Descrivere le caratteristiche, i vantaggi e svantaggi e le differenze tra *commutazione di circuito* e *commutazione di pacchetto*, anche attraverso opportuni esempi applicativi.

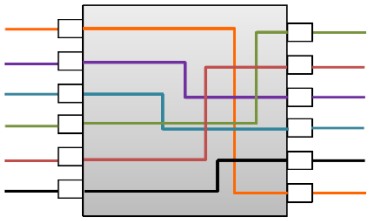
COMMUTAZIONE DI CIRCUITO

Nella commutazione di circuito si instaura un **collegamento fisico reale**, link per link, che collega due terminali e rimane **dedicato** per tutta la durata della comunicazione: le risorse per la comunicazione sono dunque riservate.

Tale operazione avviene previa opportuna **suddivisione** "in pezzi" **delle risorse di rete**: la banda, ovvero la capacità di trasmettere un certo numero di bit al secondo, viene suddivisa "in pezzi", il che può avvenire tramite **divisione in frequenza oppure divisione in tempo**.

Ciascun circuito viene allocato ai vari collegamenti, e rimane inattivo se non viene utilizzato; **non c'è dunque condivisione tra terminali**.

Modello di nodo: commutatore a circuito, caratteristiche

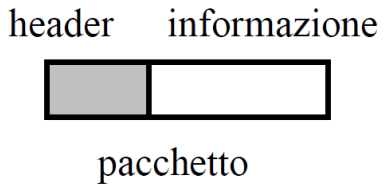


1) La capacità dei canali in ingresso è pari alla capacità di quelli in uscita
2) Non avviene alcun tipo di memorizzazione temporanea dell'informazione

COMMUTAZIONE DI PACCHETTO

A differenza della commutazione di circuito, nella commutazione di pacchetto è l'**informazione** da trasmettere a essere **divisa in** "pezzi", detti appunto **pacchetti**: tali pacchetti vengono **inviati step-by-step**, senza la necessità di instaurare un collegamento unico da terminale a terminale.

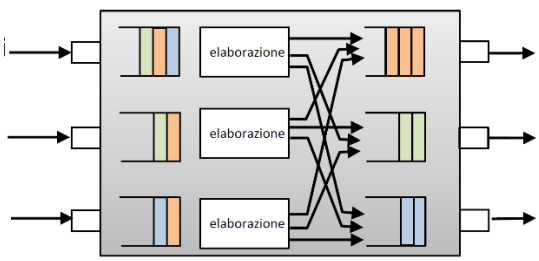
Ogni pacchetto consiste in:



- Una prima parte, detta **Header**, che contiene le informazioni necessarie all'instradamento del pacchetto, ovvero l'**indirizzo di destinazione** dello stesso: i link sono in grado di instradare il pacchetto proprio grazie alle informazioni contenute nell'header
- Una seconda parte contenente una frazione dei dati dell'**informazione** da trasmettere

In tale sistema, **i pacchetti di tutti gli utenti condividono le risorse di rete**, in quanto **ciascun pacchetto utilizza completamente il canale dove viene trasportato**.

Modello di nodo: packet switcher, caratteristiche



Nel caso della commutazione di circuito il nodo consiste in un meccanismo semplice, con n ingressi e n uscite; nella commutazione di pacchetto il nodo invece opera con un meccanismo più complesso, con le seguenti caratteristiche:
- L'arrivo dei pacchetti è asincrono
- La capacità dei collegamenti è arbitraria

- Possono esserci **conflitti temporali per la trasmissione**: dunque è necessario **memorizzare temporaneamente i dati** per analizzare l'indirizzo di destinazione e per gestire eventuali conflitti; dunque, anche la struttura fisica del nodo stesso varia tra commutazione di circuito e commutazione di pacchetto, in quanto nel secondo caso è necessario che il nodo abbia capacità di memorizzazione.

Per risolvere il problema della **contesa per le risorse** (ovvero il caso in cui più pacchetti hanno la necessità di procedere nella stessa direzione) si adottano 2 differenti strategie:

1) **STORE AND FORWARD**: il commutatore deve ricevere l'intero pacchetto prima di poter cominciare a trasmettere sul collegamento in uscita
2) **MULTIPLAZIONE STATISTICA**: accodamento dei pacchetti, attesa per l'utilizzo del collegamento

Esempi pratici di commutazione di pacchetto e di circuito

Un esempio pratico di commutazione di circuito è dato dal **sistema di chiamate telefoniche a rete fissa** che veniva adottato nel secolo scorso: la linea telefonica veniva dedicata interamente allo scopo della singola telefonata, e gli operatori nei centralini formavano il collegamento fisico reale tra i due terminali in chiamata.

Un esempio pratico di commutazione di pacchetto è proprio la **comunicazione tramite internet**.

Confronto tra pacchetto e circuito

La **commutazione di pacchetto consente a più utenti di usare la rete**, **e** inoltre generalmente consente di **scaricare le informazioni più velocemente**.

Tuttavia, la commutazione di pacchetto presenta il **problema delle code**: si ha ritardo nella trasmissione e possibile perdita di pacchetti; per questo, **sono necessari protocolli per il trasferimento affidabile** dei dati e **per il controllo della congestione**.

Nel servizio di web browsing, relativamente alla modalità di connessione tra client e server http, si descrivano le caratteristiche e le differenze tra connessione non persistente e persistente (in serie e in parallelo), riportando anche esempi applicativi.

CONNESSIONE NON PERSISTENTE

La connessione non persistente è il metodo tradizionale di connessione tra un client e un server HTTP, in cui **viene aperta una connessione separata tra client e server per ogni richiesta HTTP**. In questa modalità, la connessione viene chiusa dopo che la risposta è stata inviata dal server al client.

Le caratteristiche principali della connessione non persistente sono:

1) **Connessione separata per ogni richiesta**: **Ogni richiesta HTTP richiede** una nuova connessione con **l'apertura di un nuovo socket tra client e server**. Dopo ogni richiesta, la connessione viene chiusa.
2) **Maggior overhead di connessione**: L'apertura e la chiusura della connessione per ogni richiesta aggiunge un certo overhead che **influisce sulle prestazioni complessive**.
3) **Maggior numero di pacchetti di rete**: Dato che ogni richiesta richiede l'apertura di una nuova connessione, ci sono più pacchetti di rete inviati tra client e server, **aumentando il traffico di rete complessivo**.

Un esempio applicativo della connessione non persistente è quando un client web invia richieste multiple per ottenere differenti risorse di una pagina. Ad esempio, se una pagina web contiene immagini, fogli di stile e file JavaScript, il client dovrà aprire una connessione separata per ogni risorsa richiesta, che implica una maggiore latenza e un maggiore traffico di rete.

CONNESSIONE PERSISTENTE

la connessione persistente è una modalità di connessione tra client e server **in cui la connessione rimane aperta dopo che una risposta è stata inviata dal server al client**. In questa modalità, il server mantiene la connessione aperta per un certo periodo di tempo (come definito dalla specifica HTTP o dalle impostazioni del server) e il client invia ulteriori richieste sulla stessa connessione aperta.

Le caratteristiche principali della connessione persistente sono:

1) **Connessione riutilizzata per più richieste**: Dopo che il server ha inviato una risposta, la **connessione** rimane aperta e **può essere riutilizzata** per inviare ulteriori richieste senza la necessità di aprire una nuova connessione.
2) **Riduzione dell'overhead di connessione**: Poiché la connessione viene mantenuta aperta, non c'è bisogno di aprire e chiudere una connessione separata per ogni richiesta. Ciò riduce l'overhead e **migliora le prestazioni complessive**.
3) **Riduzione del traffico di rete**: Con meno connessioni aperte, c'è meno traffico di rete in quanto meno pacchetti di rete sono scambiati tra client e server.

Un esempio applicativo della connessione persistente è quando un client web invia richieste multiple per diverse risorse di una pagina, come immagini, fogli di stile e file JavaScript, utilizzando la stessa connessione aperta. Questo riduce la latenza complessiva e il traffico di rete, migliorando l'esperienza utente sul web.

Quale è la funzione di un server DNS? Come è organizzata la struttura dei server DNS nella rete?

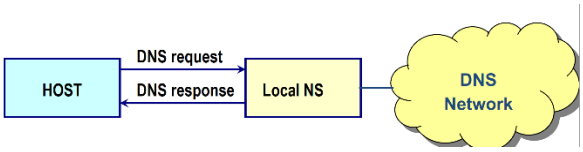
Un server DNS svolge la funzione fondamentale di **risoluzione dei nomi simbolici**: difatti, gli indirizzi IP consistono in 32 bit di codifica binaria e risultano poco adatti ad essere usati dagli applicativi, rendendo necessaria l'introduzione di indirizzi simbolici (alfanumerici), e conseguentemente di una **mappatura tra indirizzi IP** (usati dalle macchine di rete) **e nomi simbolici** (usati dagli esseri umani).

In generale, il sistema DNS è caratterizzato da 2 "ingredienti" fondamentali:

a) **Database distribuito**, costituito da molti ***name servers* con organizzazione gerarchica - Ogni livello** nella gerarchia **ha diversa "profondità" di informazione**: ad esempio, volendo cercare l'IP di www.google.com si avrà una prima ricerca da parte del r*oot server* per il dominio .com, seguita poi dalla ricerca da parte dei *name server* prima del dominio google.com e poi del nome simbolico www.google.com
b) **Protocollo applicativo tra *name server* e *host* per risolvere nomi simbolici**, ovvero tradurre nomi simbolici in indirizzi IP

La struttura dei server DNS nella rete è organizzata in una **gerarchia a livelli che include diversi tipi di server**.

1) **Root Server**: Al vertice della gerarchia ci sono i server DNS radice (Root Server), che sono responsabili di **fornire informazioni sugli indirizzi IP dei server dei domini di primo livello**, come ad esempio .com, .org, .it, ecc. . Tali Server sono **sparsi in tutto il mondo**.
2) **TLD Server**: I server dei domini di primo livello (TLD Server) sono responsabili di fornire **informazioni sugli indirizzi IP dei server dei domini di secondo livello all'interno del loro TLD**. Ad esempio, un server TLD per il dominio .com fornisce informazioni sugli indirizzi IP dei server per siti come google.com, microsoft.com, ecc.



In realtà, in generale i singoli host non sono direttamente collegati al DNS Network: **ogni ISP ha un Local Name Server**, che gestisce gli host che fruiscono di tale ISP contattando il LNS ogni volta che è richiesto risolvere un indirizzo simbolico; **è dunque il solo LNS a comunicare con il DNS**.

Inoltre, per evitare che il DNS Network venga sovraccaricato di richieste, vengono definiti **protocolli di caching per i name server**: un server, dopo aver reperito un'informazione su cui non è autoritario, può memorizzarla temporaneamente per poterla fornire all'arrivo di una nuova richiesta, senza dover risalire sino al server authoritative. Il **TimeToLive** è deciso dal server authoritative ed è usato da essi per decidere un **time-out**, sulla base della **stabilità dell'informazione relativa**.

Nel caso si voglia aggiungere un dominio al DNS Network, è necessario registrare il dominio desiderato presso un **DNS Registrar**, fornendo i **nomi simbolici** e i relativi **indirizzi IP** dei name server autoritativi.

Si descriva l'architettura a strati della rete IP, discutendo le relazioni tra i diversi livelli, l'incapsulamento della PDU e la struttura del modello OSI (e Internet).

L'architettura a strati della rete IP, concettualmente, è **organizzata in diversi livelli che collaborano per fornire la comunicazione tra i dispositivi di rete**. Questi livelli sono composti da **protocolli e funzionalità specifici**, ognuno dei quali svolge un compito specifico per facilitare la trasmissione dei dati attraverso la rete.

La **rete IP segue il modello a strati TCP/IP**, che è diverso dal modello OSI (Open Systems Interconnection) ma ha una struttura simile.

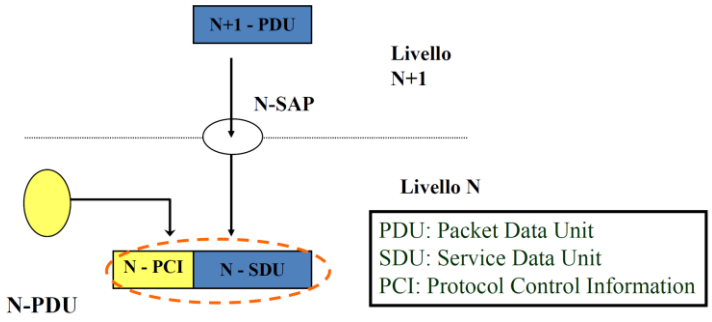
MODELLO TCP/IP

Il modello TCP/IP si compone di quattro livelli:

1) **Livello di collegamento dati**: Questo livello si occupa di trasmettere i dati su un singolo collegamento fisico, come l'Ethernet. Esso definisce i protocolli e i metodi per l'accesso al mezzo di trasmissione, l'indirizzamento fisico e la correzione degli errori.
2) **Livello di rete**: Il livello di rete si occupa del routing dei pacchetti attraverso la rete. Esso utilizza l'indirizzo IP per determinare la destinazione dei pacchetti e seleziona il percorso migliori per la consegna. Il protocollo principale a questo livello è l'IP.
3) **Livello di trasporto**: Questo livello fornisce servizi di trasporto end-to-end e si occupa del controllo degli errori, del controllo di flusso e dell'affidabilità dei dati. Il protocollo principale a questo livello è il TCP (Transmission Control Protocol) che offre un trasporto affidabile e orientato alla connessione, ma vi è anche l'UDP (User Datagram Protocol) che fornisce un trasporto non affidabile ma più leggero.
4) **Livello applicativo**: Questo è il livello più alto del modello TCP/IP e include tutti i protocolli e le applicazioni che consentono agli utenti di accedere ai servizi di rete. Alcuni esempi di protocolli a livello applicativo includono HTTP (Hypertext Transfer Protocol) per il web, SMTP (Simple Mail Transfer Protocol) per l'invio di email.

INCAPSULAMENTO PDU

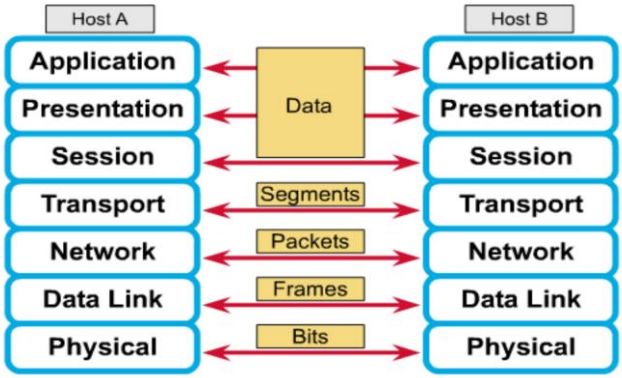
L'incapsulamento della PDU avviene quando i **dati di un livello vengono inseriti come payload all'interno del campo dati di un altro livello inferiore**: tale operazione è svolta dai **SAP** (Service Access Point), che aggiungono un'**overheader specifico del livello corrente** al pacchetto ricevuto dal livello superiore. Ad esempio, i dati del livello di trasporto (TCP o UDP) vengono incapsulati all'interno dei pacchetti IP come payload. A loro volta, i pacchetti IP vengono incapsulati all'interno dei frame del livello di collegamento dati.



MODELLO OSI

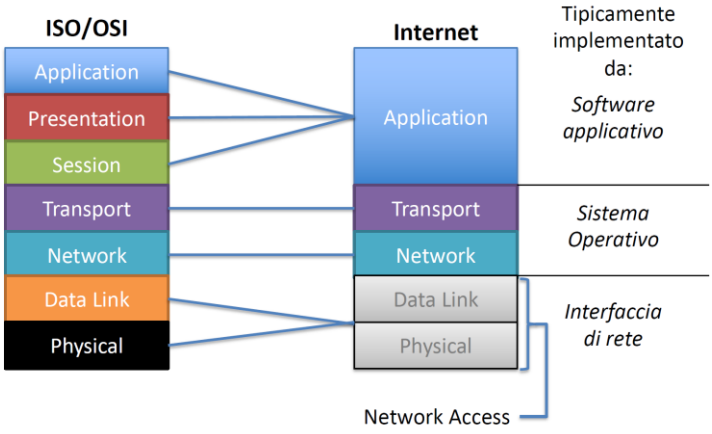
La struttura del modello OSI (Open Systems Interconnection) è simile al modello TCP/IP, ma prevede sette livelli invece di quattro:

1) **Livello fisico**: Include le specifiche per la modulazione dei mezzi e dei segnali fisici utilizzati per la trasmissione dei dati.
2) **Livello di linea**: Gestisce la trasmissione dei dati aggregando i bit in gruppi, detti trame.
3) **Livello di rete**: Si occupa del routing dei pacchetti attraverso la rete.
4) **Livello di trasporto**: Fornisce servizi di trasporto end-to-end, gestendo il controllo degli errori e il controllo di flusso.
5) **Livello di sessione**: Gestisce l'apertura, la chiusura e la sincronizzazione delle sessioni tra i dispositivi di rete.
6) **Livello di presentazione**: Si occupa della rappresentazione dei dati, inclusa la crittografia e la compressione.
7) **Livello applicativo**: Include tutti i protocolli e le applicazioni attraverso le quali gli utenti accedono ai servizi di rete.



Confronto tra i due modelli

Sebbene il modello TCP/IP non corrisponda esattamente al modello OSI, i suoi livelli possono essere correlati in questa maniera: il livello di collegamento dati e il livello di rete del modello TCP/IP spesso corrispondono ai livelli fisico e di collegamento dati del modello OSI. Il livello di trasporto del modello TCP/IP può essere correlato al livello di trasporto del modello OSI. Infine, il livello applicativo del modello TCP/IP può essere correlato ai livelli di sessione, presentazione e applicativo del modello OSI.

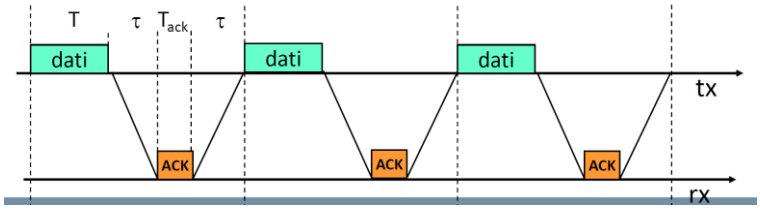


Nell'ambito del livello di trasporto, nel caso di approccio "stop e wait" si definisca e si discuta il concetto di "efficienza del protocollo" e si discuta quando tale efficienza è alta e bassa.

Nell'ambito del livello di trasporto, l'approccio "stop e wait" è un semplice metodo di comunicazione tra mittente e destinatario. **Quando il mittente invia un pacchetto al destinatario, attende la conferma di ricezione prima di inviare il pacchetto successivo**.

L'efficienza del protocollo si riferisce alla **capacità del protocollo di utilizzare in modo ottimale la larghezza di banda disponibile e di inviare i dati nel minor tempo possibile**, **minimizzando la quantità di tempo di inattività** e l'utilizzo delle risorse di rete. In generale, un protocollo è considerato efficiente se riesce a **trasferire i dati in modo rapido ed efficace**, **senza ritardi o congestionamenti**.

Nel caso specifico dello stop and wait, l'efficienza del protocollo è la **frazione di tempo in cui il canale è usato per trasmettere informazione utile** in assenza di errori.



$$\eta = \frac{T}{T + T_{ack} + 2\tau}$$

Facendo considerazioni ulteriori sulla formula, si ricava:

$$\eta = \frac{T}{T + T_{ack} + 2\tau} = \frac{1}{1 + \frac{T_{ack}}{T} + \frac{2\tau}{T}}$$

Dunque, **si ha efficienza bassa se T<<τ**.

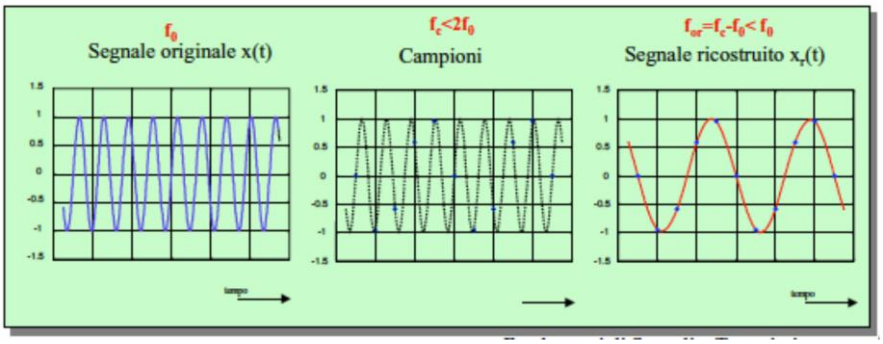
Si conclude quindi che il **protocollo non è adatto a situazioni con elevato ritardo di propagazione e/o elevato ritmo di trasmissione**.

Si enunci il teorema di Nyquist, si discutano la sua applicazione e la sua efficacia, riportando anche esempi numerici applicati a reali segnali di comunicazione.

Il teorema di Nyquist, formulato da Harry Nyquist, è un teorema fondamentale nella teoria delle telecomunicazioni che **stabilisce un limite teorico alla capacità di trasmettere informazioni in un canale di comunicazione senza perdita di dati**.

TEOREMA: Un segnale del tempo è completamente determinato dai suoi campioni presi a distanza T tale che **$T \leq 1/2B$**, dove B è la banda del segnale.
Alternativamente, si può anche dire che i campioni devono essere presi con frequenza **$f_c \geq 2B$** ; tale frequenza è definita frequenza di Nyquist.

In pratica**, i campioni presi alla frequenza di Nyquist rappresentano il contenuto informativo completo del segnale**: campioni **più frequenti sarebbero inutili**, mentre campioni **meno frequenti** "perdono informazione", **rendendo il segnale impossibile da ricostruire in maniera esatta**.
Se non rispetta il Teorema di Nyquist, si rischia di ricostruire il segnale ottenendo un contenuto informativo totalmente diverso da quello di partenza:



Alcuni **esempi di frequenze di campionamento** tipiche sono:

Segnale	Banda	Frequenza di campionamento
Segnale telefonico	300-4000 Hz	**8000 Hz**
Voce	300-8000 Hz	**16000 Hz**
Musica	100-20.000 Hz	**40 kHz**
TV (PAL)	0-5.000.000 Hz (5 MHz)	**10 MHz**
Cinema	0-500 MHz	**1 GHz**

Questa formulazione teorica del teorema di Nyquist è valida e si applica ai segnali ideali. Tuttavia, **nelle situazioni reali, possono essere presenti fattori che limitano l'efficacia del teorema di Nyquist**. Ad esempio, i **segnali reali possono contenere rumore che può alterare il segnale originale, aumentando la frequenza di trasmissione effettiva**. Inoltre, possono essere presenti dispersione del segnale e attenuazione che possono influire sulla qualità della trasmissione. In conclusione, il teorema di Nyquist fornisce una base teorica importante per la progettazione dei sistemi di comunicazione digitale, stabilendo un limite superiore alla velocità massima di trasmissione dei segnali senza perdita di informazioni. Tuttavia, l'efficacia pratica del teorema può essere influenzata da vari fattori che possono ridurre la velocità di trasmissione effettiva e la qualità della comunicazione.