intervallo di tempo) di voce e video e la compongo, metto la voce all'inizio e il video dopo. I due flussi vengono composti in modo sincrono e in modo da generare simboli di 7 bit, che abbiano la voce per prima a sinistra e il video a destra. Supponiamo di voler trasmettere questi bit, se trasmetto un certo simbolo, il fatto che sia dentro una certa nuvola impone i bit di voce, e il simbolo esatto all'interno della nuvola specifica i bit video. Se tutti i simboli sono correttamente interpretati sento e vedo bene. Se trasmettessi su un canale con rumore, il simbolo potrebbe arrivare sbagliato (ma almeno sempre dentro la nuvola grigia). Non sbaglio la voce, ma solo il video, la voce continua ad arrivare nitida. Implicitamente, 64-QAM genera due flussi di bit fusi assieme dei quali uno è più protetto rispetto all'altro. È una sorta di protezione stratificata variabile dei 2 flussi. In questo modo, i simboli possono non essere del tutto corretti, ma la nuvola (quindi la voce) è sempre corretta. Sacrifichiamo la qualità video in modo di ricevere perfettamente la voce.



## Problemi del mondo radio a livello di protocolli

1. Le onde radio sono un broadcast naturale, arrivano a chiunque
2. Il problema delle collisioni non si ha più solo quando siamo in 2 a trasmettere nello stesso istante sullo stesso cavo, ora è anche un problema di posizione, quindi di dove si trasmette. Se siamo in 2 a trasmettere nello stesso tempo ma siamo distanti, non ci sono problemi, se siamo vicini si ha una collisione
3. Se con Ethernet abbiamo una collisione, il protocollo si accorge che siamo in 2 a trasmettere sul cavo, interrompe entrambi e li fa ripartire sfalzandoli temporalmente. Col wireless come si fa? Non c'è modo, se c'è una collisione essa continua per tutto il tempo di durata delle 2 trasmissioni.

Se i due punti esclamativi vogliono comunicare coi due punti interrogativi, i due punti interrogativi sentono entrambe le trasmissioni, quindi sentono la collisione.

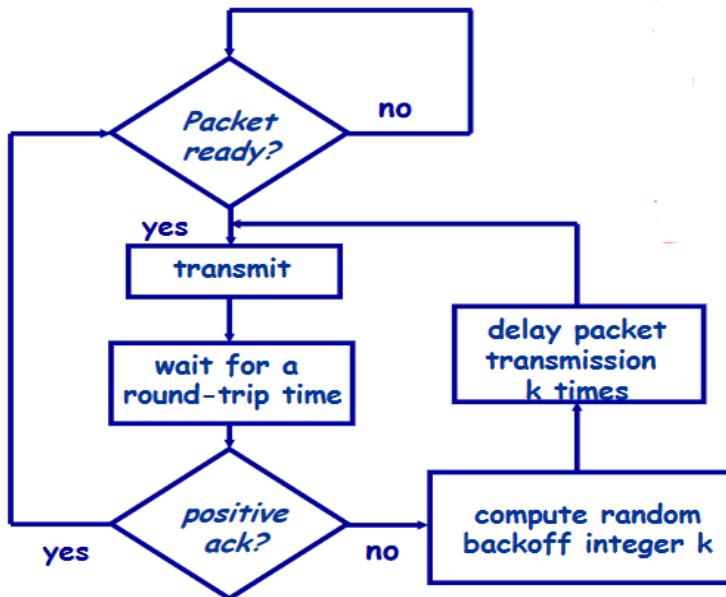


In un sistema wireless può esistere o meno un coordinatore. Se c'è, l'AP (access point) è centrale ed evita le collisioni. Se il coordinatore centrale muore/si rompe/non c'è/succede qualcosa, si ricade nello schema base (per le reti orfane del coordinatore è ancora funzionante). Lo schema ad hoc è quello in cui si cerca di accedere al canale "sgomitando" con gli altri e cercando di vincere la contesa per l'accesso. Si cerca di trasmettere il più possibile ma cercando di evitare la collisione. Abbiamo 2 domini:

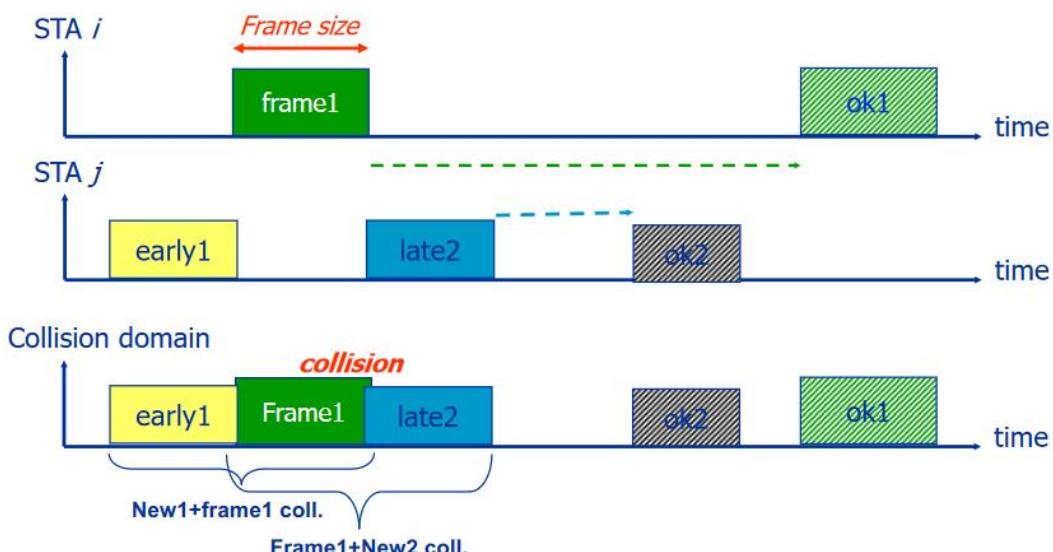
1. MAC protocoli che cercano di risolvere il problema nel tempo (chi trasmette quando?).
2. MAC protocoli che cercano di risolvere il problema nello spazio (chi trasmette dove?), che si aggiunge al "chi trasmette quando?" già esistente nelle reti cablate.

## Protocolli che risolvono il problema nel tempo

Abbiamo, come primo protocollo, il protocollo ALOHA; in pratica se vengono rilevate delle collisioni, non arriva l'ACK, e se non arriva l'ACK, entrambi i mittenti che hanno creato la collisione randomizzano un certo numero, chiamato  $k$  (il backoff, un valore random di attesa), aspettano  $k$  unità di tempo prima di riprovare la ritrasmissione e dopo  $k$  unità di tempo ci riprovano. E' probabile che i valori siano diversi, quindi uno dei due attende di meno, ci prova per primo e ci riesce, evitando così la collisione.

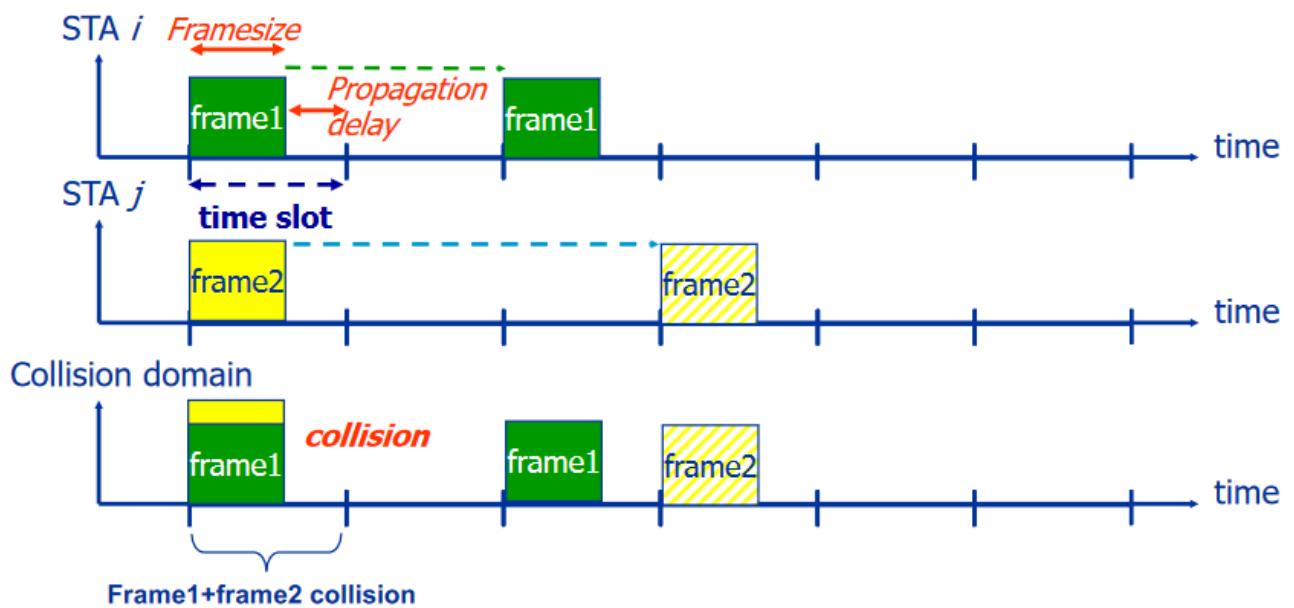
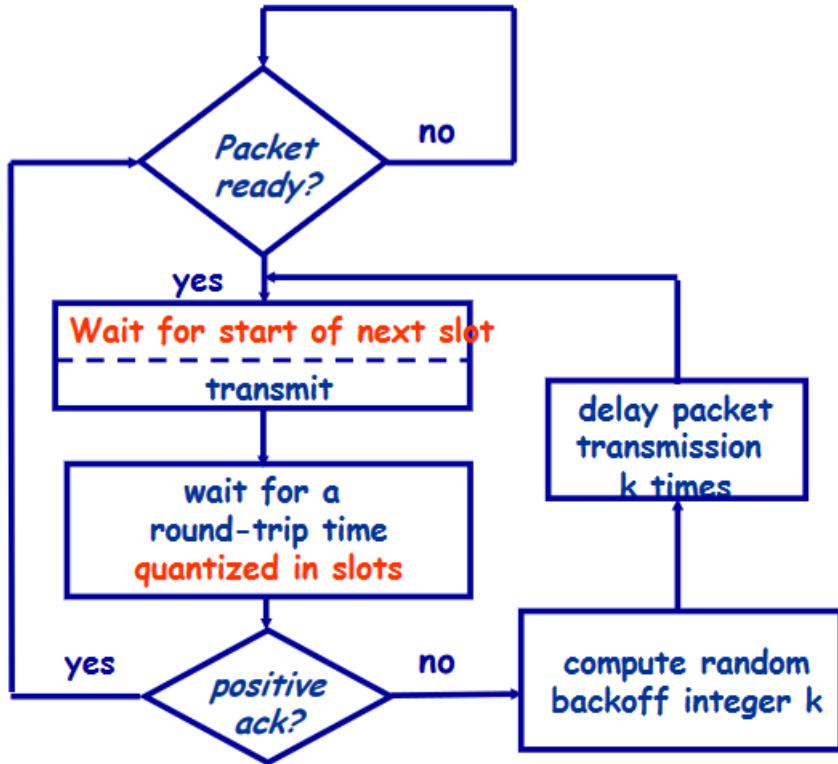


I frame possono scontrarsi per due motivi: o perché sono partiti prima o perché sono partiti dopo rispetto al frame di riferimento. La finestra durante la quale nessuno deve trasmettere mentre sta "parlando" un certo frame si chiama vulnerabilità. In particolare, con questo protocollo, la vulnerabilità del frame è pari a 2 volte la dimensione del frame. Ogni frame mandato con ALOHA rischia molto la collisione.

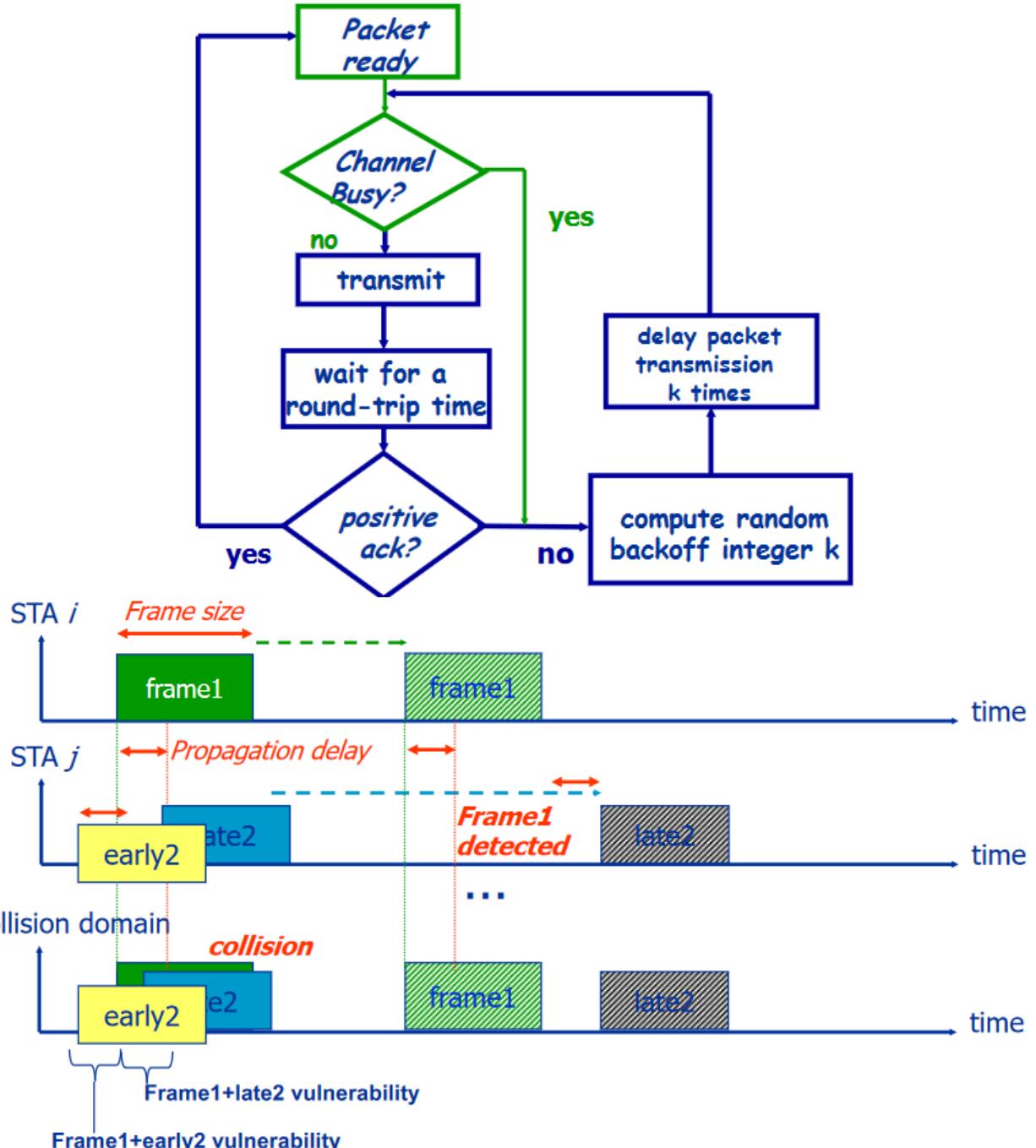


- Frame vulnerability time: twice the frame size

Per migliorare la situazione, si utilizza una versione di ALOHA slotted, cioè il tempo viene misurato con unità costanti sincrone, e la trasmissione può essere effettuata solo all'inizio di uno slot successivo. In questo modo due frame rischiano la collisione solo quando vengono trasmessi nello stesso slot. La vulnerabilità di un frame diventa, quindi, uguale alla dimensione dello slot (che può anche essere uguale alla dimensione del frame). Prima era  $2 \times \text{framesize}$ , ora è  $1 \times \text{framesize}$  (cioè slot + propagation delay). Implementazione dell'ALOHA slotted:

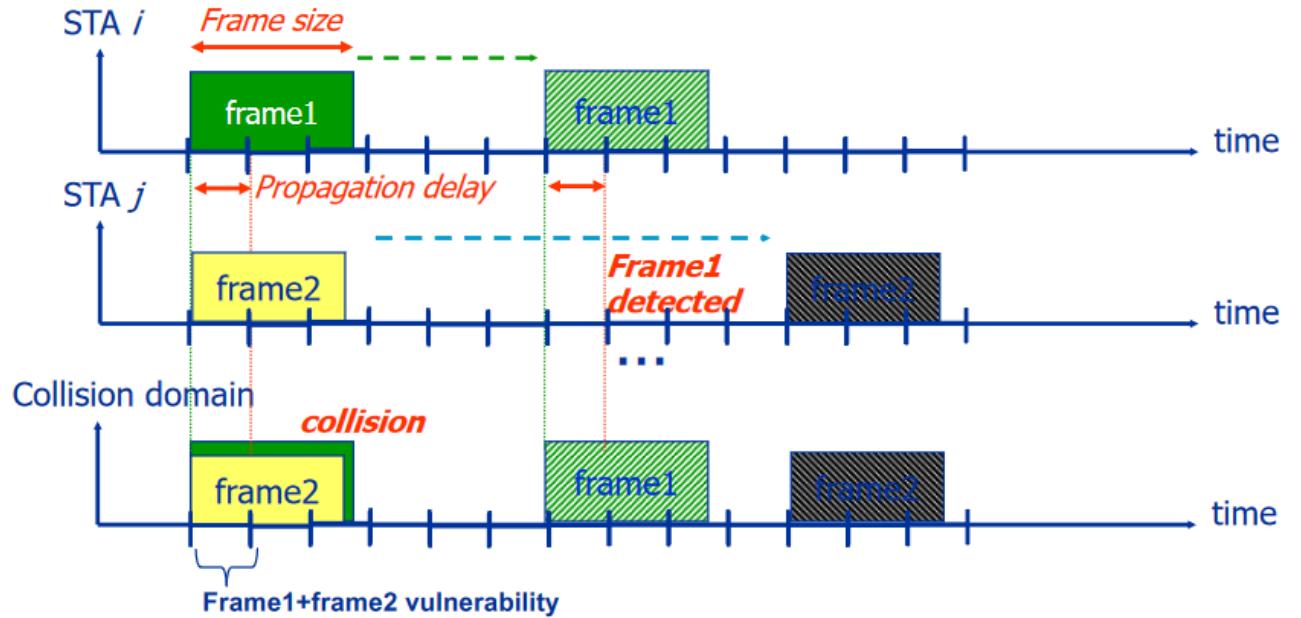


Un'altra considerazione possibile è la creazione di chip radio che possano ascoltare il canale prima di trasmettere. Se devo trasmettere, prima ascolto il canale e mi chiedo se è già occupato o meno. Ascoltare prima di parlare è proprio il principio del protocollo CSMA (Carrier Sens Multiple Access). In questo modo la vulnerabilità scende ancora, arrivando così a 2 volte il propagation delay, cioè 2 volte il tempo che impiega il segnale radio da una stazione all'altra.



- Frame vulnerability time: twice the propagation delay

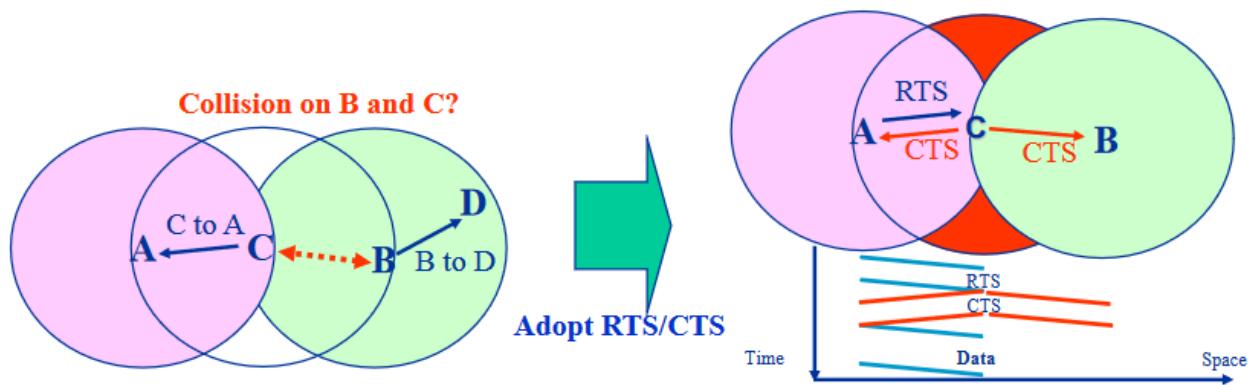
Introducendo gli slot su CSMA, la vulnerabilità è pari ad una volta sola il propagation delay (nell'immagine è la distanza fra le diverse tacche). Questo è il miglior approccio quando sul canale vogliamo essere in tanti a parlare "sgomitando" l'uno con l'altro. La base di 802.11 se sul canale siamo senza AP è proprio uno slotted CSMA.



- Frame vulnerability time: the propagation delay

## Protocolli che risolvono il problema nello spazio

Il problema del chi trasmette dove viene risolto utilizzando il meccanismo di RTS/CTS, Request to Send/Clear to Send. In pratica, un nodo A e un nodo B fanno collisione su un certo nodo C, ed A e B non riescono ad ascoltarsi. RTS/CTS risolve questa situazione facendo in modo che A deve sapere se il canale è libero per C (ricordiamo che la collisione è un problema sul destinatario); questo viene fatto mandando un RTS a C, se C riceve RTS, vuol dire che non è soggetto a collisione da parte di B, quindi C risponde col CTS ad A. Se A riceve il CTS, può comunicare i dati da A a C sicuro del fatto che C non è soggetto a collisione. Quando C manda il CTS ad A, in realtà lo manda in tutto il suo raggio, quindi anche a B. Cosa succede a B se sente, col carrier sensing, un CTS proveniente da C? B capisce che C è esposto ad una comunicazione da parte di qualcuno (A) che B non può sentire. Dal CTS, B è informato che deve rimanere in silenzio, perché C sta ricevendo da qualcun altro. RTS/CTS risolve la vulnerabilità dei terminali esposti (C) e dei terminali nascosti (A e B, uno dall'altro) in uno scenario distribuito nello spazio con le reti radio.

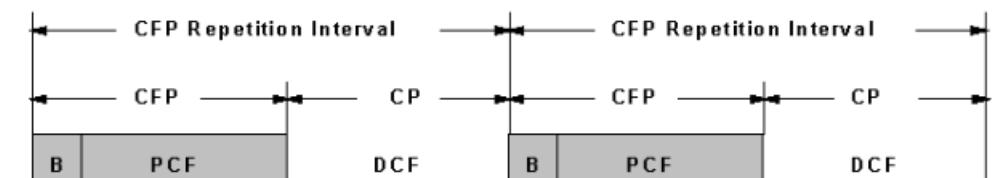


## Funzionamento di 802.11

802.11 consiste in due schemi MAC coesistenti:

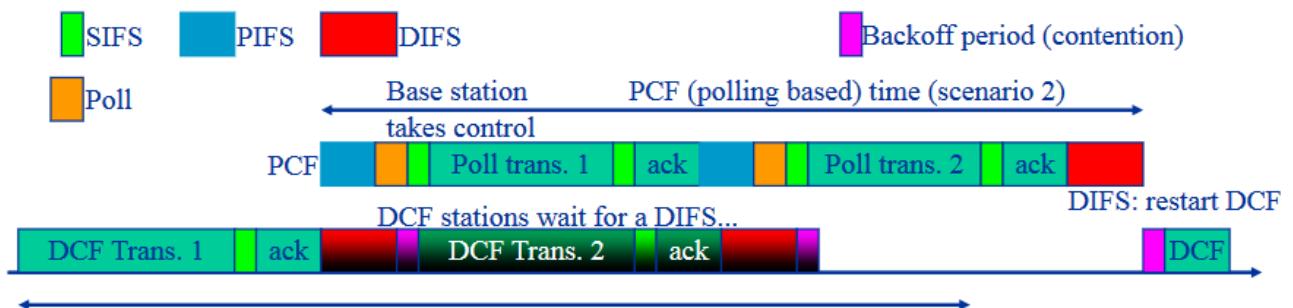
1. Uno schema distribuito basato su CSMA (il modo di distribuire gli accessi randomizzandoli si chiama binary exponential backoff). Lo schema si chiama DCF (Distributed Coordination Function).
2. Uno schema centralizzato, cioè dove è presente uno coordinatore centrale, lo schema in questo caso si chiama PCF (Point Coordination Function).

Di base abbiamo l'accesso distribuito, cioè la modalità DCF. Se e quando è presente l'AP, l'AP prende il controllo del canale e decide chi parla quando evitando così le collisioni. Lo schema è fatto a 2 piani. Quando l'AP è presente trasmette periodicamente un pacchetto, chiamato beacon, contenente alcune informazioni sulla rete (tipo il nome, il canale utilizzato, ecc). Dentro il beacon, se qualche stazione ha del traffico riservato e vuole trasmettere ad intervalli costanti (garantiti), viene schedulato a trasmettere nella fase successiva, cioè proprio la PCF. Questa fase è fatta in modo che solo chi viene chiamato dall'AP potrà parlare. Quando la PCF termina, il tempo fino al prossimo beacon è governato dalla DCF (tutti provano a trasmettere cercando di evitare le collisioni). In sostanza, è un'alternanza di fasi, prima il beacon, poi la fase centralizzata, infine quella distribuita e così via.



Se l'AP non c'è proprio, sparisce il beacon, sparisce la parte centralizzata PCF, e abbiamo solo la DCF. A qual punto le stazioni possono comunicare tra loro solo sgomitando e comunicando quando gli è concesso. Non saranno mai coordinati da nessuno. Ma le 2 fasi come si alternano nel MAC? Se abbiamo una trasmissione sul canale, alla fine della trasmissione c'è l'ACK, questo particolare è importante per tenere conto di come si alterneranno le fasi. L'alternanza di queste fasi viene dettata utilizzando 3 intervalli, Short IFS (SIFS) < Point IFS (PIFS) < Distributed IFS (DIFS). In pratica l'intervallo SIFS è un tempo vuoto sul canale (molto breve) che serve solo a dare l'autorizzazione al nodo che deve mandare l'ACK per mandare il suo ACK. Una volta che il canale rimane libero per un SIFS, se non c'è una stazione autorizzata a trasmettere nessuno comincia a parlare. E quindi a questo punto la durata di vuoto sul canale continua a crescere e diventa pari ad un PIFS, Supponiamo sia arrivato l'ACK, supponiamo che trascorra un SIFS vuoto e nessuno parla (perché nessuno può parlare), supponiamo che l'intervallo vuoto diventi PIFS, l'unica stazione che può parlare dopo un PIFS vuoto è l'AP. Se l'AP comincia a generare i poll, la stazione chiamata fa la sua trasmissione, c'è l'ACK. Trasmissione + ACK sono autorizzati dal poll. L'intervallo SIFS intende dire che c'è solo qualcuno di preautorizzato che può parlare, dopodiché il canale diventa libero, ma diventa libero al massimo per un PIFS: Dopo un PIFS, può parlare solo l'AP; l'AP spara un altro polling , che autorizza un'altra stazione a trasmettere, viene inviato eventualmente l'ACK e via di questo passo. L'AP ha il controllo del canale finchè vuole. Tiene in mano la gestione centralizzata del canale. Quando l'AP finisce di chiamare in causa le stazioni della lista di comunicazioni da effettuare in polling, lascia trascorrere il PIFS senza intervenire. A questo punto il PIFS vuoto sul canale aumenta ancora e diventa un DIFS. Il DIFS è un intervallo vuoto sul canale )dove nessuno trasmette, di dimensione massima). Implicitamente il MAC protocol è fatto in modo che se tutti vedono un DIFS vuoto sul canale, allora tutti sanno che da lì in poi l'AP non vuole più governare il canale. Tutti sanno che da lì in poi devono contendere l'accesso al canale. C'è una trasmissione sul canale, c'è il suo ACK, a quel punto trascorre un PIFS, se l'AP vuole prendere il controllo del canale trasmette il polling. Se l'AP non volesse prendere il controllo, lascerebbe scorrere il PIFS, le stazioni vedrebbero, successivamente, il DIFS e quindi tutte saprebbero di poter parlare contendendosi il canale. Al termine di ogni comunicazione sapremo dall'ACK se è andata bene oppure se è stata una collisione e trascorso questo tentativo di comunicazione (positivo o meno) ci sarà un altro DIFS e dopo il DIFS un altro tentativo, eccetera. Questa è l'alternanza di 802.11.

- **Each station performs a carrier sensing activity when accessing the channel**
- **priority is determined by Interframe spaces (IFS):**
  - Short IFS (SIFS) < Point IFS (PIFS) < Distributed IFS (DIFS)
  - after a SIFS only the polled station can transmit (or ack)
  - after a PIFS only the Base Station can transmit (and PCF takes control)
  - after a DIFS every station can transmit according to basic access CSMA/CA (DCF restarts)



## Alternanza di comunicazione nella fase distribuita

Supponiamo che il canale sia occupato da una trasmissione, improvvisamente diventa libero, e questa durata di silenzio arriva a durare un DIFS. A questo punto tutte le stazioni che hanno qualcosa da trasmettere sanno di poterlo fare, ma sanno anche di poter generare delle collisioni nel caso ci provassero insieme (non si sa in quanti sono a voler trasmettere, ognuno sa per sé). Cosa conviene fare? Ogni stazione genera un valore casuale  $k$  di attesa (una specie di pena da scontare, ascoltando gli slot vuoti sul canale). Dopo il DIFS vuoto, una stazione genera un certo  $k$ , chiamato backoff. Se per esempio il backoff  $k$  fosse uguale a 4, la stazione deve ascoltare 4 slot vuoti. Ascoltato il 4° slot vuoto,  $k$  è diventato 0, e quando il backoff è 0, può iniziare la trasmissione della stazione. Quando scatta il DIFS, tutte le stazioni dovrebbero (statisticamente) generare valori di backoff diversi e quindi dovrebbero provare/iniziare a trasmettere in uno slot diverso. Quindi ci troviamo in uno slotted CSMA. Se tentiamo di trasmettere in uno slot già occupato da qualcun altro, avremmo una collisione. In quel caso non vediamo l'ACK arrivare. Dalla mancanza di ACK sappiamo che c'è stato un tentativo ma non siamo riusciti nella trasmissione. Dobbiamo riprovare di nuovo a trasmettere lo stesso frame finché non riusciamo. Se proviamo svariate volte e facciamo sempre collisione, l'informazione implicita che il canale ci comunica è che stiamo provando e siamo in troppi a provare a trasmettere (visto che continuiamo a fare collisione). Anche se scegliamo gli slot a caso, è probabile che scegliamo in almeno 2 lo stesso slot che continuiamo a fare collisione. Il meccanismo di random backoff è adattivo, cioè la 1° volta che trasmette un frame sceglie un valore di backoff a caso tra 0 e 15. Se qualcuno sceglie il valore 6 di backoff, emagari nel punto in cui il suo  $k$  è arrivato a 2 qualcun altro trasmette, quando il canale ridiventava libero partiremo da backoff 2. Cioè congeliamo il valore, se lo abbiamo congelato perché è partita la trasmissione di qualcun altro allora nel momento in cui ripartiamo a contare (quando il canale ridiventava libero) torneremo a contare da 2 (avevamo congelato quel valore). Nel primo tentativo quindi scegliamo un valore casuale tra 0 e 15, nel secondo un valore doppio, tra 0 e 31, sempre randomicamente. Ancora collisione, tra 0 e 63, poi fino a 127, 255, 511 e infine 1023. Col Wifi possiamo arrivare quindi ad un massimo di 7 tentativi per ogni frame.

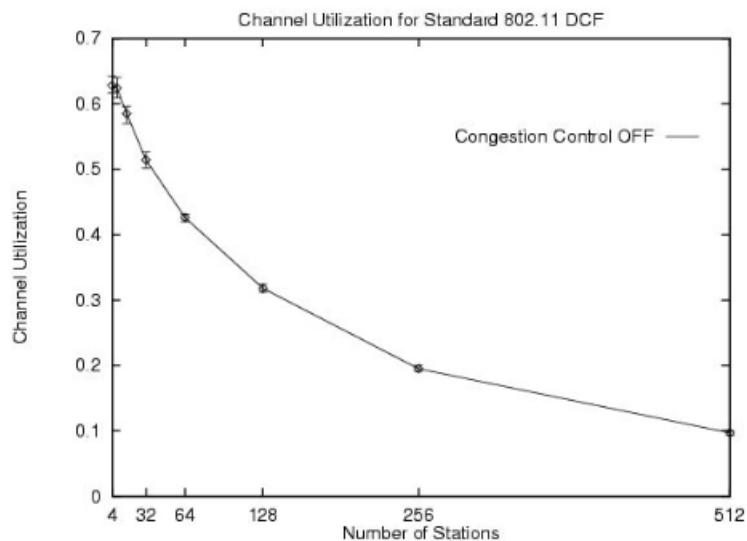
**CWi=contention window size at the i-th transmission attempt. CWi is doubled after each collision experienced (to reduce the contention)**

$$\text{BackoffTime}(i) = (\text{CWi} * \text{random}) * \text{SlotTime}$$

$i$	1	2	3	4	5	6	7
$CW_i$	15	31	63	127	255	511	1023

Ogni volta avrà un backoff più lungo in media da scontare (riduco le probabilità di collisione andando in 2 o più nello stesso slot). Se arrivassimo a 7 tentativi e facessimo ancora collisione, allora rinunciamo ad inviare di nuovo, perché? È probabile che colui che stiamo cercando di contattare per mandargli questo pacchetto in realtà non è presente/è andato via. Se dopo 7 tentativi non ho ancora l'ACK, probabilmente la causa è più radicale rispetto ad una collisione (l'interlocutore non risponde perché non c'è). A quel punto il MAC protocol rinuncia a trasmettere di nuovo quel frame (si arrende) ma notifica al livello rete (il "piano di sopra") che questa comunicazione dal nodo A al nodo B non è possibile. Il livello rete deve trovare una soluzione, la soluzione sarà trovare un cammino alternativo per far arrivare quel pacchetto a destinazione (problema di routing). Se un pacchetto non riesce a proseguire la sua strada da un certo nodo ad un certo nodo successivo, bisogna trovare una strada alternativa (è ciò che il MAC protocol chiederà di fare a livello rete con questa notifica in cui si arrende). Se il numero di stazioni che contendono per l'accesso allo stesso canale condiviso cresce, l'utilizzo percentuale del canale (cioè il tempo durante il quale il canale non sta facendo collisioni) è rappresentato dal grafico. Poche stazioni, molto tempo viene riservato a trasmettere e poco viene consumato dalle collisioni. Aumentiamo la contesa, gran percentuale del canale va sprecata in collisioni, e solo una

porzione minoritaria diventa l'utilizzo che facciamo del canale, cioè il bitrate che riusciamo a trasferire sul canale. Questa è la prova che il MAC protocol deve cercare di contrastare l'insorgere delle collisioni quando la contesa è alta sul canale.



**FIG. 1.** Channel utilization of Standard 802.11 DCF