

## RETI di CALCOLATORI

<b>INTRODUZIONE</b>	<b>7</b>
INFRASTRUTTURA DI RETE	7
CANALI DI COMUNICAZIONE	8
RETI A COMMUTAZIONE DI CIRCUITO	8
RETI A COMMUTAZIONE DI PACCHETTO	9
SERVIZI ORIENTATI ALLA CONNESSIONE E NON	9
PROTOCOLLI DI RETE	9
STANDARD ISO/OSI RM	9
PROTOCOLLI DI INTERNET	10
RENDERE UN SEGMENTO DI RETE AFFIDABILE	11
TECNOLOGIE PER SCHEDE DI RETE	11
COMPORRE SEGMENTI IN RETI LOCALI	12
IL LIVELLO RETE e IL PROTOCOLLO IP	12
SOTTORETI	13
INSTRADAMENTO (FORWARDING) DEI PACCHETTI	14
ROUTING	14
PROTOCOLLO ICMP	14
APPLICAZIONI BASATE SU ICMP	15
<b>19-20/10</b>	<b>15</b>
PROTOCOLLO ARP E RARP	15
ASSEGNAZIONE IP: DHCP	16
IPV6 E TUNNELLING IPV4	16
APPROFONDIMENTO SULLA RETE GARR	16
LIVELLO TRASPORTO	17
SLIDING WINDOW	17
NOMI DI DOMINIO E SERVIZIO DNS	18
LIVELLO APPLICAZIONE	19
SERVIZI CLIENT/SERVER E PEER TO PEER	19
<b>26/10</b>	<b>19</b>
FREQUENCY DIVISION MULTIPLEXING (FDM)	19
MEZZI DI PROPAGAZIONE DEL SEGNALE	20
CORE NETWORK	20
STORE AND FORWARD	20
RITARDO NELLA TRASMISSIONE DEI PACCHETTI	21
TIME DIVISION MULTIPLEXING	21
FREQUENCY DIVISION MULTIPLEXING	21
ALGORITMI DI ROUTING E FORWARDING	21
<b>27/10</b>	<b>21</b>
PACKET SWITCHING	21
STRUTTURA DI INTERNET	22

ESERCIZI SULLE RETI (packet delay)	22
<b>29/10</b>	<b>22</b>
CREARE UN'APPLICAZIONE DI RETE	23
Server	23
Client	23
ARCHITETTURA PEER TO PEER (P2P)	23
COMUNICAZIONE TRA PROCESSI	23
DIFFERENZE TRA P2P e SERVER	24
WELL-KNOWN-PORT NUMBERS (WKPN)	24
COSA DEFINISCE IL PROTOCOLLO DI LIVELLO APPLICAZIONE?	24
COSA SERVE ALLE APPLICAZIONI?	24
RENDERE UDP E TCP SICURI	25
WEB e HTTP	25
RTT	25
HTTP USA TCP	25
METODI DI HTTP	25
HTML	26
<b>2/11</b>	<b>26</b>
COOKIES	26
PROXY	26
E-MAIL	27
DNS (Domain Name System)	28
FUNZIONI DEL SERVIZIO	28
APPROCCIO DISTRIBUITO	28
DNS RECORD TYPE	29
POSSIBILI ATTACCHI DNS	29
FILE DISTRIBUTION TIME (FDT)	30
PEER TO PEER FDT	30
BitTorrent	31
<b>5/11</b>	<b>31</b>
STREAMING VIDEO	31
DASH	32
CONTENT DISTRIBUTION NETWORKS (CDN)	32
ENTER DEEP	32
BRING HOME	33
APPROCCIO OVER THE TOP (OTT)	33
CASO NETFLIX	33
SOCKET PROGRAMMING	34
CREAZIONE SOCKET	34
UDP CLIENT e SERVER	34
TCP CLIENT e SERVER	34

<b>9/11</b>	<b>35</b>
SERVIZI DI LIVELLO TRASPORTO	35
MULTIPLEXING (MUX) / DEMULTIPLEXING (DMUX)	35
FUNZIONAMENTO DEL DEMULTIPLEXING	36
CONNECTIONLESS DMUX	36
CONNECTION ORIENTED DMUX	36
TRASMISSIONE DATI CONNECTIONLESS (UDP)	37
HEADER DEI SEGMENTI	37
CHECKSUM	37
<b>10/11</b>	<b>38</b>
TRASMISSIONE DATI AFFIDABILE (RDT)	38
NOTAZIONE PER MACCHINA A STATI TRANSIZIONE	39
RDT 1.0	39
RDT 2.0	40
RDT 2.1	41
RDT 2.2 (NAK free protocol)	42
RDT 3.0	42
MODIFICHE DA INTRODURRE	43
PIPELINING	44
GO-BACK-N	44
GO-BACK-N: SENDER	44
GO-BACK-N: SENDER EXTENDED	45
GO-BACK-N: RECEIVER EXTENDED	45
SELECTIVE REPEAT	46
PROBLEMA DI NUMERAZIONE	46
<b>12/11</b>	<b>47</b>
TCP	47
SEGMENTO TCP	47
ESEMPIO	48
RTT e SETTING del TIMEOUT	48
TCP SENDER	49
SCENARI DI RITRASMISSIONE TCP	49
TABELLA SULLA GENERAZIONE DEGLI ACK	49
TCP FAST RETRANSMIT	50
CONTROLLO DI FLUSSO	50
<b>16/11</b>	<b>50</b>
THREE WAY HANDSHAKE	50
TERMINARE UNA CONNESSIONE	51
CONTROLLO DELLA CONGESTIONE	52
SLIDING WINDOW DI TCP	52
SLOW START	52

CAPIRE SE C'È DAVVERO CONGESTIONE	53
ESEMPIO	53
EXPLICIT CONGESTION NOTIFICATION (ECN)	53
LIVELLO RETE: DATA PLANE e CONTROL PLANE	54
<b>19/11</b>	<b>55</b>
COME FUNZIONA UN ROUTER AL SUO INTERNO?	55
INPUT PORT	55
LONGEST PREFIX MATCHING	56
SWITCHING FABRICS	57
SWITCHING VIA MEMORIA	57
SWITCHING VIA BUS	58
SWITCHING VIA CROSSBAR	58
INPUT PORT QUEUING	58
PORTE DI OUTPUT	58
MECCANISMI DI SCHEDULING	59
PRIORITY	59
ROUND ROBIN (RR)	60
WEIGHTED FAIR QUEUE (WFQ)	60
LIVELLO RETE	60
PACCHETTO IPV4	61
<b>23/11</b>	<b>62</b>
PROTOCOLLO IPV4 - FRAMMENTAZIONE E REASSEMBLY	62
INDIRIZZAMENTO IP	62
DHCP	63
NETWORK ADDRESS TRANSLATION (NAT)	63
<b>26/11</b>	<b>64</b>
IPv6	64
TUNNELLING	65
APPROCCIO SDN (Software Defined Network)	65
OPENFLOW	66
ESEMPI	66
MATCH + ACTION	67
CONTROL PLANE	67
CLASSIFICAZIONE DEGLI ALGORITMI ROUTING	68
ALGORITMO DI DIJKSTRA (link state)	68
ALGORITMO DI BELLMAN- FORD (distance vector)	69
DIFFERENZE TRA LINK STATE E DISTANCE VECTOR	69
ROUTING SCALABILE	69
InteriorGatewayProtocols (IGP)	70
BGP (Border Gateway Protocol)	70
BASI DI BGP	70

MESSAGGI CONTROLLO BGP	70
SDN CONTROL PLANE	71
TRAFFIC ENGINEERING	71
SNMP (Simple Network Management Protocol)	71
ICMP (Internet Control Message Protocol)	71
<b>30/11</b>	<b>71</b>
SICUREZZA	72
CRITTOGRAFIA	72
CRITTOGRAFIA A CHIAVE SIMMETRICA	73
DES (data encryption standard)	73
AES (Advanced Encryption Standard)	74
MECCANISMI A CHIAVE PUBBLICA E PRIVATA	74
<b>1/12</b>	<b>75</b>
FATTORIZZAZIONE DEI NUMERI PRIMI	75
RSA: Rivest, Shamir, Adelson algorithm	75
AUTENTICAZIONE	76
INTEGRITÀ DEI MESSAGGI - FIRMA DIGITALE	77
<b>3/12</b>	<b>77</b>
MESSAGE DIGEST	77
ALGORITMI DI HASHING	77
MD5	77
SHA-1	77
CERTIFICATION AUTHORITY	77
RENDERE SICURA LA MAIL	78
RENDERE SICURO TCP TRAMITE SSL/TLS	79
SSL cipher suite	79
SSL record	79
IP Sec	79
<b>7 - 10- 11/12</b>	<b>80</b>
FIREWALL	80
ACCESS CONTROL LIST (ACL)	81
ESEMPIO DI APPLICATION GATEWAY PER TELNET	81
ZONA DEMILITARIZZATA (DMZ)	82
RETI WIRELESS	83
PROPRIETÀ DELLE RADIO FREQUENZE	83
PROPAGAZIONE RADIO FREQUENZE	84
RANGE DI PROPAGAZIONE DEL SEGNALE WIRELESS	84
FASE DI UN SEGNALE	85
POLARIZZAZIONE	86
COMPORTAMENTO RADIO FREQUENZE	87

TRASMISSIONE WIRELESS	88
VSWR e INTENTIONAL RADIATOR	90
EIRP	90
Esempio EIRP	91
MISURAZIONE DELL'ENERGIA (dB, W ecc...)	91
dBm	93
DIPOLO e dBi	93
ANTENNE OMNIDIREZIONALI	94
ANTENNE SEMI-DIREZIONALI	95
ANTENNE HIGHLY-DIRECTIONAL	95
FRESNEL ZONE	96
ANTENNE SETTORIZZATE (Selectorized, a Settore)	97
LETTURA PERSONALE (Non obbligatoria)	98
DIVERSITY (EFFETTO)	99
PATH LOSS (Perdita di segnali in funzione della distanza)	99
LINK BUDGET	100
ESEMPIO ESERCIZIO DI APPLICAZIONE DEI VARI ARGOMENTI DI QUESTO CAPITOLO	101
 WIRELESS PARTE 2	 102
SPETTRO DELLE Onde WIRELESS	102
VELOCITÀ DI TRASMISSIONE DEI BIT	102
CODIFICHE	103
LINK BIDIREZIONALE (GENERICO)	104
MULTIPLEXING	106
MODULAZIONE E DEMODULAZIONE	108
PRINCIPALI CODIFICHE DEI SEGNALI	109
CODIFICA TRAMITE SIMBOLI	110
MODULAZIONE GERARCHICA	113
PROBLEMI DEL MONDO RADIO (A LIVELLO DI PROTOCOLLI)	114
PROTOCOLLI CHE RISOLVONO IL PROBLEMA NEL TEMPO	115
IMPLEMENTAZIONE DELL'ALOHA SLOTTED	117
PROTOCOLLI CHE RISOLVONO IL PROBLEMA NELLO SPAZIO	119
FUNZIONAMENTO DI 802.11	119
ALTERNANZA DI COMUNICAZIONE NELLA FASE DISTRIBUITA	121
 APPENDICE 1 - POTENZE DI 2	 123
APPENDICE 2 - FLOW vs CONGESTION CONTROL	123
APPENDICE 3 - HOW TO ESERCIZI	124
APPENDICE 4 - MASCHERE DI RETE	124

## INTRODUZIONE

Definizione di **Rete di Calcolatori**: insieme di dispositivi di calcolo, autonomi e interconnessi

Classificazione delle reti:

- Reti personali (Personal Area Network), PAN
- Reti locali (Local Area Network), LAN
- Reti metropolitane (Metropolitan Area Network), MAN
- Reti geografiche (Wide Area Network), WAN

Ogni dispositivo è dotato di:

- Dispositivi o schede di rete
  - Dispositivi hardware per codificare, trasmettere, ricevere e decodificare dati dal calcolatore alla rete, e viceversa
  - Mezzo di trasmissione
  - es. cavi elettrici, fibra ottica
- Connettore di rete
  - es. connettore RJ45
- Protocolli di rete
  - Insieme di regole software univocamente definite per garantire compatibilità e corretta gestione della comunicazione

## INFRASTRUTTURA DI RETE

Un'infrastruttura di rete rappresenta l'insieme dei collegamenti o connessioni fisiche esistenti tra tutti i dispositivi di una rete.

Tipologie di infrastrutture di rete:

- Punto a punto
- Completamente connesse con cammini ridondanti
- Parzialmente connesse
- Partizioni di rete
  - Gruppo di componenti isolato da altri (reti con connessioni multiple riducono il rischio di partizioni della rete)



Topologia di rete: schemi di connessione dei dispositivi

Esempi di topologie (usate principalmente per LAN e PAN):

- Anello (ring)
  - Ogni componente può comunicare con ogni altro componente inviando i segnali attraverso la sequenza di connessioni in senso orario o antiorario.
- Stella (star)
  - Ogni componente abbia una connessione verso un bus condiviso (cioè una connessione condivisa da tutti).
- Bus

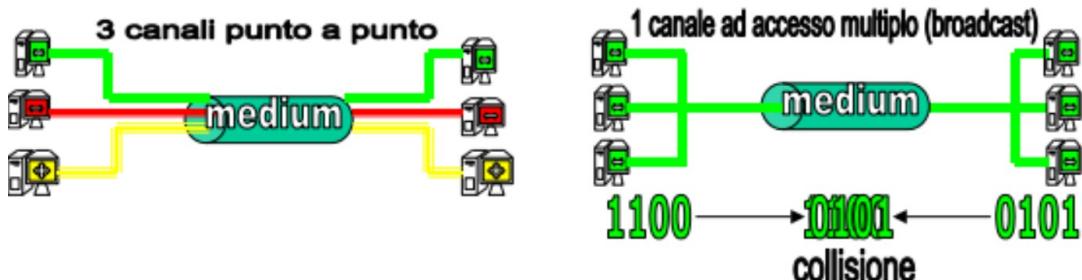
- Albero (tree)
  - prevede un'organizzazione gerarchica delle connessioni come se fosse un albero genealogico: il nodo nonno connette i nodi figli e questi connettono i nipoti e così via



## CANALI DI COMUNICAZIONE

Possiamo vedere un canale di comunicazione come una visione astratta del mezzo di trasmissione. Esistono due tipologie di canale:

- Canale punto a punto
  - Riservato alla comunicazione di due soli dispositivi
- Canale broadcast
  - Tutti i dispositivi ricevono e trasmettono sullo stesso canale
  - Il canale richiede arbitraggio per evitare le collisioni, ovvero la sovrapposizione di trasmissioni sul canale che porta alla distruzione dei dati
  - Le informazioni trasmesse richiedono un indirizzamento (mittente e destinatario)
    - A tal fine si utilizza l'indirizzo MAC (Medium Access Control) delle schede di rete (che è univoco per ogni scheda)



## RETI A COMMUTAZIONE DI CIRCUITO

Una serie di canali in cascata può realizzare un servizio di trasferimento dell'informazione tra due dispositivi anche molto distanti.

Nelle reti a commutazione di circuito (come ad esempio la classica linea telefonica) viene riservato un circuito di canali punto a punto su ogni connessione lungo il cammino dal mittente al destinatario.



## RETI A COMMUTAZIONE DI PACCHETTO

Questo tipo di rete è molto usato quando si usano canali ad accesso multiplo e su internet.

In questo caso i dati vengono divisi in pacchetti indipendenti che vengono spediti sul canale indipendentemente, in particolare:

- Ogni pacchetto deve contenere gli indirizzi di mittente e destinatario
- Il canale viene condiviso tra più flussi di pacchetti, quindi sono richieste meno risorse e sono meglio utilizzate.
- I pacchetti vengono immagazzinati dagli host intermedi e inoltrati verso il destinatario finale
- È possibile che alcuni pacchetti arrivino nell'ordine sbagliato o vengano persi
- La rete subisce un maggior ritardo



## SERVIZI ORIENTATI ALLA CONNESSIONE E NON

Nell'ambito delle reti a commutazione di pacchetto, se consideriamo un servizio non orientato alla connessione (*connectionless*), i pacchetti potrebbero seguire strade diverse per arrivare a destinazione e arrivare disordinati rispetto all'ordine di invio o addirittura non arrivare.

I servizi orientati alla connessione (*connection oriented*), invece, garantiscono la consegna ordinata dei pacchetti ricevuti, secondo l'ordine di invio e la ri-trasmissione di eventuali pacchetti perduti.

## PROTOCOLLI DI RETE

Permettono la compatibilità di dispositivi conformi allo stesso standard attraverso regole e procedure semantiche di gestione dei processi di comunicazione e regole e formati sintattici per definire scambio di messaggi non ambigui.

La definizione dei protocolli di rete deve prevedere e supportare diverse finalità di comunicazione. Non ha quindi senso definire un protocollo rigido, ma ha senso invece definire classi di protocolli, deputate a svolgere e gestire determinate funzioni della comunicazione.

## STANDARD ISO/OSI RM

Questo standard (Open System Interconnection Reference Model) rappresenta un insieme di **sette livelli** per supportare la comunicazione di rete.

Ogni livello si occupa contemporaneamente di gestire alcune problematiche di rete e di fornire ai livelli superiori una visione semplificata della rete.

Il dialogo tra livelli paritari avviene astraendo l'architettura di rete sottostante, mentre quello tra livelli sovrapposti sfrutta un'interfaccia comune per tutti i protocolli.

I sette livelli dello standard sono:

- 7) Livello Applicazione → Contiene le primitive dei dati delle applicazioni
- 6) Livello Presentazione → Traduce i formati dati dell'utente
- 5) Livello Sessione → Controlla lo stato del collegamento
- 4) Livello Trasporto → Rende la connessione affidabile e si occupa del controllo congestione
- 3) Livello Rete → Si occupa di frammentazione, indirizzamento e instradamento dei pacchetti
- 2) Livello MAC / LLC → Regola l'accesso al mezzo trasmissivo e controlla eventuali errori
- 1) Livello Fisico → Codifica, trasmette e riceve i dati



La necessità di accordarsi su regole e servizi comuni per la comunicazione di rete ha lo scopo di permettere una completa compatibilità e supporto alla comunicazione su sistemi, tecnologie e dispositivi eterogenei.

E' molto importante non violare lo stack ISO/OSI in quanto questo permette un corretto ordine di esecuzione delle istruzioni. Senza questo stack si potrebbe creare una struttura di controllo complessa e incomprensibile andando di fatto a compromettere la struttura dello stack, rendendo eventuali modifiche molto difficili e creando il cosiddetto "spaghetti code".

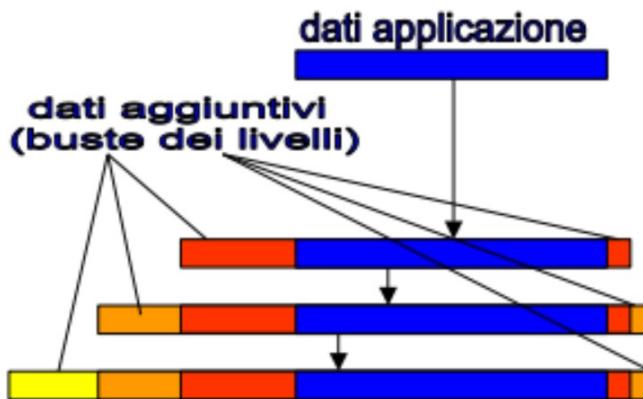
## PROTOCOLLI DI INTERNET

L'architettura dei protocolli di internet utilizza solo cinque dei sette livelli dello standard, escludendo il livello Presentazione e il livello Sessione.

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

In ricezione, ogni livello verifica i dati della busta, agisce di conseguenza, e passa il contenuto della busta ai livelli superiori:

- Il livello trasporto imbusta i dati aggiungendo informazioni utili all'ordinamento e controllo della velocità di invio delle buste
- Il livello rete frammenta i dati in pacchetti, decide il cammino sul quale inviare il pacchetto a seconda dell'indirizzo del destinatario
- Il livello MAC/LLC esegue la consegna finale dei dati a dispositivi di una rete locale.



Vediamo che caratteristiche ha la rete ad ogni livello e le relative problematiche:

- Livello Fisico: la rete è solo un segmento = un mezzo di trasmissione condiviso tra dispositivi
  - regole per codificare e trasmettere dati, con una tecnologia in comune
- Livello MAC/LLC: la rete è locale, può integrare mezzi trasmittivi e tecnologie diverse
  - regole per gli indirizzi dei dispositivi, i tempi di accesso al mezzo, e gestione errori
- Livello Rete: la rete è una collezione di reti, e assume struttura gerarchica
  - regole per indirizzi di rete che nascondano i dettagli locali
  - si definiscono nuovi dispositivi (router) che smistano i pacchetti di dati tra rete e rete
- Livello Trasporto: la rete è una collezione di reti organizzate gerarchicamente
  - regole per la spedizione affidabile di pacchetti e controllo della congestione
- Livello Applicazione: la rete esiste, funziona, ed è utilizzabile dalle applicazioni dell'utente

### RENDERE UN SEGMENTO DI RETE AFFIDABILE

Dato un segmento di rete non affidabile, cioè soggetto a collisioni e altri problemi, è possibile renderlo affidabile implementando un sistema che permetta di sapere se un frame di livello MAC è stato ricevuto dal destinatario:

- Se il frame è ricevuto correttamente il destinatario invia un frame di conferma della corretta ricezione al mittente
- Se il frame viene perso o comunque non ricevuto, destinatario non fa nulla
  - Se il mittente non riceve conferma entro un tempo fissato, trasmette di nuovo il frame continuando a provare finché non ottiene la conferma del successo

### TECNOLOGIE PER SCHEDE DI RETE

- Ethernet
  - Adattatore di tipo cablato, la scheda di rete ascolta il canale e trasmette solo se nessuno sta già trasmettendo. Se viene rilevata una collisione la trasmissione è interrotta per riprovare più tardi.
- IEEE 802.11 (Wi-Fi)
  - Adattatore di tipo wireless, la scheda di rete ascolta il canale e trasmette solo se nessuno sta già trasmettendo, per prevenire le collisioni si dilazionano le trasmissioni nel tempo.
- Token ring

- Utilizza ethernet su dispositivi collegati tra loro in una topologia ad anello, viene utilizzato un frame (**Token**) che viene passato tra i dispositivi come "testimone", Solo chi detiene il token ha diritto di trasmettere, poi deve passare il token.

Lo scopo dei protocolli del livello 2 è nascondere ai livelli superiori i dettagli del mezzo fisico, e mostrare il canale condiviso sul segmento di rete locale come se si trattasse di un canale affidabile. A tal fine il frame di dati viene delimitato mediante particolari etichette di bit, poste all'inizio e alla fine del frame più altri campi di dati utili al protocollo.

### COMPORRE SEGMENTI IN RETI LOCALI

Per unire più segmenti di reti locali è possibile avvalersi di alcuni dispositivi hardware che operano al livello 1 o 2:

NOME	LIVELLO	FUNZIONE
Ripetitore	Fisico (1)	Evita che il segnale si degradi a causa della distanza amplificando e rigenerando il segnale ricevuto
Hub	Fisico (1)	(o repeater multiporta) Funziona come un ripetitore ma amplifica il segnale su più segmenti di rete contemporaneamente
Bridge	MAC (2)	Connette segmenti di una rete locale con tecnologie MAC diverse traducendo da un formato all'altro
Switch	MAC (2)	Simile al bridge ma permette di connettere più segmenti, filtrando i frame e inviandoli sul segmento giusto grazie ad una tabella interna che associa ogni MAC conosciuto al segmento su cui si trova



### IL LIVELLO RETE e IL PROTOCOLLO IP

Le reti locali sono connesse attraverso collegamenti organizzati in modo gerarchico, basati su calcolatori rappresentanti della rete locale (router) a loro volta collegati da linee dati veloci o dorsali (backbone).

All'interno del livello Rete è definito il protocollo IP per la comunicazione.

Questo protocollo richiede nuovi indirizzi: gli indirizzi IP:

- Un indirizzo IP viene associato a una e una sola interfaccia di rete (scheda di rete) con un'associazione univoca tra indirizzo MAC e indirizzo IP
- Può essere statico (sempre lo stesso) o dinamico (può cambiare l'associazione MAC-IP)
- Gli indirizzi IP attualmente usati si riferiscono al protocollo IP versione 4, (IPv4)
  - Un indirizzo IPv4: 32 bit (4 Byte) = sequenza di 4 valori decimali separati da punto
  - Ogni valore decimale può essere compreso tra i valori 0 e 255
- Un indirizzo IP è sempre composto da due parti:
  - Numero della rete IP alla quale appartiene la scheda (network number)
  - Numero dell'interfaccia di rete (host number) all'interno della rete
- Il valore dell'indirizzo IP determina la classe della rete: A,B,C:



## SOTTORETI

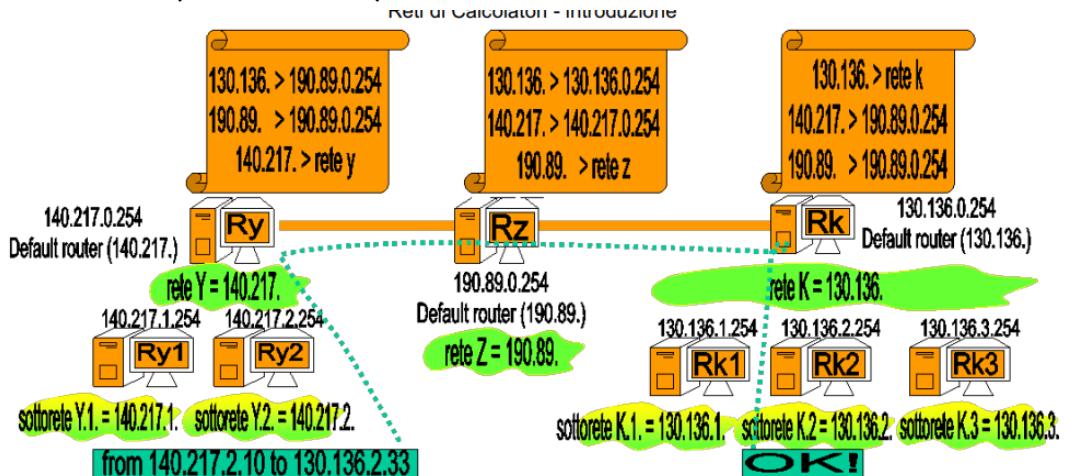
Reti IP e router sono organizzati secondo una gerarchia rete e sottorete:

- Indirizzo di sottorete: bit a 1 nella maschera
- Numero dell'host nella sottorete: bit a 0 nella maschera

ogni sottorete è amministrata da un default router. Indirizzo IP, netmask e default router sono informazioni di configurazione del livello rete



## INSTRADAMENTO (FORWARDING) DEI PACCHETTI



Esempio: l'host appartenente alla sottorete 140.217 invia un pacchetto all'host 130.136.2.33 il pacchetto viene raccolto dal default router 140.217.0.254 e si accorge che non appartiene alla sottorete, quindi invia il pacchetto al default router al livello successivo.

Questo guarda le tabelle di instradamento e invia il pacchetto al default router della rete 130.136.

Questo router sa che il pacchetto è per uno degli host appartenente ad una delle sue sottoreti e spedisce il pacchetto al default router della sottorete indicata nel pacchetto.

L'instradamento dei pacchetti è in sostanza la verifica che fa ogni router per verificare se il pacchetto ricevuto è destinato a lui, alla sua sottorete o al default router.

## ROUTING

Il routing è l'aggiornamento delle tabelle di forwarding dei router, per fare questo bisogna tenere conto dei seguenti problemi:

- Modifiche dei cammini per i dati nella rete: possibili soprattutto in reti senza fili a causa della mobilità degli host o causate da modifiche agli accordi di servizio tra gestori di sistemi autonomi (AS).
- le modifiche dei cammini rendono sbagliate le tabelle di forwarding dei router di conseguenza i pacchetti possono andare perduti, o seguire strade diverse e arrivare disordinati.
- Occorrono reazioni da parte dei router per scoprire nuovi cammini utilizzando protocolli (algoritmi) di routing:
  - invio richieste: qualcuno conosce il modo per arrivare al destinatario?
  - Aggiornamento della tabella di forwarding con il cammino migliore

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 (ICMP) è un protocollo standard di messaggi su internet per definire la comunicazione di informazioni utili alla gestione di Internet. Viene usato da host, router e gateway per scambiare informazioni di livello rete, usando pacchetti definiti con il protocollo IP per notificare errori di configurazione e gestione dei collegamenti:

- Rete di destinazione non raggiungibile (possibile interruzione di rete?)
- Rete di destinazione sconosciuta (indirizzo di rete male specificato?)

- Host destinazione non raggiungibile (host spento o scollegato?)
- Host destinazione sconosciuto (indirizzo di host male specificato?)
- Host destinazione sconosciuto (indirizzo di host male specificato?)
- Ricerca di un cammino alternativo per la destinazione (se esiste)

## APPLICAZIONI BASATE SU ICMP

Applicazione **PING**: test di verifica di connessione tra due host

Comando → ping (indirizzo IP o sito web)

Applicazione **Traceroute**: mostra la sequenza di router attraversati da un pacchetto per arrivare all'host destinazione

Comando → tracert (indirizzo IP o sito web)

Può succedere con tracert che alcuni router in mezzo al cammino cambino. Per questo motivo a volte i pacchetti arrivano disordinati (potrebbero "sorpassarsi" prendendo strade diverse con maggiore velocità o minor traffico).

I router di partenza e di arrivo di solito non cambiano (a meno che il sistemista non crei percorsi ridondanti).

## 19-20/10

### PROTOCOLLO ARP E RARP

((reverse) Address Resolution Protocol)

Se un router riceve un Pacchetto destinato ad un IP della sua sottorete, deve produrre un frame MAC/LLC che riporti il MAC del destinatario (NON L'IP) da passare al livello 2 per trasmetterlo sulla rete locale.

**ARP** viene usato se router non conosce MAC corrispondente a IP → genera frame spedito a tutti su rete locale dove chiede il MAC corrispondente all'IP.

Se il dispositivo esiste, invia un frame a livello MAC indirizzato al router nel quale specifica il suo indirizzo MAC.

**RARP** funziona come ARP ma al contrario, ovvero richiede l'IP corrispondente al MAC.

Questi protocolli servono a riempire la tabella ARP del router.



Se l'invio di un pacchetto fallisce, il router ritenta fino ad un massimo prestabilito di volte (nel WIFI di solito è 8), se fallisce ancora notifica il fallimento ai livelli superiori.

A volte solo l'utente può risolvere il problema (es. router spento, ricevitore troppo distante ecc), altre volte il router cerca altre strade alternative per far arrivare i pacchetti a destinazione (servizio best effort).

## ASSEGNAZIONE IP: DHCP

Alcuni enti nazionali (es. il CNR) assegnano alle organizzazioni che ne fanno richiesta i numeri di rete di classe A, B o C.

L'amministratore di rete assegna manualmente IP e MAC staticamente (sempre uguali e quindi sempre riservati anche quando la macchina è spenta) OPPURE un server **Dynamic Host**

**Configuration Protocol** (DHCP) assegna dinamicamente gli IP attraverso un metodo automatico (per reti wireless, locali e domestiche).

Un server DHCP è un host che implementa il servizio di assegnazione di IP agli host richiedenti.

Un DHCP dispone di un blocco di numeri di host liberi per la sua rete.

Il protocollo associa IP a MAC dei dispositivi che scelgono di affidarsi al servizio.

Gli IP possono cambiare, in base alla configurazione, quando la macchina si accende, quando si connette o quando vuole il server DHCP.

Gli indirizzi dinamici vengono dati con un tempo limite (**LEASE**). Quando il leasing scade il server verifica che la macchina sia ancora in rete: se non lo è, l'indirizzo torna disponibile.

Gli host mandano una richiesta in broadcast per accedere al DHCP, che a sua volta comunica numero di host, rete, maschera e default gateway.

Esiste una tabella DHCP dove sono segnate le corrispondenze tra host e MAC.

## IPV6 E TUNNELLING IPV4

IPv6 è attivo dal 1990 visto che gli indirizzi IPv4 sarebbero finiti tra il 2008 e il 2018.

Gli indirizzi IPv6 sono composti da 16 byte (128 bit) e non sono compatibili con IPv4.

Esiste una nuova struttura dei campi della busta di livello rete (IP) → identificazione di parametri per differenziare flussi di dati con priorità diverse (QoS) → più code in uscita dal router secondo priorità.

IPv4 può frammentare i pacchetti in più parti (problemi per ricevente ma anche mittente);

IPv6 non frammenta i pacchetti.

IPv6 non può sostituirsi a IPv4 "in corsa", quindi IPv6 oggi funziona su reti diverse con appositi router.

Ci sono casi di integrazione tra IPv4 e IPv6 usando tecniche di tunnelling (ovvero spedire pacchetti IPv6 in buste IPv4), in particolare il router IPv6 invia il pacchetto ad un router in grado di effettuare un bridging a livello rete tra IPv6 e IPv4 (dual stack); all'arrivo del pacchetto a destinazione, un altro router equivalente fa lo "strip out" del pacchetto (ovvero elimina la busta IPv4) e lo fa proseguire.

Con il tunneling **non è garantita la QoS** perché la priorità non è prevista in IPv4.

IPv6 resta best effort.

Esistono servizi che garantiscono la QoS → internet 2 o servizi integrati

## APPROFONDIMENTO SULLA RETE GARR

[www.garr.it](http://www.garr.it) rete della ricerca.

PoP → centri che fanno il routing (Point of Presence)

Dorsali → dove viaggiano segnali tra PoP

GEANT → rete di reti di ricerca europee

Internet exchange points → punti di scambio dati tra "isole PoP" diverse

Cogent → rete tier 1

## LIVELLO TRASPORTO

**User Data Protocol (UDP)** → connectionless (per applicazioni che tollerano perdite ecc) e

**Transmission Control Protocol (TCP)** → rende servizio trasporto affidabile: servizio connection oriented

**Socket** = coppia <IP, numero di porta> dell'applicazione che deve ricevere i dati a livello trasporto

1) TCP configura i port number del socket TCP;

2) I pacchetti vengono numerati sequenzialmente, riordinati e i duplicati eliminati; vengono inviati ACK di livello trasporto: i pacchetti non ricevuti entro un certo timeout vengono spediti di nuovo;

3) Gestione delle congestioni: se pacchetti in arrivo sono più della capacità di output del router o se router che riceve è già congestionato non cattura più pacchetti (DROP), gli altri router iniziano a congestionarsi a loro volta → l'unico che può decidere velocità di trasmissione è il mittente → TCP gestisce la velocità di trasmissione trovando un compromesso;

4) Controllo flusso pacchetti grazie ad un meccanismo a finestra scorrevole (sliding window)

Su internet si applica connubio **TCP/IP** che consente smistamento pacchetti verso applicazioni in ascolto sulle varie porte. TCP richiede **attivazione della connessione punto a punto tra due socket** mettendo in comunicazione le applicazioni in attesa sul socket (ogni applicazione ha una porta). es. PC1 (client) invia richiesta TCP di connessione sul socket del PC2 (server) → se il socket esiste e non è occupato, TCP di PC2 risponde ok → se riceve ok da PC2, il TCP di PC1 può inviare i dati di configurazione (three way handshake) → avviene scambio dati a livello TCP → quando la connessione viene rilasciata si liberano le porte TCP usate.



## SLIDING WINDOW

Viene incrementata gradualmente la velocità di invio dei pacchetti finché non si riceve un sintomo di congestione, a questo punto il protocollo TCP ricomincia a trasmettere a partire dalla velocità minima.

La sliding window è un valore intero e rappresenta il numero di segmenti di livello trasporto spedibili consecutivamente dal mittente (quindi ricevibili dal destinatario); dopo aver spedito questo numero massimo, il client deve aspettare l'ACK (che può essere singolo per ogni pacchetto o cumulativo).

Se non lo riceve probabilmente c'è un router congestionato quindi il client ricomincia a trasmettere dalla velocità minima.

Si comincia con 1 pacchetto e si raddoppia la finestra mano a mano che si ricevono gli ACK.



## NOMI DI DOMINIO E SERVIZIO DNS

Lato utente si utilizzano i nomi per identificare le risorse in rete e non gli IP.

I nomi (mnemonici e gerarchici) sono assegnati in modo univoco come i numeri di rete, sono arbitrari ma non duplicabili entro il dominio stesso.

I protocolli di rete e i router pretendono di usare indirizzi IP quindi si ricorre al **Domain Name System (DNS)**, basato su una catena di server DNS organizzati gerarchicamente.

Ogni host in rete deve conoscere almeno un DNS server e ogni server DNS conosce almeno un DNS server superiore.

I server ricevono richieste (protocollo DNS) e forniscono indirizzi IP (se un server non conosce la risposta inoltra la richiesta DNS a un server superiore).

I server DNS radice (DNS root server) conoscono tutti i domini e i loro IP! (es. .it .com ecc)

Una richiesta DNS può propagarsi secondo due modalità:

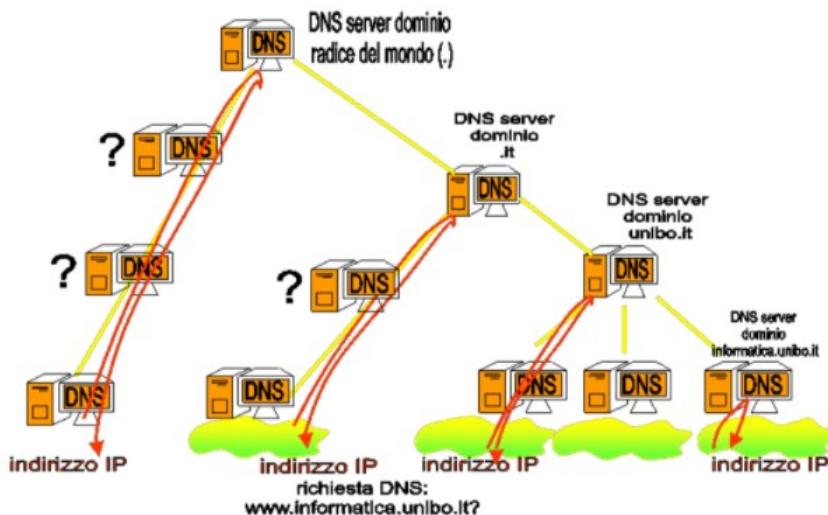
- 1) Il DNS locale propaga la richiesta al server superiore senza far lavorare il client (catena dei server lavora molto ma dopo una richiesta la risposta può essere messa in cache) - DNS ricorsivo.

Quando viene ottenuta la risposta, essa viene ritrasmessa al richiedente.

- 2) Il client propaga la sua richiesta autonomamente (chiede ad un DNS, se non sa rispondere invia IP del DNS successivo) - DNS iterativo.

Il server lavora meno ma il sistema è meno efficiente.

(Il DNS Google è 8.8.8.8 (welcoming host che smista richieste)).



## LIVELLO APPLICAZIONE

Contiene le primitive e i protocolli per spedire e ricevere dati delle applicazioni:

- I livelli Sessione e Presentazione sono raramente implementati sugli host di Internet
- Il livello applicazione si appoggia sul livello trasporto (in particolare sul protocollo TCP)

Esempi di protocolli con relative primitive (servizi supportati da internet (web)):

Posta elettronica (E-mail): basati su protocolli di livello applicazione SMTP, POP3, IMAP

- Simple Mail Transfer Protocol (SMTP): per la spedizione e trasporto dei messaggi
- Post Office Protocol 3 (POP3): per la consegna dei messaggi all'utente
- Internet Mail Access Protocol (IMAP): alternativa migliorata a POP3

Il World Wide Web (WWW) è basato sul livello applicazione e protocollo HTTP

Hyper Text Transfer Protocol (HTTP) è il protocollo per trasferire pagine web

DNS: Domain Name Service (protocollo DNS)

## SERVIZI CLIENT/SERVER E PEER TO PEER

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

- Architettura Client/Server:  
I Client sono host che spediscono richieste di servizio e i Server sono host sui quali sono in esecuzione i servizi che soddisfano le richieste.
- Architettura Peer to Peer (P2P)  
Tutti gli host sono contemporaneamente sia client che server:  
Ogni host agisce da Server cercando di soddisfare le richieste ricevute e agisce da Client quando spedisce ad altri host le richieste (per se stesso o per soddisfare richieste di terzi) ("Se lo so sono server, altrimenti sono client").
- Servizi ibridi: esistono server che aiutano solo a trovare in fretta i giusti host Peer to Peer.

## 26/10

### FREQUENCY DIVISION MULTIPLEXING (FDM)

Oggi i telefoni di casa sono connessi al modem e fanno viaggiare le informazioni sulla rete dell'ISP eliminando la rete telefonica (VoIP).

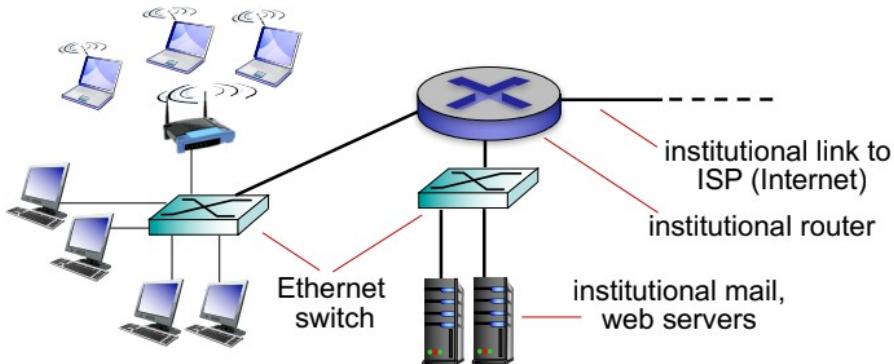
Il mercato americano realizza la backbone dell'edge network attraverso cavo coassiale cioè un mezzo condiviso in cui viaggiano le info di tutti gli host di un quartiere (per esempio).

Le trasmissioni si distinguono separando dei canali, che utilizzano diverse frequenze per trasmettere diverse cose come: video, dati e informazioni relative al canale di controllo cioè un canale dedicato alla gestione dell'architettura.

Ricapitolando: in un'architettura domestica abbiamo un modem collegato al mezzo che porta alla centrale di commutazione, che a volte fa anche da router / switch.

In un'architettura aziendale o istituzionale abbiamo un router che divide i dati in entrata e in uscita su due strade:

- A uno Switch che gestisce gli host collegati e possibilmente anche un access point wireless.
- A uno Switch che porta alla rete DMZ (DeMilitarized Zone) a cui sono collegati gli host a cui è possibile accedere dall'esterno come siti web, caselle di posta aziendali etc...



Come si crea un'architettura del genere che separa host pubblici e host privati ("istituzionali")?  
Il router istituzionale separa i pacchetti in entrata attraverso l'IP destinatario e se è diretto alla rete privata può bloccare direttamente i pacchetti. Se la trasmissione proviene dall'interno, il router funziona normalmente, quindi possono essere aperte delle connessioni di socket tra server e client.  
Questo meccanismo è detto **Port Filtering o Mapping**. → vedi firewall

## MEZZI DI PROPAGAZIONE DEL SEGNALE

- Mezzi guidati:
  - Ethernet TP (twisted Pair): fatto con doppino di rame intrecciato e isolato per schermare dalle interferenze, esistono di due categorie:
    - Categoria 5: fino a 100 mbps; categoria 6: fino a 1 gbps
  - Cavo Coassiale: fatto con due conduttori di rame concentrici attraverso cui possono essere creati diversi canali.
  - Fibra Ottica: Fatta di vetro che trasmette impulsi di luce, trasmissioni molto veloci e di larga capacità, con tasso di errori molto ridotto.
- Mezzi non guidati: wifi, radio etc...
  - Vantaggi: non dovere avere sempre un cavo attaccato.
  - Svantaggi: interferenze dovute all'ambiente, consumo di energia elevato, elevato rischio di intrusione

## CORE NETWORK

E' una mesh (grafo) di router interconnessi che prendono i pacchetti che viaggiano tra le varie reti di accesso e ne fanno il forwarding, gli archi del grafo sono link in fibra ottica.

## STORE AND FORWARD

Quando un host sorgente spedisce dati a un host destinazione, abbiamo un tratto di mezzo trasmissivo, diversi router intermedi e infine il destinatario.

Quando abbiamo un pacchetto, questo viene immagazzinato dal router in un buffer (coda) di ingresso, prima di inoltrare le informazioni in entrata si aspetta che arrivi tutto il pacchetto poi, se il buffer di uscita è libero, si procede all'inoltro.

Se il buffer di ingresso dovesse saturarsi i pacchetti in arrivo verrebbero scartati (drop).

Un pacchetto non è mai diviso tra più link successivi.

## RITARDO NELLA TRASMISSIONE DEI PACCHETTI

Spesso i messaggi che le applicazioni devono scambiarsi vengono divisi in porzioni più piccole dette pacchetti. Supponiamo di avere pacchetti da  $L$  bits trasmessi con un rate  $R$ : il ritardo di trasmissione dei pacchetti, ovvero il tempo necessario a trasmettere un pacchetto (quindi  $L$  bits) sul mezzo

trasmissivo sarà pari a  $\frac{L(\text{bits})}{R(\text{bits/sec})}$ .

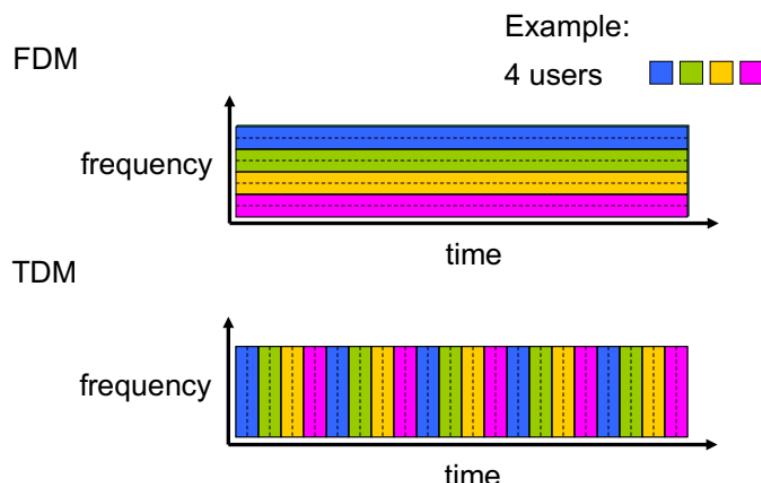
Quindi il tempo che ci mette un pacchetto ad arrivare da mittente a destinatario è di  $L/R * \text{tutti i router intermedi da cui passa} + \text{il ritardo di propagazione del segnale}$  (trasmissione non istantanea ma dipendente dal mezzo trasmissivo).

## TIME DIVISION MULTIPLEXING

TDM → tempo limitato dedicato alla trasmissione sui singoli canali e pattern ciclico

## FREQUENCY DIVISION MULTIPLEXING

FDM → indifferente al tempo, si basa sul riconoscimento delle diverse frequenze su cui avviene la trasmissione dei vari host



## ALGORITMI DI ROUTING E FORWARDING

Gli algoritmi di Routing creano e aggiornano le tabelle, gli algoritmi di Forwarding le utilizzano.

**27/10**

## PACKET SWITCHING

Nelle tabelle di routing di solito abbiamo una lista di destinazioni associate a una porta di uscita del router, questo richiede molto tempo per scorrere e comparare il pacchetto corrente con le destinazioni nella tabella.

Esistono quindi le tabelle di instradamento che richiedono un solo ciclo di clock per essere controllate.

Il packet switching è molto utile quando abbiamo un traffico cosiddetto a burst (ovvero con molti utenti contemporaneamente) poiché mentre uno non trasmette, il canale non è occupato e un altro può inviare i suoi pacchetti (condivisione delle risorse).

Quindi le risorse sono allocate in modo dinamico e non si necessita di tempo per il call setup.

Tuttavia, può esserci comunque congestione che porta a ritardo e perdita di pacchetti.

Es: Supponiamo di avere n flussi in ingresso; ogni utente ha 2 stati: attivo e inattivo, gli utenti attivi possono trasmettere 100 kbps.

Supponiamo che siano attivi il 10% del tempo: la capacità del link in entrata sarà di 1 Mbps.

Le risorse sono allocate dinamicamente tramite la formula  $2L/R$  con L bit per pacchetto e R Bps

## STRUTTURA DI INTERNET

Per connettere milioni di Internet Service Providers (ISP) non possiamo creare una connessione per ogni coppia poiché sarebbe estremamente costoso e avremmo  $N^2$  collegamenti mentre noi puntiamo ad averne  $\log N$  o, al massimo, N.

Un'alternativa valida è creare vari livelli di architetture di reti gerarchiche dove abbiamo le ISP globali che connettono gli ISP regionali che a loro volta connettono le reti di accesso locali

Gli ISP globali sono più di uno e sono connessi tra loro tramite IXP (Internet Exchange Point).

Alcune società, in particolare i content providers si inseriscono nella rete per portare i loro contenuti agli utenti.

Ad esempio Google paga per mettere su la sua infrastruttura e poi guadagna dal content deliver.

Esempi Tier 1 ISP americani: NTT, AT&T etc..



## ESERCIZI SULLE RETI (packet delay)

Ingresso = uscita = 500 → primo pacchetto perso dopo 112

$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$

delay totale = delay processing + delay accodamento + delay trasporto + delay propagazione

$d_{\text{proc}}$ : verificare eventuali errori, determinare link di output (tempo < msec)

$d_{\text{queue}}$ : ritardo causato dall'accodamento dei pacchetti (dipende dal livello di congestione router)

$d_{\text{trans}}$ : ritardo del trasferimento →  $L/R$

$d_{\text{prop}}$ : ritardo di propagazione →  $d/s$  dove d è la lunghezza del link e s è la velocità di propagazione

R = larghezza banda (bit/s)

L = dimensione pacchetti (bit)

a = numero medio pacchetti che arrivano

$L * a$  = numero medio di bit che arrivano o che devono essere trasmessi

$La / R < 1$  (circa 0): meno entrate di quanto esce → basso ritardo

$La / R \rightarrow 1$  (tende a 1): ritardo di accodamento sempre crescente

$La / R > 1$  : più entrate che uscite → infinito ritardo (medio)

**Throughput:** velocità effettiva di trasferimento dei bits tra mittente e destinatario.

Definiamo **bottleneck link** il tratto del collegamento dove il throughput è minore.

## 29/10

### CREARE UN'APPLICAZIONE DI RETE

- Creare un'applicazione di rete significa scrivere programmi che funzionino su sistemi (host finali) differenti e comunichino attraverso la rete
- Non serve sviluppare software per i dispositivi del network-core (non eseguono applicazioni utente)

#### Server

- Sempre disponibile
- Ha IP permanente
- Sono presenti data centers per rendere i sistemi scalabili
- Esistono server deputati allo smistamento di pacchetti provenienti dall'esterno

#### Client

- Comunica con il server
- Può non essere sempre connesso
- Può avere IP dinamico
  - (client chiede a server e quindi comunica il suo IP attuale → se cambia IP durante TCP, il socket viene chiuso)
- Non comunica con altri client

### ARCHITETTURA PEER TO PEER (P2P)

- Non ci sono server sempre online
- I nodi non sono gerarchizzati unicamente sotto forma di client o server fissi, ma anche sotto forma di nodi equivalenti o 'paritari' (peer), potendo fungere al contempo da client e server verso gli altri nodi terminali (host) della rete
- Diversi host possono comunicare tra loro
- I peers chiedono e forniscono servizi agli altri peers
- Auto scalabilità → più peer si interconnettono per mantenere una connessione stabile e funzionale
- I peer interconnessi possono cambiare IP
- Le macchine tengono traccia di quali client sono connessi, quindi tracciano IP
- Come i virus: la tua richiesta diventa virale e viene diffusa su diverse macchine
- La gestione di questa architettura è particolarmente complessa

### COMUNICAZIONE TRA PROCESSI

- Processo = programma eseguito su 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.
- Si rimane nella stessa macchina grazie ad indirizzo di loopback 127.0.0.1 ma si utilizza comunque lo stack ISO/OSI
- host diversi (scambio di messaggi (message passing) - socket based communication)

Processi:

client: inizia la comunicazione

server: attende di essere contattato

## DIFFERENZE TRA P2P e SERVER

SERVER:

- Ha tutte le informazioni necessarie per rispondere ai client
  - Se Host1 ha una richiesta questa viene trasmessa al server, Host1 otterrà la risposta
  - Se Host2 ha la stessa richiesta di H1 la chiede comunque al server e questo gliela fornisce.
  - Ora H1 e H2 possiedono la stessa informazione non condivisa

P2P:

- Host1 ha una richiesta → **il server trova** un altro Host che conosca la risposta;
- Questo nuovo Host diventa SERVER e spedisce la risposta;
- Se Host2 dovesse avere la stessa richiesta, Host1 potrebbe accorgersi di avere già la risposta e quindi fungere da server e fornire la risposta prima che Host2 chieda al server P2P.
- Ora la richiesta è su due possibili server

## WELL-KNOWN-PORT NUMBERS (WKPN)

- HTTP 80
- Mail 25
- da 0 a 1023 molti sono WKPN
- In totale ci sono 16 bit per identificare le porte (64.000 porte totali)
- Porte alte (da 16/32 mila) sono create estemporaneamente per creare comunicazioni tra processi
- Si usa virtualizzazione per superare il limite del numero di porte

## COSA DEFINISCE IL PROTOCOLLO DI LIVELLO APPLICAZIONE?

- Tipi di messaggi scambiati
- Sintassi dei messaggi (quali campi ci sono nei messaggi e come sono compilati)
- Semantica dei messaggi (significato dei campi nei messaggi)
- Regole per decidere quando e come processare invii dei messaggi e risposte

Esistono protocolli aperti (univoci ma conosciuti grazie alle RFC) e proprietari (oscuri).

## COSA SERVE ALLE APPLICAZIONI?

- Integrità dei dati;
  - Non sempre → per transazioni bancarie serve sicuramente ma per servizi di streaming audio può non essere necessaria
- Timing;
  - In base al Jitter è possibile modificare la dimensione del buffer (es. streaming)
  - Alcune app richiedono poco ritardo per funzionare (es. VoIP)
- Throughput;
  - Alcune app richiedono un throughput minimo per funzionare, altre (app elastiche - es.email) funzionano con qualunque throughput
  - Viene fatta la richiesta del flusso con cui effettivamente si trasmette
- Sicurezza;

- Es. dati criptati
- Protocolli di livello applicazione
  - HTTP: es. YouTube (protocollo trasmissione TCP o UDP)
  - SMTP: email (protocollo trasmissione TPC)
  - RTO: skype (proprietario, protocollo trasmissione TCP o UDP)

## RENDERE UDP E TCP SICURI

TCP e UDP non sono criptati, si ricorre quindi al Secure Socket Layer (SSL) che crea una connessione TCP protetta grazie all'autenticazione e assicura data integrity.

## WEB e HTTP

Le pagine web sono formate da oggetti che possono essere in formato HTML, JPEG o molti altri.

HTML: file di base che usano le pagine web: la pagina scarica le informazioni riguardanti come devono essere visualizzati le scritte e gli oggetti (attraverso un codec sono elaborate e mostrate).

Ogni oggetto è indirizzabile grazie ad un URL (Uniform Resource Locator), nella forma:

(www).nomehost.dominio/percorso

Il file HTML contiene le references (URL) per trovare gli oggetti da mostrare che possono anche essere presenti su diversi server nel mondo.

Tutti i browser visualizzano le pagine in modo **conforme allo standard**.

## RTT

Definiamo RTT 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 di trasmissione del 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).

## HTTP USA TCP

Funzionamento del protocollo HTTP tramite socket TCP:

- Il client avvia una connessione TCP alla **porta 80**
- Il server accetta la connessione TCP
- HTTP invia file richiesto
- Connessione TCP si chiude:
  - Per ogni file con HTTP 1.1 → connessione non persistente;
  - HTTP 1.2 usa invece una connessione persistente
- I server non tengono traccia delle richieste passate dei client, quindi deve pensarci il browser

## METODI DI HTTP

- GET → Trova e restituisce la risorsa richiesta
- POST → si usa di solito quando il client compila un form per far variare il contenuto della pagina in base all'input dell'utente nel form
- URL

I comandi utilizzati per realizzare le funzioni (es. GET) sono implementati al livello 7.

Per connettersi ad un server da browser è necessario digitare: IP:port/nomefile.html

- esempio visto a lezione → <http://130.136.5.36:8427/pippo.html>
  - porta 8427: welcoming socket del server
- Se si accede ad un dominio senza specificare il file richiesto si viene reindirizzati a sito/index.html (splash page)

## HTML



## 2/11

### COOKIES

Molti server implementano i cookies per tenere traccia dei client e di alcuni status che vengono raggiunti durante la comunicazione.

Per implementare i cookies in HTML si usa la parola riservata cookie seguita da un intero.

La prima volta che client contatta il server, avviene la richiesta HTTP(GET...):

Se il server non ha mai visto il client gli assegna un numero e lo registra nel suo backend (in caso di registrazione memorizza anche ulteriori dati) e il cookie viene inserito con set-cookie nell'header della risposta.

Il browser associa i cookies ai siti dentro una cache locale (su disco) e quando fa una richiesta successiva (sullo stesso browser) scrive sempre il cookie per far sì che il server lo riconosca, anche se la connessione TCP si è chiusa.

I cookie non cambiano a meno che non si cancellino.

Quando si accede da un altro browser il sito riconosce il cookie in base ai dati dell'utente stesso: il server fornisce un secondo cookie e tiene traccia di entrambi.

I cookie vengono riciclati: quando si rimane inattivi per troppo tempo questi scadono e passano ad altri client. Fare richieste con i cookies di altri utenti costituisce violazione della privacy.

## PROXY

Verifica se le pagine del server sono in cache e risponde all'host senza passare direttamente dal server web remoto → aumenta la scalabilità: basta inoltrare una volta a livello locale e tutti gli host allo stesso livello ci accedono

- Se l'host invia una richiesta di accesso a qualcosa non presente in cache, il proxy invia una richiesta al server (e mette in cache la risposta)
- Analizzando il proxy durante il passaggio dei dati si vedono diverse informazioni (pericolo di spyware)
  - Replay attack: rimandare indietro pacchetti intercettati ("se ha funzionato prima, funziona anche dopo")

Web cache: molto poco costoso, riduce il collo di bottiglia: risolve la richiesta localmente grazie all'utilizzo del codice 304: not modified → la richiesta appena effettuata non è stata modificata (il file già in cache è corretto)

Se il file fosse stato modificato il campo <data> presente nella risposta conterrebbe tutto il contenuto del file stesso.

## E-MAIL

La gestione di una casella di posta elettronica coinvolge tre componenti principali:

- User Agent
  - La pagina web o l'applicazione dove è possibile scrivere e leggere le mail
- Server Mail
  - Il server dove vengono memorizzate le mail, diviso in due parti:
    - Mailbox, dove sono presenti i messaggi per l'utente
    - Coda dei messaggi, dove si trovano i messaggi in uscita
- Protocollo SMTP
  - Adottato tra server per far avvenire lo scambio delle mail
    - Il client diventa il mittente mentre il server il destinatario
    - Utilizza TCP sulla porta 25 per garantire il trasferimento affidabile delle mail
    - I comandi vengono dati in ASCII mentre le risposte sono dei codici
      - Il messaggio deve essere in ASCII da 7 bit



- Protocollo POP3
  - Serve a permettere la lettura della posta elettronica, è stato sostituito da IMAP
  - Diviso in due fasi:
    - Fase di autorizzazione: è necessario indicare username e password;
    - Fase "operativa": grazie ad una serie di comandi è possibile visualizzare la posta, cancellare dei messaggi o uscire.

## DNS (Domain Name System)

Servizio O protocollo

Traduce gli indirizzi nominativi (es. www.youtube.it) in IP

- Sfrutta un database distribuito: implementato grazie ad una gerarchia di tanti server;
- Utile anche per conoscere il server SMTP del dominio;
- Application-Layer protocol: host e server dei nomi comunicano per risolvere i nomi (traduzione IP/nomi).

## FUNZIONI DEL SERVIZIO

- Traduzione dei nomi
- Aliasing (anche dei mail server)
- Distribuzione del carico: possiamo avere più IP corrispondenti ad uno stesso nome

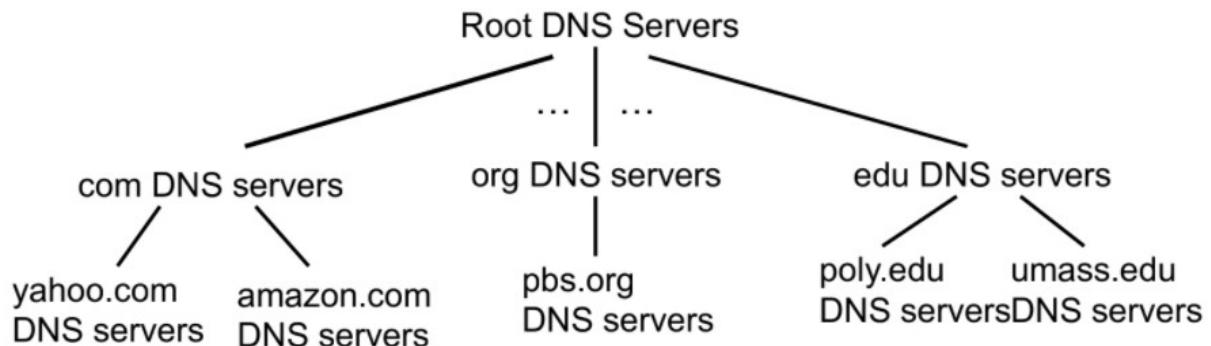
Il DNS non è centralizzato ma distribuito per non avere un solo point of failure e per avere un servizio più efficiente, che ci porta ad avere un approccio distribuito.

## APPROCCIO DISTRIBUITO

Organizzato come un albero, quando una richiesta non può essere soddisfatta viene mandata al DNS superiore (in modo simile al gateway predefinito nelle tabelle di instradamento).

Se nessun DNS riesce a risolvere un nome, lo rimanda al root che lo inoltra dalla parte giusta verso il basso dell'albero.

Al livello più basso dell'albero ci sono i DNS authoritative (es yahoo, amazon); al piano superiore ci sono i server authoritative (.com, .org, .edu ecc...); il livello più alto è il dominio radice (esistono almeno 13 server (cluster) root).



Il server DNS mantiene le cache che sa essere corrette (non vengono mai cambiate!), anche se tutte le righe della tabella cache del server hanno un TTL in quanto soggette ad obsolescenza.

Alle subnet potrebbe corrispondere un DNS locale (che non fa parte della gerarchia sopra indicata) per evitare colli di bottiglia nella parte alta dell'albero.

Il DNS locale tiene una cache di tutti i nomi/IP forniti recentemente così da essere più veloce nel caso gli venisse chiesto un'altra volta e per evitare una nuova propagazione della richiesta nell'albero DNS.

Nel caso in cui il DNS locale non sappia risolvere un nome procede con un'interrogazione ricorsiva o iterativa (a seconda dell'implementazione) al suo **default name server**.

Le richieste inoltrate ai DNS che vengono perdute non sono recuperate.

## DNS RECORD TYPE

I server DNS memorizzano degli elenchi di risorse detti RR nelle loro tabelle.

Questi dati hanno il seguente formato: **RR (name, value, type, ttl)**, interpretato a seconda di type:

TYPE = ...	
A	<b>name</b> è l'hostname e <b>value</b> è l'indirizzo IP
NS	<b>name</b> è il dominio e <b>value</b> l'hostname dell'autoritative name server per quel dominio
CNAME	<b>name</b> è un alias per un nome convenzionale e <b>value</b> è il nome canonico
MX	<b>value</b> è il nome del mailserver associato a <b>name</b>

Le richieste e le risposte hanno entrambe lo stesso formato:



## POSSIBILI ATTACCHI DNS

- DDos
  - Bombardamento TLD o del Root server
- Reindirizzamenti
  - Man in the middle:
    - È possibile fare spoofing fingendosi il DNS, spegnendo, interrompendo o dirottando il servizio dopo aver però convinto diversi host e router.
  - DNS poisoning:
    - Consiste nell'"avvelenare" un DNS cambiando le tabelle nella cache: le richieste verranno mandate nella direzione sbagliata e scadranno (drop causato da out-of-date).
- Sfruttare il DNS per il DDoS (DNS amplification):
  - Invio di richieste con un indirizzo ottenuto tramite spoofing verso un IP bersaglio.

### FILE DISTRIBUTION TIME (FDT)

$F$	Dimensione del file	$u_i$	Capacità di upload del i-esimo host
$u_s$	Capacità di upload del server	$d_i$	Capacità di download del i-esimo host

Tempo distribuzione file con l'approccio client-server

Server:

- Spedire 1 copia =  $F/U_s$
- Spedire N copie =  $F*N/U_s$

Client (download):

- $D_{min} =$  minimo download rate del client
- Tempo minimo di download del client =  $F/D_{min}$

Quindi il tempo di distribuzione di un file di dimensione  $F$  a  $N$  client sarà:

$$D_{cs} \geq \max\{N*F/U_s, F/D_{min}\}$$

Non importa quanto grande sia la banda del server perchè potrebbe verificarsi un bottleneck in qualunque parte della rete.

Un'architettura client server per questo tipo di servizio non è scalabile, il fattore di scala è lineare.

### PEER TO PEER FDT

Nell'approccio Peer to Peer il server termina il suo lavoro non appena ha fornito il file almeno ad un client, che a questo punto può diventare server e offrire il file agli altri peers.

Tempo distribuzione file con l'approccio P2P:

Server:

- Deve condividere almeno una copia del file
- Tempo di upload di una copia:  $F/U_s$

Client:

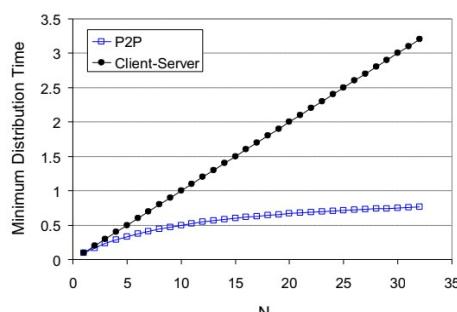
- Ogni client deve scaricare una copia del file
- Tempo di download minimo del client:  $F/D_{min}$
- Tutti i client insieme devono scaricare un totale di  $N*F$  bits
- La velocità massima di caricamento sarà quindi  $u_s + \sum u_i$

Quindi il tempo di distribuzione di un file di dimensione  $F$  a  $N$  client sarà:

$$D_{p2p} \geq \max\{F/U_s, F/D_{min}, N*F/(u_s + \sum u_i)\}$$

Questa modalità di distribuzione è ampiamente più scalabile: su un grafo, al crescere dei client il tempo di distribuzione tende a diminuire.

$$\text{client upload rate} = u, F/u = 1 \text{ hour}, u_s = 10u, d_{min} \geq u_s$$



## BitTorrent

Nei torrent il file è diviso in chunks di pochi Kb (nel caso di BitTorrent 256).

Tracker: tiene traccia degli IP connessi in quel momento e quali file/servizi sta offrendo.

Quando un client si unisce alla rete di peer si registra su un register indicando l'IP e i chunks in suo possesso (spesso vengono accumulati durante la permanenza sul torrent) e ottiene la lista di peer nel suo stesso sottoinsieme di host ("i vicini") dal tracker.

Mentre un peer scarica un file procede ad inviarne alcune parti (chunks) ad altri host (variabili).

Quando il peer ha il file intero può lasciarlo (altruistically) o rimuoverlo (selfish) dal torrent.

- Per richiedere un file il peer chiede ad ogni vicino la lista di chunks in suo possesso e procede poi a richiedere quelli a lui mancanti partendo da quello meno frequente.
- Per inviare dei chunks i peer adottano l'approccio tit-for-tat, ovvero li inviano ai quattro vicini che gli stanno a loro volta trasmettendo dei file con il rate più alto (lista aggiornata ogni 10 secondi). I peers a cui non si invia nulla sono "choked".
  - Ogni 30 secondi viene scelto randomicamente un altro peer a cui inviare chunks.

## 5/11

### STREAMING VIDEO

Come costruire applicazioni di streaming video da una sorgente a migliaia di client all'interno dell'architettura attuale di internet?

Oggi anche un vecchio televisore con USB o HDMI può essere reso smart grazie ad appositi dispositivi (chiavette) che si connettono alla rete domestica permettendo di avere una "smart TV" connessa al proprio dispositivo.

Il problema che si è venuto a creare con il passare del tempo è l'intasamento di router e delle dorsali dovuto alla grande quantità di dati messa in movimento dagli streaming.

Per avere una buona qualità degli streaming è necessario mantenere basso e costante il jitter perché maggiore è il jitter maggiore deve essere la grandezza del buffer che riceve i dati.

Come si può trasmettere ad esempio un evento ripreso da una singola telecamera a milioni di utenti?

Non si può usare P2P sia per motivi di copyright sia perché non si può far ricondividere a tutti gli utenti ciò che stanno vedendo.

Serve rendere scalabile sia l'architettura di rete che il protocollo (ovvero l'applicazione) e considerare che i vari utenti fruiscono del servizio da dispositivi diversi (es schermo del cellulare o grande TV).

A livello di protocollo si deve far scalare la qualità dello streaming in base al dispositivo ricevente, sulla base della capacità di visualizzazione e di riproduzione del suono.

- Un **video** è un flusso di immagini trasmesse a velocità costante (es. 24 FPS), l'occhio umano vede circa 10 immagini al secondo (ad una velocità maggiore si ottiene un'interpolazione del flusso). Quindi trasmettendo a 24 fps nessuno dovrà accorgersi di scatti ecc.
- Un'**immagine** è un array di pixel, ognuno rappresentato da bits (64000 colori = 16 bit ecc.)
- Per quanto riguarda la **codifica** si fa uso della ridondanza all'interno delle immagini (spaziale) e tra le immagini (temporale) per ridurre il numero di bit utilizzati per codificarla: i bit ridondanti vengono eliminati.

Infatti analizzando due frame successivi, molti pixel sono dello stesso colore quindi è possibile indicare che un pixel è di un determinato colore e quel colore è propagato per altri N pixel.

Questo spesso vale per tutto il frame: il delta di variazione tra i due frames è molto piccolo ed è quindi possibile risparmiare molti bit di trasmissione trasmettendo solo i pixel variati.

Constant Bitrate Communication (CBR) → Il formato di codifica è costante nel tempo

Variable Bitrate (VBR) → La codifica varia e così il flusso di bits che codificano il video (se il frame trasmesso cambia drasticamente viene ricodificato tutto il frame ma poi si riprende a trasmettere solo i pixel variati)

In caso di VBR è utile avere un playout buffer in cui immagazzinare le informazioni se il flusso di informazioni è molto grande (es. nel caso di un nuovo frame) per poi trasmetterle con flusso costante

Esempi di buffer:

- MPEG 1 (CD-ROM) → 1.5 Mbps
- MPEG 2 (DVD) → 3 - 6 Mbps
- MPEG 4 - mp4 (spesso usato su internet) → <1 Mbps

Il video server contiene il video e tramite internet lo trasmette ai dispositivi che lo richiedono.

## DASH

Dynamic, Adaptive Streaming over HTTP → trasmissione tramite HTTP (simile a server web ma pubblica dei frame video)

Server:

- Divide i video in chunks e ogni chunk è memorizzato e codificato con rate differenti (diverse qualità audio e video a seconda delle caratteristiche del client);
- Si può creare sul server un **manifest file** che fornisce un URL diverso per ogni frame e per ogni formato;
- Fornendo al browser il CODEC si riescono a vedere gli streaming grazie al protocollo HTTP

Client:

- Misura i bit/s trasmessi con successo (benchmark e relative valutazioni sullo stato della rete)
- Consultando il manifest chiede un chunk per volta adattandosi al ritmo della rete
- Sceglie il massimo coding rate dato dalla banda (dynamic) e può scegliere coding diversi in momenti differenti secondo lo stato della rete (adaptive).
- Al crescere della demand della rete la qualità diminuisce.

In un sistema DASH decide **tutto il client** (quando chiedere, quale encoding usare e dove richiedere i chunk → possono esserci più server dislocati, in tal caso si chiede al migliore o al più vicino).

Non ci sono compiti aggiuntivi per il server, questo perché ogni client conosce autonomamente tutti i parametri necessari a prendere le decisioni.

## CONTENT DISTRIBUTION NETWORKS (CDN)

Ora la sfida è: come trasmettere milioni di video a miliardi di utenti con client diversi e arrivati in punti differenti del video (trasmissioni unicast)?

Non si può creare un singolo megaserver perchè oltre ad essere un single point of failure, sarebbe soggetto ad uno sforzo troppo elevato, inoltre alcuni client potrebbero essere molto distanti dal server e si dovrebbe allungare la catena dei router per la trasmissione (rischio congestione elevato). Si potrebbero **memorizzare diverse copie dei video in alcuni siti geograficamente distribuiti** (CDN).

## ENTER DEEP

Ad esempio AKAMAI ha 1700 server potenti delocalizzati che possono servire molteplici utenti vicini

## BRING HOME

Qualche decina di clusters con POP (Point of Presence) vicini alla rete degli ISP, che viene poi utilizzata dagli utenti.

## APPROCCIO OVER THE TOP (OTT)

L'approccio over the top delle CDN ha il compito di evitare la congestione della rete.

Over the top sfrutta l'intera Host Communication per condividere i contenuti in maniera scalabile.

Le sfide da affrontare sono:

- Da quale nodo CDN prendere i contenuti?
- Comportamento del client in caso di congestione?
- Quale contenuto mettere in quale nodo CDN?

Funzionamento pratico di CDN:

1. Un utente da casa sua sceglie i video dal catalogo in base alla locandina che gli piace che corrisponde ad un URL con un codice ben preciso, in questo modo, ottiene URL del video.
2. Questo URL viene risolto tramite local DNS che inoltra la richiesta al DNS authoritative del sito di streaming.
3. Il DNS autoritario restituisce un alias del server su cui si trova il file al local DNS.
4. Il local DNS a questo punto risolve URL che gli è appena stato fornito e ottiene l'IP del server che contiene il file.
5. Ogni chunk verrà richiesto da quel server seguendo il DASH protocol.
6. Se c'è uno strozzo della grandezza di banda è possibile chiedere il next chunk al DNS con un formato peggiore

## CASO NETFLIX

L'utente si connette al server e ottiene un URL di un film.

Il sito indica un cammino (molto breve) verso un server POP vicino che contiene una copia del film.

Se il cammino è congestionato si può chiedere un chunk a più server.

Il manifest file viene fornito all'utente e grazie allo streaming DASH il film comincia.

Quando su Netflix arriva un nuovo film viene messo in un server e viene poi condiviso con gli altri server durante le varie fasi enter deep.



## SOCKET PROGRAMMING

Socket: definito a livello trasporto → Grazie alla porta aperta al medesimo livello instrada verso l'IP tramite un protocollo end to end.

I raw socket sono programmati in modo non standard e messi a livello 2 MAC (giallo).

Il sistema operativo controlla i socket.

Tipi di socket: TCP e UDP → in base alla scelta effettuata per il livello trasporto.

## CREAZIONE SOCKET

Realizziamo un programma dove l'utente scrive una stringa e la invia ad un server che la converte da maiuscolo in minuscolo e risponde con la stringa convertita.

Una volta spedita richiesta si attende una risposta, quando arriva si chiude il socket.

API per spedire dati su socket = sendto (si invia messaggio encoded)

UDP: deve specificare socket di partenza e invio, non è criptato, non garantisce QoS.

**sock\_dgram = UDP**

TCP: come UDP ma criptato e ordinato

**SOCK\_STREAM = TCP**

## UDP CLIENT e SERVER

<i>Python UDPClient</i>	<i>Python UDPServer</i>
<pre>include Python's socket library  create UDP socket for server get user keyboard input Attach server name, port to message; send into socket read reply characters from socket into string print out received string and close socket</pre>	<pre>from socket import *  serverName = 'hostname' serverPort = 12000  clientSocket = socket(AF_INET,                       SOCK_DGRAM)  message = raw_input('Input lowercase sentence: ')  clientSocket.sendto(message.encode(),                     (serverName, serverPort))  modifiedMessage, serverAddress = clientSocket.recvfrom(2048)  print modifiedMessage.decode() clientSocket.close()</pre>
<pre>serverName = 'hostname' serverPort = 12000  serverSocket = socket(AF_INET, SOCK_DGRAM) serverSocket.bind(("", serverPort)) print ("The server is ready to receive")  while True:     message, clientAddress = serverSocket.recvfrom(2048)     modifiedMessage = message.decode().upper()     serverSocket.sendto(modifiedMessage.encode(),                         clientAddress)</pre>	<pre>create UDP socket bind socket to local port number 12000  loop forever Read from UDP socket into message, getting client's address (client IP and port) send upper case string back to this client</pre>

## TCP CLIENT e SERVER

<i>Python TCPClient</i>	<i>Python TCPServer</i>
<pre>from socket import *  serverName = 'servername' serverPort = 12000  clientSocket = socket(AF_INET, SOCK_STREAM) clientSocket.connect((serverName,serverPort)) sentence = raw_input('Input lowercase sentence: ')  clientSocket.send(sentence.encode()) modifiedSentence = clientSocket.recv(1024) print ('From Server:', modifiedSentence.decode()) clientSocket.close()</pre>	<pre>from socket import *  serverPort = 12000  serverSocket = socket(AF_INET,SOCK_STREAM) serverSocket.bind(("",serverPort)) serverSocket.listen(1) print 'The server is ready to receive'  connectionSocket, addr = serverSocket.accept()  sentence = connectionSocket.recv(1024).decode() capitalizedSentence = sentence.upper() connectionSocket.send(capitalizedSentence.encode()) connectionSocket.close()</pre>

Il client deve contattare server (il processo del server deve essere già in esecuzione e aver già creato il welcoming socket dove arrivano tutte le richieste dei client e vengono messe in coda).

Il server:

- Attende le richieste di connessione ed eventualmente le **accetta**
- Riceve su connection socket il messaggio (non sul welcoming socket)
- Crea socket specifico per ogni client

- Se creo più socket paralleli, il server può comunicare con più client alla volta
- Chiude la connessione

I codici di server e client sono simili, si usa SOCK\_STREAM invece di SOCK\_DGRAM.

## 9/11

### SERVIZI DI LIVELLO TRASPORTO

Mentre il livello rete permette la sola comunicazione tra host, il livello trasporto implementa la comunicazione logica tra processi su host differenti (end to end), facendo affidamento sui servizi di livello rete.

I pacchetti a livello trasporto sono detti segmenti.

Per aprire un tunnel logico tra chi trasmette e chi riceve, i dati delle applicazioni devono essere trasferiti in segmenti che il ricevente riassembra e passa al livello applicazione.

Esistono più protocolli di livello trasporto (ad esempio TCP e UDP).

Non esistono sistemi per recuperare i contenuti dei pacchetti perduti.

Il livello trasporto NON IMPLEMENTA la garanzia del ritardo e la larghezza di banda minima (internet rimane un servizio best effort anche a livello trasporto) → ISP danno solo limite massimo.

Per ottenere anche queste garanzie si usano i servizi integrati (internet 2).

### MULTIPLEXING (MUX) / DEMULTIPLEXING (DMUX)

Lato client posso avere più applicazioni di rete in esecuzione (es. Teams + Chrome + client posta ecc.).

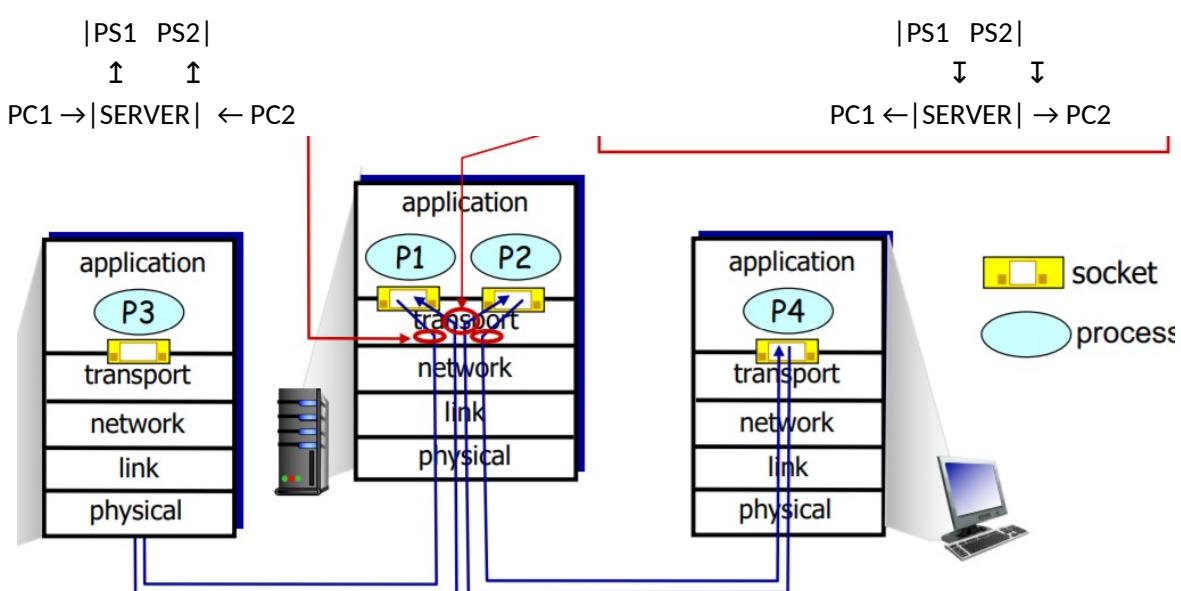
Lato server potrei avere tante richieste di vari client per avere un servizio.

Le varie richieste di un client ad un certo punto devono convogliare in un bus e partire, quindi potrebbe crearsi un effetto collo di bottiglia dovuto alle varie partenze simultanee.

Il livello trasporto si occupa di gestire questo aspetto tramite i socket.

Quando un processo client (sia PC1 ) apre un socket con il server, quest'ultimo apre a sua volta un socket per decidere a quale processo server di livello applicazione (siano PS1 e PS2) smistare le richieste provenienti dai livelli inferiori (DMUX).

Se un unico flusso di dati viene spedito da un server a più client abbiamo il MUX.



## FUNZIONAMENTO DEL DEMULTIPLEXING

- Host riceve datagram (segmenti);
- Ogni datagram contiene IP e un segmento livello trasporto di 4 byte, contenente porta di partenza, porta di destinazione, dati vari e payload (dati applicazione);
- L'host utilizza IP e porta per rendirizzare il segmento nel socket giusto.

Quindi se ricevo tanti pacchetti da internet destinati a diverse applicazioni, lo smistamento avviene tramite demultiplexing.

Se invece invio tanti dati sulla rete provenienti da tante applicazioni differenti, il processo da utilizzare è il multiplexing.

## CONNECTIONLESS DMUX

- Scelgo una porta alta ( $> 16000$ )
  - Quando l'host riceve il pacchetto UDP (DataGram\_socket → servizio datagram) controlla il numero di porta di destinazione e dirige i dati verso quella porta
    - Ciò significa che i pacchetti con la stessa porta di destinazione ma IP mittente o porta mittente diversi saranno inviati ugualmente alla stessa porta
      - Anche se l'host è lo stesso, i processi diversi vengono riconosciuti come tali dall'header, quindi non rischiano di mischiarsi i dati.
- OGNI PROCESSO RICHIENDE UNA PORTA DIVERSA

## CONNECTION ORIENTED DMUX

Sono sempre necessari IP e porte di sorgenti e destinazioni (4-tupla).

Per garantire che il servizio sia connection oriented uso TCP con 3-way-handshake.

Il server può supportare più socket TCP contemporaneamente e quindi più flussi di pacchetti di richieste e risposte verso client diversi.

Mantenere aperte varie connessioni TCP richiede la gestione dei vari socket.

I web servers hanno socket differenti per ogni client che si connette.

Il collettore dei dati in partenza e in arrivo è solo uno.



I due P3 sono la stessa applicazione (processo) in esecuzione su due client diversi.

Non è possibile sbagliare ad inviare la risposta perché anche se si usa la stessa porta resta comunque diverso l'IP.

Le nuove richieste vengono mandate al welcoming socket che accoglie le richieste, il server crea poi un client socket (su una porta alta) unico per ogni processo.

C'è la possibilità che il server esegua un solo processo che risponde a tutte le richieste (threaded server), in questo caso le accept avvengono tutte insieme.

## TRASMISSIONE DATI CONNECTIONLESS (UDP)

Caratteristiche:

- Servizio best effort
  - I segmenti potrebbero essere persi o arrivare disordinati
- Connectionless
  - Nessun handshake per l'avvio della connessione
  - Ogni segmento viene gestito indipendentemente dagli altri
- Gli utilizzi principali di UDP sono:
  - Servizi di streaming loss tolerant ma rate sensitive
  - DNS
  - Simple Network Management Protocol (SNMP)
- Esiste la versione più affidabile di UDP chiamata ReliableUDP

Spesso si utilizza UDP perchè TCP è troppo pesante e dispendioso.

Inoltre, in molti casi, anche se con UDP si perde qualche bit il destinatario capisce comunque e non se ne accorge.

## HEADER DEI SEGMENTI

L'header di UDP ha dimensione 32 bit e contiene

- Porta sorgente (16 bit);
- Porta destinazione (16 bit);
- Lunghezza totale in byte del segmento (16 bit);
- Checksum (16 bit per il controllo della correttezza, se sbagliato il pacchetto viene cestinato);
- Payload (dati applicazione).

Utilizzando UDP si ha un throughput più alto rispetto a TCP perchè gli header dei segmenti sono più piccoli, non sono richieste procedure complesse per stabilire la connessione e non avviene il controllo di congestione.

Non andrebbe usato sopra un certo volume di dati da inviare perché si rischia di perdere gran parte se non tutto il contenuto se lungo la strada c'è un router congestionato.

## CHECKSUM

Si tratta di una "somma di controllo" che serve per controllare se a causa di errori sono cambiati dei bit all'interno dei segmenti durante il trasporto.

Il sender elabora tutto il contenuto del segmento come una serie di sequenze di 16 bit, poi crea il checksum facendo la somma in complemento a uno di queste sequenze.

Il receiver esegue le operazioni al contrario e verifica se il checksum corrisponde:

- Se corrisponde significa che probabilmente non ci sono errori.
- Se **non** corrisponde significa che probabilmente ci sono errori e il segmento viene cestinato.

	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
wraparound	1	1	0	1	1	1	0	1	1	0	1	1	1	0	1	1
sum	1	0	1	1	1	0	1	1	1	0	1	1	1	0	0	0
checksum	0	1	0	0	0	1	0	0	0	1	0	0	0	1	1	1

10/11

**TRASMISSIONE DATI AFFIDABILE (RDT)**

A partire da un canale inaffidabile di livello sottostante (ad esempio di livello rete) rendiamo la comunicazione affidabile con TCP a livello trasporto:



Partiamo con un canale non affidabile:



Serve una primitiva per il trasferimento affidabile, chiamiamo da livello applicazione `rdt_send()` che a sua volta chiamerà da `udt_send()` implementato dal livello rete.

Lato ricevente incontriamo a livello rete `rdt_rcv()` che controlla la correttezza dei dati e poi chiama `deliver_data()` per il passaggio dei dati al livello applicazione.



## NOTAZIONE PER MACCHINA A STATI TRANSIZIONE

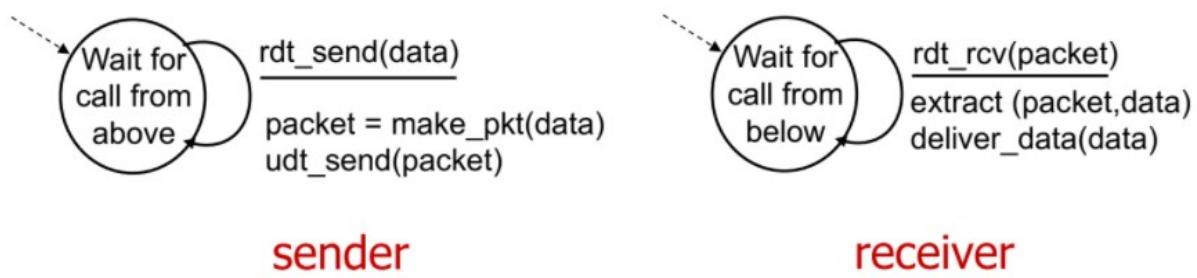
Stato con archi uscenti con sopra tipologia dell'evento che causa il cambio di stato e sotto le azioni che vengono eseguite. Quindi: **SE** sono nello stato N; **SE** avviene l'evento indicato sopra l'arco **ALLORA** eseguo quello che c'è scritto sotto l'arco per passare allo stato M.



### RDT 1.0

ASSUNZIONE (assurda): ho già un canale affidabile → no bit errati, no pacchetti persi.

	Sender	Receiver
EVENTI	Aspetta finché da sopra non viene chiamata rdt_send(data)	Aspetta finché da sotto non viene chiamata rdt_rcv(packet)
AZIONI	Crea un pacchetto con packet = mkt_pkt(data) Chiama udt_send(packet)	Estrae dal pacchetto il campo data con extract(packet,data) → controllo campo destinaz. I dati vengono passati al livello superiore con deliver_data(data)
STATO	Ritorna allo stato iniziale (freccia circolare)	Ritorna allo stato iniziale (freccia circolare)



**RDT 2.0**

Cominciamo a considerare la possibilità che ci siano bit errati (canale non totalmente affidabile)

Utilizziamo due tipi di acknowledgement: ACK (positivo) o NAK (negativo).

In caso di NAK avviene la ritrasmissione.

	Sender	Receiver
EVENTI	Aspetta finché da sopra non viene chiamata rdt_send(data)	//
AZIONI	Crea un pacchetto con sndpkt = make_pkt(data, <b>checksum</b> ) Chiama udt_send(sndpkt)	//
STATO	Attende ACK o NAK	//
ifEVENTI	Se ricevo un pacchetto && è un NAK	Ricevo un pacchetto corrotto
thenAZIONI	Chiamo udt_send(sndpkt)	Invio un NAK
STATO	Continua ad aspettare ACK (freccia circolare)	Continua ad aspettare pacchetti (freccia circolare)
elseEVENTO	Se ricevo un pacchetto && è un ACK	Ricevo un pacchetto non corrotto
thenAZIONE	NULLA: passa solo allo stato successivo	Estraggo e invio i dati; invio un ACK
STATO	Torna allo stato iniziale	Continua ad aspettare pacchetti (freccia circolare)
	<u>rdt_send(data)</u> <u>sndpkt = make_pkt(data, checksum)</u> <u>udt_send(sndpkt)</u>  <u>rdt_rcv(rcvpkt) &amp;&amp; isACK(rcvpkt)</u> Λ	<u>rdt_rcv(rcvpkt) &amp;&amp; corrupt(rcvpkt)</u> <u>udt_send(NAK)</u>  <u>rdt_rcv(rcvpkt) &amp;&amp; notcorrupt(rcvpkt)</u> <u>extract(rcvpkt,data)</u> <u>deliver_data(data)</u> <u>udt_send(ACK)</u>

**RDT 2.1**

Se sono gli ACK o i NAK ad essere corrotti potrei decidere che tutto quello che non è un ACK lo considero un NAK ma sarebbe svantaggioso.

Il sender non sa cosa sia successo lato ricevente, tuttavia non può semplicemente ritrasmettere il pacchetto perché se il precedente fosse stato ricevuto correttamente si genererebbe un duplicato. Quindi il sender ritrasmette se l'acknowledgement è corrotto, aggiungendo però il numero di sequenza di ogni pacchetto al suo interno in modo che i duplicati vengano scartati immediatamente

- Si utilizzano solo 0 e 1 perché quando un pacchetto viene ricevuto correttamente, il destinatario dovrà memorizzare il numero dell'ultimo pacchetto ricevuto per sapere quale aspettarsi come successivo.

Si adotta dunque un approccio Stop and Wait: il mittente invia un pacchetto e aspetta la conferma di avvenuta ricezione.



### RDT 2.2 (NAK free protocol)

Non ci sono più i NAK, vengono inviati solo gli ACK con il numero del pacchetto a cui si riferiscono (ovvero l'ultimo ricevuto correttamente).

Un ACK duplicato (ovvero dello stesso pacchetto) crea lo stesso risultato di un NAK ovvero la ritrasmissione del pacchetto.

Se non si utilizza lo stop and wait bisogna utilizzare un sistema di numerazione a 2 bit dato che il pacchetto potrebbe essere, ad esempio, il numero 0 attuale ma anche il numero 0 già ricevuto.



Lato ricevente, se viene ricevuto il pacchetto con numero di sequenza 1 , costruito l'ACK corrispondente e spedito, ora il prossimo pacchetto atteso avrà numero di sequenza 0.

Se arriva di nuovo il pacchetto 1 o un pacchetto corrotto, viene ritrasmesso l'ACK di 1.

Lato sender: Bisogna ricordare il numero di sequenza dell'ultimo ACK ricevuto.

Se viene ricevuto un ACK per un pacchetto sbagliato, si ritrasmette il pacchetto.

### RDT 3.0

Rimane da risolvere il problema dei pacchetti persi.

Per farlo, si introduce il concetto di timer:

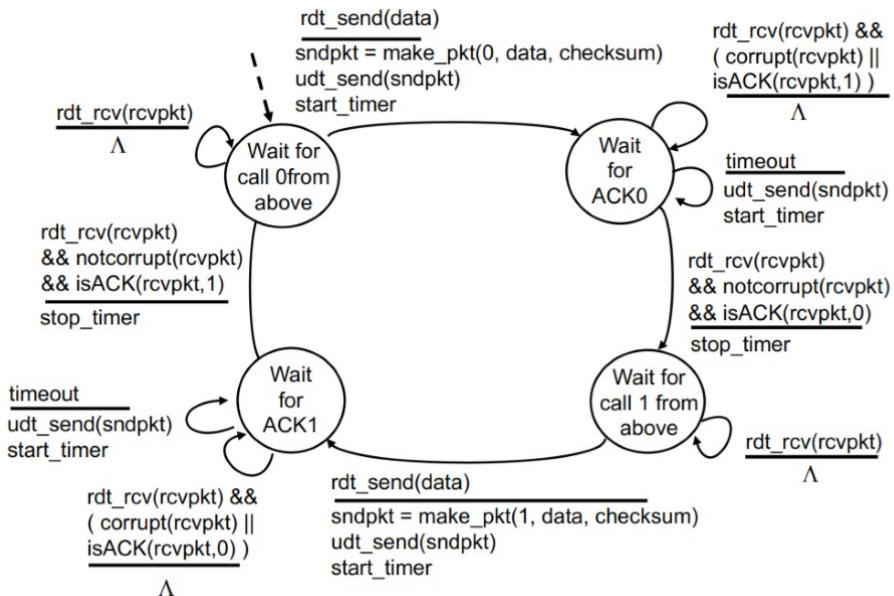
- Il sender aspetta un tempo ragionevole per la ricezione dell'ACK
  - Se non arriva il timer scade quindi il pacchetto potrebbe essere stato perso;
  - Se arriva in ritardo la gestione dei duplicati è già gestita inserendo il numero del pacchetto al suo interno.

Bisogna quindi tarare correttamente il timer per il timeout.

### MODIFICHE DA INTRODURRE

Bisogna pensare che il ricevente non abbia ricevuto ACK.

Quindi bisogna introdurre un time\_out e tararlo correttamente (non deve fare conflitto con il RTT).



## PIPELINING

L'utilizzo stop and wait permette di inviare 1 pacchetto alla volta fornendo prestazioni molto basse.

Esempio: con una velocità di collegamento di 1 gbit/s; 15ms di ritardo e 8000 bit/pacchetto

$$\text{Delay\_trans} = L/R = 8000/10^9 = 8\text{usec}$$

$$\text{Utilizzo_rete} = (L/R) / (\text{RTT} + L/R) = 0.00027$$

Grazie al pipelining è possibile inviare più pacchetti da confermare con ACK.

Il sender o il receiver ha un buffer (quindi si rischia la congestione se si inviano troppi pacchetti).

Il pipelining incrementa sensibilmente l'utilizzo della rete

Due forme di pipelining: go-back-N o selective repeat:

### GO-BACK-N

Si possono inviare max N pacchetti non confermati prima di terminare l'invio, il ricevente invia solo un ACK cumulativo (quello del pacchetto N-esimo).

Es: se invio pacchetti a gruppi di 10 invio ACK(10).

Con questa politica il sender imposta il timer per il primo pacchetto (il più anziano).

Se si perde un pacchetto, gli altri dovrebbero essere inseriti in un buffer ma ciò richiederebbe molta memoria, quindi se si ricevono i pacchetti in ordine sbagliato o alcuni non arrivano, si lascia scadere il timeout del pacchetto più "anziano" e si fa reinviare tutto al mittente.

### GO-BACK-N: SENDER

K bit rappresentano il sequence number (numero di sequenza dei byte contenuti in un segmento)

La "Finestra (scorrevole) di trasmissione" invia fino a N pacchetti non ACKd



Il segmento giallo più a sinistra è il più anziano di cui non abbiamo l'ACK e viene identificato con il puntatore `send_base`.

Il segmento blu più a sinistra è il primo che potremmo trasmettere se l'applicazione ci avesse trasmesso i dati ed è puntato da `nextseqnum`.

Giallo + Blu = N (dimensione della finestra).

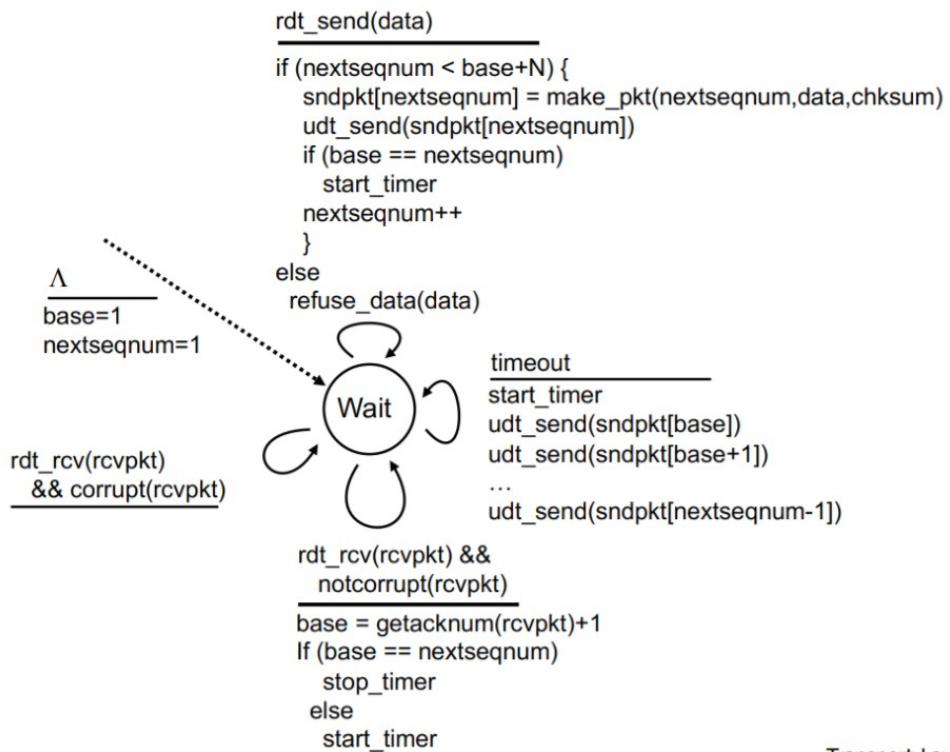
I segmenti bianchi non potremmo spedirli neanche se li avessimo disponibili perché ne abbiamo già N in sospeso.

Se scatta il timeout del segmento giallo più a sinistra il sender ritrasmette tutti i segmenti gialli.

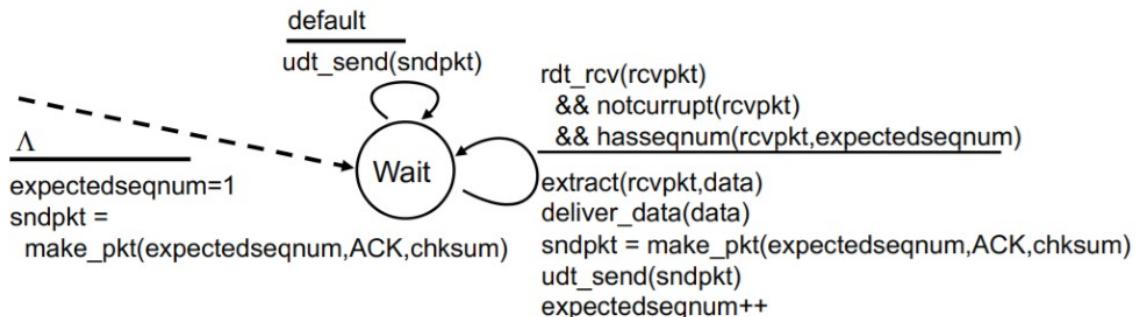
Se il receiver riceve pacchetti disordinati li droppa in attesa che il sender gli rimandi tutto (NO buffer).

Tutti i pacchetti sono confermati in maniera cumulativa (i segmenti gialli confermati diventano verdi e la finestra scorre) → l'ACK viene inviato per l'ultimo segmento ricevuto in ordine senza che ne manchino altri precedenti.

## GO-BACK-N: SENDER EXTENDED

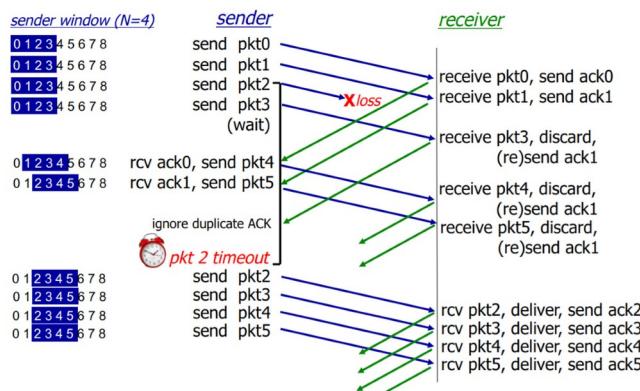


## GO-BACK-N: RECEIVER EXTENDED



Servono gli stati del three way handshake per conoscere l'expected sequence number.

Mandando lo stesso ACK più volte il mittente capisce che un pacchetto è stato perso ma la rete funziona.

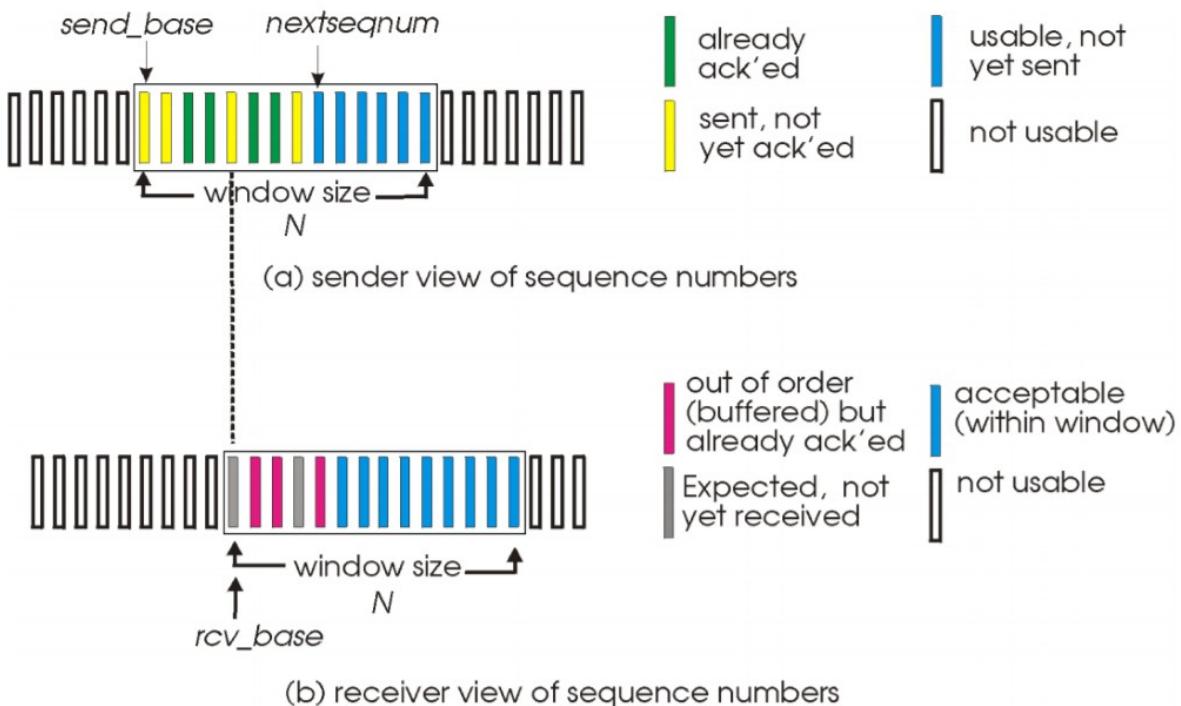


Delayed ACK: di solito 500ms → aspetto di vedere se ricevo pacchetto successivo per ACK cumulativo, altrimenti invio ACK ultimo pacchetto

## SELECTIVE REPEAT

Si possono inviare max N pacchetti non confermati prima di stoppare l'invio, il ricevente però ogni singolo ACK. Il timer è mantenuto per ogni pacchetto non confermato.

ACK inviati singolarmente, i pacchetti sono messi in un buffer e ordinati prima di essere consegnati. Il sender reinvia solo pacchetti non confermati.



Mano a mano che si ricevono gli ACK più a sinistra si può iniziare la trasmissione di quelli più a destra  
Se si perde un ACK il pacchetto viene ritrasmesso finché non è confermato altrimenti si blocca la finestra.

## PROBLEMA DI NUMERAZIONE



In questo caso vengono ricevuti due "Pacchetti zero" diversi → bisogna che la numerazione dei segmenti sia molto più grande di quella della finestra.

**12/11****TCP**

- Un mittente e un destinatario
- Flusso continuo di bit
- Usa pipelining
- Controllo congestione, flusso e dimensione della finestra
- Full duplex → flusso contemporaneo e bidirezionale sulla stessa connessione
  - MSS= max segment size
- Connection oriented
  - 3 way handshake e messaggi di controllo per inizializzare sender e receiver
- Controllo di flusso

**SEGMENTO TCP**

- 16 bit per porta sorgente e 16 per porta destinazione;
- Il sequence number è il numero ordinale dei bytes contenuti nel campo dati su 32 bit (ultimo byte inviato dal segmento);
- ACK number è il valore del prossimo byte atteso;
- Lunghezza header; non usati o per usi futuri; 6bit "di stato"; Receive window (dimensione buffer destinatario) ;
  - U = URGENTE;
  - A= è un ACK;
  - P= push, i dati sono necessari all'applicazione e vanno consegnati subito;
  - R= reset, server vuole buttare giù o non accettare connessione;
  - S= SYN, per richiesta nuova apertura connessione TCP;
  - S+A= SYN-ACK, risposta a richiesta connessione TCP;
  - F= FIN, o client o server richiede chiusura connessione da confermare;
    - Utilizzando questi 6 bit in modo improprio si possono creare danni o ritardi
- Checksum; Urg data pointer se U=1 punta ai dati urgenti
- Opzioni (lunghezza variabile)
- Dati per l'applicazione (lunghezza variabile)

## ESEMPIO

Sender invia byte da 0 a 999

Sequence number: 999 (ultimo inviato)

Receiver invia ACK con campo A = 1

Acknowledgment number: 1000 (prossimo atteso).



La gestione dei segmenti disordinati è lasciata all'implementazione (TCP standard sia con GoBackN che con selective repeat)

## RTT e SETTING del TIMEOUT

- Il timeout deve essere più lungo del RTT (variabile)
  - Se è troppo breve scade e si reinvianno inutilmente i pacchetti
  - Se è troppo lungo le reazioni ai pacchetti persi sono troppo lente

Per stimare RTT serve un sampleRTT, ovvero la misura del tempo di arrivo dei singoli pacchetti.

Tuttavia questa misura è troppo soggetta a variazioni.

Si usa quindi una media "a finestre mobili" → esitmatedRTT ottenuto tramite:

$$(1-\alpha) * \text{EstimatedRTT} + \alpha * \text{SampleRTT}$$

Dove alpha è compreso tra 0 e 1 → di solito 0.125 (così sampleRTT vale meno).

Il timeout si valuta con EstimatedRTT + un margine di sicurezza, in particolare:

$$\text{TimeOut} = \text{EstimateRTT} + 4 * \text{DevRTT} \quad (4 \text{ è arbitrario})$$

Dove DevRTT è il valore assoluto della differenza tra sampleRTT e EstimatedRTT e si calcola come:

$$\text{DevRTT} = (1-\beta) * \text{DevRTT\_precedente} + \beta * \text{abs}(\text{SampleRTT} - \text{EstimatedRTT})$$

con beta solitamente = 0.25.

**TCP SENDER**

Transport Layer 3-67

**SCENARI DI RITRASMISSIONE TCP****TABELLA SULLA GENERAZIONE DEGLI ACK**

Cosa succede al receiver	Azione del TCP receiver
Arrivo di segmenti in ordine con il numero di sequenza atteso e i segmenti precedenti ACKd	Delayed ACK: si aspettano 500ms per vedere se arrivano altri segmenti, altrimenti si invia l'ACK
Arrivo di segmenti in ordine con il numero di sequenza atteso e un segmento da confermare	Invio immediato di un ACK cumulativo per i segmenti in ordine
Arrivo di segmenti in disordine con il numero di sequenza maggiore di quello atteso	Invio immediato di un ACK duplicato con il numero di sequenza atteso
Arrivo di segmenti che riempiono totalmente o parzialmente un gap	Se il segmento arrivato si trova nella parte inferiore del gap si invia ACK

## TCP FAST RETRANSMIT

Se ad un certo punto vengono ricevuti tre ACK duplicati, significa che la rete ha perso un segmento ma dopo quel segmento perduto sono arrivati tre pacchetti in disordine, ciò significa che la rete funziona ma il pacchetto perduto va rimandato in fretta perché il buffer del receiver si sta riempiendo, tuttavia la velocità di trasmissione non deve essere rallentata.

## CONTROLLO DI FLUSSO



**RWND** = receive window, ovvero il numero di byte che il ricevente (o il mittente se invia un ACK) ha a disposizione nel suo buffer.

Esiste il 2 way handshake dove un host A chiede ad un host B di stabilire una connessione e B risponde semplicemente ok, tuttavia non funziona sempre bene perché potrebbero esserci dei ritardi nella rete o altri problemi.

## 16/11

### THREE WAY HANDSHAKE

- I socket e i processi sono stati creati e sono in uno stato di **LISTEN** in attesa della richiesta di apertura connessione.
- Quando il client vuole aprire connessione sceglie un numero di sequenza iniziale che rappresenta il numero da cui si iniziano a contare i pacchetti; si tratta di un numero casuale per evitare di confondere due connessioni diverse.
- Successivamente viene inviato un pacchetto con SYN=1 e il numero di sequenza X precedentemente scelto.
- Nel momento in cui il WELCOMING socket (in listen) riceve la richiesta viene creato il relativo **CLIENT** socket
- Il server che è in Listen riceve la richiesta e sceglie anche lui un numero di sequenza iniziale per i suoi pacchetti (il flusso di dati può essere bidirezionale) e invia un pacchetto con SYN e ACK a 1 (pacch. SYNACK) e il suo numero di sequenza Y.
  - In questo caso il numero di ACK sarà X+1

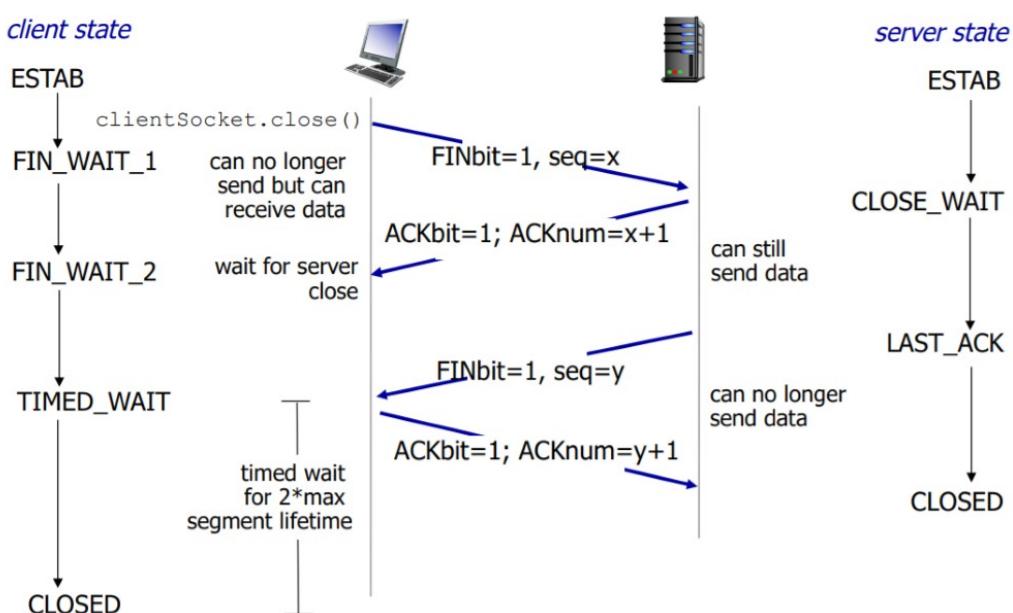
- Ora il server entra in stato **SYN RCVD**.
- Il ricevente invia il bit SYN a 1 , ack=1 e ack num = X+1
- Il client riceve il SYNACK (che vuol dire che il server è attivo) e invia un ACK per il SYNACK che potrebbe anche contenere alcuni dati – il client entra in stato ESTAB.
- Il server riceve quest'ultimo ACK (numerato Y+1) e capisce che anche il client è attivo, la connessione è stabilita (**ESTAB**)



Per una connessione P2P bisogna implementare tutti e due i lati, client e server.

## TERMINARE UNA CONNESSIONE

Il lato che vuole terminare una connessione invia un pacchetto con FIN = 1, se (e quando) l'altro lato risponde con un segmento con FIN e ACK a 1 (FINACK) si può chiudere la connessione.



## CONTROLLO DELLA CONGESTIONE

Se troppe sorgenti inviano dati troppo in fretta la rete non riesce a propagarli tutti, spesso a causa della presenza di un router più lento degli altri.

La congestione causa perdite di pacchetti dovute a buffer overflow sui router.

Indichiamo con:

- $\lambda_{in}$  i pacchetti inviati
- $\lambda_{out}$  il throughput del destinatario
- $\lambda'_{in}$  i dati originali + eventuali ritrasmissioni

Il massimo throughput per ogni connessione è  $R/N$  dove  $R$  è la capacità del mezzo trasmissivo e  $N$  il numero di dispositivi che sfruttano il link.

Su alcune reti vige la **fairness** → 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.

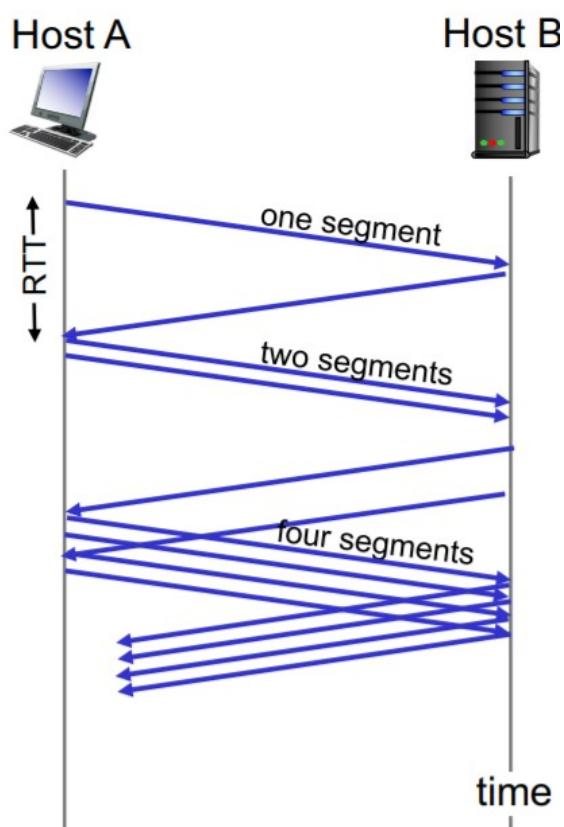
## SLIDING WINDOW DI TCP

Il transmission rate iniziale è basso (contention window piccola), se si ricevono gli ACK di questi primi pacchetti si incrementa la dimensione della finestra.

Appena non arriva un ACK si riduce drasticamente la dimensione della finestra → crollo del throughput quindi il router più lento svuota il buffer → (additive increase, multiplicative decrease).

In questo modo il rate sarà circa  $cwnd/RTT$  byte/sec → si aspetta un ACK ogni RTT

## SLOW START



## CAPIRE SE C'È DAVVERO CONGESTIONE

Se si riceve per TRE volte un ACK duplicato un pacchetto è considerato perso.

TCP RENO: In questo caso la cwnd viene divisa a metà per permettere al router che ha quel pacchetto nel buffer di elaborarlo.

TCP TAHOE: riporta sempre la cwnd a 1

Viene inoltre fissata una variabile ssthresh che indica più o meno fin dove di solito va tutto bene, superata la quale non si raddoppia più la finestra ma si aumenta di 1 (congestion avoidance)



L'andamento del throughput di TCP è a dente di sega.

- Sia  $W$  la dimensione della finestra
- avg. window size equivale a  $\frac{3}{4} W$  ;
- avg. throughput equivale a  $\frac{3}{4} W$  per RTT;
- avg TCP throughput =  $\frac{3}{4} W * \text{RTT bytes/sec}$ ;

## ESEMPIO

(1.22 è una costante)

- example: 1500 byte segments, 100ms RTT, want 10 Gbps throughput
- requires  $W = 83,333$  in-flight segments
- throughput in terms of segment loss probability,  $L$  [Mathis 1997]:

$$\text{TCP throughput} = \frac{1.22 \cdot \text{MSS}}{\text{RTT} \sqrt{L}}$$

→ to achieve 10 Gbps throughput, need a loss rate of  $L = 2 \cdot 10^{-10}$  – a very small loss rate!

## EXPLICIT CONGESTION NOTIFICATION (ECN)

Nell'header dei pacchetti IP ci sono dei campi ToS (type of service) che vengono compilati dai router per indicare una congestione.

## LIVELLO RETE: DATA PLANE e CONTROL PLANE

Forwarding → instradare i pacchetti nella direzione giusta grazie alle tabelle di instradamento

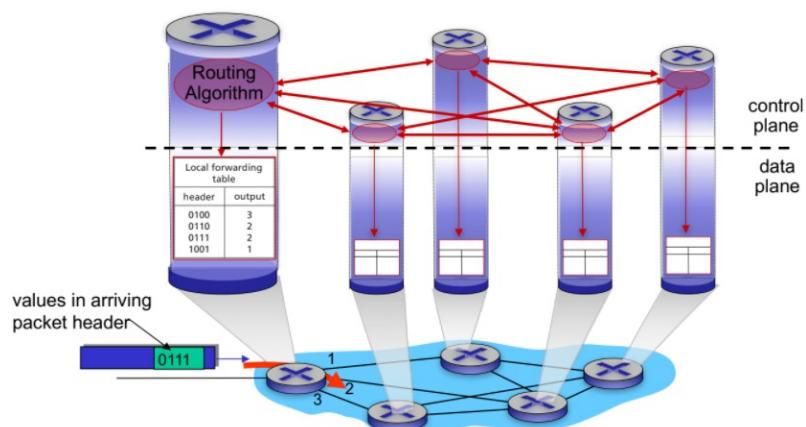
Routing → aggiornamento delle tabelle di instradamento

### DATA PLANE:

- Funzione implementata nei singoli router
- Determina come i pacchetti in input sui router devono essere inviati alla porta di output
  - Mentre il pacchetto viene ricevuto il router decide dove mandarlo, successivamente viene copiato il buffer di ingresso in quello di uscita non appena possibile.
- Implementa il forwarding dei pacchetti
  - Nel data plane di ogni router c'è la forwarding table che indica le corrispondenze tra header ricevuti e porte di uscita

### CONTROL PLANE:

- Si tratta di una logica diffusa su tutta la rete
- Determina come i pacchetti devono essere instradati tra i vari router intermedi
- Implementa il routing dei pacchetti
- Due approcci distinti:
  - Algoritmi di routing tradizionali implementati nei router
    - Singoli algoritmi di routing interagiscono in ogni router



- Software defined networks (SDN) implementate in server remoti
  - Un controllore esterno interagisce con i control agents (CA) locali



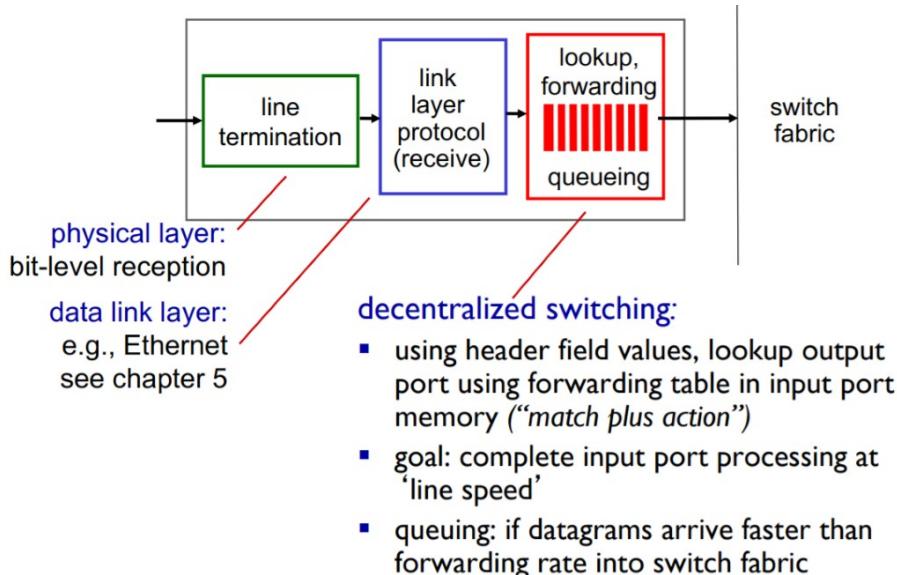
19/11

## COME FUNZIONA UN ROUTER AL SUO INTERNO?



- A sinistra ci sono le code di input (verde, blu, rosso)
- Al centro c'è una "centrale di commutazione" ad alta velocità controllata da un processore logico in cui viene implementato l'algoritmo di routing.
- A destra ci sono le code di uscita.
  - Non tutti i router funzionano in questo modo, alcune macchine che non sono state progettate specificamente per essere un router sono meno efficienti.

## INPUT PORT



- Verde: line termination → connettore RJ45 o ricevitore wifi ecc
- Blu: Link layer (livello 2) → protocolli di ricezione per il protocollo livello MAC utilizzato
- Rosso: coda di input di livello rete
  - Viene fatto il lookup del pacchetto e si controlla nella tabella di forwarding per decidere in che direzione l'high-speed switching fabric deve inoltrarlo.

**Match and action:** trova la corrispondenza ed esegui l'azione associata

Es. Pacchetto per 130.136:

- Match → cerco in tabella 130.136
- Action → guardo da quale porta deve uscire il pacchetto per 130.136

<i>forwarding table</i>	
Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

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.

## LONGEST PREFIX MATCHING

*longest prefix matching*

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

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

Il funzionamento base di una tabella di instradamento prevede di indicare un blocco di IP e la porta da cui far uscire i pacchetti destinati a un IP di quel blocco e una porta dedicata a tutti gli IP non contenuti nella tabella.

Per fare ciò è necessario eseguire operazioni matematiche lunghe e potrebbe succedere che due blocchi contigui non contengano IP contigui, quindi in presenza di un buco è necessario analizzare tutti i blocchi per poi terminare con l'inoltro sulla porta dedicata agli IP sconosciuti.

Si usa quindi il **Longest Prefix Matching** che prevede che nella tabella ci sia solo un indirizzo per porta che termina con degli asterischi. Si applica la regola di matching che prevede di controllare quale sia l'indirizzo esplicito nella tabella più simile all'indirizzo di destinazione.

Si prosegue cercando se esistono righe con un match più lungo (prefisso di bit espliciti corrispondente a indirizzo destinatario più lungo).

Si analizza così velocemente (and logico) ogni riga che possa corrispondere una sola volta, se non si trovano match si manda sulla porta "altrimenti", se no si sceglie il match più lungo.

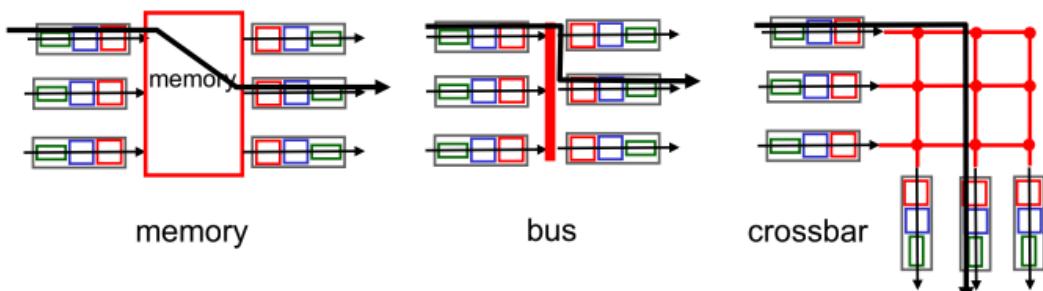
Gli indirizzi sulla tabella sono in ordine e ogni riga è memorizzata in un registro quindi si possono anche eseguire controlli in parallelo (memoria associativa).

Di solito nei router locali si fa in modo che la prima riga corrisponda alla link interface del router verso il basso.

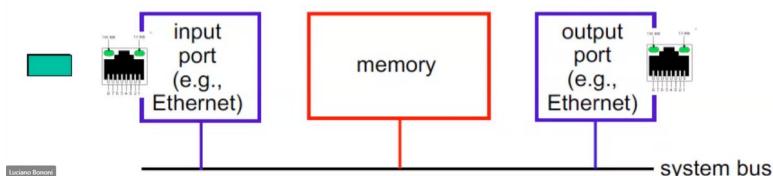
## SWITCHING FABRICS

- Questa è la parte del router che permette di trasferire i pacchetti dal buffer di input al buffer di output corretto.
  - Definiamo lo **switching rate** come la velocità con cui i pacchetti possono essere trasferiti dal buffer di input a quello di output.
    - Dipende dal tipo di struttura, di tecnologia e materiali, Idealmente se abbiamo  $N$  linee di input dovremmo avere uno switching rate pari a  $N$  per il rate delle linee in ingresso

Esistono tre tipi di switching fabric che differiscono tra loro per velocità di ingresso e uscita



## SWITCHING VIA MEMORIA



Nei computer convenzionali lo switching è gestito direttamente dalla CPU:

- I pacchetti sono copiati nella memoria di sistema
- Ci sono due accessi in memoria: uno per scrivere in input e uno per leggere in output.
- Il limite del sistema è rappresentato dalla lentezza di scrittura: il bus condiviso è impegnato sia dai pacchetti in entrata che da quelli in uscita.

## SWITCHING VIA BUS

- I pacchetti passano dalla porta di ingresso a quella di uscita tramite un bus.
- Il limite di questo sistema è la velocità del bus (bus contention) che può rappresentare un collo di bottiglia in caso di porte di input molto veloci.

## SWITCHING VIA CROSSBAR

Questo sistema riesce a superare le limitazioni imposte dal bus.

Ogni incrocio è un programmabile array che consente ai bit di passare o meno.

Le crossbar erano state inizialmente pensate per connettere architetture multiprocessore, ma permettono di attuare operazioni più complesse come la fragmentation switch, dove i pacchetti vengono frammentati in celle e spediti.

Risolve i problemi che avevamo con il bus condiviso, infatti è possibile il parallelismo finché i segmenti non si incrociano (più percorsi possibili contemporaneamente).

## INPUT PORT QUEUING

Head of line (HOL): il primo pacchetto accodato impedisce agli altri di avanzare: tutti i pacchetti che potrebbero passare vengono bloccati da uno fermo.



## PORTE DI OUTPUT

Se il fabric fa arrivare i pacchetti troppo velocemente, questi vengono accodati.

Si sfrutta lo scheduling per decidere come far passare i pacchetti dalla coda di output alla line termination (es. FIFO).

Qualunque algoritmo di scheduling usiamo si creano o dei ritardi o delle situazioni vantaggiose solo per alcuni pacchetti.



Per decidere la capacità del buffer di output si usa la **regola del pollice** che dice:

- RFC 3439 rule of thumb: average buffering equal to “typical” RTT (say 250 msec) times link capacity C
  - e.g., C = 10 Gbps link: 2.5 Gbit buffer
- recent recommendation: with  $N$  flows, buffering equal to

$$\frac{RTT \cdot C}{\sqrt{N}}$$

## MECCANISMI DI SCHEDULING

Per scegliere quale pacchetto e quando spedirlo si possono sfruttare diversi meccanismi:

- Politica FIFO (inviati in ordine di arrivo)

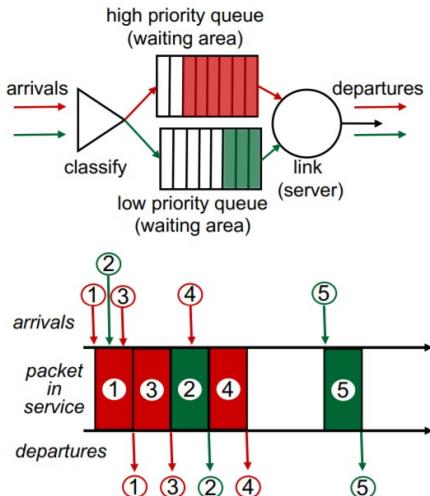
Discard policies: quando arriva un pacchetto ma la coda è piena, quale bisogna droppare?

- Tail drop: l'ultimo
- Priority: quello a priorità più bassa
- Random: Un pacchetto a caso: in caso di congestione abbiamo tutti la stessa possibilità di essere perduti

Il random è applicato quando tutti hanno la stessa priorità: infatti quello che spedisce tanti pacchetti ha più probabilità di venire dropato. Lo stesso vale per TCP prioritario perché avrà più pacchetti e statisticamente avrà più probabilità di perderne uno.

## PRIORITY

Sono presenti due buffer a seconda della priorità, scritta nel pacchetto

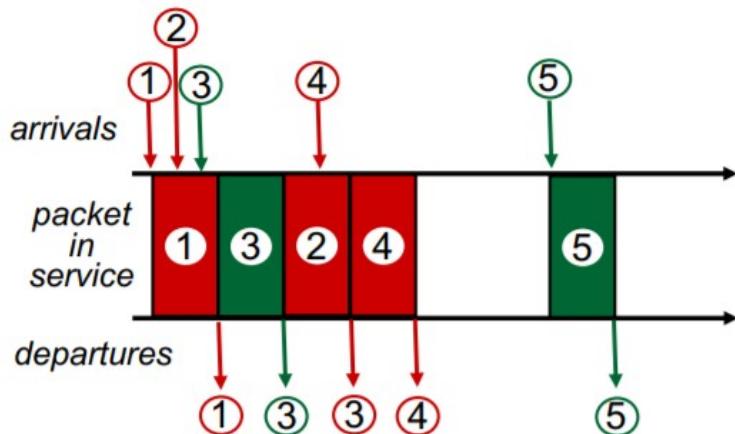


C'è tuttavia il rischio che i pacchetti a bassa priorità non vengano mai inviati (accodati per tempo indefinito), potrebbe quindi scadere il timer per gli ACK e avremmo dei duplicati.

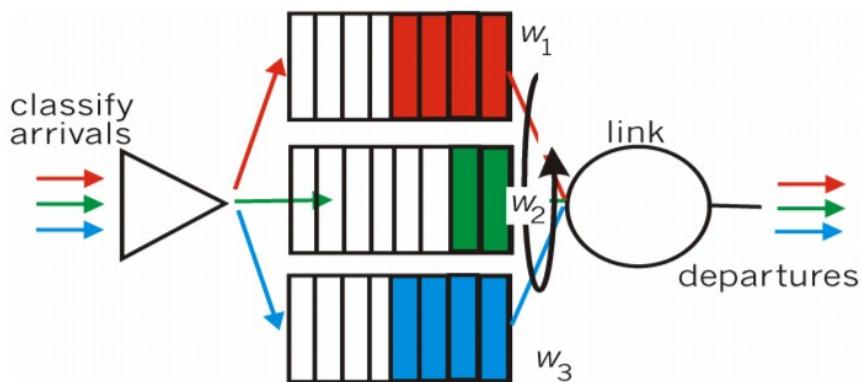
## ROUND ROBIN (RR)

Un pacchetto dal buffer priorità e uno dall'altro.

Vantaggio: non vengono bloccati i pacchetti a bassa priorità

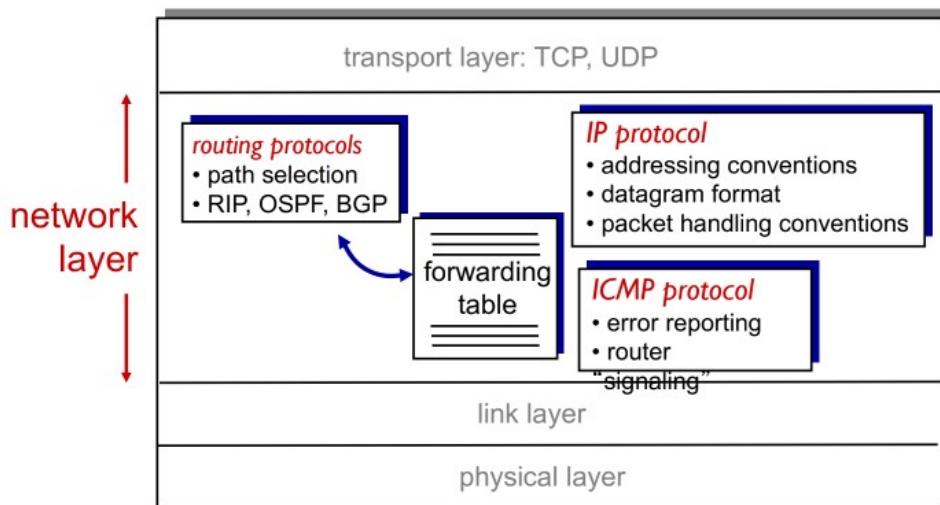


### WEIGHTED FAIR QUEUE (WFQ)



Funziona in modo analogo al RR ma se un buffer ha maggiore priorità, questa viene rispettata  
Es. Se rosso ha priorità doppia rispetto agli altri quando è il suo turno invia il doppio

### LIVELLO RETE



### PACCHETTO IPV4

Grandezza massima:  $2^{16}$  byte



23/11

**PROTOCOLLO IPV4 – FRAMMENTAZIONE E REASSEMBLY****Frammentazione:** spezzare in più pacchetti un pacchetto grande.

I frammenti avanzeranno autonomamente e saranno ricomposti dal ricevente finale

**Reasembly:** i pacchetti precedentemente frammentati vengono riuniti dal ricevente

### **example:**

- ❖ 4000 byte datagram
  - ❖ MTU = 1500 bytes

*one large datagram becomes several smaller datagrams*

1480 bytes in  
data field

	length =1500	ID =x	fragflag =1	offset =0
--	-----------------	----------	----------------	--------------

offset =  
1480/8

	length	ID	fragflag	offset
	=1500	=x	=1	=185

length =1040	ID =x	fragflag =0	offset =370
-----------------	----------	----------------	----------------

Indichiamo con **MTU** la max transfer unit (che varia in base al livello MAC considerato).

La somma delle dimensioni dei frammenti è maggiore della dimensione del pacchetto originale perché ogni frammento ha un suo header.

La dimensione minima di un frammento è 8 byte.

## **INDIRIZZAMENTO IP**

- IP: identificatore di 32 bit per host o router;
  - Interfaccia: connessione tra host o router e mezzo fisico
  - Subnet (sottorete): insieme di dispositivi collegati tra di loro senza necessità di passare per un router
    - Tramite IP si capisce a quale HOST di quale sottorete si è interessati
    - La maschera di rete è assegnata nel formato CIDR (Classless InterDomain Routing) che prevede che l'indirizzo sia espresso come a.b.c.d/x, dove x è il numero di bit dedicati alla sottorete.



200.23.16.0/23

## DHCP

## Metodi del DHCP:

- DHCP discover → host cerca in broadcast MAC un server DHCP (opzionale)
  - DHCP offer → risposta del server (offer) (opzionale)
  - DHCP request → host chiede un IP
  - DHCP ack → server fornisce IP

Tutti questi messaggi vengono inviati in broadcast con il numero di porta del DHCP server (67 - 68) tramite UDP



## NETWORK ADDRESS TRANSLATION (NAT)

Si tratta di un processo implementato dai router utile a sopperire alla mancanza di indirizzi IP.

- Il router ha "da una parte" un suo IP connesso a Internet e dall'altra un indirizzo diverso per la LAN di domestica:
- Tra loro gli host interni alla LAN possono comunicare
- Per comunicare su internet gli host interni alla LAN mandano il pacchetto al router che lo manda verso internet
- Tutti gli host che vogliono comunicare dall'esterno mandano pacchetti all'IP del router
  - Per capire a quale host interno sono destinati i pacchetti il NAT capisce chi aveva fatto la richiesta grazie a dei finti socket
  - Esiste una NAT translation table



Interfaccia WAN: interfaccia verso l'esterno

Tutti gli host gestiti tramite DHCP non sanno di essere in un NAT.

NAT translation table: quello usato lato rete internet viene tradotto in quello usato in LAN

- Praticità della soluzione: non aggiunge overhead e gli host non si accorgono di nulla, infatti avviene una conversione 1 a 1 tra Local IP e IP in rete.
- Per identificare le porte all'interno dei NAT si utilizzano 16 bit quindi si possono avere più di 60.000 host connessi attraverso un solo indirizzo IP.

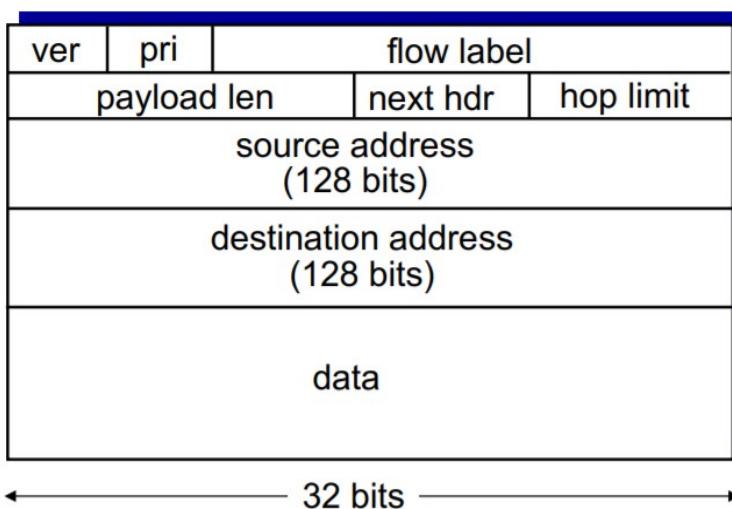
- C'è una controversia legata al NAT che realizza, grazie ai numeri di porta, un legame con il livello trasporto, violando così lo standard per cui ogni livello deve essere separato dagli altri.

**26/11**

### IPv6

L'header resta sempre di 40 byte ma è cambiato per facilitare la QoS, rendendo più rapidi elaborazione e forwarding dei pacchetti.

Inoltre non è permessa la frammentazione



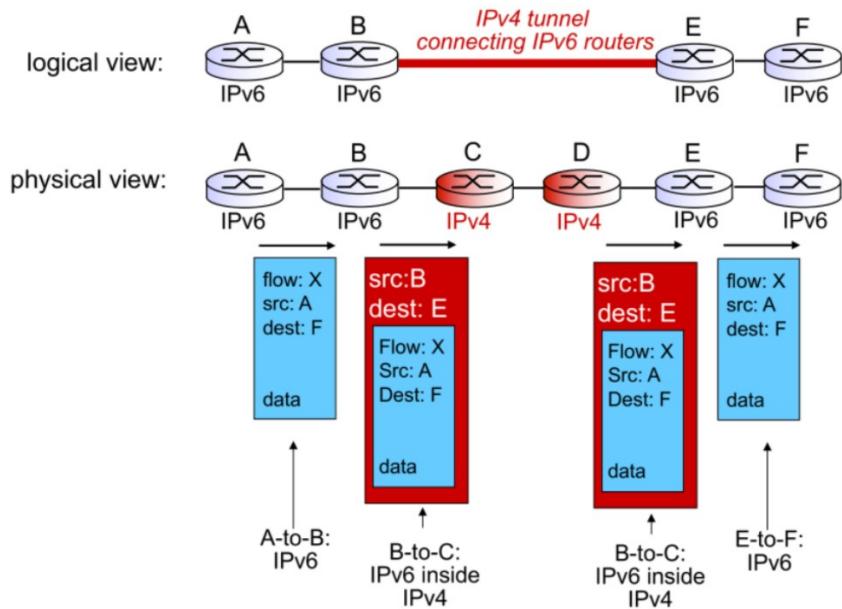
- Versione; priorità; flow label (identifica pacchetti nello stesso flusso)
- Lunghezza del payload; next header; hop limit
  - Next hdr → punto in cui nel campo dati si trova l'header del protocollo di livello superiore (TCP o UDP)
  - Hop limit → equivalente al TTL
  - È possibile aggiungere campi addizionali senza dover modificare la dimensione dell'header (compatto e scalabile)
- Mittente (128 bit)
- Destinatario (128 bit)
- Dati
  - Il checksum è stato rimosso e le opzioni si inseriscono in next hdr
  - In ICMPv6 sono stati introdotti nuovi messaggi tipo packet too big

### TUNNELLING

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.

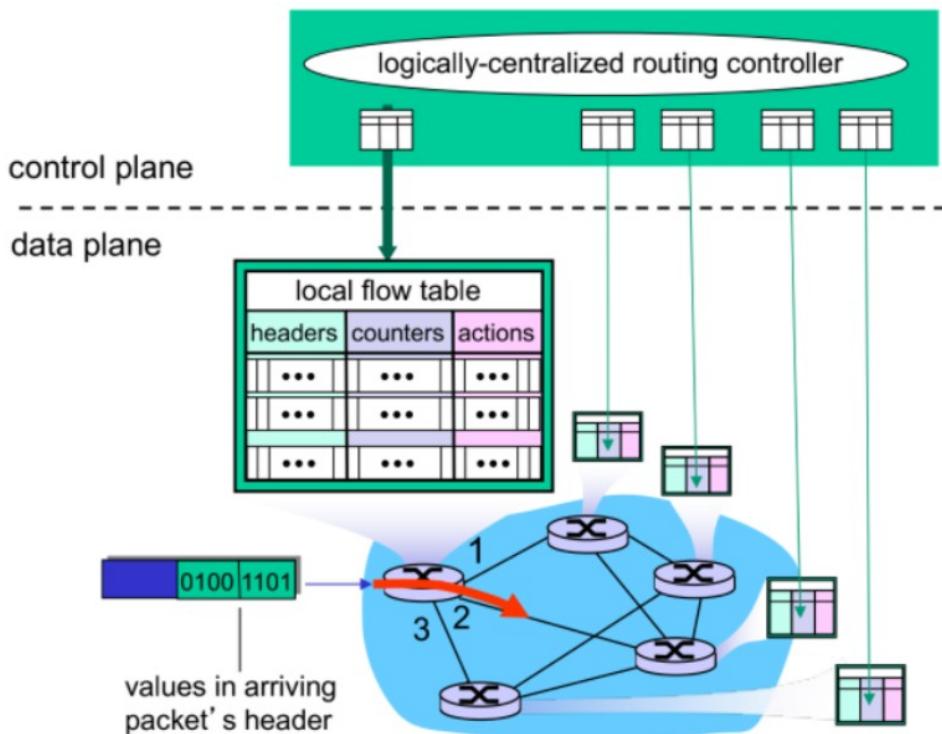


### APPROCCIO SDN (Software Defined Network)

Ogni router contiene una tabella di flusso che è controllata e distribuita da un software.

**Flusso:** pacchetti provenienti da un IP e destinati nello stesso posto.

Queste tabelle diventano l'equivalente delle tabelle di routing, permettono comunque di usare switch, firewall ecc. e contengono headers, counters e actions



### OPENFLOW

I flussi sono definiti dai campi dell'header.

Regole di gestione:

- Pattern: riconosce i flussi tramite gli header e in base al matching esegue una action:
  - DROP
  - FORWARD

- MODIFY (router cambia dati al volo)
  - SEND (invio al controller)
  - I campi priority distinguono i pacchetti urgenti da quelli non urgenti
  - Counters (numero bytes e pacchetti)



## ESEMPI

## Destination-based forwarding:

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

*IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6*

## Firewall:

*do not forward (block) all datagrams destined to TCP port 22*

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
-------------	---------	---------	----------	---------	--------	--------	---------	-----------	-----------	---------

\* \* \* \* \* 128.119.1.1 \* \* \* \* drop

*do not forward (block) all datagrams sent by host 128.119.1.1*

## Destination-based layer 2 (switch) forwarding:

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

\* 22:A7:23:11:E1:02 \* \* \* \* \* \* \* \* \* port3

*layer 2 frames from MAC address 22:A7:23:11:E1:02  
should be forwarded to output port 6*

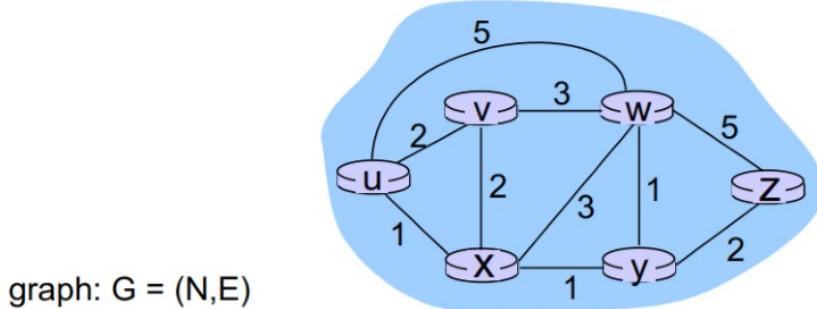
### MATCH + ACTION

DISPOSITIVO	MATCH	ACTION
ROUTER	Prefisso IP più lungo	Inoltra su un link
SWITCH	Indirizzo MAC di destinazione	Inoltra o manda in broadcast
FIREWALL	IP e porte TCP o UDP	Consenti o blocca
NAT	IP e porta	Riscrive indirizzo e porta

### CONTROL PLANE

- 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

Visualizzazione della rete come grafo:



$$N = \text{set of routers} = \{ u, v, w, x, y, z \}$$

$$E = \text{set of links} = \{ (u,v), (u,x), (v,w), (v,x), (v,y), (w,z), (x,y), (x,z), (y,z) \}$$

*aside:* graph abstraction is useful in other network contexts, e.g., P2P, where  $N$  is set of peers and  $E$  is set of TCP connections

## CLASSIFICAZIONE DEGLI ALGORITMI ROUTING

- Globali
  - Chi decide gli algoritmi conosce tutta la rete → algoritmi **link state**
- Decentralizzati
  - Chi decide gli algoritmi conosce solo una parte della rete → **distance vector**
- Approccio statico
  - I percorsi cambiano lentamente nel tempo
- Approccio dinamico
  - I percorsi cambiano velocemente e sono aggiornati periodicamente

È ora necessario costruire un algoritmo che da un grafo come il precedente trovi i percorsi migliori e li metta nelle tabelle di instradamento:

## ALGORITMO DI DIJKSTRA (link state)

*notation:*

- $c(x,y)$ : link cost from node  $x$  to  $y$ ;  $= \infty$  if not direct neighbors
- $D(v)$ : current value of cost of path from source to dest.  $v$
- $p(v)$ : predecessor node along path from source to  $v$
- $N'$ : set of nodes whose least cost path definitively known

```

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  $D(v) = \infty$ 
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  $N'$ 

```

**ALGORITMO DI BELLMAN- FORD (distance vector)***Bellman-Ford equation (dynamic programming)*

let

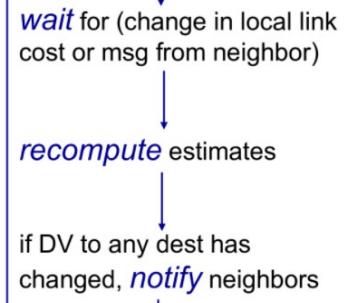
 $d_x(y) := \text{cost of least-cost path from } x \text{ to } y$ 

then

$$d_x(y) = \min_v \{ c(x,v) + d_v(y) \}$$

↓                    ↓                    ↓  
 cost from neighbor v to destination y  
 cost to neighbor v

min taken over all neighbors v of x

*each node:*

- Approccio di tipo gossip: appena un nodo scopre qualcosa lo dice agli altri.  
In questo caso l'evento da diffondere sarebbe un cambiamento sul grafo.

**DIFFERENZE TRA LINK STATE E DISTANCE VECTOR***message complexity*

- **LS:** with n nodes, E links,  $O(nE)$  msgs sent
- **DV:** exchange between neighbors only
  - convergence time varies

*speed of convergence*

- **LS:**  $O(n^2)$  algorithm requires  $O(nE)$  msgs
  - may have oscillations
- **DV:** convergence time varies
  - may be routing loops
  - count-to-infinity problem

*robustness: what happens if router malfunctions?***LS:**

- node can advertise incorrect *link* cost
- each node computes only its own table

**DV:**

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagate thru network

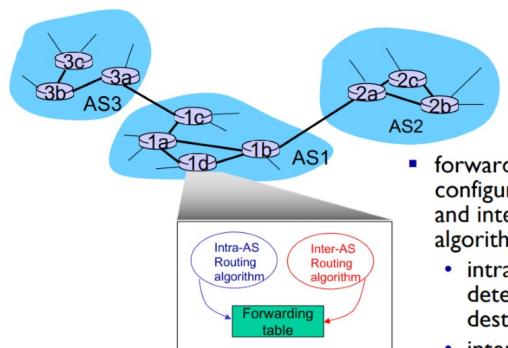
**ROUTING SCALABILE**

Per rendere routing scalabile dobbiamo considerare una rete gerarchica, dove sono presenti vari domini (autonomous system - AS), dove i router utilizzano le stesse regole.

**Intra AS routing:** tutti i router dell'AS adottano lo stesso protocollo intra-domain.

I router in AS differenti possono eseguire protocolli intra-domain diversi

**Inter AS routing:** Routing tra AS, gestito da algoritmi super partes



- forwarding table configured by both intra- and inter-AS routing algorithm
  - intra-AS routing determine entries for destinations within AS
  - inter-AS & intra-AS determine entries for external destinations

## Interior Gateway Protocols (IGP)

È un altro modo di chiamare gli Algoritmi intra-AS

I più comuni sono:

- RIP: Routing Information Protocol
- OSPF: Open Shortest Path First (link state con Dijkstra)
  - Contiene elementi di sicurezza come l'autenticazione
  - Sono consentiti più cammini con costo uguale
  - Supporta uni- e multi- cast
  - Protocollo gerarchizzato con due livelli di gerarchia: local area, backbone
    - Router ai bordi (boundary) possono essere connessi con interfacce esterne
    - Router backbone eseguono algoritmi di routing limitatamente alla backbone
- IGRP: Interior Gateway Routing Protocol (proprietario CISCO fino al 2016)

## BGP (Border Gateway Protocol)

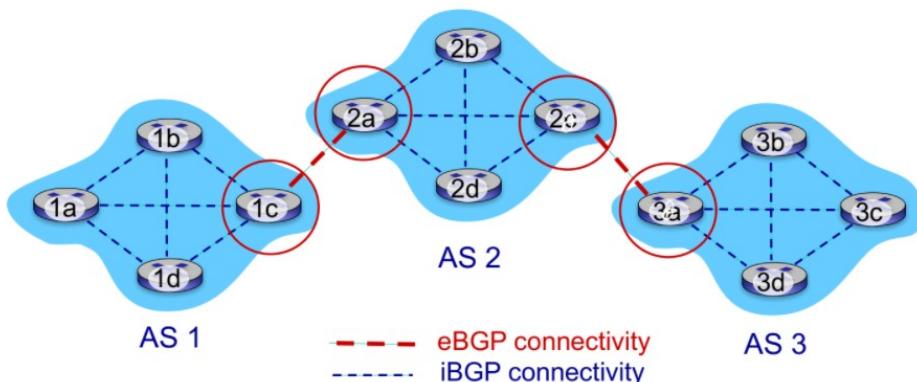
È stato definito come "la colla che tiene insieme internet".

**eBGP:** (external BGP) ottiene le informazioni di raggiungibilità dagli AS vicini

- I percorsi migliori sono determinati dalle informazioni sulla raggiungibilità e sulle policies economiche e politiche vigenti tra i vari AS.

**iBGP:** (internal BGP) propaga le informazioni di raggiungibilità a tutti i router interni dell'AS

Esiste un meccanismo per permettere ad ogni rete di indicare la sua presenza a tutta la rete.



## BASI DI BGP

All'interno di una sessione BGP due router (peers) si scambiano messaggi BGP tramite una connessione TCP semi permanente.

Grazie a eBGP un border router riceve l'advertising di un altro router e tramite iBGP lo fa sapere a tutti gli altri router dell'AS.

Ora tutti gli altri border router di quel AS tramite eBGP diffondono l'informazione verso un altro AS.

- Il sistema basato su SDN è migliore ma è un single point of failure e rischia la congestione.
- Per decidere dove far passare il traffico si sfrutta OpenFlow con specifiche regole.

## MESSAGGI CONTROLLO BGP

OPEN: apre una connessione TCP con un peer BGP e procede con l'autenticazione

UPDATE: notifica nuovi advertisements o ritira i vecchi

KEEPALIVE: mantiene aperta la connessione, anche senza update o conferma (ACK) un Open

Notification: notifica gli errori nei messaggi precedenti o viene usato per chiudere una connessione

## 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:** non è possibile modificare la rete a proprio piacimento, quindi è necessario utilizzare sistemi come quelli OpenFlow permessi da SDN per dare comandi specifici ai singoli router.

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 e altri fattori.

## 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 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.

**30/11**

## SICUREZZA

Gli aspetti parzialmente garantiti dai firewall, dagli IDS (Intrusion Detection Systems) e dai livello applicazione, trasporto, rete ecc... sono:

- Crittografia
- Autenticazione (persona e macchina)
- Integrità del messaggio (messaggio non modificato)

Garantire la sicurezza delle comunicazioni di rete significa garantire:

- Confidenzialità
  - Solo sender e receiver comprendono il messaggio
  - Sender cifra (encrypt) il messaggio e il ricevente deve decriptarlo (decrypt)
- Autenticazione
  - Il sender deve essere sicuro che il receiver sia proprio chi deve essere e viceversa
  - Mutua autenticazione → sender e receiver si autenticano l'un l'altro
- Integrità del messaggio
  - Bisogna accertarsi che il messaggio non sia modificato durante il transito
  - Non si può verificare con certezza → si fa solo una verifica finale per vedere se anche un solo bit è stato modificato → se è stato modificato si scarta
- Garantire l'accesso ai dati
- Garantire la disponibilità del servizio di comunicazione

ALICE (A – sender sicuro), BOB (B – receiver sicuro), TRUDY (T - intruso)

A e B potrebbero essere:

- Persone reali
- Browser web o servizi per transazioni elettroniche
- Server o client per transazioni bancarie
- Server DNS
- Router che si stanno scambiando gli update delle tabelle di routing
- ecc ecc...

I tipi di attacco che T potrebbe effettuare sono:

- Eavesdrop (intercettare i messaggi)
- Inserire nuovi messaggi (falsi) in rete
- Furto di identità (spoofing di IP ecc)
- Hijacking (dirottamento dei pacchetti verso sé stesso)
- Denial of service (DOS – impedisce di usare un servizio)

## CRITTOGRAFIA

Terminologia:

- Plaintext – messaggio in chiaro (non cifrato)
- Blocco di cifratura → cifra il plaintext tramite algoritmo di cifratura
- Ciphertext - testo cifrato
- Blocco di DEcifratura → DEcifra il plaintext tramite apposito algoritmo

$m$  = plaintext

$K_A(m)$  = testo cifrato con chiave  $K_A$

$m = K_B(K_A(m)) \rightarrow$  messaggio decifrato

La forza degli algoritmi di (de)cifratura non è la loro segretezza, essi infatti sono di dominio pubblico, ma le chiavi di (de)cifratura **da 128 o più bit** che sono praticamente impossibili da ricavare in tempi utili tramite algoritmi brute force.

Per trovare la chiave, T potrebbe:

- Analizzare  $K_A(m)$
- Provare tutte le chiavi (improbabile)
- Fare un'analisi statistica sulle chiavi per provare ad indovinarla (attacco dictionary)
- Se T conosce  $m$  e ottiene  $K_A$  potrebbe ricavare la chiave
- T cerca  $K_A(m)$  per un testo che conosce

### CRITTOGRAFIA A CHIAVE SIMMETRICA

B e A hanno la stessa chiave  $K_S$  simmetrica → per decifrare si usa  $K_S(K_S(m))$

Es. Permutazione delle lettere dell'alfabeto (**Cifrario di Cesare**)

<b>plaintext:</b>	<b>abcdefghijklmnopqrstuvwxyz</b>
	↓                      ↓
<b>ciphertext:</b>	<b>mnbvcxzasdfghjklpoiuytrewq</b>

e.g.: **Plaintext:** bob. i love you. alice  
**ciphertext:** nkn. s gktc wky. mgsbc

**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.

Il problema è ora comunicare la chiave senza che T la intercetti.

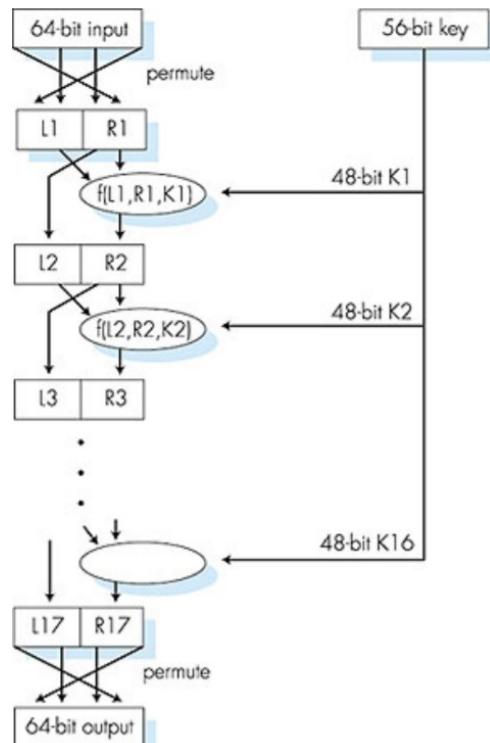
### DES (data encryption standard)

- Standard USA 1993
- 56 bit di chiave simmetrica
- 8 bit di controllo
- 64 bit di input per plaintext
- Oggi si trova la chiave in meno di un giorno
- Non esistono backdoor per questo algoritmo
- Esiste anche 3DES
  - Applicazione per tre volte dell'algoritmo con tre chiavi differenti
- La funzione è una funzione di hashing
  - Per ogni output non è ricostruibile l'input

## Symmetric key crypto: DES

### *DES operation*

- initial permutation
- 16 identical “rounds” of function application, each using different 48 bits of key
- final permutation

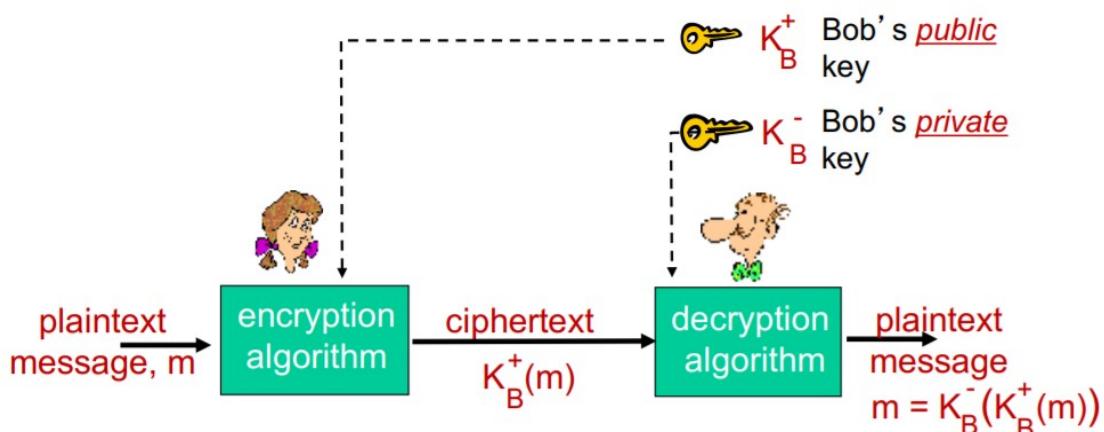


### AES (Advanced Encryption Standard)

- Ha sostituito DES nel 2001
- I dati sono processati in blocchi da 128 bit
- Le chiavi sono a 128, 192, 256 bit
- Con la forza bruta è improbabile decifrare la chiave in tempi utili
  - Ci vorrebbero 149 trilioni di anni se per ogni chiave servisse un solo secondo

### MECCANISMI A CHIAVE PUBBLICA E PRIVATA

- A e B hanno due chiavi a testa, una pubblica ( $K_A^+$  e  $K_B^+$ ) e una privata ( $K_A^-$  e  $K_B^-$ );
- La chiave pubblica è conosciuta da tutti e serve ad scambiare messaggi tra A e B
- La chiave privata è segretissima e serve a decifrare i messaggi;
- Data una chiave pubblica deve essere impossibile ricavare quella privata.



## 1/12

In generale, il meccanismo a chiave pubblica e privata è molto dispendioso in termini di risorse quindi è **utilizzato per scambiare chiavi simmetriche**, poi utilizzate per la comunicazione. Esistono due schemi principali di encryption: la fattorizzazione dei numeri primi e le curve ellittiche.

### FATTORIZZAZIONE DEI NUMERI PRIMI

Prendendo due numeri primi molto grandi e moltiplicandoli tra loro ottengo un numero che può essere solo il frutto di quella moltiplicazione.

Cosa serve:

- Chiave pubblica e chiave privata di B in modo che  $K_{priv}(K_{pubb}(m)) = m$
- Data la chiave pubblica dovrebbe essere impossibile risalire a quella privata

### RSA: Rivest, Shamir, Adelson algorithm

- Sappiamo che  $x \bmod n = \text{resto di } x/n$
- Sappiamo inoltre che:
  - $[(a \bmod n) + (b \bmod n)] \bmod n = (a+b) \bmod n$
  - $[(a \bmod n) - (b \bmod n)] \bmod n = (a-b) \bmod n$
  - $[(a \bmod n) * (b \bmod n)] \bmod n = (a*b) \bmod n$
  - Quindi  $(a \bmod n)^d \bmod n = a^d \bmod n$

Un messaggio è una sequenza di bit che può essere rappresentata univocamente da una sequenza di numeri quindi possiamo applicare la crittografia ai vari byte del messaggio.

Es:  $10010001 = 145 \rightarrow$  cifriamo 145 per ottenere un nuovo numero

Procedimento:

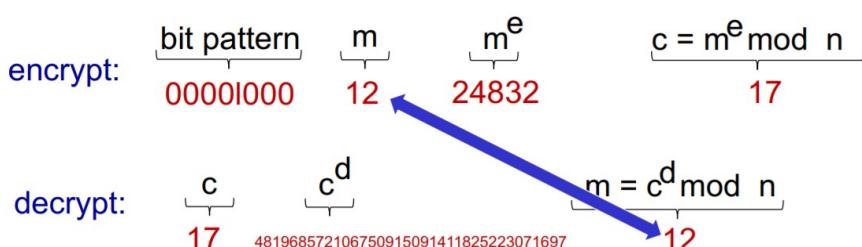
- Scegliamo due numeri primi molto grandi da circa 1024 bit l'uno, siano p e q
- Calcoliamo  $n = p * q$  e  $z = (p-1) * (q-1)$ 
  - z sarà pari
- Scegliamo un  $e < n$  che non abbia fattori in comune con z (sono primi tra loro)
  - e sarà dispari
- Scegliamo un  $d$  in modo che  $(e * d) - 1$  sia divisibile per z (ovvero  $e * d \bmod z = 1$ )
- La chiave pubblica diventa  $(n, e)$ ; la chiave privata diventa  $(n, d)$ 
  - Ovvero i bit di n seguiti da e o d

Ciò che rende segreta la chiave privata è il fatto che i bit di d non siano ricavabili da n ed e

Esempio:

Bob chooses  $p=5$ ,  $q=7$ . Then  $n=35$ ,  $z=24$ .  
 $e=5$  (so e, z relatively prime).  
 $d=29$  (so  $ed-1$  exactly divisible by z).

encrypting 8-bit messages.



Per le proprietà della funzione modulo si ha che  $K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$ .

Grazie a questo possiamo implementare il meccanismo di firma digitale dei messaggi.

RSA può anche essere usato semplicemente per scambiare le chiavi simmetriche.

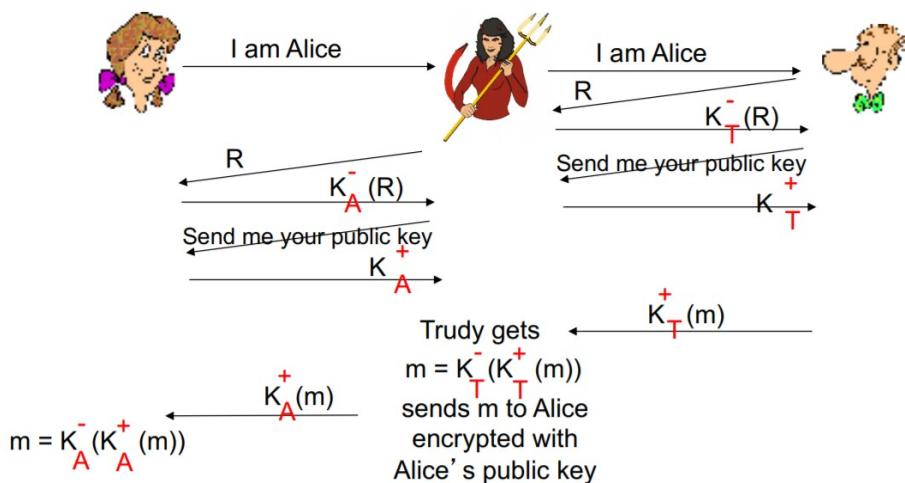
## AUTENTICAZIONE

Per evitare che T possa sfruttare un replay attack o un IP spoofing di A, si usa nonce, ovvero un valore casuale R da usare una volta sola (once in a lifetime).

### ap5.0: use nonce, public key cryptography



Anche in questo caso sarebbe però possibile fare un attacco man in the middle, inserendosi nella comunicazione (ad esempio in un router intermedio tra A e B) e inoltrando ogni messaggio al suo destinatario, copiandolo o modificandolo.



T invia un R da lui criptato e la sua chiave pubblica in modo da autenticarsi come A quindi se B volesse mandare un messaggio ad A, sarebbe ricevuto invece da T.

Nel frattempo T inganna allo stesso modo anche A.

La debolezza sta nel fatto che B deve chiedere ad A la sua chiave pubblica e che A debba fornirgliela, permettendo a T di conoscerla e di (de)cifrare i messaggi di conseguenza.

L'unico modo per essere sicuri di avere la giusta chiave pubblica è chiederla alla certification authority tramite scambio di messaggi autenticato con più authority nel mezzo.

I certificati che vengono ad esempio installati nel browser hanno una data di scadenza per evitare che un algoritmo forza bruta riesca a decifrare la chiave privata dell'authority in tempi utili.

## INTEGRITÀ DEI MESSAGGI - FIRMA DIGITALE

- Il sender B firma digitalmente con la propria chiave privata il documento stabilendo che lui è il creatore e il proprietario;
- Il ricevente A può provare a chiunque (incluso A stesso) che B ha firmato quel documento applicando la chiave pubblica di B;
- Il messaggio è quindi verificabile e NON modificabile.
  - In particolare, il messaggio viene firmato (cifrato) con la propria chiave privata e inviato sia in chiaro che cifrato con Kpriv.

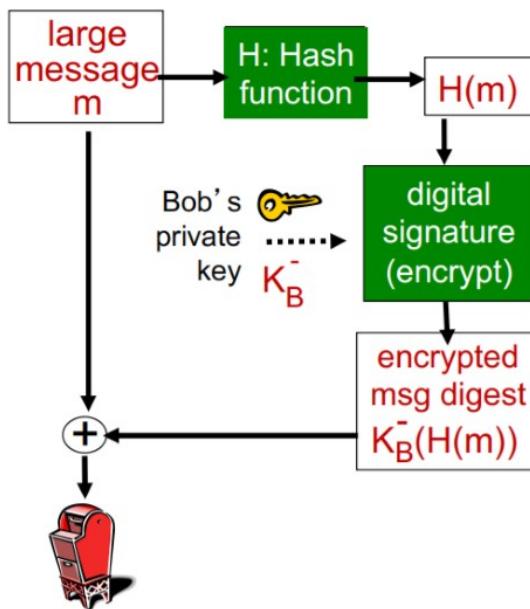
3/12

### MESSAGE DIGEST

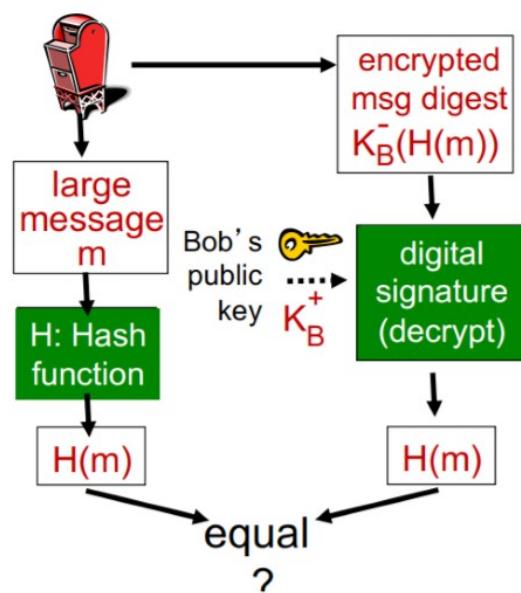
Nel caso in cui il messaggio da cifrare sia troppo lungo, esso viene passato ad una funzione di hashing H che produce una sequenza di bit di lunghezza fissa  $H(m)$  da cui non è possibile risalire a m. Tuttavia è possibile che m diversi collidano su un risultato.

Per questo la funzione è fatta in maniera che gli m che collidono siano totalmente diversi tra loro.

**Bob sends digitally signed message:**



**Alice verifies signature, integrity of digitally signed message:**



### ALGORITMI DI HASHING

#### MD5

- Calcola l'hash di un messaggio da 128 bit con digest in 4 step.
- Una qualunque stringa x da 128 bit passata a MD5 è molto difficile da ricostruire

#### SHA-1

- Calcola l'hash di un messaggio da 160 bit

### CERTIFICATION AUTHORITY

Abbinare le chiavi ad entità (es. Router, persone ecc) che forniscono prova della loro identità. Crea un certificato digitale che collega l'entità alla sua chiave pubblica firmato digitalmente.

## RENDERE SICURA LA MAIL

Alice wants to send confidential e-mail,  $m$ , to Bob.



**Alice:**

- generates random symmetric private key,  $K_S$
- encrypts message with  $K_S$  (for efficiency)
- also encrypts  $K_S$  with Bob's public key
- sends both  $K_S(m)$  and  $K_B(K_S)$  to Bob

**Bob:**

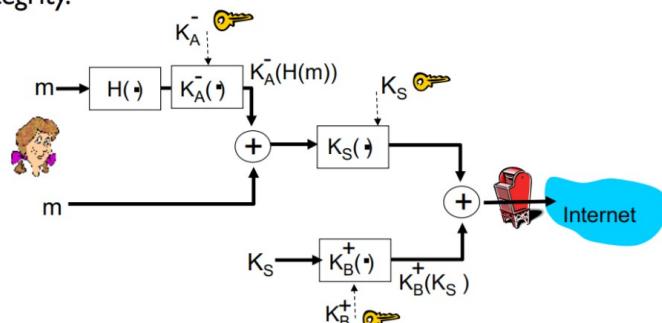
- uses his private key to decrypt and recover  $K_S$
- uses  $K_S$  to decrypt  $K_S(m)$  to recover  $m$

Alice wants to provide sender authentication message integrity



- Alice digitally signs message
- sends both message (in the clear) and digital signature

Alice wants to provide secrecy, sender authentication, message integrity.



**Alice uses three keys:** her private key, Bob's public key, newly created symmetric key

## RENDERE SICURO TCP TRAMITE SSL/TLS

SSL: secure socket layer

- Si interpone tra il livello trasporto (TCP) e il livello applicazione, fornendo delle API al livello applicazione).

TLS è una variazione del protocollo SSL.

Sia SSL che TLS servono a fornire un livello trasporto sicuro dove sono garantiti segretezza, autenticazione e integrità dei dati:

- Permette di avere su internet confidenzialità, integrità delle informazioni e autenticazione.
- In origine è nato per gli acquisti online (es. criptare numeri carte di credito), per l'autenticazione nei web server e altri scopi di sicurezza.
- SSL è disponibile per tutte le applicazioni che si avvalgono di TCP

Funzionamento:

- Handshake
  - A e B utilizzano i loro certificati e le loro chiavi private per autenticarsi e scambiarsi un segreto condiviso
- Derivazione delle chiavi (Key derivation)
  - A e B utilizzano il segreto appena scambiato per derivare le rispettive chiavi
- Trasferimento dei dati
  - I dati da trasferire sono suddivisi in record di dimensioni più piccole contenenti ognuno un nonce diverso per evitare replay attack.
- Chiusura della connessione
  - Si utilizzano messaggi speciali per chiudere la connessione in modo sicuro, impedendo che T possa farla terminare anticipatamente.

### SSL cipher suite

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

Soltanente 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.

È alla base delle VPN (virtual 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 grazie alla VPN.

## 7 - 10- 11/12

### FIREWALL

Il Firewall isola la rete interna di un'organizzazione (o comunque degli host connessi) da ciò che si trova all'esterno, permettendo di filtrare i pacchetti in ingresso, facendone passare alcuni e bloccandone altri.

Perchè serve il firewall:

- Previene gli attacchi DOS (es SYN flooding)
- Previene l'accesso non autorizzato e la modifica dei dati interni
- Permette l'accesso alla rete ai soli autorizzati tramite autenticazione
- Filtraggio di pacchetti
  - Stateless filter → pacchetto per pacchetto senza distinzione
    - Alcune regole di filtraggio stateless potrebbero essere:
      - IP mittente, Ip destinatario
      - Tipo di connessione (TCP / UDP)
      - Messaggi ICMP
      - SYN e ACK di TCP (per evitare attacchi)
  - Stateful filter → filtra in base allo stato attuale (es connessione stabilita) e al contesto attuale
  - Application gateway → Firewall speciali che filtrano o regolano l'utilizzo di una particolare app
    - Es. Telnet potrebbe non essere consentito in rete aziendale perché non criptato ma un application gateway potrebbe tradurre da ssh a telnet per avere dati criptati

Esempio:

<i>Policy</i>	<i>Firewall Setting</i>
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

### ACCESS CONTROL LIST (ACL)

Elenco di regole applicate dalla prima all'ultima ai pacchetti in entrata e relative azioni da compiere (nel formato azione, condizione).

L'ultima riga vieta tutto ciò che non è esplicitamente consentito.

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

Lo **stateful filtering** monitora e traccia le connessioni TCP (in particolare i bit SYN e FIN) e filtra i pacchetti in base a se hanno senso in quel contesto oppure no.

action	source address	dest address	proto	source port	dest port	flag bit	check connexion
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	X

### ESEMPIO DI APPLICATION GATEWAY PER TELNET



- Tutti coloro che vogliono connettersi tramite telnet devono passare per l'application gateway.
- Per gli utenti autorizzati ad utilizzare telnet, il gateway imposta la connessione con il destinatario permettendo la trasmissione dei dati.
- I filtri presenti nel router bloccano tutte le connessioni non originate dal gateway
- In alcuni contesti aziendali il gateway potrebbe annotare su log chi si connette tramite telnet

### ZONA DEMILITARIZZATA (DMZ)

Area di un dominio o di una rete dove vengono forniti servizi a coloro che si trovano fuori dalla rete (es. server, mail server ecc) per cui il firewall autorizza gli accessi.

Se qualcuno dall'esterno entrasse nella zona demilitarizzata non ci sarebbe alcun modo attraverso cui potrebbe compromettere la parte restante del dominio, questo grazie al firewall.



## RETI WIRELESS

### PROPRIETÀ DELLE RADIO FREQUENZE

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 saranno messi in movimento.

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

Una carica che si sposta nello spazio induce un'emissione di onde elettromagnetiche.

Il mezzo di propagazione delle onde radio è il vuoto.



- **Aampiezza**

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

Wavelength =  $c / \text{frequency}$

Esempio: il WiFi ha lunghezza d'onda 12,5 cm.

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

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, devono 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 pari ad un sottomultiplo binario della lunghezza d'onda.

## PROPAGAZIONE RADIO FREQUENZE

Una problematica nel mondo delle radiofrequenze ha a che fare con la loro propagazione, non posso ottenere una copertura generale ovvero non posso 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, tuttavia il segnale decade in modo esponenziale quindi per poter raddoppiare la distanza al quale trasmettere, dobbiamo quasi quadruplicare la potenza trasmittiva del segnale in partenza (e quindi la potenza con il quale il segnale viene generato).



La Signal Detection Limit 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 della Signal Detection Limit.
- La circonferenza formata da questa distanza  $d$  è pari quindi alla Radio Transmission 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.

Ci sono due modi per risolvere questo problema: o avvicinare il client o aumentare la potenza trasmittiva del trasmettitore.

Formule importanti (in una sfera):  $V = (4 \pi r^3 / 3)$   $S = (4 \pi r^2)$

dove  $V$  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!**

Schema che rappresenta il range di propagazione del segnale wireless. L'area esterna (range di interferenza) è potenzialmente infinita, riesco a stenti a capire che ci sia un segnale. Questo schema dipende molto dalla Sensibilità del ricevente (Receiver sensitivity).



Ogni scheda radio ha una sensibilità differente, quindi il detection range ad esempio, può essere più ampio per alcuni dispositivi che hanno sensibilità maggiore.

Altri invece, con sensibilità minore, possono avere un detection range veramente piccolo.

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).

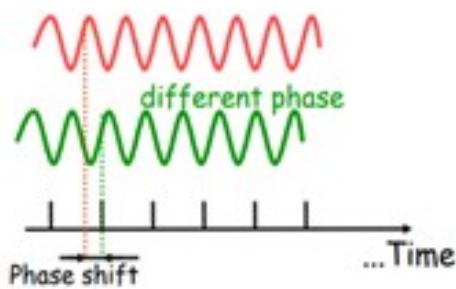
#### ▪ Radio transmission coverage



- 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

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

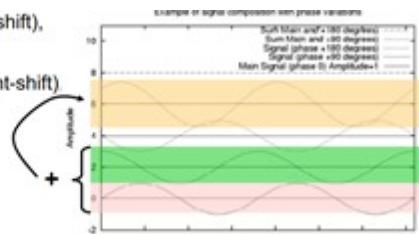
Due segnali con stessa ampiezza e frequenza, ma fase differente. Anticipo e ritardo corrispondono a quantificare gli slittamenti dell'onda.

Problema: i segnali che ricevo ad una certa distanza arrivano sia per via diretta che facendo una strada più lunga ed hanno sfasamento maggiore (rimbalzando su degli ostacoli, ad esempio).

Il segnale che parte è lo stesso, il segnale che arriva (lato ricevente) è pari alla somma vettoriale tra segnale diretto e segnale che ha fatto una strada più lunga.

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 waveform
- Negative phase (right-shift) late waveform

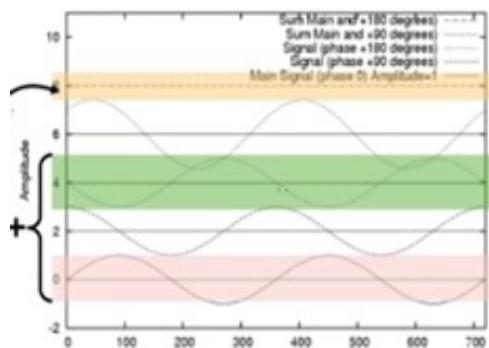


Sommendo punto a punto (vettorialmente) il valore della sinusoide verde e della sinusoide rosa otteniamo la sinusoide arancione.

In questo caso, avrei un guadagno di ampiezza.

Lato ricevente, ovviamente preferisco ricevere segnali con ampiezze maggiori, attraverso somma di segnali. In alcuni casi però, la somma vettoriale (punto per punto) di più segnali ha come risultante 0 (anche se sono a pochi cm dal trasmittente, posso avere una somma di segnali che si autoannulla).

Possiamo immaginare questo anche attraverso i suoni, immaginando due segnali acustici opposti emessi da una sorgente, che sovrapposti si annullano. Questo fenomeno viene chiamato Opposizione di fase.

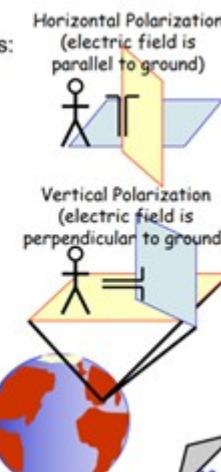


← Opposizione di fase, fasi sfasate di circa metà periodo (180 gradi).

La somma del segnale diretto e del segnale sfasato produce 0 (si auto annulla).

## POLARIZZAZIONE

- **Polarization: (physical orientation of antenna)**
  - RF waves are made by two perpendicular fields:
    - Electric field and Magnetic field

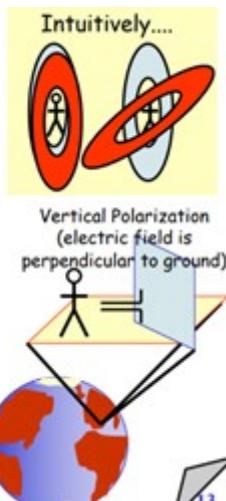
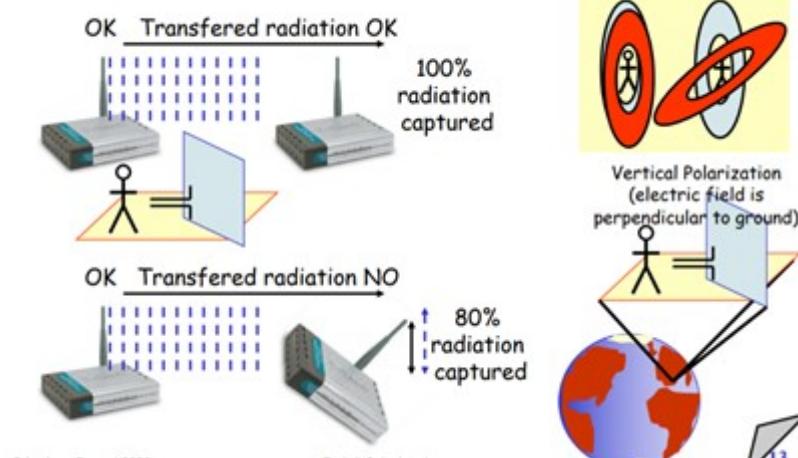


La polarizzazione è fisicamente l'orientamento dell'antenna. I piani su cui si trasmettono le onde radio sono 2, Magnetico (perpendicolare all'antenna) ed Elettrico (parallelo all'antenna). Questi due piani si autosostengono e viaggiano ruotati di 90 gradi. Se l'antenna viene posizionata in modo orizzontale 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 di ha quando l'antenna del ricevente è polarizzata come l'antenna del trasmittitore (altrimenti, avremmo una perdita di energia del segnale trasferito, e rischiamo di scendere proprio a causa di questa perdita dovuta alla polarizzazione sotto la Signal Detection Limit (linea azzurra 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

**COMPORTAMENTO RADIO FREQUENZE**

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



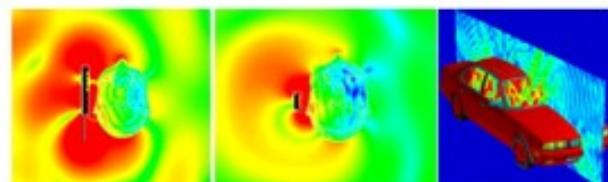
Lato ricevente, quello che posso captare è dinamicamente la composizione di fasi e frequenze di tanti segnali che si sommano vettorialmente generando interferenze.

Sono i protocolli che riescono a portare ordine in questo sistema (e quindi, filtrando le varie frequenze).

Effetto radiofrequenze in auto: simile a gabbia di Faraday, annulla il valore del campo elettromagnetico al di fuori della gabbia stessa.

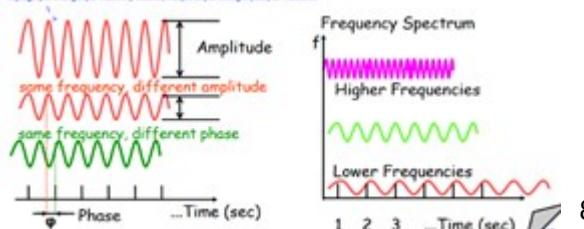
Un parallelo intuitivo è immaginare l'auto come una barriera fonoassorbente, lo smartphone che cerca di collegarsi 'urla' energia, che viene abbattuta dall'auto.

Il cellulare si scarica prima perché utilizza più energia per cercare di connettersi al trasmettitore.

**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 prendono in input la sinusoide e restituiscono in output la stessa sinusoide variando solo l'ampiezza (attraverso una sorgente di energia, ad esempio una batteria)

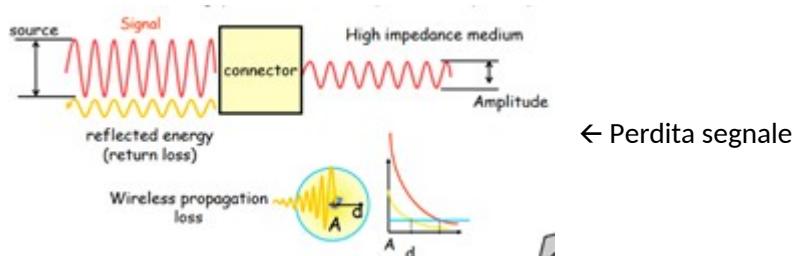


- Guadagno passivo di energia: un altro modo di catturare energia, ad esempio attraverso una parabola.
  - La parabola tende a catturare energia in quantità direttamente proporzionale alla grandezza del disco e questa viene catturata e focalizzata in un punto.

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

- 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 può percorrere il 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



**Shadowing:** impedisce che l'onda

proseguo Riflessione: l'onda viene riflessa

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

**Diffrazione 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

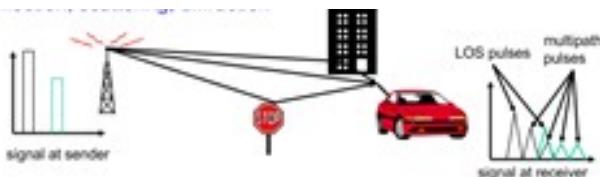


A seconda di dove ci poniamo, il segnale ha energia diversa.  
 La prima immagine raffigura la propagazione del segnale in una zona montuosa.  
 Possiamo avere dei coni d'ombra, come ad esempio ai piedi della montagna, che corrispondono a dei punti in cui abbiamo assenza di segnale.  
 Le striature in foto rappresentano i rimbalzi dell'onda verso l'alto.

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

La 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 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 quello che ha rimbalzato sul grattacielo arriva con un evidente ritardo di fase.

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.

(**Dispersione temporale:** il segnale viene disperso nel tempo)



Variazione di segnale che posso avere spostandomi 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 sparisce e poi riappare.

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

## VSWR e INTENTIONAL RADIATOR

### 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)
- 

Quando creiamo un sistema di comunicazione, ogni volta che mettiamo in collegamento un dispositivo con un altro successivo dobbiamo fare attenzione al valore di impedenza nominale ovvero alla resistenza della corrente perché se i due componenti hanno delle impedenze 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 Così non fosse avremmo un effetto

di dissipazione di energia e ciò non è mai positivo perché abbiamo consumato dell'energia per generare quell'energia, senza poi avere degli effetti positivi dal punto di vista della comunicazione. Quando abbiamo un sistema di comunicazione, tutto il blocco (esclusa l'antenna) si chiama **Intentional Radiator** (Radiatore intenzionale, IR).

Quando misuriamo l'energia di un sistema di trasmissione radio l'energia a cui facciamo riferimento è l'energia che arriva all'antenna. Questa 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 del trasmettitore meno tutte le perdite causate da collegamenti, connettori, etc..., fino all'ultimo elemento prima dell'antenna.

### EIRP

Se noi abbiamo una antenna le cui caratteristiche sono quelle di concentrare le energie in un punto dello spazio (una parabola, ad esempio) è come dire che se dai dell'energia a questa parabola, non viene sparata in tutte le direzioni omogeneamente (come una lampadina che illumina in tutte le direzioni). La lampadina è l'equivalente ad un radiatore isotropico, ovvero un'antenna che emette energia omogeneamente nello spazio.

Un'antenna che concentra l'energia in una (almeno) direzione preferenziale, allora noi 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 quella energia che servirebbe a un radiatore isotropico per generare lo stesso effetto di onda elettromagnetica nella direzione preferenziale.

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 EIRP

Se do 16mW direttamente all'antenna isotropica, l'energia irradiata in tutte le direzioni è 16mW.

Se do 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 dovrei 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.

### MISURAZIONE DELL'ENERGIA (dB, W ecc...)

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 questa energia decade velocemente con la distanza con la legge di propagazione, e questa 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 e guadagni di potenza. Il dB è nato soprattutto per esprimere le perdite, che possono arrivare a 11, 12, 13 cifre dopo la virgola, che diventano difficili da gestire per fare dei calcoli.

dB usa una Scala logaritmica per mappare i mW ed esprimere le grandezze.

**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.

Esempio:

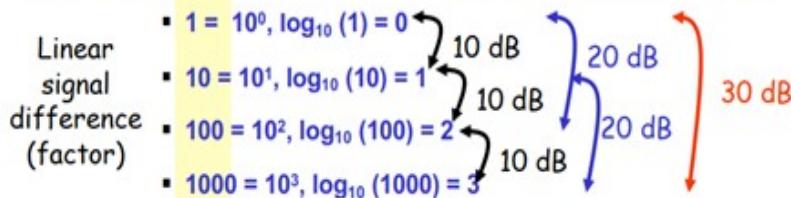
"A quanto corrisponde un segnale forte 10dB" → Non vuol dire nulla, non ha senso

"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 ( $1 \text{ dB} = 1/10 \text{ Bel}$ )

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



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 quindi 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.

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

- Il rapporto è minore di 1 poiché 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.

-3 dB	$\frac{1}{2} \text{ power in mW (/ 2)}$
+3 dB	$2x \text{ power in mW (* 2)}$
-10 dB	$\frac{1}{10} \text{ power in mW (/ 10)}$
+10 dB	$10x \text{ power in mW (* 10)}$

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.  $100 \text{ mW} - 6 \text{ dB} = 100 \text{ mW} \cdot 3 - 3 \text{ dB} = 100 / 2 / 2 = 25 \text{ mW}$
- E.g.  $100 \text{ mW} + 7 \text{ dB} = 100 \text{ mW} + 10 - 3 \text{ dB} = 100 * 10 / 2 = 500 \text{ mW}$
- E.g.  $10 \text{ mW} + 5 \text{ dB} = 10 \text{ mW} (+10+10-3-3-3-3)\text{dB} = 1000/32 = 31.25 \text{ mW}$

**dBm**

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

Assumiamo che 1mW = 0 dBm

Attenzione:

**Tabella conversione da mW a dBm**

-40 dBm	-30 dBm	-20 dBm	-10 dBm	0 dBm	+10 dBm	+20 dBm	+30 dBm	+40 dBm
100 nW	1 μW	10 μW	100 μW	1 mW	10 mW	100 mW	1 W	10 W

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

**DIPOLO e dBi**

dBi = 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 a 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 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 usati 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 usata: 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, ottengo un'antenna tilt, ovvero un'inclinazione da dare all'antenna in modo da ottenere una migliore copertura.

Se inclino l'antenna verso il basso, ottengo un Down Tilt.

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

Esempio di Tilt:



Anche se comunque ormai i router attuali hanno più di un'antenna, nei router di ultima generazione è utilizzata una tecnologia chiamata MIMO che 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.

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 HIGHLY-DIRECTIONAL

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

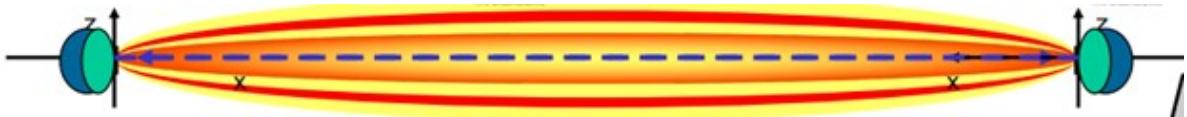
E' molto importante centrare correttamente le antenne (e vincolarla bene in modo che non si muova ed il vento non faccia oscillare la parabola), altrimenti i raggi possono andare fuori bersaglio e il segnale non arriva.



## FRESNEL ZONE

Line Of Sight = Linea di vista

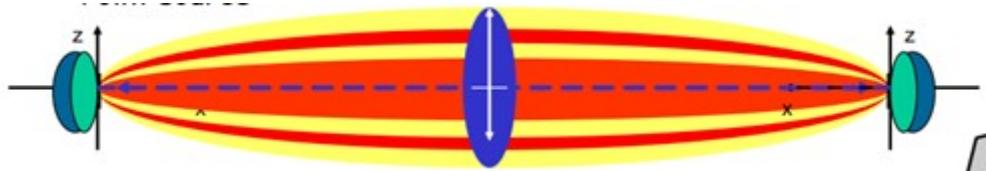
(Linea Blu nel grafico seguente, tra le due antenne)



Immaginiamo di avere due antenne. Queste due funzionano meglio a trasferire energia quando la Line Of Sight è libera da ostacoli (mettendo in mezzo un ostacolo, 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 perdiamo energia inutilmente.

La regola che devono progettare i progettisti di sistemi radio è calcolare il raggio del disco blu con le appropriate formule e poi 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:

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

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

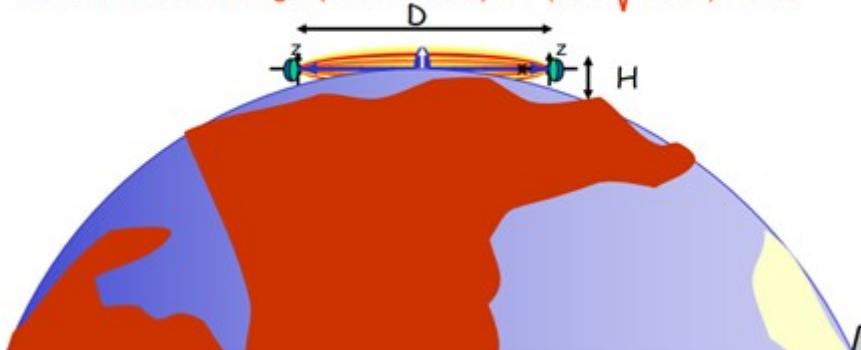
A volte la Fresnel Zone potrebbe essere ostruita da ostacoli tra cui la curvatura terrestre, in caso di comunicazione a grandi distanze.

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

Per questo, nella pratica, vengono utilizzati dei tralicci per mettere (ad esempio per i ripetitori tv/telefonia).

Importante, 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) =  $D^2/8$
- 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 (Selectorized, a Settore)

Utilizzando delle antenne direzionali o semi-direzionali posso coprire tutti i 360 gradi ambiente intorno dividendo i raggi di emissione radio in 6 diversi separati raggi. Ogni stecca nella foto è un'antenna patch, emette segnale su un angolo aperto 30 gradi, questi tralicci fanno copertura rete cellulare.

Con il cellulare quindi, quando utilizziamo la rete, usiamo solo una delle 6 antenne, se qualcuno usa un'antenna affianco, può comunicare i suoi dati 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 (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 usare più volte la stessa frequenza verso più clienti 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 all’interno del sistema cellulare, opportunità che abbiamo utilizzando questi array di antenne).

Come fa il telefono 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 usano tutti i telefoni), ascoltano quella frequenza e capiscono quali frequenze 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 che trasmetto 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**

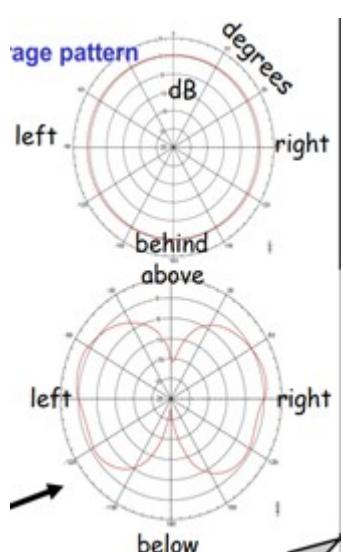


sectorized antenna

#### **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 troviamo nel foglio tecnico delle antenne, e consente al progettista di saper 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 che a noi interessa.

## DIVERSITY (EFFETTO)

I router moderni, hanno gruppi di due o più antenne (Dipoli).

Queste antenne sono posizionate tutte a distanza Lambda/2. Questa distanza non è casuale, ma è stabilita in modo che a quella distanza non posso avere opposizione di fase su tutte le antenne (avrò sempre un segnale su un'antenna che non sia in opposizione di fase).

Sempre per composizione di fase dalla diversity, estraggo 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); Estraggo 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)

Quest 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 noi otteniamo il risultato in dB, qualunque potenza parta dalla trasmissione, dalla forma della curva che dipende da frequenza e distanza possiamo dire 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 trasmettitore, 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).

- 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)



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

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 anche che 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 in qualsiasi condizione atmosferica, ad esempio).

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.

Trasmittitore: Il trasmittore 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 trasmittore 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 catturerebbe un isotropico). In questo modo utilizzo 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 la ricevo, in alternativa se su 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.

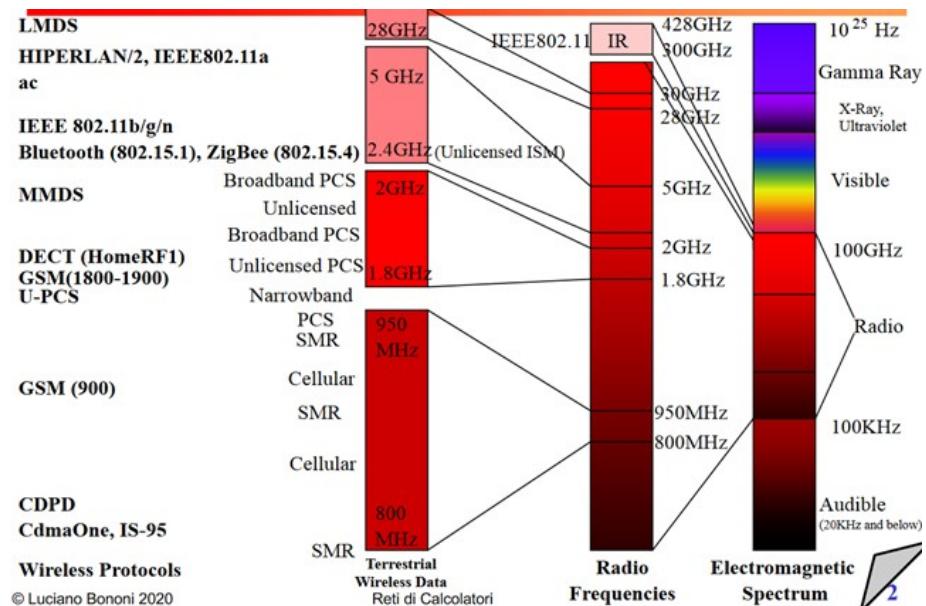
Possibile domanda esame:

-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 trasmittitore (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 perdite di energia minori.

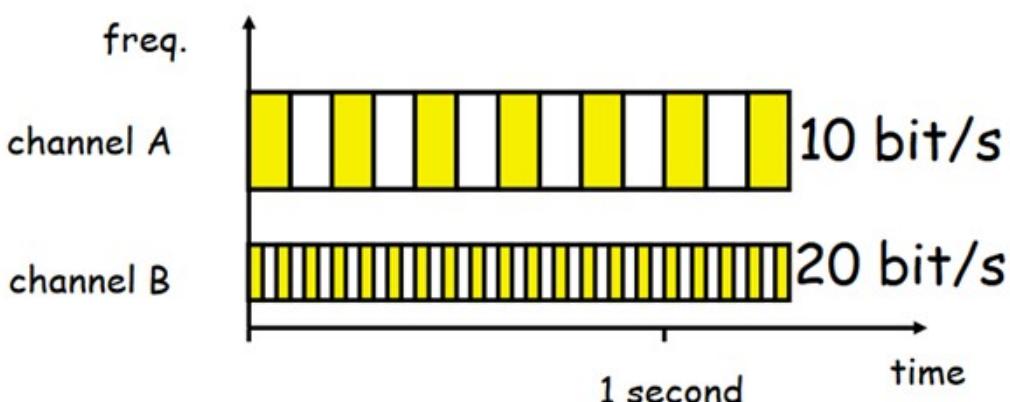
## **WIRELESS PARTE 2**

### **SPETTRO DELLE ONDE 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.



#### VELOCITÀ DI TRASMISSIONE DEI BIT

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 1Mb/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 un certo canale, 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).

### CODIFICHE

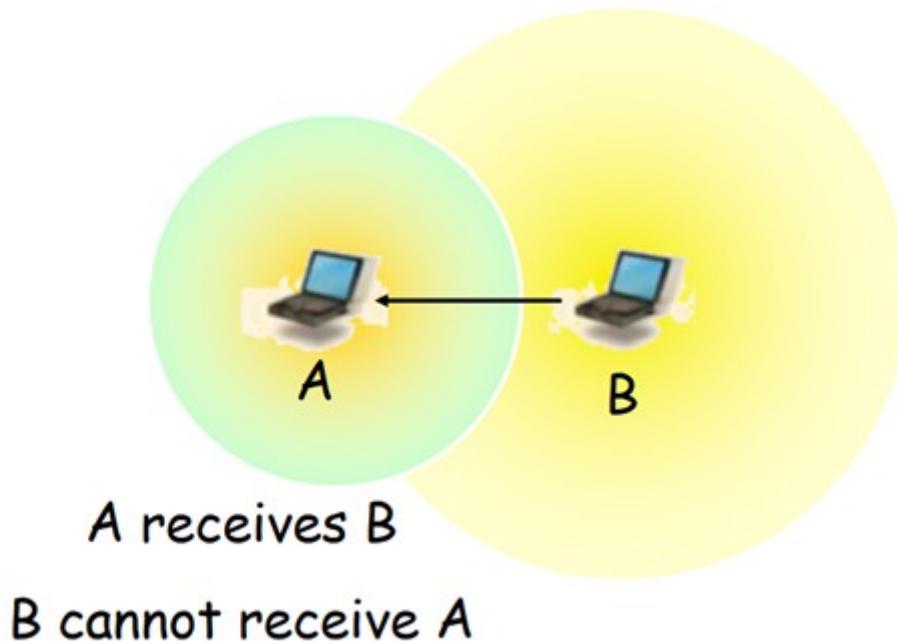
Le tecniche di codifica di bit realizzano un compromesso tra ciò si vuole trasmettere bene e ciò che si vuole trasmettere velocemente. Ci possono essere diverse situazioni, dipendentemente dai range (detti dalla potenza trasmittiva).

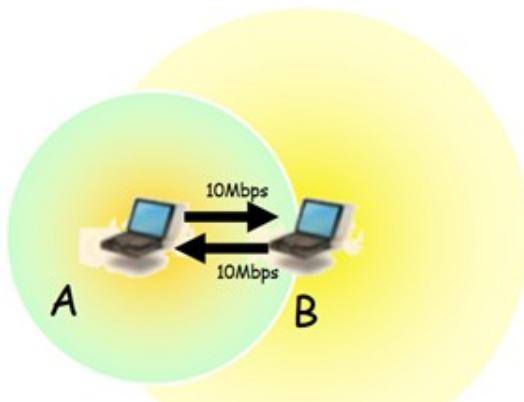
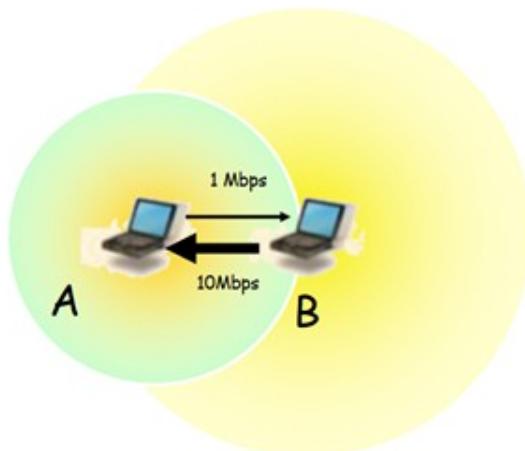
Link possibili durante la creazione di canali radio

Caso 1, sia A che B isolati:



Caso 2, A riceve da B ma B non può ricevere da A, link unidirezionale.



**bidirectional symmetric link****bidirectional asymmetric link**

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)

Creazione dei canali radio:

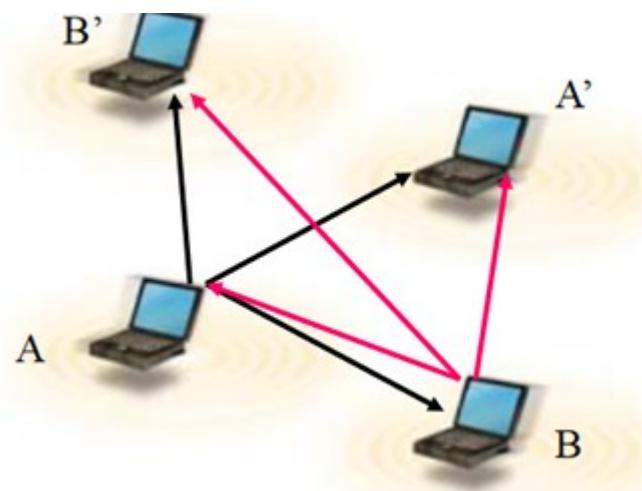
La creazione di diversi canali radio può essere di 3 tipi:

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 vengono evitate perché riusciamo ad estrarre dal canale che ci interessa. Le collisioni radio si hanno quando si vuole usare una frequenza per comunicare tra più di 2 interlocutori.

→ **frequency X**  
 → **frequency Y**





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).

È 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.

## MULTIPLEXING

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

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.



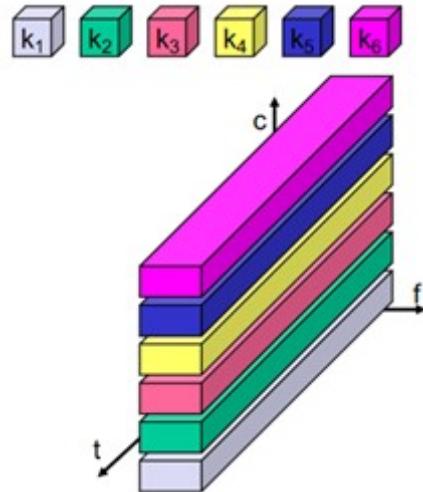
Ogni canale di rete che usa un canale diverso avrà il proprio intervallo di tempo nella quale può trasmettere, e quando può trasmettere usa tutto lo spettro radio.

**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).



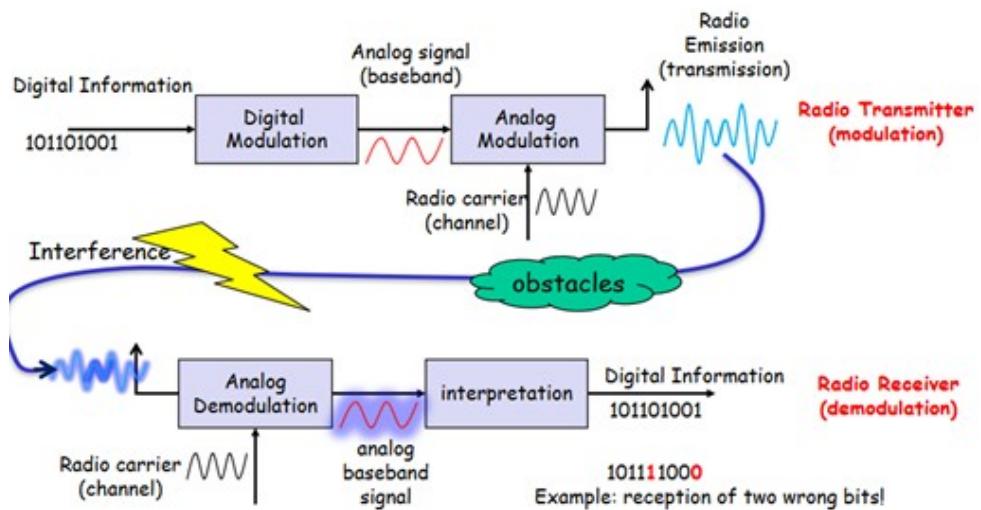
## MODULAZIONE E DEMODULAZIONE

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 infatti 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 trasmettitore) 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

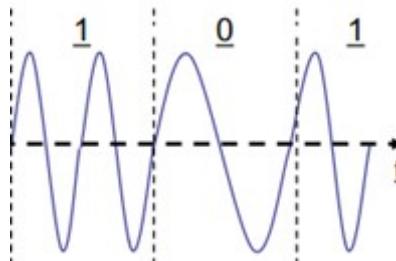
ASK, FSK e PSK (Amplitude, Frequency e Phase shift keying) sono 3 modi per rappresentare 0 e 1 tramite una sinusoide.

**ASK**, bit 1, rappresentato da un’ampiezza elevata (ampiezza = differenza di potenziale tra picco positivo e negativo), bit 0 trasmetto una sinusoide piatta, cioè 0 energia. Questa implementazione è molto sensibile alle interferenze ma usa una sola frequenza (narrow band).

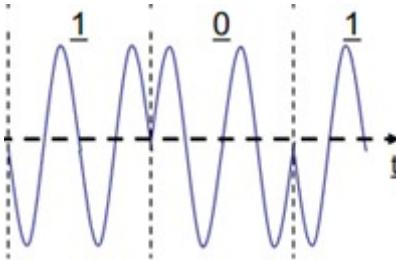


**FSK**, sinusoide ad una certa frequenza  $f_1$  può essere variata in un’altra frequenza  $f_2$  inferiore. Possiamo fare in modo che  $f_1$  rappresenti il bit 1,  $f_2$  il bit 0. Lo svantaggio di questa soluzione è che

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).

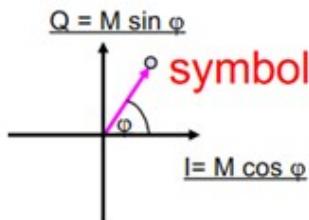


**PSK**, variamo la fase. L'ampiezza è la stessa, la frequenza è la stessa, però varia la fase, infatti il segnale parte con la sinusoide positiva per lo 0 e con quella negativa per l'1.



### 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 infatti 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 trasporta 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 sceglieremo 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 solo bit sbagliato, non mi consente di trovarlo se i bit sbagliati sono 2 o più.

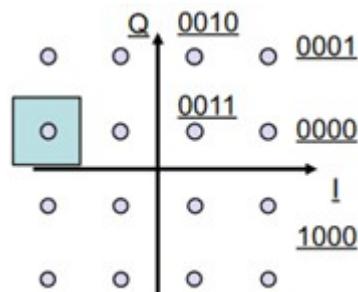


In questo caso, un qualsiasi bit diverso avrebbe reso dispari il numero di bit ad 1, consentendoci così di rilevare l'errore su singolo bit (ma, ricordiamo, non possiamo accorgercene quando l'errore è su 2 bit). E' possibile creare una struttura (matrice di bit di parità) che permette anche la correzione 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? 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 solo la fase).



### MODULAZIONE GERARCHICA

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. E' 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.

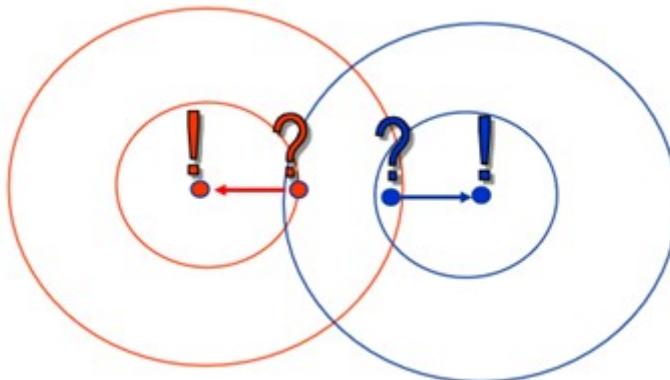
**100010 | 011001 | 111100 | 000101**



### 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 sfalsandoli 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.



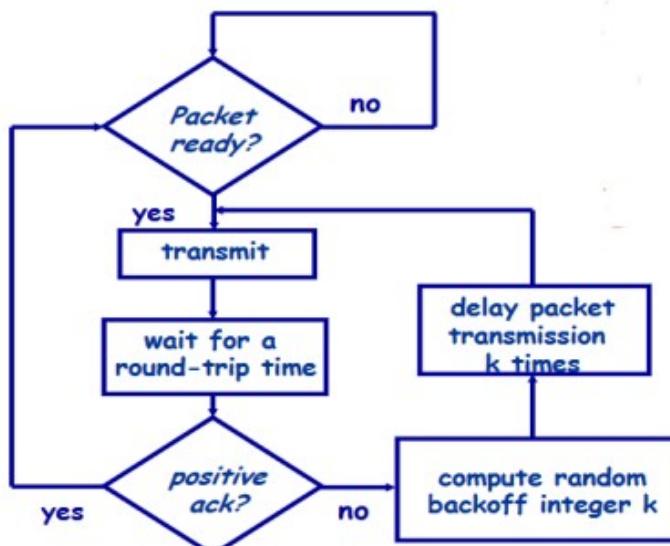
In un sistema wireless può esistere o meno un coordinatore. Se c'è, l'AP (access point) è centrale ed evita le collisioni. Se il coordinatore centrale muore/si rompe/non c'è/succede qualcosa, si ricade nello schema base (per le reti orfane del coordinatore è ancora funzionante). Lo schema ad hoc è

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 protocol che cercano di risolvere il problema nel tempo (chi trasmette quando?).
2. MAC protocol 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

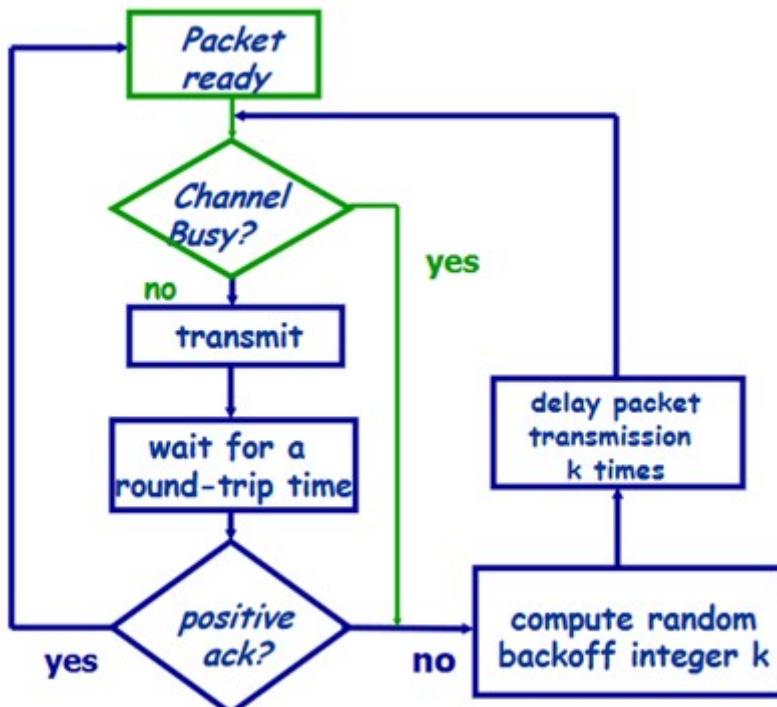
Per migliorare la situazione, si utilizza una versione di ALOHA slotted, cioè il tempo viene misurato con unità costanti sincrone, e la trasmissione può essere effettuata solo all'inizio di uno slot successivo. In questo modo due frame rischiano la collisione solo quando vengono trasmessi nello stesso 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 Sense 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.

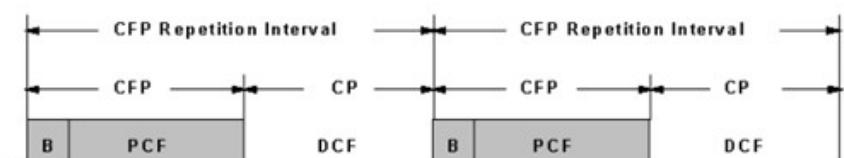


## FUNZIONAMENTO DI 802.11

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 un 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

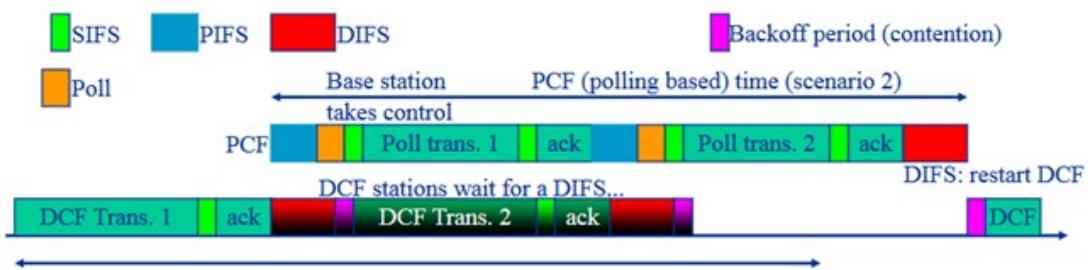
importante per tenere conto di come si alterneranno le fasi. L'alternanza di queste fasi viene dettata utilizzando 3 intervalli, Short IFS (SIFS) < Point IFS (PIFS) < Distributed IFS (DIFS). In pratica l'intervallo SIFS è un tempo vuoto sul canale (molto breve) che serve solo a dare l'autorizzazione 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, e magari nel punto in cui il suo  $k$  è arrivato a 2 qualcun'altro trasmette, quando il canale ridiventà 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 ridiventà 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.

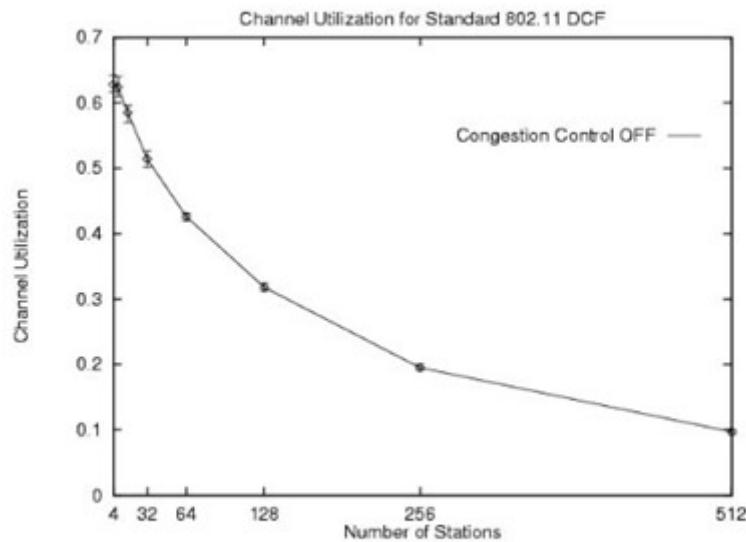
**CWi=contention window size at the i-th transmission attempt. CWi is doubled after each collision experienced (to reduce the contention)**

$$\text{BackoffTime}(i) = (\text{CWi} * \text{random}) * \text{SlotTime}$$

$i$	1	2	3	4	5	6	7
$\text{CW}_i$	15	31	63	127	255	511	1023

Ogni volta avrò 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? E' 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

**FINE**

**APPENDICE 1 - POTENZE DI 2**

0	1	5	32	10	1024	15	32768	20	1048576
1	2	6	64	11	2048	16	65536	21	2097152
2	4	7	128	12	4096	17	131072	22	4194304
3	8	8	256	13	8192	18	262144	23	8388608
4	16	9	512	14	16384	19	524288	24	16777216

**APPENDICE 2 - FLOW vs CONGESTION CONTROL**

<https://www.geeksforgeeks.org/difference-between-flow-control-and-congestion-control>

FLOW CONTROL	CONGESTION CONTROL
Viene controllato il traffico tra sender e receiver	Viene controllato il traffico entrante nella rete
Gestito da lv. data link e trasporto	Gestito da lv. rete e trasporto
Si cerca di non far perdere pacchetti al receiver	Si cerca di non far perdere pacchetti alla rete
Il sender è il responsabile del traffico	Il lv trasporto è il responsabile del traffico
Il sender invia i dati lentamente per evitare la congestione del receiver	Il lv. trasporto invia i dati lentamente per evitare la congestione della rete
Se c'è buffer overflow, avviene nel receiver	Se c'è buffer overflow, avviene nei router intermedi

## APPENDICE 3 - HOW TO ESERCIZI

- Quante sottoreti sono individuate dato IP e maschera di rete?
  - Converto IP e maschera in binario
  - In base alla classe di appartenenza dell'indirizzo (A, B, C) conto gli N bit che estendono la classica maschera (/8, /16 o /24).
  - Sono individuate  $2^N$  sottoreti

**ES:** IP 130.136.128.128; MASCHERA: 255.255.128.0 (/17)

IP in binario            10000010 10001000 10000000 10000000

Maschera in binario    11111111 11111111 10000000 00000000

L'indirizzo è di classe B quindi la sua maschera è estesa con un solo bit.

Le sottoreti individuate sono due: la 0 e la 1

- A quale sottoretete appartiene IP data la maschera
  - Converto IP e maschera in binario
  - Non considero i bit degli host nell'IP
  - Calcolo il valore decimale del byte da considerare (si parte da 0)

**ES:** IP 130.136.9.1; MASCHERA: 255.255.248.0 (/21)

IP in binario            10000010 10001000 00001001 00000001

Maschera in binario    11111111 11111111 11111000 00000000

I bit corrispondenti agli host nel terzo byte sono 001.

Per questo calcolo non li considero e ottengo 00001 che ha come valore decimale 1.

Questa sarà quindi la sottorette 1 della rete 130.136.x.y (ma visto che si parte da 0 sarà la seconda)

- Quale numero di host ha la macchina dati IP e maschera
  - Converto IP e maschera in binario
  - Considero il valore decimale associato ai bit dell'IP corrispondenti agli zero della maschera di rete.

**ES:** IP 130.136.249.0; MASCHERA: /23

IP in binario            10000010 10001000 11111001 00000000

Maschera in binario    11111111 11111111 11111110 00000000

I bit a zero della maschera corrispondono sull'IP a 1 00000000.

Il valore decimale associato sarà quindi 256.

Si tratta dunque dell'host 256.

## APPENDICE 4 - MASCHERE DI RETE

Calcolo veloce della maschera in formato CIDR

128	+1	248	+5
192	+2	252	+6
224	+3	254	+7
240	+4	255	+8

ES:  $255.255.224.0 = 8 + 8 + 3 = 19 \rightarrow /19$