PROGRAMMAZIONE DI RETI

Canale e Rete

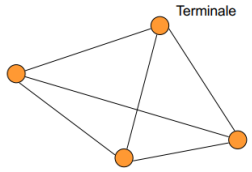
- Telecomunicazioni utilizzano **canali di comunicazione**:
- Entità logica o fisica che permette trasporto dei flussi informativi fra nodi remoti
    - **Monodirezionale**: informazione trasferita in una sola direzione
    - **Bidirezionale**: informazione trasferita in entrambe le direzioni
    - **Punto-Punto**: un nodo è collegato con un solo nodo
    - **Punto-Multipunto**: un nodo può comunicare con tanti altri:
        - **Broadcast**: nodo trasmette a tutti i nodi della rete
        - **Multicast**: nodo trasmette a un sottoinsieme di nodi della rete

Componenti della rete:

- Terminali:
    - Fungono da interfaccia con l'utente finale
    - Codificano l'informazione in modo consono ad essere trasferita in rete
- Collegamenti:
    - Permettono il trasferimento di uno o più flussi di informazione fra punti remoti
- Nodi di Commutazione:
    - Utilizzano i mezzi trasmissivi per creare canali di comunicazione sulla base degli richieste degli utenti

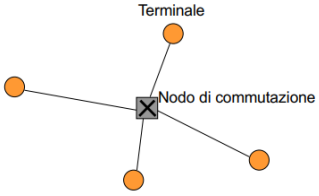
Topologie di rete:

- Maglia completa:



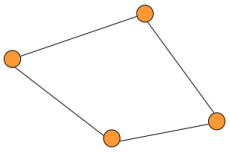
Un collegamento per ogni coppia di nodi
N nodi implicano N(N-1) / 2 collegamenti
Grande resistenza ai guasti ma complessa e costosa

- Stella:



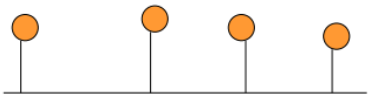
N collegamenti
Centro stella deve smistare informazioni
Minor costo ma minor resistenza ai guasti

- Anello:



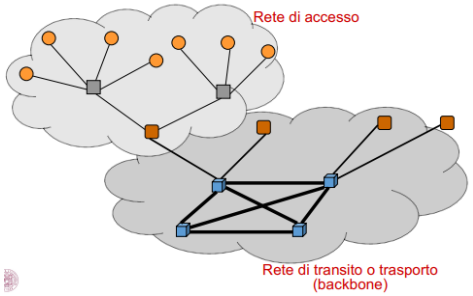
Monodirezionali: se un collegamento si interrompe la rete si guasta
Bidirezionali: maggiore complessità ma maggiore resistenza ai guasti

- Bus:



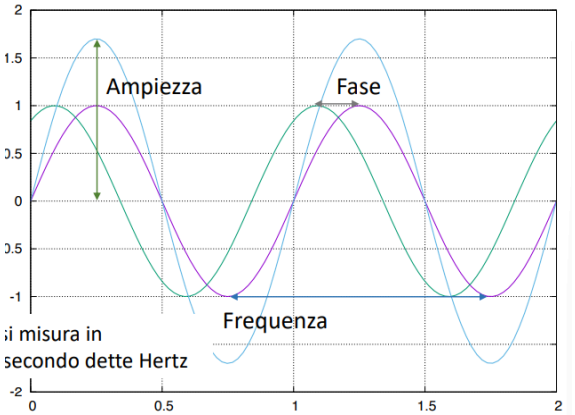
bus è bidirezionale
mezzo di trasmissione è condiviso:
necessario definire protocollo di accesso (MAC)

- Collegamenti realizzati tramite mezzi trasmissivi: etere, fibre ottiche ecc…
- Collegamento è oggetto fisico per realizzazione di canali; può essere che si usino più collegamenti per fare 1 canale
- Rete suddivisa in rete di Accesso e di Transito (backbone):
  - Terminali si connettono alla rete tramite quella di accesso che comunicano con quella di transito, che comunicano tra loro tramite collegamenti più veloci
  - La rete di transito collega più reti di accesso



Informazione, Segnali, Digitalizzazione

$A \sin(2\pi f t + \phi)$



Si ricava un grafico ampiezza/frequenza (Spettro del segnale) (quello non è il grafico dello spettro)

- Noto lo spettro sono note anche la frequenza minima e massima (f_m e f_M)
- Banda del segnale: $B = f_m - f_M$
- Più è grande B, maggiore è la complessità del segnale

Conversione analogico / digitale

- **Campionamento**:
  - Segnale analogico misurato in determinati istanti di tempo
  - Si produce una serie temporale di numeri corrispondenti alle misure fatte
- **Quantizzazione**:
  - Numeri risultanti dal campionamento rappresentati in forma binaria

- Problema principale: usare giusto intervallo di campionamento, T_S è la larghezza
- **Teorema di Shannon-Nyquist:**
  - Fornisce un criterio per determinare il giusto intervallo di campionamento

$$f_s \geq 2f_M$$

$$T_s = \frac{1}{f_s} < \frac{1}{2f_M}$$

Esempio: banda fonica ha f_m = 0 Hz e f_M = 4000 Hz
Quindi B = 4 KHz
In base al teorema:

$$T_s < \frac{1}{2f_M} = \frac{1}{8000} = 125\mu s$$

Quindi il campionamento di un segnale telefonico genera una sequenza di 8000 campioni al secondo

- Vantaggi del digitale:
  - Integrazione: formato dell'informazione unificato
  - Computazione: segnali che diventano bit possono essere elaborati da calcolatori → Compressione, cifratura ecc…

Reti e Servizi

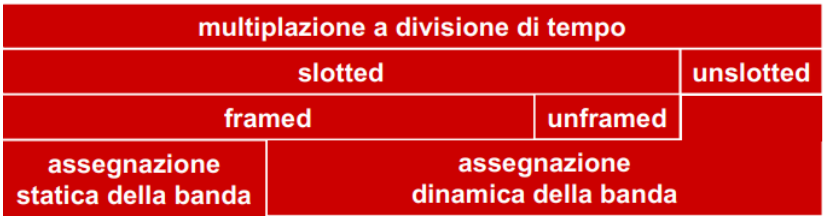
- **Servizi Interattivi**: esiste interazione tra sorgente e destinazione
  - Conversazione: scambio informativo in tempo reale (telefonia)
  - Messaggistica: scambio informativo in tempo differito con memorizzazione
  - Consultazione: scambio informativo con flusso controllato dall'utente (WWW)
- **Servizi distributivi**: sorgente diffonde informazioni alle destinazioni
  - **Senza** controllo di presentazione:
    - Utente non controlla l'ordine con cui ricevere le informazione (radio)
  - **Con** controllo di presentazione: viceversa (televideo)

- Flusso informativo:
  - **Punto-Punto**: Trasferimento informativo 1 a 1 (posta elettronica)
  - **Punto-Multipunto**: Trasferimento informativo 1 a molti (sms a gruppi)
  - Diffusivo (**Broadcast**): Trasferimento informativo da 1 a tutti (radio)
  - **Monodirezionale**: Trasferimento informativo in una direzione (radio)
  - **Bidirezionale Simmetrico**: Uguale capacità per ogni direzione (telefonia)
  - **Bidirezionale Asimmetrico**: Diversa capacità per ogni direzione (ADSL)

- **Quality of Service(QoS):** qualità della comunicazio. percepita dall'utente del servizio
  - E' funzione della trasparenza della rete
- Servizi **isocroni**: serve trasparenza temporale per la corretta interpretazione dell'info
- **Trasparenza Semantica**:
  - Riguarda integrità delle info trasportate
  - Richiede di attuare procedure di recupero in caso di errore
- **Trasparenza Temporale**:
  - Riguarda variabilità dei ritardi di transito

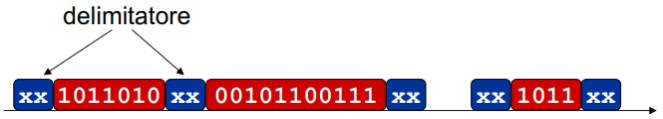
Multiplazione, Codifica e QoS

- **Multiplazione**: tecnica che permette di trasmettere più segnali su un unico canale di comunicazione, ottimizzando l'uso della banda disponibile
- Si evita di dedicare un canale separato per ogni comunicazione
- Il metodo più importante è il **TDM** (**Time Division Multiplexing**):

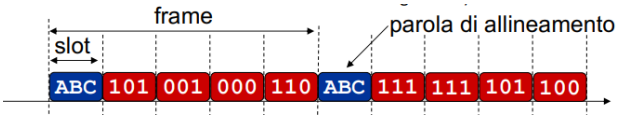


- **TDM Slotted:**
  - Asse dei tempi suddiviso in intervalli di durata prefissata (**slot**)
  - Unità informative hanno tutte stessa lunghezza commisurata allo slot

- **TDM Unslotted:**
  - Asse dei tempi non suddiviso a priori
  - Si possono adottare unità informative di lunghezza variabile dal delimitatore



- TDM Slotted **Framed**:
  - Slot strutturati in **frame**
  - Si sincronizza il frame e non il singolo slot



- TDM Slotted **Unframed**:
  - Si sincronizzano i singoli slot

- Assegnazione **Statica** della banda: (S-TDM)
    - La banda non può cambiare a comunicazione in corso
- Assegnazione **Dinamica** della banda: (A-TDM)
    - La banda può cambiare a comunicazione in corso

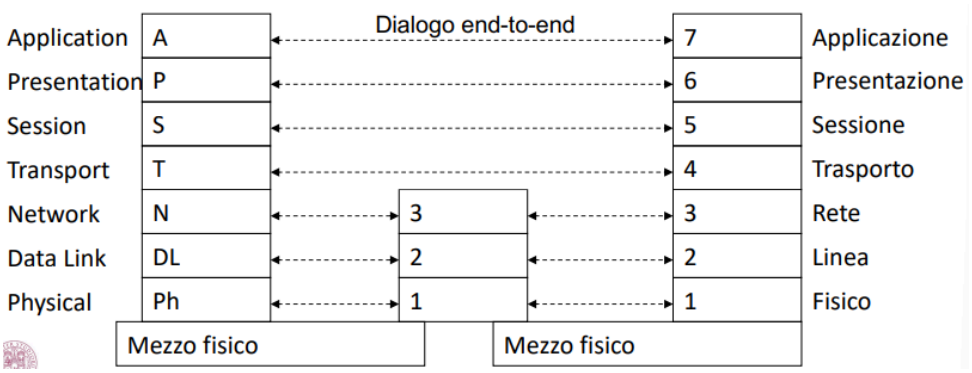
Commutazione

- Instradamento delle informazioni all'interno di una rete
- **Commutazione di Circuito** (rete telefonica):
    - Informazione analogica o digitale
    - Rete crea canale di comunicazione dedicato tra due terminali (**circuito**)
    - Circuito riservato ad uso esclusivo dei terminali (circuito **end-to-end**)
    - Esiste un ritardo iniziale dovuto all'instaurazione del circuito(**call set-up time**)
    - Alla fine della comunicazione si rilascia il circuito
    - Pro:
        - Circuito dedicato garantisce sicurezza e affidabilità
        - Trasparenza temporale
    - Contro:
        - Se sorgenti hanno basso tasso di attività il circuito è sottoutilizzato
        - Capacità del canale è fissata dalla capacità del circuito
- **Commutazione di pacchetto** (reti di calcolatori):
    - Informazione digitale
    - Messaggi suddivisi in sotto-blocchi (**pacchetti**)
    - Pacchetti trasmessi da un nodo di commutazione all'altro usando risorse comuni in tempi diversi
    - Tecniche:
        - Circuito Virtuale (**Connection Oriented**):
            - Prima viene stabilito il percorso dei pacchetti
            - Si associa al percorso un numero di circuito virtuale
            - I pacchetti contengono solamente il numero di circuito virtuale
            - Tutti i pacchetti seguono il percorso
        - Datagramma (**Connectionless**):
            - Ogni pacchetto gestito e instradato in modo indipendente
            - Ogni pacchetto porta tutte le info di indirizzamento utili per raggiungere la destinazione
            - Possono seguire percorsi differenti in momenti diversi
    - Pro:
        - Efficienza nell'utilizzo dei collegamenti
        - Facile controllo degli errori (trasparenza semantica)
    - Contro:
        - Difficile garantire predeterminato tempo di transito

ARCHITETTURA DI INTERNET E INDIRIZZAMENTO

Architettura delle Reti a Pacchetto

- **ISO-OSI**: standard per la realizzazione di reti di calcolatori aperte
- Architettura a 7 strati, dal basso verso l'alto da 1 a 7
    - 1, 2, 3 sono **network oriented layers**
    - 5, 6, 7 sono **application oriented layers**
    - 4 funge da tramite tra i due strati
    - Si possono avere funzioni di ripetizione (**relay**) ai livelli 1, 2, 3, che si dice operano **link-by-link**
    - Gli strati dal 4 in su operano solo **end-to-end**



Strato 1: **Fisico**:
- Specifica le modalità di invio dei singoli bit sul mezzo fisico di trasmissione

Strato 2: **Linea**:
- Rende affidabile il collegamento tra i nodi di rete
- Divide il flusso di dati in frame, controlla e gestisce errori di trasmissione

Strato 3: **Rete**:
- Si occupa della commutazione, nelle reti di calcolatori detto **routing**

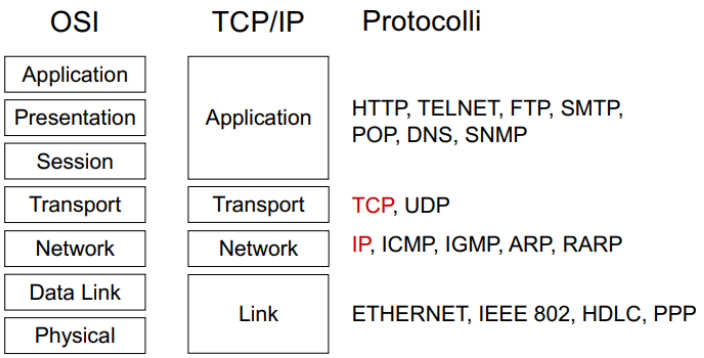
Strato 4: **Trasporto**:
- Fornisce canale end-to-end, adatta la dimensione dei file a quella dei pacchetti

Strato 5: **Sessione**:
- Suddivide il dialogo fra applicazioni in unità logiche chiamate **sessioni**

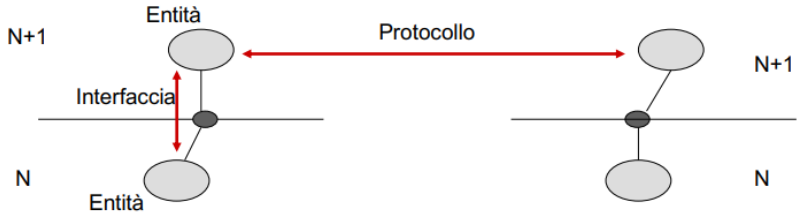
Strato 6: **Presentazione**:
- Adatta il formato dei dati preservandone il significato

Strato 7: **Applicazione**:
- Rappresenta l'applicazione, non deve fornire servizi a nessuno

Per creare rete universale serve che livello di trasporto e livello di rete siano unici.
OSI definisce protocolli: **IP** (Rete) e di Trasporto (**TCP**)
ISO/OSI modello teorico come guida generale, **TCP/IP** modello pratico usato su Internet

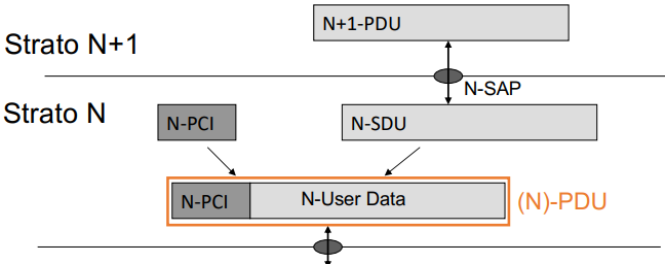


- **Entità**: ogni elemento attivo in uno strato, identificata da un **title** (nome simbolico)
- **Protocollo**: regole di dialogo tra entità dello stesso livello
- **Interfaccia**: regole di dialogo tra entità di livelli adiacenti



Trasferimento dei dati:

- **N-Protocol Data Unit** (**PDU**): dati trasferiti tra entità di strato N
- **N-Service Data Unit** (**SDU**): dati passati dallo strato N+1 allo strato N
- **N-Service Access Point** (**SAP**):indirizzo di identificazione di flusso dati tra N+1 ed N
- **N-Protocol Control Information** (**PCI**): informazioni aggiuntive per il controllo del dialogo a livello N
- **Encapsulation**: N-PDU = N-PCI + N-SDU
- Non è permesso connettere più N-user allo stesso N-SAP
- Possibile dividere una SDU in più PDU (Segmentazione e Riassemblamento)



Modalità di dialogo:

- **Confermato**: prevede conferma dal destinatario
- **Non Confermato**: non prevede conferma
- **Parzialmente Confermato**: richiesta confermata dal service-provider

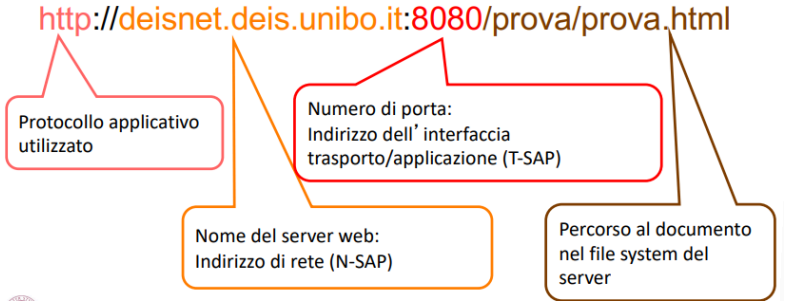
- **Multiplazione**:
    - Più connessioni di strato N mappate in una di strato N-1
    - Obiettivo: condividere risorse
- **Splitting**:
    - Inverso alla multiplazione, aumenta flessibilità e velocità trasferimento dati

Internet

- Utilizza come standard i protocolli TCP/IP
- I vari protocolli sono definiti in documenti chiamati **Request For Comment** (**RFC**)

Indirizzamento (meccanismo per identificare e raggiungere un entità in rete):

- **Identifier**: identificativo di una certa risorsa di rete
- **Locator**: indirizzo necessario per localizzare tale risorsa
- In internet:
    - Uniform Resource Name o **URN**
    - Uniform Resource Identifier o **URI**
    - Uniform Resource Locator o **URL** (orientato alla posizione)
- **Indirizzo Globale:**
    - Valido per tutta la rete, deve essere univoco
- **Indirizzo Locale:**
    - Valido in una sottoporzione della rete, può non essere univoco



Livello Data Link:

- Obiettivo: connettività locale
- MAC address: indirizzi locali nelle LAN
    - 48 bit (6 byte), nella scheda di rete, univoci nel mondo

Livello Network (IP):

- Obiettivo: connettività globale, ponte tra tecnologie locali
- Indirizzo IP:
    - 32 bit, sequenza di 4 numeri decimali da 0 a 255
    - Non identifica un host ma una delle sue interfacce di rete
    - Calcolatore con 3 interfacce ha bisogno di 3 indirizzi IP (importante)

Livello Trasporto (TCP):

- Obiettivo: garantire dialogo affidabile tra applicazioni, protocollo end-to-end
- Numero di porta:
    - 16 bit (valori decimali da 0 a 65535)
    - HTTP usa la 80, HTTPS la 443 e la SSH la 22

- Per identificare singolo flusso serve sapere:
    - IP sorgente e destinazione
    - Porta sorgente e destinazione

Livello Application (HTTP):
Server e Client:

- Server:
    - Applicazione che rende disponibile un servizio mediante interfaccia
- Client:
    - Applicazione che è in grado di utilizzare servizi del server
- Apertura:
    - Il server si predispone a ricevere una connessione facendo apertura passiva
    - Il client esegue apertura attiva, tenta di collegarsi al server
- Caratteristiche:
    - Modello uno (server) a molti (client)
    - Sincrono Bloccante:
        - Se server non risponde client non avanza
        - Si implementa nel client come reagire
    - Binding dinamico:
        - A ogni invocazione il client può scegliere il server
        - Se server non disponibile dove atteso, rete restituisce errore

Peer to Peer (P2P):

- Host in rete tutti equivalenti e fungono sia da client che da server

DOMAIN NAME SYSTEM (DNS)

- Per comodità degli utenti, ai numeri IP sono associati dei nomi simbolici (sequenza di stringhe alfanumeriche separate da punti)
- Porta Well Known 53
- Per eseguire ricerca degli indirizzi a partire dai nomi si usa il DNS:
    - Database distribuito che associa a ogni nome il relativo indirizzo di rete
- Consultazione avviene tramite server DNS (browser sa cosa fare per consultare il DNS senza doverlo chiedere all'utente finale)
- Servizio Whois del Registro .it per vedere se dominio .it già registrato ed a chi
- Il gestore di un dominio si occupa dei suoi sottodomini e questi non sono registrati
    - (.unibo.it si occupa di disi.unibo.it dei.unibo.it ecc…)

Composizione dei Nomi

deisnet.deis.unibo.it

- Componenti del nome riflettono l'organizzazione dei domini
    - it → dominio Italia; unibo → dominio università di Bologna ecc…
- Domini spesso suddivisi in sottodomini: unibo lo è di it, deis lo è di unibo
- Nomi dei domini assegnati da IANA

it → dominio di 1°livello; unibo → dominio di 2° livello; deis → dominio di 3°livello; deisnet → nome specifico dell'host entro il dominio deis

Name Resolver

- Programma del SO che converte un nome in numero IP, l'host ne deve essere equipaggiato

2 Casi:

- Name Resolver può risolvere il nome localmente (archivio locale, cache o file):
    - Comunica direttamente il numero IP all'applicazione
- Name Resolver non può risolvere il nome localmente:
    - Interroga il name server della zona a cui appartiene l'host
    - **Name Server** (.it, .com, .edu ecc…) della zona risolve il nome cooperando con server DNS di altre zone
        - Contatta prima il name server del dominio di 1°livello
        - Eventualmente gli altri

Risposta ricorsiva e iterativa:

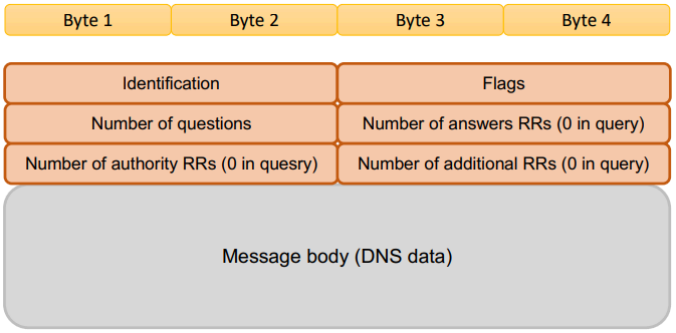
- **Ricorsiva**:
    - Name Server interrogato si preoccupa di risolvere il nome interrogando eventuali server di sotto-dominio e risponde alla richiesta
- **Iterativa**:
    - Name Server interrogato risponde indicando un name server di sottodominio a cui delega la risoluzione della richiesta

DNS Recursive e Authoritative:

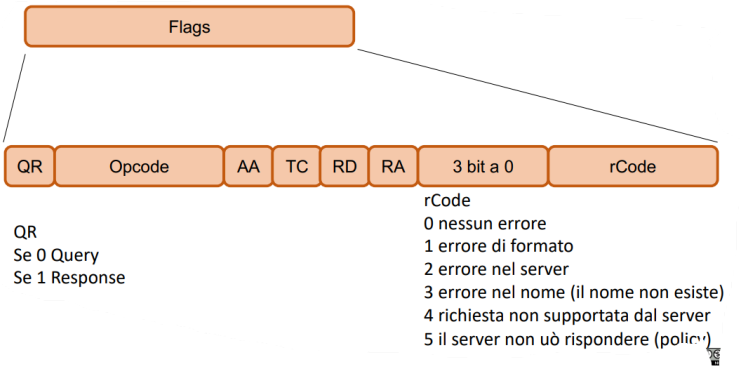
- Recursive:
  - Server che può produrre una risposta per la domanda
  - Primo server raggiungibile nel dominio
- Authoritative:
  - Server possessore di dominio, fornisce l'ultima parola su una risposta DNS
  - Massima affidabilità anche di prestazioni

Le PDU DNS:
Per il trasporto usa UDP

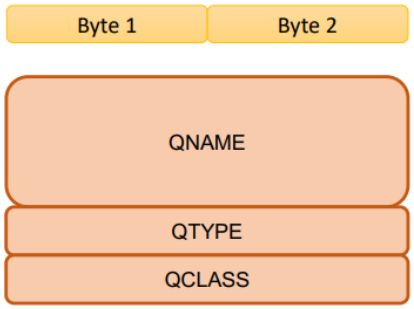
- Query:
  - Suddiviso in:
    - HEADER (PCI)
    - QUESTION (domande al server DNS)
- Response:
  - Suddiviso In:
    - HEADER (PCI)
    - QUESTION (copia delle domande della query)
    - Records
    - Answer records
    - Authoritative records
    - Additional records



- Campi delle PCI:
  - **Flags**: singoli bit a valore binario e piccoli gruppi di bit a più valori



- Formato della domanda:

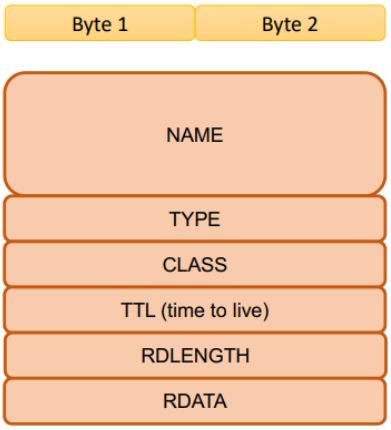


QNAME: nome di dominio per cui si effettua la richiesta

QTYPE: tipo della richiesta (codificato in due byte)

QCLASS: classe della domanda

- Formato della risposta:



NAME: nome di dominio per cui si effettua la richiesta

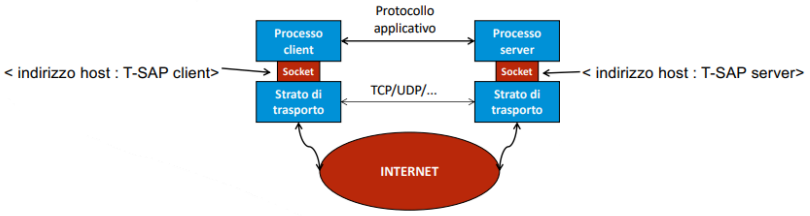
TYPE: tipo della risposta(significato del contenuto in **RDATA**)

TTL: durata in secondi del tempo per il quale la risposta può essere mantenuta in memoria

- Richieste e risposte utilizzano i **Resource Record** (RR):
  - Formato RR: (*name, value, type, ttl*)
  - Tipi di risposte da parte del DNS:
    - A: restituisce l'indirizzo IPv4 a 32 bit
    - NS: indica un server DNS autorevole
    - CNAME: collega un nome DNS a un altro
    - MX: collega un nome di dominio a una lista di server di posta autorevoli per quel dominio

PROTOCOLLI APPLICATIVI

- Protocolli utilizzati dalle applicazioni per scambiare informazioni in rete
- HTTP (web), SMTP (email), FTP (trasf. file), DNS (domini), SSH (accesso remoto)
- **Socket**:
    - Interfaccia che le applicazioni usano per interagire con i protocolli dello strato di trasporto
    - Fornita dal SO dell'host e accessibile tramite primitive
    - Rappresenta l'implementazione del T-SAP
    - Indirizzo socket: < indirizzo host : T-SAP > (< indirizzo IP : porta TCP/UDP >)



Primitive Berkeley Socket: (funzioni usate per comunicazione tra processi in rete)

- **Stream Socket**: connessioni affidabili e connection oriented (**TCP**)
    - Primitive usate dal processo server:
        - socket(): crea una nuova entità T-SAP
        - bind(): associa l'indirizzo (< indirizzo host : T-SAP >) alla socket creata
        - listen(): si mette in ascolto sulla socket creata
        - accept(): pone il server in attesa di accettare una richiesta da un client, a valle della quale crea un processo separato per gestirla (fork) e torna in ascolto sulla socket
        - send/receive(): trasmette/riceve dati sulla connessione stabilita
        - close(): chiude la connessione e rilascia l'indirizzo della socket
    - Primitive usate dal processo client:
        - socket(): crea una nuova entità T-SAP
        - connect(): blocca il processo client e tenta di aprire una connessione verso il server; sblocca il client a connessione instaurata
        - send/receive(): trasmette/riceve dati sulla connessione
        - close(): chiude la connessione e rilascia l'indirizzo della socket
- **Datagram Socket**: connessioni non affidabili e connectionless (**UDP**)
    - Primitive usate dal processo server:
        - socket(): crea una nuova entità T-SAP
        - bind(): associa l'indirizzo (< indirizzo host : T-SAP >) alla socket creata
        - sendto/recvfrom():trasmette/riceve dati da una socket remota specifica
        - close(): chiude la connessione e rilascia l'indirizzo della socket
    - Primitive usate dal processo client:
        - socket(): crea una nuova entità T-SAP
        - sendto/recvfrom():trasmette/riceve dati da una socket remota specifica
        - close(): chiude la connessione e rilascia l'indirizzo della socket

- Lo Stream Socket (TCP) può avere:
    - **Server Iterativo**:
        - Ciclo infinito permette al server di rispondere a più richieste di connessione successive, ma in sequenza
        - Nuova connessione non viene servita finché non termina il servizio di quella eventualmente in corso e di altre eventualmente già in attesa
    - **Server Concorrente**:
        - Ciclo infinito permette al server di rispondere a più richieste
        - Si genera un processo o un thread separato che gestisce ogni nuova connessione

Hyper-Text Transfer Protocol (HTTP)

- Protocollo utilizzato dal World Wide Web
- Client HTTP: browser web (chrome, safari)
- Server HTTP: Apache, Nginx
- Consultazione di contenuti web consiste nel richiedere l'invio di oggetti individuabili ciascuno attraverso un URL
- In WireShark, all'inizio si ha TCP three way handshake per creare la connessione TCP, Quindi ogni volta che si usa TCP ci sono 3 righe, SYN → SYN ACK → ACK

Formato dei messaggi:

- HTTP si basa su un dialogo di tipo **stateless**: non si tiene memoria delle transazioni
- Implementazione in modalità **stateful** grazie all'uso dei cookie
- I messaggi sono di due tipi: **richiesta** e **risposta**
- Ciascun messaggio è formato da un'intestazione (**header**) seguita dal corpo (**body**) del messaggio
- Header:
    - Intestazione composta da righe di testo terminate da CRLF
    - Richiesta inizia con riga di richiesta, poi una o più righe di intestazione
    - Risposta inizia con una riga di stato, seguita da una o più righe di intestazione
- Body:
    - Corpo contiene dati da trasferire (pagina HTML)
- Es: GET localhost:8080/index.html HTTP/1.1\r\n HTTP/1.1 200 OK\r\n

URL in HTTP:

- Solitamente un URL in HTTP è fatto così: **/<directory>/<file>?<query>#fragment**
    - **/<directory>/<file>** viene utilizzata per rappresentare la posizione "fisica" della risorsa nel filesystem del server
    - **<query>** è facoltativa. E' un dizionario/mappa (python, java) (var1=0&var2=1)
    - **#fragment**:
        - Informazione riservata ai client
        - Client non lo inviano mai al server, se lo fanno il server lo ignora
    - "#" usato per i segmenti e il server ignorerà ogni carattere successivo
    - "&" usato per separare variabili nel percorso URL

Codifica URL:

- Si antepone "**%**" davanti alla codifica esadecimale del carattere da codificare
- Ogni carattere riservato e non stampabile deve essere **urlencoded**
- Spazi rappresentati con **%20** o **+**

http://foobar.com/?var=hello's world → http://foobar.com/?var=hello+%26%23+world

Cookie:

- Per rendere HTTP stateful sono stati introdotti i **cookie**
- Informazioni di testo che identificano il browser nei confronti di un server
- Usati per: gestione delle sessioni, personalizzazione, monitoraggio ecc…
- Server HTTP possono impostare i cookie con il campo di intestazione della risposta **Set-Cookie**
- I cookie possono anche essere impostati lato client tramite JavaScript
- Composti da un **nome**, **valore** e **meta-informazioni**:
  - Origine
  - Data di scadenza
  - Politiche di sicurezza
- Se un cookie denominato "foo" viene impostato da "www.google.com", non può essere inviato a "www.microsoft.com", ma solo a "www.google.com"

File Transfer Protocol (FTP)

- Permette di lavorare con un file system remoto

Funzionamento:

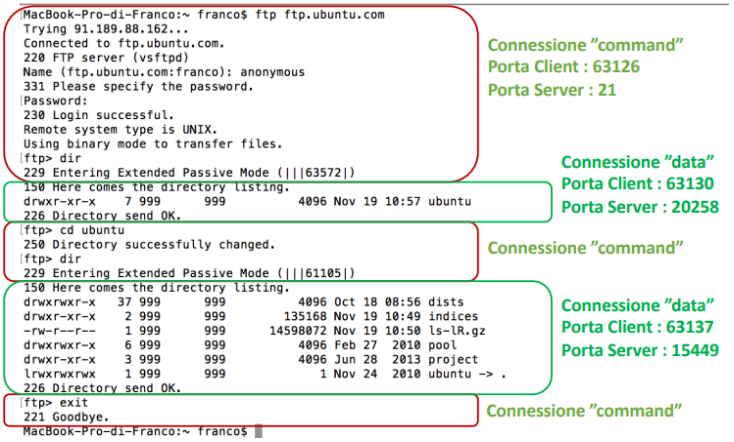
- Il client FTP contatta il server FTP alla porta 21, specificando TCP come protocollo di trasporto stabilendo la **connessione di controllo**
- Il client ottiene l'autorizzazione sulla connessione di controllo
- Il client cambia la directory remota inviando i comandi sulla connessione di controllo
- Quando il server riceve un comando per trasferire un file, apre una nuova connessione dati TCP con il client
- Dopo il trasferimento del file, il server chiude la connessione
- Per ogni richiesta di trasferimento, il server apre una connessione dati TCP verso il client

Sessione e Connessione:

- 2 entità colloquiano utilizzando una sessione di dialogo
- Singola sessione può includere numerose connessioni di trasporto contemporanee
- FTP usa 2 connessioni:
  - Connessione **command** (per usare una shell dei comandi)
  - Connessione **data** (per download / upload di file)

Active e Passive Mode:

- La connessione command viene aperta dal client che chiama il server sulla porta well known 21
- La connessione data:
    - **Active Mode**:
        - Server apre la connessione data
        - Se client schermato da firewall la richiesta di connessione viene bloccata
    - **Passive Mode**: client apre la connessione data dalla porta detta dal server



Posta Elettronica

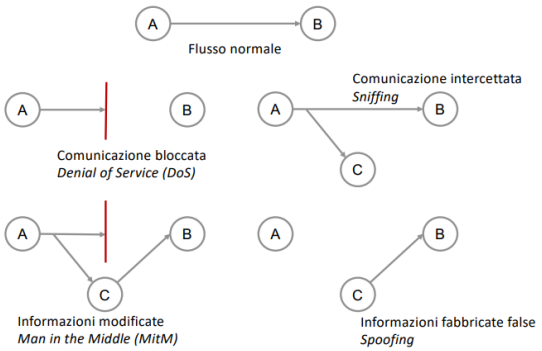
- Protocollo SMTP (Simple Mail Transfer Protocol) usato per trasmettere messaggi dal client al server o tra server
- Usa TCP nella porta 25
- Trasferimento diretto: server trasmittente contatta direttamente il server ricevente
- Pacchetto SMTP: Intestazione, Campi tipici(To, From, Subject) e message body
- 3 fasi per trasferimento:
    - Handshaking (saluto)
    - Trasferimento dei messaggi
    - Chiusura
- Messaggi devono essere in ASCII a 7 bit

Lettura e-mail (3 protocolli):

- **HTTP**: web mail (Gmail ecc…) applicazione eseguite tramite browser direttamente sul server
- **POP** (Post Office Protocol):
    - Modalità “**scarica e cancella**”:
        - Messaggi scaricati in locale e cancellati sul server
    - Modalità “**scarica e mantieni**”:
        - Messaggi scaricati in locale e rimangono anche sul server
        - Possibile copiare i messaggi su più client
        - Se cancelli messaggio su un client, gli altri client lo avranno ancora
    - Protocollo **stateless**: non so cosa si è fatto in precedenza

- **IMAP**:
    - Mantiene tutti i messaggi sul server
    - Conserva lo stato dell'utente tra le varie sessioni
    - Usa il **MIME** (**Multipurpose Internet Mail Extensions**):
        - Estensione di messaggi di posta multimediali
        - Contenuto multimediale codificato in testo
        - Nell'intestazione del pacchetto SMTP si dichiara il tipo di contenuto MIME così che il ricevente possa decodificarlo correttamente

HTTPS



Difendere l'informazione:

- **Integrità**: impedire la modifica non autorizzata delle informazioni
- **Riservatezza**: impedire l'accesso all'informazione da utenti non autorizzati
- **Disponibilità**: garantire la possibilità di usare le informazioni a chi è autorizzato
- **Paternità**: impedire a un utente di ripudiare un suo messaggio

Certificato Digitale:

- Chiave pubblica (usata per decodificare messaggi) pubblicata in un messaggio certificato da un'autorità, questo messaggio è il **certificato digitale**
- Redatti secondo lo **standard ITU X.509**
    - Numero identificativo, versione, Certification Authority, Algoritmi usati, Firma, Validità,

HTTPS:

- Autenticazione del server, cifratura delle transazioni HTTP
- Necessita di un ulteriore RTT per stabilire la sessione sicura tramite il protocollo **TLS (Transport Layer Security)**

VoIP & VoLTE

VoIP

- VoIP: telefonia digitale su rete IP
- Richiede:
  - Digitalizzazione e compressione del segnale vocale
  - Terminali e segnalazione connessi a rete IP
  - Rete IP per il trasporto

Realizzata in 2 modi:

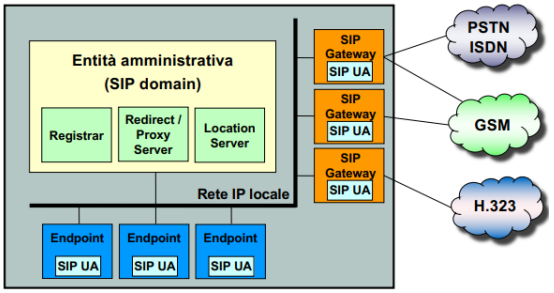
- IP Trunking:
  - Tecnologia IP (commutazione di pacchetto) su collegamenti della rete di trasporto
- Telefonia IP (attuale):
  - Tecnologia IP per fornitura del servizio di telefonia

Standard VoIP:

- Standard **ITU** (obsoleto)
- Standard **IETF** basati su protocollo **SIP**:
  - Struttura gerarchica a domini
  - Ogni organizzazione controlla il proprio dominio telefonico
  - Domini telefonici comunicano tramite rete IP
  - Integrato in IMS e parte del VoLTE
- Whatsapp, Skype ecc… basati su protocolli proprietari
  - Tecnologia P2P, non gerarchico, non adatto in azienda

SIP

Architettura:



Componenti principali:

- Entità Amministrativa (SIP Domain):
  - Gestisce routing delle chiamate SIP
- Rete IP Locale:
  - Contiene gli endpoint(dispositivi che possono avviare/ricevere chiamate VoIP)
- SIP Gateway:
  - Permette l'interconnessione tra la rete SIP e altre reti di telecomunicazione

Transazioni e Dialoghi:

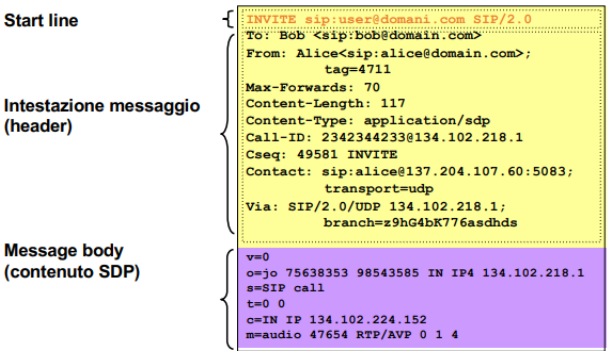
- SIP si basa su concetto di **Transazione** (sequenza di operazioni)
  - Ha un **Transaction ID** univoco con origine, destinazione, unique token…
  - Completamento della Transaction è indipendente da altre comunicazioni
- Entrambi gli endpoint creano **transaction state** e alla fine entrambi la distruggono
- Un endpoint fa una **Request Iniziale** a un altro e attende la **Final Response**
- L'altro può dare delle **Provisional Responses** (risposte intermedie)
  - Con informazioni aggiuntive, inaffidabili

- **Dialogo**: sessione di comunicazione tra endpoint
  - **Signaling Session**: controlla comunicazione (creare, terminare sessione)
  - **Media Session**: flusso di dati effettivo
- Gli endpoint mantengono uno "**stato**" della comunicazione (se attivo, in modifica…)
- Una Transaction indica un cambiamento di stato, nessun cambiamento se c'è errore
- Ogni dialogo è identificato da **Unique Dialog Identifier**

Indirizzamento in SIP:

- Segue la base degli URI
- Separazione tra **Naming** (permanente) e **indirizzi di localizzazione** (temporaneo)
  - Consente la mobilità
- 2 ruoli del SIP URI:
  - Definire nominalmente un utente (Naming)
    sip:user:password@host:port;uri-parameters?headers
  - Fornire info per contattare un utente (indirizzo IP, numero porta, protocollo…)
    sip:bob@137.204.57.10

Sintassi Richiesta:



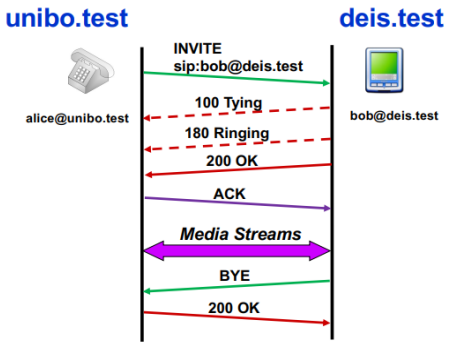
- Start line → **Request line**:
  - <Method> <Request-URI> <Versione protocollo SIP> <CRLF>
- **Header**: specifica intestazioni del messaggio, transaction, dialog …
- **Body**: contenuto messaggio SIP
- Il **Method** è il tipo di messaggio che si intende inviare:
  - INVITE: avvia una chiamata (crea dialogo)
  - ACK: conferma di un messaggio ricevuto
  - BYE: termina chiamata
  - CANCEL: stato di ringing
  - REGISTER: registra presso un server
  - UPDATE: aggiorna dialogo non confermato

Sintassi Risposta:

- Start line → **Status line**:
  - <Versione SIP> <Status Code> <Indicazione Ragione> <CRLF>
- **Header**: specifica intestazioni del messaggio, transaction, dialog …
- **Body**: contenuto messaggio SIP (generalmente omesso)
- **Status Code**:
  - Risultato del tentativo di interpretare e soddisfare la richiesta
  - Composto da 3 cifre decimali:
    - 1xx Provisional: in ricerca, squillo… (100 Trying, 180 ringing)
    - 2xx Success (200 OK)
    - 3xx Redirection: forwarding (302 Moved temporarily)
    - 4xx Client Error: (404 User not found)
    - 5xx Server Error: (500 Internal Server Error)
    - 6xx Global Failure: occupato, rifiutato… (603 Decline)

    - Tipi di risposte:
        - **Provvisorie**: non terminano transaction (1xx)
        - **Definitive**: terminano transaction (2, 6 chiudono/avviano dialog, gli altri danno info provvisorie)
- **Indicazione Ragione** (Reason-Phrase): descrizione testuale del codice di stato

Chiamata Diretta:



Il caller conosce il nome host o l'indirizzo IP del callee

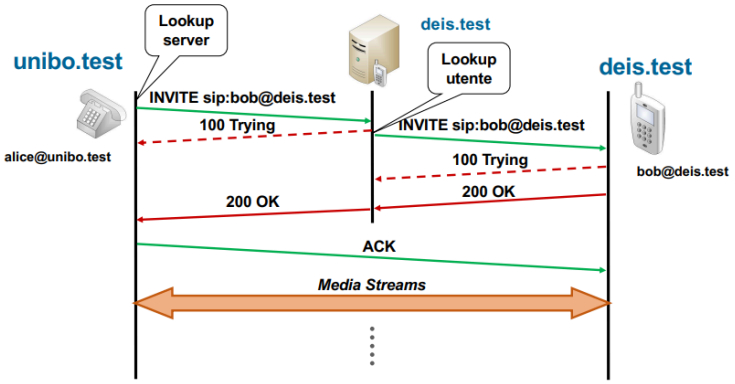
L'UA destinatario riporta i cambiamenti di stato

Quando Bob accetta la chiamata viene inviato l'OK

L'UA chiamante conferma, la chiamata è stabilita

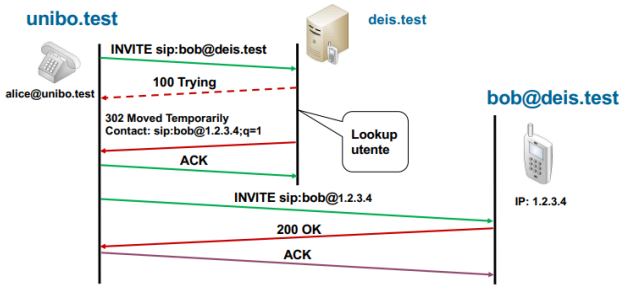
Si scambiano info multimediali
La chiamata è terminata da uno dei due

Proxied Call:



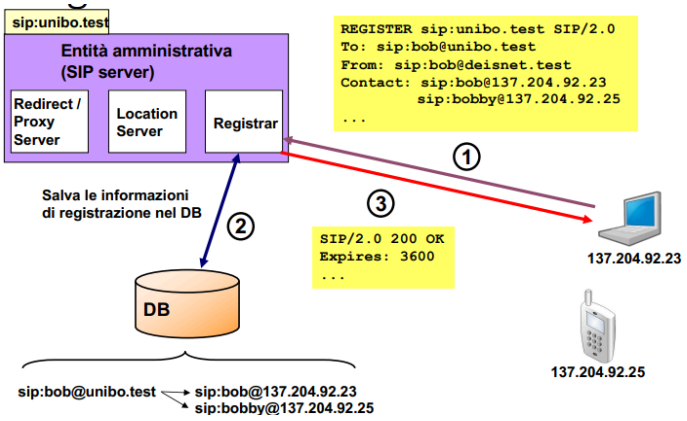
- Chiamata che avviene per mezzo di un server intermediario, perché per esempio non si sa se gli endpoint riescono a comunicare tra loro fisicamente
- In seguito, il server può dire ad alice il nome di Bob e viceversa e possono instaurare chiamata diretta

Redirected Call:



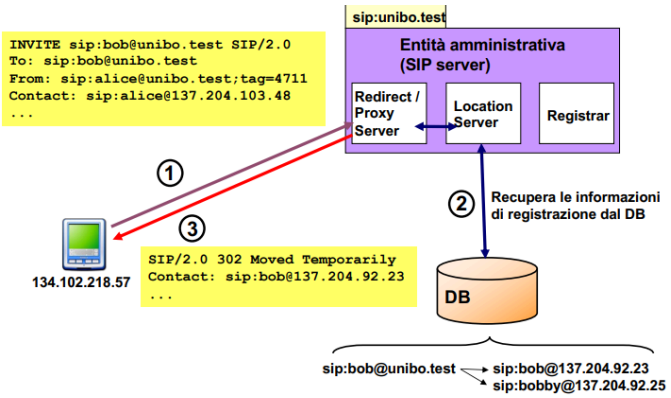
- Bob si sposta e manda un messaggio **redirect** al server che lo comunica ad alice che ha cambiato indirizzo
- Ora Alice instaura obbligatoriamente una chiamata diretta con Bob

User Registration:



- User effettua operazione di REGISTER al Registrar del SIP domain
- SIP domain salva le info di registrazione nel database
- Il Registrar effettua operazione di OK allo user
- Nell'esempio: bob dice chiamami al computer o al cellulare, ha salvato due indirizzi fisici, sip:bob@unibo.test è un indirizzo logico

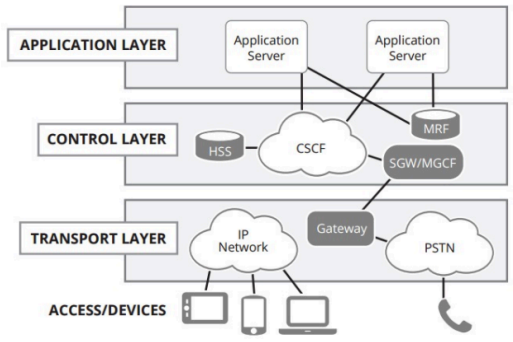
User Location:



- Utente INVITA a chiamata un indirizzo logico SIP tramite Redirect / Proxy server
- Proxy chiede a Location Server le info di registrazione nel DB
- Poi al contrario, le info vanno date allo user che ha invitato

IP Multimedia Sub-system (moderne reti mobili):

- Architettura funzionale per realizzare servizi multimediali su reti IP
- Implementazione della teoria precedente



Control Layer equivale al SIP domain
HSS è il database di prima
CSCF comprende il proxy, redirect, location e registrar

VoLTE terminal:

- **ISIM** (nella SIM card): identificativo
    - IP Multimedia Private Identity (**IMPI**)
        - Identità privata che contiene info sull'operatore con cui si ha contratto
    - IP Multimedia Public Identity (**IMPU**)
        - Identità pubblica per essere raggiunti da altri
        - Stessa IMPU può essere attribuita a terminali diversi
- **SIP UE** (nello smartphone): lo User Agent
    - Gestisce segnalazione SIP

Home Subscriber Server (**HSS**):

- Database contenente info per gestire chiamate

Call Session Control Function (**CSCF**):

- Proxy-CSCF (**P-CSCF**) equivale al proxy di prima: (forse nella visited network)
    - Primo punto di contatto, redirige opportunamente i messaggi SIP
- Interrogating-CSCF (**I-CSCF**) equivale al Location server di prima:
    - Assegna alle richieste il relativo S-CSCF
- Serving-CSCF (**S-CSCF**) equivale al Registrar: (sempre nella home network)
    - Gestisce registrazione degli utenti e comunicazione con HSS

CONTROLLO DEL CANALE

Controllo del Canale al livello 2

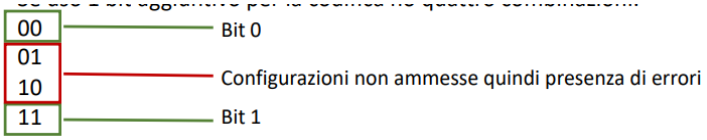
- Funzioni svolte dall strato 2 per il controllo del canale:
  - Strutturazione del flusso dati: PDU strato 2 sono detti frame
  - Controllo e **gestione degli errori**
  - Controllo di flusso, di sequenza, gestire protocollo di accesso per collegamento punto-multipunto

Controllo dell'Errore

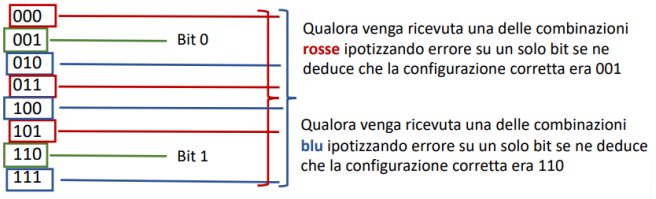
- Ci si riferisce a **codici a blocco**: si applica codifica a blocchi di k bit
- Vengono calcolati r bit di ridondanza e trasmessi n = k + r bit
- Rivelazione di un errore costa meno bit della correzione

Esempi:

- Trasmetto 1 bit
- Se uso 1 bit aggiuntivo per la codifica ho 4 combinazioni
- Nell'ipotesi che l'errore possa coinvolgere un solo bit , se ricevo 10 o 01 so che il messaggio è stato modificato e si è verificato un errore ma … quale era la configurazione iniziale? Posso rilevare ma non posso correggere gli errori



- Se uso 2 bit aggiuntivi ho 8 combinazioni:
- Nell'ipotesi che l'errore possa coinvolgere un solo bit è possibile rilevare l'errore e risalire alla corretta sequenza di bit, correggendo l'errore stesso

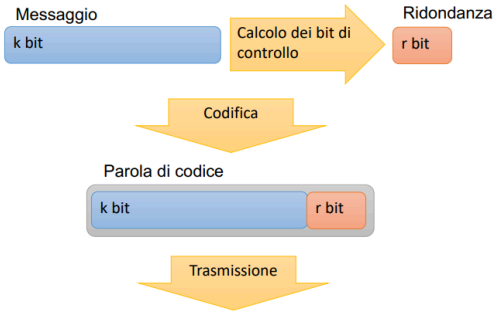


In generale si sceglie in base al canale trasmissivo:

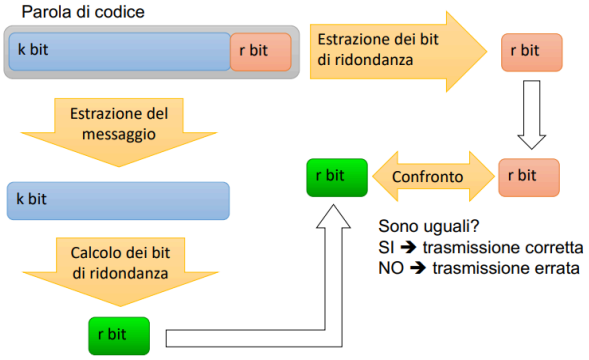
- **Correzione di errore (FEC):**
  - Richiede numero elevato di bit aggiuntivi
  - Permette correzione dei dati errati in base ai soli dati ricevuti
- **Rivelazione di errore:**
  - Richiede un numero limitato di bit aggiuntivi
  - Rende necessaria ritrasmissione dei dati errati
- Conviene rivelazione se canale è affidabile con pochi errori e viceversa
- Solitamente si usano codici a correzione di errore nello strato fisico e rivelazione dei livelli di linea e trasporto

Rivelazione d'errore:

- Si usano **codici sistematici**: Se trasmetto n bit, k bit sono info e r di ridondanza
- K e r bit vengono mantenuti distinti
- In trasmissione:

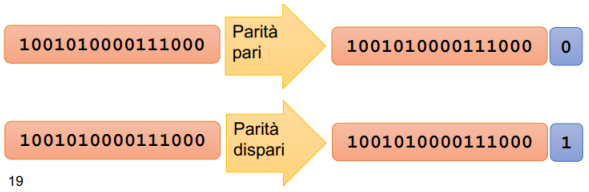


- In ricezione:



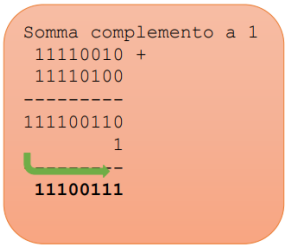
Bit di parità:

- Dati k bit di info b_0, b_1, …, b_{k-1}
- Xor dà 1 se bit diversi, 0 se uguali
- Abbiamo due tipi di parità:
  - Pari: $b_k = b_0 \oplus b_1 \oplus b_2 \oplus \ldots \oplus b_{k-1}$
  - Dispari: $b_k = NOT\,[b_0 \oplus b_1 \oplus \ldots \oplus b_{k-1}]$
- Rivela sempre un numero dispari di errori, fallisce con errori in numero pari
- Se r = 1:

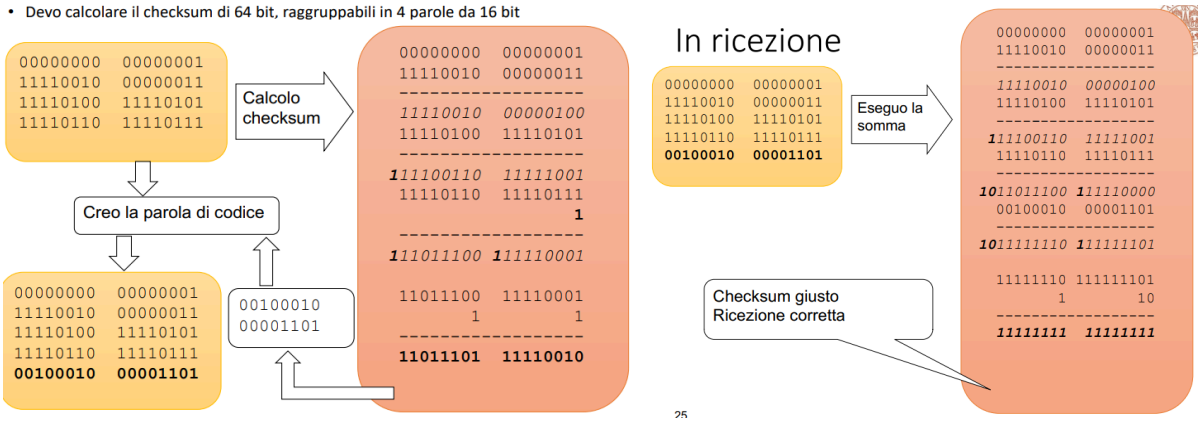


Internet Checksum:

- Meccanismo definito in RFC che dice come costruire un blocco di bit per rivelazione d'errore nella rete internet su parole di 16 bit
- Usa la somma a complemento a 1: se somma genera riporto, si somma al risultato



- Proprietà (blocco dati di 1 byte: A, B, C… Parole di 16 bit [A, B], [C, D]):
  - Commutativa, associativa
  - Indipendenza dall'ordine dei byte: rende calcolo indipendente da "big-endian" "little-endian"
- Esempio:



Codici Polinomiali:

- Utilizzati per rivelare e correggere errori nella trasmissione dati
- k bit posti in corrispondenza con polinomio di grado k-1

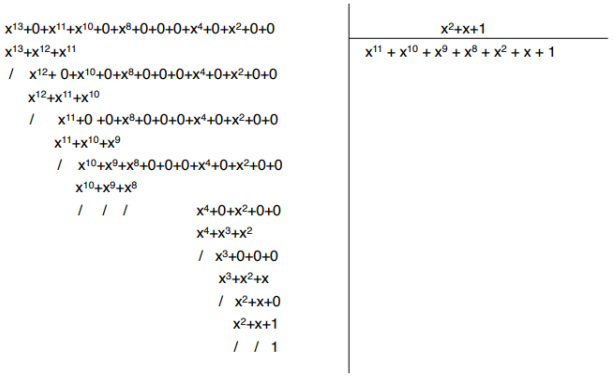
$$P_{k-1}(x) = b_0 + b_1x + b_2x^2 + \ldots + b_{k-1}x^{k-1}$$

- Polinomio **generatore**:
  - Polinomio di grado r noto a trasmettitore e ricevitore
  - $G_r(x)$ determina la proprietà di rivelazione del codice
- Polinomio **trasmesso** $T_{n-1}(x)$:
  - Si moltiplica il polinomio $P_{k-1}(x)$ per x^r (si spostano i bit a sinistra per fare spazio alla ridondanza)
  - Si esegue divisione polinomiale tra $P_{k-1}(x)\ x^r$ e $G_r(x)$ ottenendo un quoziente e un resto: $P_{k-1}(x)\ x^r = G_r(x)\ Q_{k-1}(x) \oplus R_{r-1}(x)$
  - Si trasmette: $T_{n-1}(x) = P_{k-1}(x)\ x^r \oplus R_{r-1}(x) = G_r(x)\ Q_{k-1}(x)$
  - $T_{n-1}(x)$ è divisibile per G(x)

- Polinomio **ricevuto**:
  - Ricevitore riceve un polinomio T' che può contenere errori
  - Se si verifica errore:
    - T'(x) = T(x) + E(x) con E(x) polinomio di errore
    - T'(x) ≠ T(x)
  - E(x) ha coefficienti non nulli in corrispondenza dei bit in cui T' differisce da T
  - Ricevitore esegue la divisione T'(x) / G(x) → (T(x) + E(x)) / G(x)
  - Si ha T(x)/G(x) che dà sempre resto 0 e E(x)/G(x)
  - Per rilevare errori si deve evitare che Resto[E(x) / G(x)] = 0
- Scelta di G(x):
  - Un solo errore: $E(x) = x^i$
    - Rilevato se G(x) ha almeno due bit a 1
  - Numero dispari di errori:
    - Se G(x) è multiplo di (x + 1) riesce a rilevarli
  - Due errori: $E(x) = x^i + x^j = x^j(x^h + 1)$
    - Esistono diversi polinomi capaci di rilevarli
    - ITU ha proposto il seguente: $G_{16}(x) = x^{16} + x^{12} + x^5 + 1$
- Errori a burst:
  - Nelle reti di gli errori non sono casuali ma tendono a raggrupparsi (**Burst**)
  - Burst di bit è lungo k e rappresentato da polinomio di grado k - 1
  - Si possono avere i seguenti casi:
    - k - 1 < r → errore viene sempre rilevato
    - k - 1 = r → si ha resto nullo se E(x) = G(x) con prob. di $1/2^{r-1}$
    - k - 1 > r → resto è casuale e l'errore sfugge se resto è nullo $1/2^r$

Procedimento esercizi:

- Si vuole trasmettere da S a D la stringa di bit: 101101000101
- $P(x) = x^{11} + 0 + x^9 + x^8 + 0 + x^6 + 0 + 0 + 0 + x^2 + 0 + 1$
- Polinomio generatore: $G(x) = x^2 + x + 1$ → $r = 2$
- Poiché r = 2, moltiplichiamo P(x) per x^2
  - $P(x)x^2 = x^{13} + 0 + x^{11} + x^{10} + 0 + x^8 + 0 + 0 + 0 + x^4 + 0 + x^2 + 0 + 0$
- Divido $P(x)x^2$ per G(x) per ottenere $T_{n-1}(x)$:
  - Divido la x con il grado massimo del dividendo con quello di grado max del divisore e scrivo il risultato sotto il divisore
  - Moltiplico il risultato per ogni termine del divisore e scrivo i risultati a sinistra, sotto i termini di quel grado. Se è presente lo elimino se no lo aggiungo
  - Mi fermo quando il resto ha grado inferiore al divisore
  - Invece sotto al divisore è il quoziente



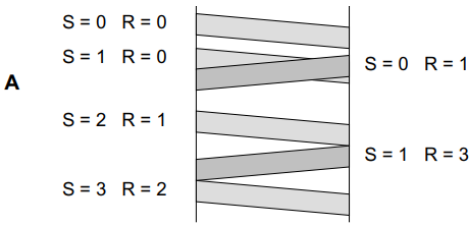
- Ottengo R(x) = 1 e Q(x) = $x^{11} + x^{10} + x^9 + x^8 + x^2 + x + 1$
- Trasmettiamo T(x) = $x^{13} + 0 + x^{11} + x^{10} + 0 + x^8 + 0 + 0 + 0 + x^4 + 0 + x^2 + 0 + 1$
  - Ho fatto lo XOR tra $P(x)x^2$ e R(x)
  - Sequenza di bit da trasmettere: 10110100010101
- Consideriamo i seguenti E(x):
  - $E(x) = x^9 + x^8$
  - $E(x) = x^4 + x^3 + x^2$
- T'(x) = T(x) + E(x) = $x^{13} + 0 + x^{11} + x^{10} + x^9 + 0 + 0 + 0 + 0 + x^4 + 0 + x^2 + 0 + 1$
- Per verificare se l'errore viene rilevato divido T'(x) per G(x)
  - Con il 1° E(x) il resto della divisione R(x) ≠ 0 quindi errore rilevato
  - Con il 2°, il resto R(x) = 0, quindi errore NON rilevato

Automatic Repeat reQuest (ARQ)

- Protocolli utilizzati nello strato di linea ed in quello di trasporto in sinergia con una codifica a rivelazione di errore
- Obiettivo: rendere affidabile il canale
  - Identifica errori di trasmissione e innesca ritrasmissione
- Funzioni di controllo: di errore, di flusso, di sequenza
- Meccanismo utilizzato è quello della **numerazione a finestra scorrevole**

Numerazione:

- Prot. ARQ numerano sequenzialmente le unità informative da dare ai prot. superiori
  - Le PDU e le UI standard (bit)
- Trasmettitore e ricevitore mantengono due contatori:
  - **S** conta in modo sequenziale le UI **inviate**
  - **R** conta le UI **ricevute** in <u>modo corretto</u>
- S permette posizionamento nel flusso e R di confermare ricezione



canale full-duplex
(bidirezionale asimmetrico)

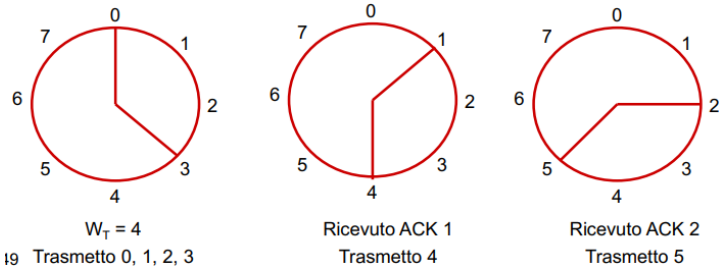
- Controllo errori:
  - Alle PDU applicata una codifica di canale
  - Ricevitore:
    - Verifica correttezza delle PDU grazie a rivelazione d'errore
    - Ignora PDU errate
    - Può far partire le procedure di ritrasmissione
  - Trasmettitore:
    - Ritrasmette frame non ricevuti correttamente (su indicazione o a time-out)
- Time out: Un orologio parte al termine della trasmissione di ciascun frame. Se si raggiunge il time out senza avere conferma si ritrasmette il frame

Conferma (**Acknowledge**):

- Corretta ricezione confermata dal ricevitore, invia al trasmettitore il proprio valore R
- Acknowledge può essere:
  - **Esplicita**: Ogni PDU ricevuta correttamente genera una conferma
  - **Implicita**: una PDU di conferma con R = n conferma la ricezione fino a n-1
  - **Piggybacking**: viaggia inserita in una PDU
- **ACK** sono pacchetti PDU che portano solo info di controllo per il protocollo
- ACK pacchetto trasporta il contatore R dal ricevitore al trasmettitore
- Conferma ricezione del pacchetto n inviando il numero n+1 al trasmettitore

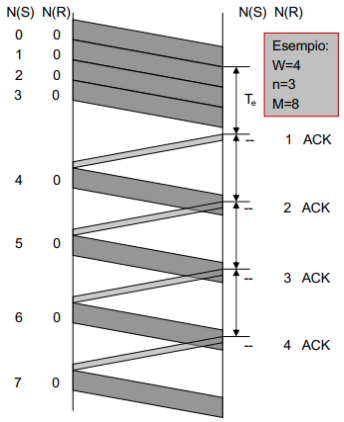
Finestra di trasmissione:

- **W_T**=numero massimo di frame che trasmettitore può inviare senza ricevere conferma
- Numerazione frame fatta in modulo M = 2^n dove n sono i bit usati per numerazione
- Si procede con trasmissione di nuovi frame solo al ricevimento delle conferme
  - Numerazione dei frame trasmessi scorre nel tempo (**sliding window**)
- Imporre W finito e sospendere trasmissione dei frame garantisce:
  - Unicità di numerazione: se si trasmettesse all'infinito i frame con numero uguale sarebbero indistinguibili
  - Protocollo che si auto-adatta alla velocità del canale / ricevitore/ trasmettitore

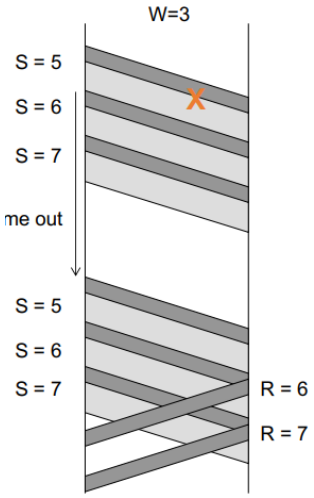


Controllo di flusso:

- Accorda velocità del trasmettitore alla capacità del ricevitore
- Ricevitore:
  - In grado di gestire una finestra: memorizzazione e elaborazione di W frame
  - Accorda flusso di frame in arrivo tramite conferme
- Un nuovo frame ogni **T_e** = tempo necessario per elaborare una trama



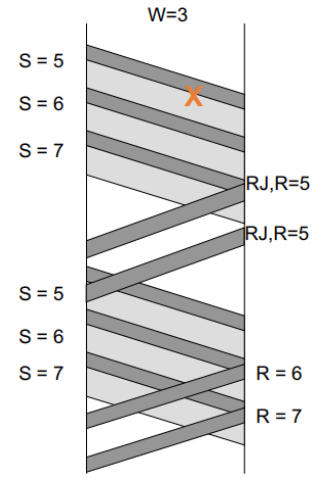
Recupero dell’errore: Go - Back - n ARQ (metodo attuale)



Viene perso il frame N
Il ricevitore:
- Scarta tutti i frame successivi
- O segnala mancata ricezione o rimane in silenzio

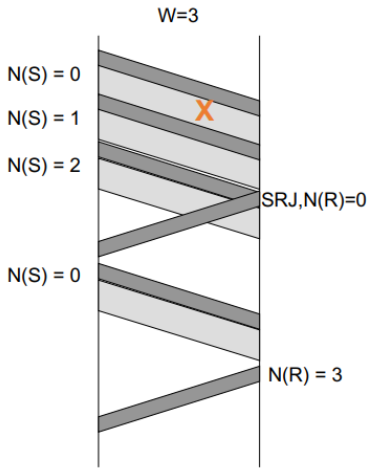
Trasmettitore ritrasmette tutti i frame a partire dall’N
Vantaggi:
- Semplice

Svantaggi:
- Inefficiente



CON SEGNALAZIONE

Selective Repeat ARQ:



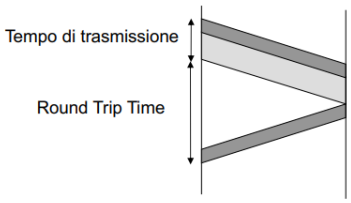
Viene perso il frame N
Il ricevitore:
- Scarta solo frame errato
- Segnala mancata ricezione del frame N

Trasmettitore ritrasmette solo frame N
Vantaggi:
- Maggiore efficienza

Svantaggi:
- Complessità del ricevitore

Round Trip Time (RTT):

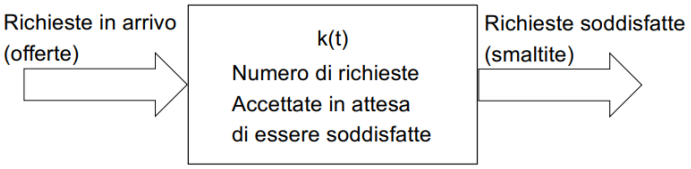
- Tempo necessario per effettuare un’andata e ritorno sul canale:
    - Tempo trascorso tra partenza dell’ultimo bit di un frame e ricezione del relativo ACK



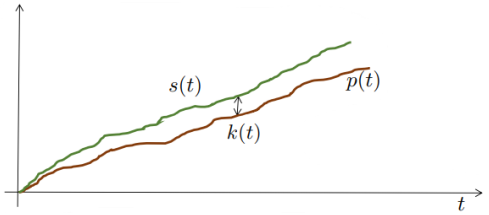
- Time Out deve essere poco più grande del RTT
- Time Out deve essere poco più grande del doppio del tempo di percorrenza
- Time Out troppo breve: non si attende arrivo di ACK e invio inutile di duplicati
- Time Out troppo lungo: Inutile attesa prima di ritrasmettere frame errati

PROTOCOLLI: PRESTAZIONI

- Protocolli progettati per garantire:
  - Funzionalità: trasmissione deve avvenire risolvendo problemi del canale
  - Prestazioni: trasmissione deve avvenire con successo usando la capacità dello strato fisico

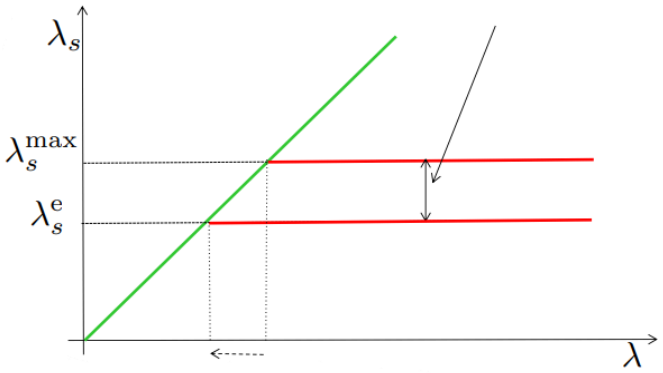


- k(t) = s(t) - p(t)
  - s: richieste accettate al tempo t; p: partenze dal sistema al tempo t; a: richieste di servizio giunte al tempo t



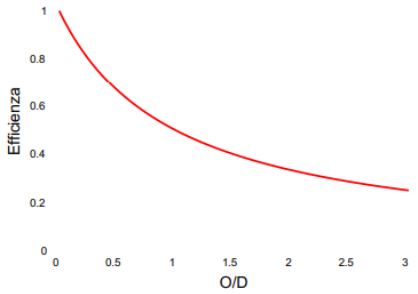
- Frequenza media delle richieste offerte: $\lambda = \lim_{t \to \infty} \frac{a(t)}{t}$
- Frequenza media delle richieste smaltite $\lambda_s = \lim_{t \to \infty} \frac{p(t)}{t}$
- Se il sistema non produce lavoro ma lo riceve solo dall'esterno: $\lambda_s \leq \lambda$
- $\lambda_s = \lambda$ significa a(t) = s(t)
- $\lambda_s < \lambda$ significa r(t) = a(t) - s(t) dove r rappresenta le richieste che non vengono accettate e sono rifiutate o perdute dal sistema
- $\lambda_p = \lim_{t \to \infty} \frac{r(t)}{t}$ da cui consegue $\lambda = \lambda_s + \lambda_p$
- In una rete il riferimento è il tempo di servizio dell'intero pacchetto che solo se completato produce un risultato "utile" per l'utente
- Non si considera il semplice bit o il bit rate
- θ Tempo richiesto dal servizio di un generico cliente (pacchetto dati, PDU)
  - Servizio aleatorio: tempo di servizio varia casualmente e si fa riferimento al valore medio
  - Servizio deterministico: tempo di servizio costante ed uguale al valore medio
- $\theta = \frac{L}{C}$ dove L lunghezza del pacchetto in bit, C capacità del canale in bit al secondo
- Frequenza media di servizio $\mu = \frac{1}{\theta}$:
  - Inverso del tempo medio di servizio
  - Legata alla presenza di utenti del sistema, se non vi sono richieste la frequenza di servizio è nulla
  - Interpretato come capacità massima del servitore
  - Se θ = 0.5 s il servitore smaltira al max. μ = 2 pacchetti/s $\rightarrow \mu = \lambda_s^{max}$

- In un sistema a coda l'utente permane per un tempo che tiene conto dell'attesa in coda e del tempo di esecuzione del servizio
    - δ tempo medio totale speso dall'utente nel sistema a coda composto da:
        - θ tempo effettivo di servizio
        - T_A tempo speso in coda, tempo di attesa prima di essere servito
    - $\delta = \theta + T_A$
- Traffico: numero medio di utenti presenti nel sistema
    - Teorema di Little: $A = \lambda\delta$
    - Prodotto tra frequenza di arrivo e tempo medio di permanenza dà il traffico
    - $A_0 = \lambda\theta$ traffico offerto (occupazione media di un sistema ideale che serve subito tutti gli utenti senza attesa)
    - $A_s = \lambda_s\theta$ Traffico smaltito (occupazione media dei servitori del sistema)
    - $A_p = \lambda_p\theta$ Traffico perduto (occupazione media di un sistema che serve gli utenti che invece sono stati rifiutati)
    - Se server sono m allora $0 \leq A_s \leq m$ chiamato *throughput* indica quanta capacità di servizio viene usata dal sistema
    - Si misura in Erlang
    - Vedere esempi nelle slide 18-19 delle PRESTAZIONI
- Efficienza del protocollo: $\eta = \frac{\lambda_s^e}{\lambda_s^{max}} \leq 1$

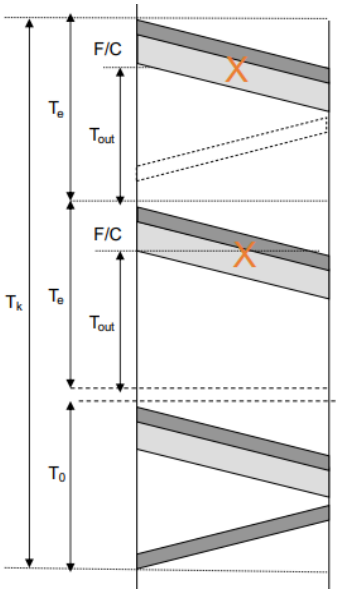


Efficienza dei Protocolli ARQ

- Tempo intercorso tra invio di due frame successivi:
  - $T_0 = \frac{F}{C} + I + \frac{A}{C'} + I'$
  - F: lunghezza frame, C: capacità canale di andata, I: tempo di andata del frame, A: lunghezza ACK, C': capacità canale ritorno, I': tempo di ritorno
- Tempo necessario per trasmissione dei dati utente:
  - $T_d = \frac{D}{C}$
  - D: dimensione dei dati da trasmettere in bit, C: capacità canale
- **Efficienza**:
  - $\eta = \frac{T_d}{T_0} = \frac{D}{D+2H+2IC} = \frac{D}{D+O}$ **Overhead**: $O = 2H + 2IC$
  - Viene così perchè per semplicità poniamo I = I' e C = C' e A = H poichè ACK composto quasi solo da PCI
  - Efficienza diminuisce al crescere di H (molti bit per PCI), C (linea molto veloce) e I (grandi distanze)
- Overhead:
  - Quantità di dati aggiuntivi introdotti dal protocollo
  - Grandezza in bit
  - Efficienza diminuisce al crescere di O
  - Tiene conto delle PCI e del tempo non usato dalla trasmissione per ARQ



Caso con Errore:



Prima di trasmettere bene una trama possono avvenire k errori
Tempo per trasmettere trama dati k errori: $T_k = kT_e + T_0$

Dove $T_0 = \frac{D+O}{C}$ e $T_e = \frac{D}{C} + \frac{H}{C} + T_{out}$

Se P_k è la prob. di avere k errori, il tempo medio per trasmettere una trama vale $E[T_k] = \sum_{k=0\ldots\infty} T_k P_k = \sum_{k=0\ldots\infty} (kT_e + T_0)P_k$

P_F = prob. di ricevere una trama errata

Numero medio di errori consecutivi: $E[k] = \frac{P_F}{1-P_F}$

P_e = prob. di errore per bit

$E[T_k] = T_0 + T_e \frac{P_F}{1-P_F}$

- Se bit indipendenti: P_F = 1 - prob{trama corretta} = DP_e
- Nelle reti telefoniche gli errori sono a burst: $P_F = \alpha F^{\beta}$ con $\beta > 1$
- In conclusione:
  - $\eta = \frac{D}{\frac{C}{E[T_k]}}$
  - $T_0 = \frac{D+O}{C}$
  - $T_{out} = 2I + \frac{H}{C}$
  - $T_e = \frac{D}{C} + \frac{O}{C}$
  - Efficienza massima: $\eta_{max} = \frac{D}{D+O+D^2P_e}$
  - Dimensione dei dati ottima: $D_{ott} = \sqrt{\frac{O}{P_e}}$
  - Efficienza Ottima: $\eta_{ott} = \frac{D_{ott}}{D_{ott}+2O}$

Caso con Finestra W > 1

- Caso senza errori:
  - Se $WF < CT_0 \quad \rightarrow \quad \eta = \frac{WD}{D+2H+2IC}$
  - Se $WF > CT_0 \quad \rightarrow \quad \eta = \frac{D}{D+H}$
- Caso con errori:
  - Caso Selective Repeat ARQ:

$$\eta = \frac{D}{D+H+D^2P_e} \qquad D_{ott} = \sqrt{\frac{H}{P_e}} \qquad \eta_{ott} - \frac{D_{ott}}{D_{ott}+2H}$$

  - Caso Go-Back-N:

$$\eta = \frac{D}{D+H+wD^2P_e} \qquad D_{ott} = \sqrt{\frac{H}{wP_e}} \qquad \eta_{ott} = \frac{D_{ott}}{D_{ott}+2H}$$

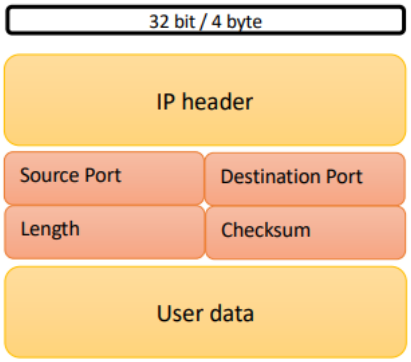
GUARDARE ESERCIZI NELLE SLIDE

PROTOCOLLI DI TRASPORTO

- 2 protocolli principali: TCP (connection oriented), UDP (connectionless)
- Strato di trasporto consente multiplazione

User Datagram Protocol

- Protocollo **connectionless** (no connessione, ogni pacchetto indipendente dagli altri)
- Pensato per invio di pochi dati e non richiede controllo qualità del trasporto(DNS)

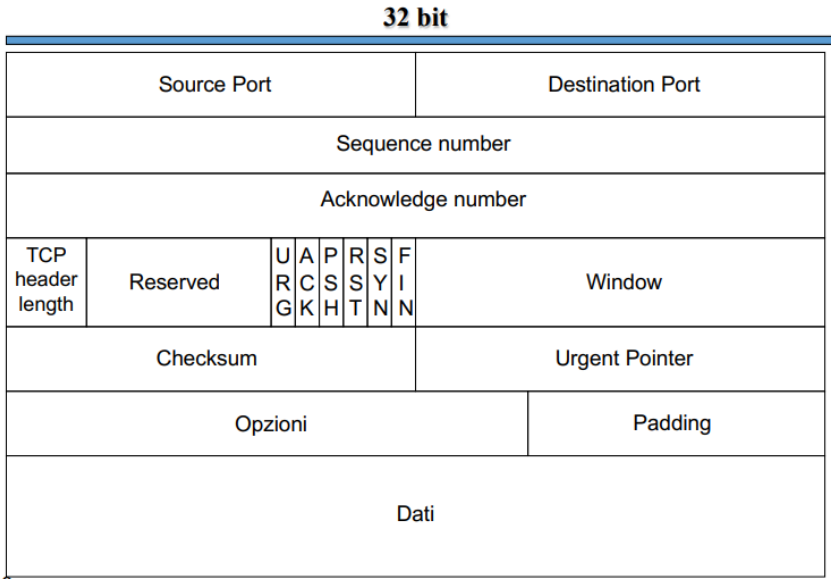


Messaggio UDP

Transmission Control Protocol

- Incapsula dati delle app in pacchetti detti segmenti
- Segmento TCP prevede:
  - Header standard di 20 byte
  - Header variabile per negoziare delle opzioni
  - Payload di dimensione variabile contenente i dati dell'app
- Segmento TCP ha dimensione massima detta **Maximum Segment Size** (MSS)
  - Massima dimensione del blocco dati di app che può essere contenuto nel segmento
  - MSS deve essere inferiore alla max. dimensione del payload IP meno un Header TCP
  - Deve rispettare limiti imposti ai pacchetti dalle reti e che hanno una Maximum Transfer Unit (MTU) (1500 byte imposti da Ethernet)
  - MSS è configurabile
- Non possibile sapere MTU di ogni rete intermedia
  - Definito algoritmo "Path MTU Discovery" basato su ICMP per efficienza dei pacchetti

Formato del segmento TCP:

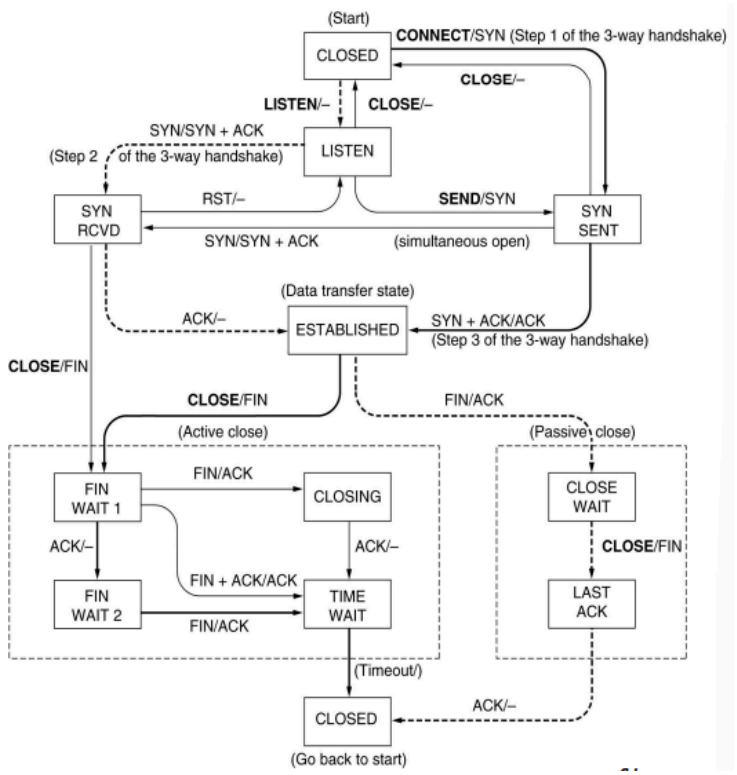


- **Source / Destination Port**
- **Sequence Number**: numero di sequenza del primo byte del pacchetto; se è presente il bit SYN questo è il numero di sequenza iniziale su cui sincronizzarsi
- **Acknowledge number**: se bit di ACK è 1, questo numero contiene il numero di sequenza del blocco di dati che il ricevitore si aspetta di ricevere
- **TCP Header Length (4 bit)**: numero di parole di 32 bit dell'intestazione TCP; indica dove iniziano i dati
- **Reserved**: sei bit riservati per uso futuro
- Control Bit: 6 bit di controllo:
    - **URG** posto a 1 se si deve considerare il campo Urgent Pointer
    - **ACK** posto a 1 se si deve considerare il campo Acknowledge
    - **PSH** posto a 1 serve per la funzione di push
    - **RST** posto a 1 per resettare la connessione
    - **SYN** posto a 1 per sincronizzare i numeri di sequenza
    - **FIN** posto a 1 per indicare la fine dei dati
- **Window**: finestra del ricevitore, cioè il numero di byte che il ricevitore può ricevere
- **Checksum**: controllo dell'errore sul segmento
- **Urgent Pointer**: puntatore a dati urgenti se presenti nel pacchetto
- **Options**: contiene opzioni per la connessione
- **Padding**: bit aggiuntivi per fare intestazione multipla di 32 bit

Checksum:

- Calcolato usando Internet Checksum a:
    - Pseudo-header: garantisce dati arrivino dal mittente al destinatario giusto
    - Intestazione TCP
    - Dati del segmento TCP

Macchina a Stati Finiti del TCP:

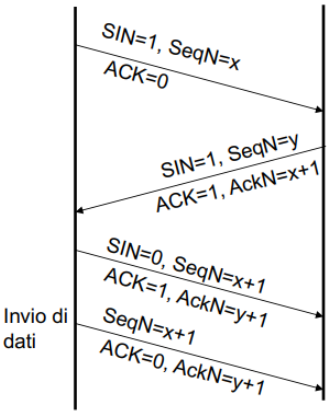


Linee tratteggiate: azioni tipiche server
Linee nere: azioni tipiche client
Linee chiare: eventi inusuali
Transizioni: causa/effetto

Stato: condizione che descrive il software del protocollo di un calcolatore in un istante

Transizione: passaggio da uno stato a un altro

Se mezzo trasmissivo inaffidabile impossibile avere scambio di info con conferma certa:
A invia messaggio a B → Se A non riceve conferma non sa se B ha ricevuto il mess. o meno → Perdita del messaggio o della conferma? → necessario decidere dove fermarsi



Per questo motivo si usa la **Three Ways Handshake** (molto robusto). Resiste all'instaurazione contemporanea di 2 connessioni e ignora pacchetti di apertura ritardatari.
Il 1° pacchetto dati ha numero di sequenza = all'ACK precedente.

Svolgimento del Dialogo:

- Usa protocollo ARQ per rendere affidabile il dialogo
- Nello stato ESTABLISHED si trasferiscono dati utilizzando protocollo ARQ:
    - Meccanismo di ritrasmissione è simile a Go-back-N
- Chiusura connessione TCP in **Soft Release**: cerca di chiudere senza perdita dati
- Usa modalità **Simplex**: Le due direzioni vengono rilasciate in modo indipendente. Il TCP che intende terminare la trasmissione emette un segmento con FIN=1
- Se dopo del tempo non arriva l'Ack il mittente del FIN rilascia comunque la conn.

Numerazione in TCP:

- Per flessibilità si danno numeri non ai segmenti ma ai byte trasportati nei segmenti
    - Dati trasportati pensati come unico flusso di byte (byte stream)
    - Si comincia a numerare da un numero x scelto all'apertura della connessione
    - Seq.Number numera il primo byte del segmento
- Conferma di ricezione data mettendo nel campo Ack Number il numero del byte successivo all'ultimo ricevuto: primo byte che ci si aspetta di ricevere
    - 1°pacchetto da 75 byte (x+1), 2°pacchetto sarà x + 76
- Nelle slide alla fine di wireshark, i numeri Seq e Ack dei pacchetti seguono quello che si è detto

Maximum Segment Lifetime:

- Numeri di sequenza possono essere riutilizzati solo se si è sicuri che non esistano più in rete vecchi segmenti numerati con tali numeri
- Massimo tempo di vita dei segmenti (MSL) deve essere noto: MSL = 2 min

Initial Sequence Number:

- All'apertura della connessione si deve scegliere il numero di sequenza iniziale (ISN)
    - Numero prefissato uguale per tutti, casuale o legato a contatore
- ISN:
    - Deve garantire che non ci sia duplicazione nell'uso dei numeri di sequenza
    - Qualora non sia prefissato e costante viene concordato fra i due host
- Se un host viene riavviato a causa di un problema:
    - Si ricreano nuove incarnazioni di vecchie connessioni
    - Scelti nuovi ISN
    - Nuova incarnazione può usare numeri già usati dall'incarnazione precedente
        - Problema: segmenti ritardati della vecchia incarnazione possono essere ricevuti erroneamente, confusione tra dati vecchi e nuovi
        - Meglio usare ISN in funzione del tempo per esempio
- Nella RFC, **ISN è funzione del tempo** usando sistema di conteggio:
    - Contatore a 32 bit
    - Incremento ogni 4 μsec → contatore ripete sequenza ogni 2^{32} * 4μsec = 4.77h
- Inoltre si usa **TCP Quiet Time**:
    - Dopo un qualunque riavvio un host attende almeno un MSL prima di riaprire le connessioni TCP

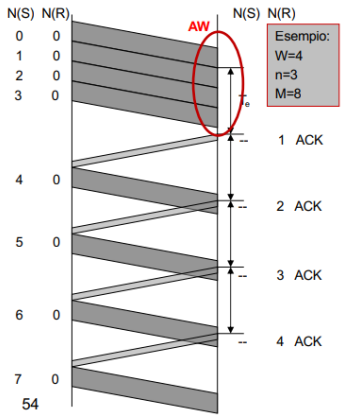
Messaggi di Conferma (ACK):

- In TCP gli ACK sono **cumulativi**: se ACK dice ackN=1500, tutti i byte fino al 1499 sono stati ricevuti correttamente
- Piggybacking:
    - ACK può essere inviato da solo o agganciato a un altro messaggio TCP
    - Se contiene solo l'intestazione TCP e il flag ACK=1, non porta dati dell'app
    - Se trasporta anche dati, fa da risposta e conferma insieme, ottimizza traffico
    - Se non contiene dati, datagramma IP=40 byte (20 intest. IP + 20 intest. TCP)

- Di default, ricevitore invia un ACK per ogni segmento ricevuto
- Tuttavia, può ritardare l'invio dell'ACK per minimizzare ACK ma può scattare time-out
    - ACK ritardato se si riceve pacchetto e si aspetta che ne arrivi un altro a breve
    - Scatta un time out (solitamente 200 ms)
    - Si riduce traffico di ACK
- Ricevitore TCP se arrivano pacchetti fuori ordine o duplicati:
    - Ricevitore ha ricevuto fino a SeqN = N
    - Attende un segmento con SeqN = N+1 mod M
    - Riceve un segmento con SeqN = X ≠ N+1 mod M
        - Se X < N → è un duplicato ritardato, viene scartato
        - Se X > N → Segmento fuori sequenza
    - Può essere successo che:
        - Segmento precedente sia andato perso
        - Un segmento trasmesso dopo un altro lo ha superato a causa dei diversi percorsi possibili e dei ritardi variabili in rete
    - Se X rientra nella finestra di ricezione (W_R) viene memorizzato
    - Ricevitore ritrasmette l'ultima conferma inviata (ACK duplicato)
    - Quando segmento mancante arriva, ricevitore conferma ackN = X + 1
- Mittente TCP:
    - Invia segmento seqN = N
    - Se riceve un ACK con ackN = N+1: segmento ricevuto, toglie il segmento dalla memoria e fa scorrere la finestra di trasmissione
    - Se non riceve ackN = N+1 allo scadere di **RTO** (Retransmission Timeout) ritrasmette il segmento
    - ACK duplicati vengono ignorati

CONTROLLO DI FLUSSO IN TCP

- Si utilizza come nel livello 2 un meccanismo a finestra scorrevole
  - Dimensione deve essere coerente con memoria di trasm. e ricevitore
  - Ricevitore comunica al trasm. le dimensioni della sua memoria di ricezione
  - Nell'intestazione TCP è contenuto il campo advertised window (AW)
- W_T = finestra di trasmissione(segmenti inviabili da trasm.senza ricevere ACK)
- W_R = finestra di ricezione (Insieme di segmenti memorizzabili fuori sequenza)
- M = spazio di numerazione (in TCP = 2^{32})
- Se $W_T = W_R$ allora $W_T + W_R \leq M$ ($W_T \leq 2^{31}$ e $W_R \leq 2^{31}$)
- W può essere misurata in byte (w) o in numero di segmenti (W)
- Normalmente si misura W in segmenti di dim. massima (full sized segments)
- W o w messe a punto dinamicamente con informazioni:
  - Provenienti dal ricevente (AW advertised window)
  - Correlate alla congestione della rete dal trasm. (CW congestion window)
- AW e CW sono funzione del tempo, a un certo t: W = min[AW, CW]



Trasmettitore veloce e ricevitore lento.
Con AW ricevitore indica al trasm. la dimensione del suo buffer.
Così si è certi che possa ricevere l'intera prima finestra di dati.
Si può creare deadlock se:

- ricevitore ha buffer pieno e comunica AW = 0
- Trasmettitore sospende trasmissione
- Ricevitore non ha più dati da inviare e non invia nuovi segmenti TCP che potrebbero aggiornare AW

Soluzione:TCP consente sempre invio di 1 byte anche se AW=0

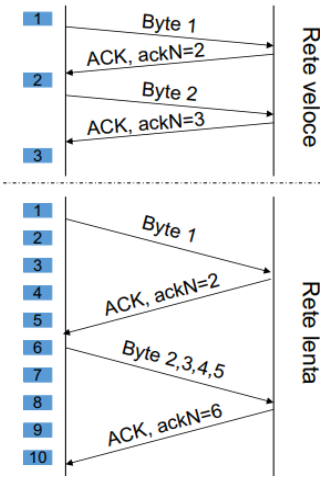
- Quando trasmettitore riceve ACK con AW = 0 fa partire il **Persist Timer**:
  - PT = 1,5 sec per LAN
  - Quando PT scade si invia un segmento di 1 byte (seqN = X + 1)
  - Ricevitore deve rispondere:
    - Invia ACK con ackN = X + 2 e AW > 0 (trasmissione riprende)
    - Invia ACK con ackN = X + 1 e AW = 0
      - Non può ricevere byte X + 1 e PT = 2PT e si attende
    - Max PT è 60 sec
  - Tutto ciò può essere inefficiente

Ricevitore lento: **Silly Window Syndrom**

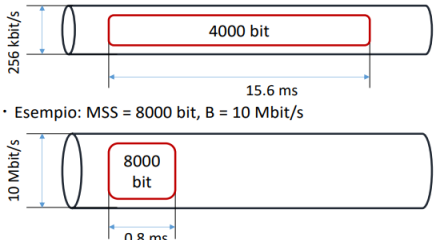
- Buffer si riempie (AW = 0), l'app legge un byte e trasmette AW = 1
- Trasmettitore manda un segmento di un byte e buffer di ricezione si riempie (AW=0)
- Quindi viene trasmesso un byte alla volta
- Soluzione:
    - Ricevitore non può aumentare AW a meno che:
        - Nuovo valore di AW sia almeno pari a MSS
        - Nuovo valore di AW sia almeno pari a metà del buffer di ricezione

Trasmettitore lento: **Algoritmo di Nagle**

- Trasmettitore passa a TCP un byte per volta (Telnet)
- Vengono trasmessi dei **Tinygram** (segmenti di un solo byte)
    - Ogni byte richiede almeno 40 byte di header e 40 byte di ACK
    - Overhead per byte molto elevato
- Si deve aumentare dimensione del messaggio
- Soluzione: Algoritmo di Nagle
    - Trasmettitore invia un nuovo segmento solo se è vera una delle seguenti condizioni:
        - Segmento è di dimensioni pari a MSS
        - Segmento è di dimensioni almeno pari a metà del valore di AW
        - Non ci sono ACK pendenti ed è possibile trasmettere tutto ciò che è in attesa nel buffer di trasmissione
    - Effetto: si può avere un solo segmento pendente per il quale non si è ricevuto ACK → Più veloci arrivano gli ACK più velocemente si trasmette



- Massimo throughput (byte/sec) vale: S = w/RTT
- TCP dovrebbe poter «indovinare» i valori di banda e ritardo per poter dimensionare correttamente CW
- **Diagramma di Jackobson**: orizzontalmente il tempo e verticalmente la banda



- CW va adattata dinamicamente in base al cambiamento della banda disponibile
- Definite due fasi: (W_{id} dimensione ideale della finestra)
    - **Slow Start:**
        - Per raggiungere velocemente un W prossimo a W_{id}
        - CW parte da valore piccolo e raddoppia ad ogni RTT
    - **Congestion Avoidance:**
        - Fa sì che W sia il più prossimo possibile a W_{id}
        - Una volta raggiunta una soglia ssthr, TCP aumenta CW più lentamente (linearmente)
        - Evita congestione e stabilizzare trasmissione

Slow Start:

- A inizio connessione: $w \leq 2 * MSS$ e $W \leq 2$
- Per ogni ACK ricevuto senza scadenza di RTO: $W = W + 1$ ($w \leq w + MSS$)
- Slow Start termina quando:
    - Si verifica congestione (no ACK in RTO)
    - w > ssthr (**Slow Start Threshold**)
- Se w = ssthr si può usare o Slow Start o Congestion Avoidance
- W ha crescita esponenziale in Slow Start(al termine di ogn RTT la finestra raddoppia)
- Slow Start dura circa: $T_{ss} = RTT \log_2 SSTHR$

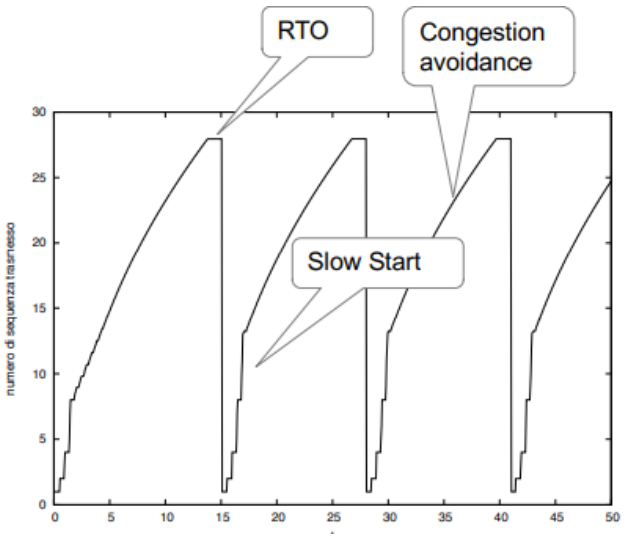
Congestion Avoidance:

- Si passa da crescita esponenziale di Slow Start a una crescita lineare
- w viene incrementata di un MSS per ogni RTT
    - FIno a quando si verifica congestione o si raggiunge AW
- Implementazione dell'incremento:
    - Ricevuto ACK i-esimo: W = W + 1 / W
    - Ricevuti ACK di un'intera finestra risulta W = W + 1
    - Normalmente si implementa il calcolo in byte $w = w + MSS^2 / w$

- Loss Window (LW):
  - Quando scade un RTO il trasmettitore ritiene perso un segmento
  - Il segmento deve essere ritrasmesso
  - Si pone CW $\leq$ LW (LW = 1 solitamente)
- **Flightsize**:
  - Quantità di byte trasmessi ma non confermati
  - E' la quantità di dati presenti in rete
  - Non necessariamente uguale a W

Scade RTO: (segmento non viene riscontrato o ACK non arriva in tempo)

- TCP in Slow Start:
  - Si riparte da capo ponendo W = 1
  - Si impone ssthr = max(W/2, 2MSS)
- TCP in Congestion Avoidance:
  - Termina la fase di Congestion Avoidance e riparte Slow Start
  - Si impone ssthr = max(W/2, 2MSS)



- Se rete stabile $T_{ss} < T_{ca}$; durate slow start e congestion avoidance