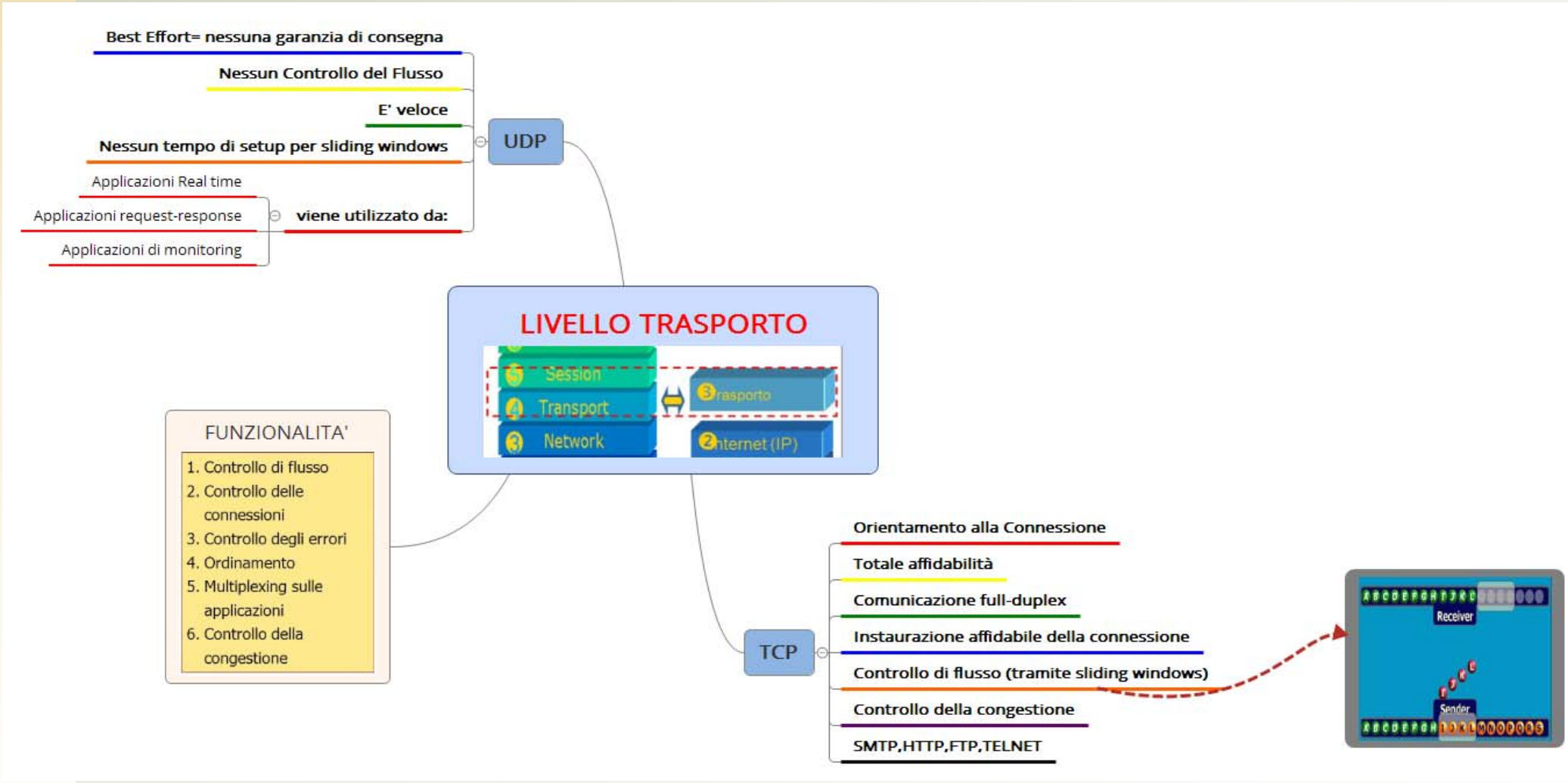
IIS Carafa-Giustiniani
Cerreto Sannita- Benevento

# Corso di Sistemi e Reti 3

## Classe V – Informatica

Prof. Tullio Parcesepe

# Lezione precedente



# Elenco degli argomenti

## UNITÀ DI APPRENDIMENTO 1

## Il livello delle applicazioni

### L1 Il livello delle applicazioni nei modelli ISO/OSI e TCP

Il livello di applicazione fornisce diversi servizi agli utenti di Internet, la sua flessibilità consente di aggiungere nuovi protocolli, pertanto non è più possibile indicare il numero preciso dei protocolli esistenti, in quanto ne vengono aggiunti costantemente di nuovi.

Il livello applicazione implementa i vari protocolli, tra cui:

- SNMP: Simple Network Management Protocol;
- SMTP: Simple Mail Transfer Protocol;
- POP3: Post Office Protocol;
- FTP: File Transfer Protocol;
- HTTP: HyperText Transfer Protocol;
- DNS: Domain Name System;
- SSH: Secure SHell;
- Telnet.

Nel livello delle applicazioni la comunicazione client-server avviene tramite una connessione logica, dove i livelli applicazioni agiscono come se esistesse un collegamento diretto tra client e server. Un programma (applicazione) che vuole comunicare con un altro programma, deve gestire i primi quattro livelli TCP/IP per aprire la connessione, inviare/ricevere dati e chiudere la connessione. Blocchi di istruzioni di questo tipo vengono chiamate API (Application Programming Interface) o socket.

# Architetture di rete

## Architetture delle applicazioni di rete

Il primo passo che il programmatore deve effettuare per progettare una applicazione di rete è la scelta della architettura dell'applicazione; le principali architetture attualmente utilizzate sono le seguenti:

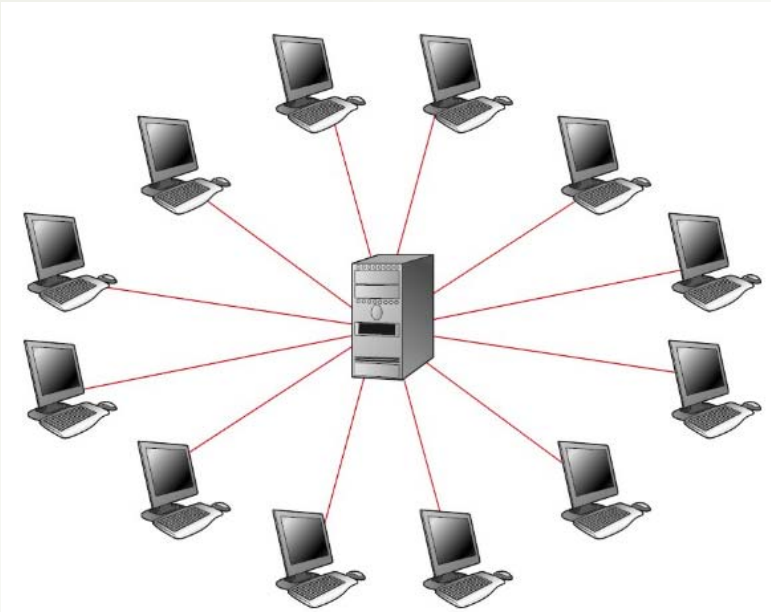
- client-server;
- peer-to-peer (P2P);
- architetture ibride (client-server e P2P).

# 1) Client-Server

## Architettura client-server

Nella architettura client-server la caratteristica principale è che deve sempre esserci un server attivo che offre un servizio, restando in attesa che uno o più client si connettano a esso per poter rispondere alle richieste che gli vengono effettuate.
Un tipico esempio di questa architettura è il www, dove molteplici server (al limite uno per ogni sito pubblicato) possiedono le pagine (statiche o dinamiche) che saranno inviate ai client che ne fanno richiesta tramite i browser.

Il server deve sempre essere attivo e deve possedere un indirizzo IP fisso dove può essere raggiunto dagli host client: quindi l'indirizzo IP deve essere statico, contrariamente a quello dei client che generalmente è dinamico.
Un client non è in grado di comunicare con gli altri client ma solo con il server: più client possono invece comunicare contemporaneamente con lo stesso server.



# 2) Peer-to-peer

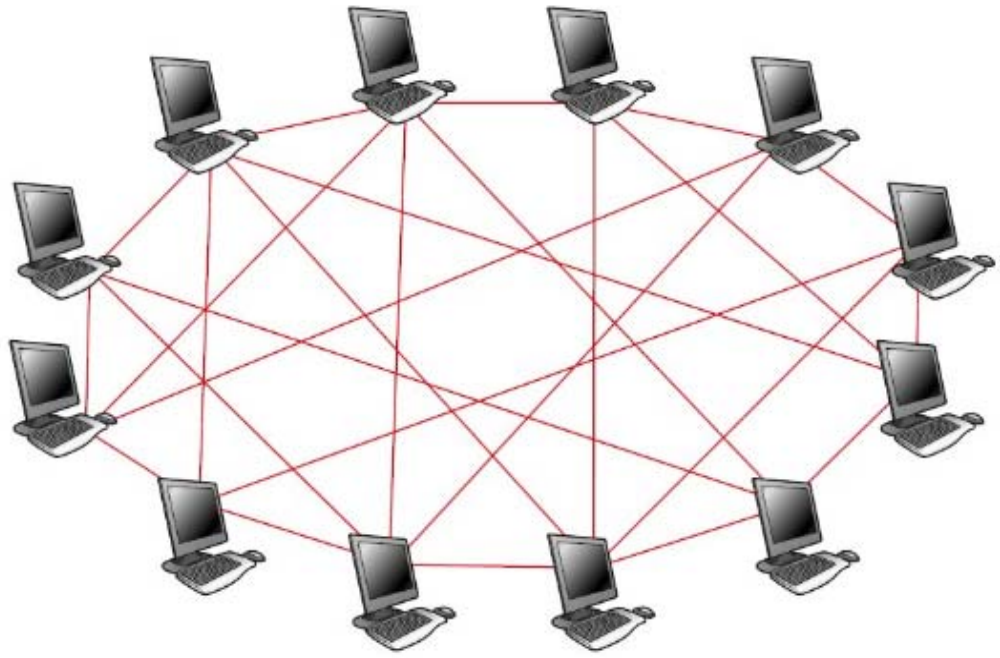
## Architettura peer-to-peer (P2P)

Nelle architetture ◄ peer-to-peer ► (P2P) abbiamo coppie di host chiamati *peer* (persona di pari grado, coetaneo) che dialogano direttamente tra loro.
Nei sistemi P2P gli host possono essere visti come una comunità che collabora con il binomio dare e ricevere: ogni peer fornisce una risorsa e ottiene in cambio altre risorse.
Gli esempi più noti sono rappresentati dalle reti peer to peer in ambito di condivisione di file, come per esempio eMule, oppure Gnutella.

# 2.1) Peer-to-peer decentralizzato

## P2P decentralizzato

Nella architettura completamente decentralizzata un peer ha sia funzionalità di client che di server (funzionalità simmetrica = servent), ed è impossibile localizzare una risorsa mediante un indirizzo IP statico: vengono effettuati nuovi meccanismi di indirizzamento, definiti a livello superiore rispetto al livello IP.
Le risorse che i peer condividono sono i dati, la memoria, la banda ecc.: il sistema P2P è capace di adattarsi a un continuo cambiamento dei nodi partecipanti (churn) mantenendo connettività e prestazioni accettabili senza richiedere l'intervento di alcuna entità centralizzata (come un server).

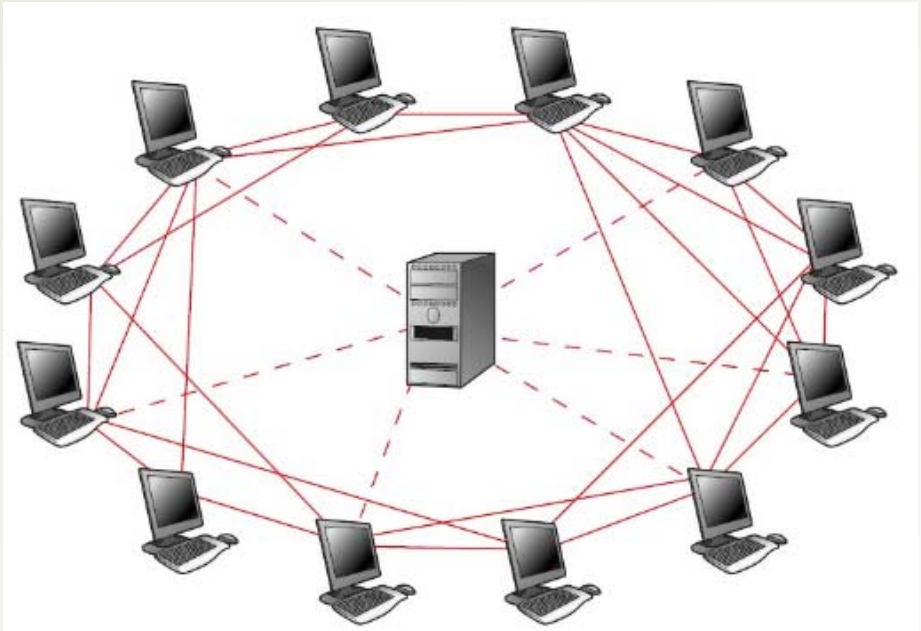


# 2.2) Peer-to-peer centralizzato

## P2P centralizzato

Il P2P centralizzato è un compromesso tra il determinismo del modello client server e la scalabilità del sistema puro: ha un server centrale (directory server) che conserva informazioni sui peer (index, cioè il mapping resorse-peer) e risponde alle richieste su quelle informazioni effettuando quindi la ricerca in modalità centralizzata.

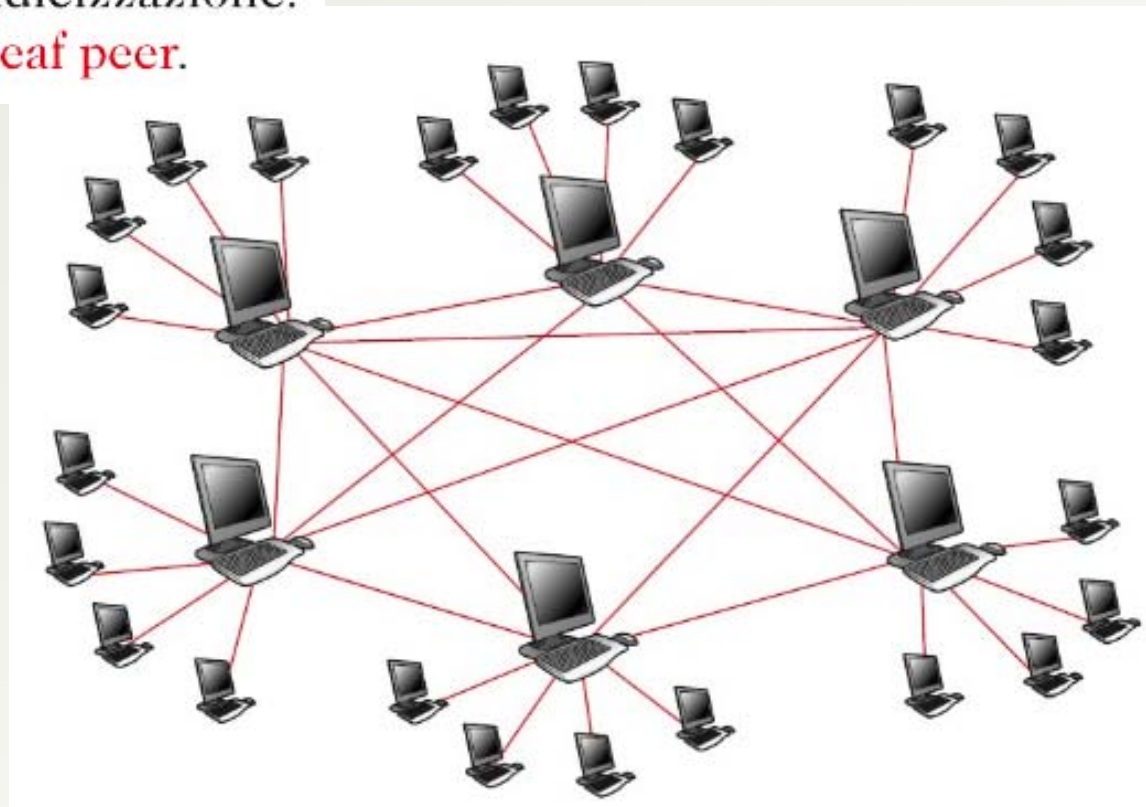
I peer sono responsabili di conservare i dati e le informazioni perché il server centrale non memorizza file, di informare il server del contenuto dei file che intendono condividere e di permettere ai peer che lo richiedono di scaricare le risorse condivise. La sua implementazione più famosa è Napster, dove gli utenti si connettono a un server centrale dove pubblicano i nomi delle risorse che condividono.

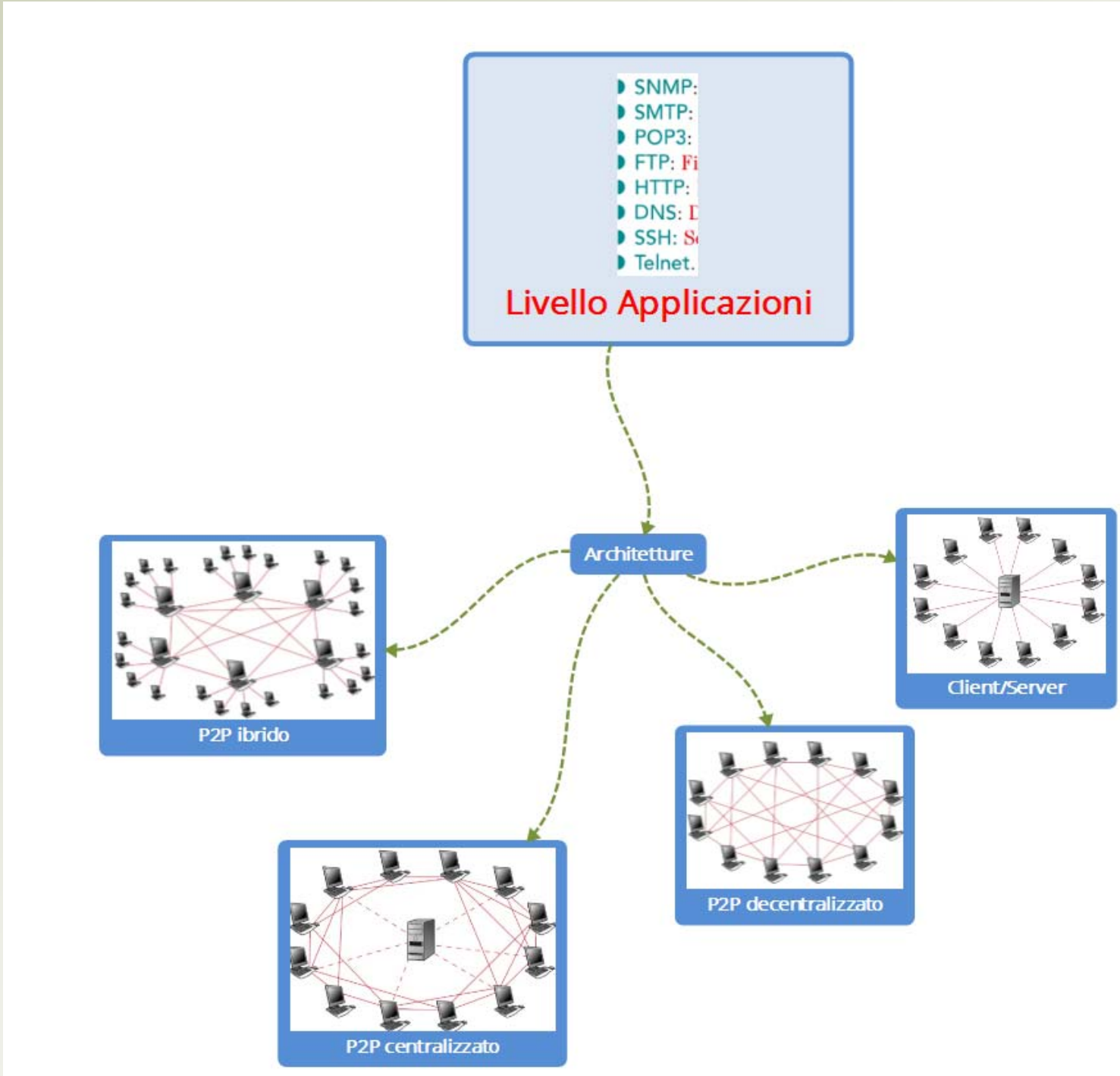


# 2.3) Peer-to-peer ibrido

**P2P ibrido (o parzialmente centralizzato)**

Il P2P ibrido è un P2P parzialmente centralizzato dove sono presenti alcuni peer (detti super-nodi o super-peer o ultra-peer) determinati dinamicamente (tramite un algoritmo di elezione) che hanno anche la funzione di indicizzazione: gli altri nodi sono anche chiamati leaf peer.




SNMP:
SMTP:
POP3:
FTP: Fi
HTTP:
DNS: D
SSH: Se
Telnet.
Livello Applicazioni
Architetture
P2P ibrido
Client/Server
P2P decentralizzato
P2P centralizzato

## Servizi offerti dallo strato di trasporto alle applicazioni

Le applicazioni richiedono allo strato di trasporto un insieme di servizi specifici oltre ai protocolli necessari per realizzarli, che possono essere standard o realizzati ad hoc.

Tutti i protocolli, sia standard che specifici, hanno in comune una particolarità: trasferire dei messaggi da un punto a un altro della rete. Ogni applicazione deve scegliere tra i protocolli di trasporto quale adottare per realizzare un protocollo applicativo in base ai servizi che sono necessari alle specifiche esigenze della applicazione, che possono essere riassunte in:

- trasferimento dati affidabile;
- ampiezza di banda;
- temporizzazione;
- sicurezza.

## Trasferimento dati affidabile

Con trasferimento di dati affidabile intendiamo un servizio che garantisce la consegna completa e corretta dei dati: da una parte sappiamo che alcune applicazioni, come per esempio quelle di audio/video possono tollerare qualche perdita di dati senza compromettere lo scopo dell'applicazione mentre altre, come per esempio il trasferimento di file, richiedono un trasferimento dati affidabile al 100%. A tale scopo il livello di trasporto mette a disposizione due protocolli:

- UDP User Datagram Protocol: il protocollo di trasporto senza connessione da utilizzarsi quando la perdita di dati è un fatto accettabile;
- TCP Transfer Control Protocol: il protocollo orientato alla connessione da utilizzarsi quando la perdita di dati è un evento inaccettabile, ovvero quando il trasferimento deve essere affidabile.

## Ampiezza di banda (bandwidth) o throughput

Alcune applicazioni, come per esempio quelle multimediali, per poter "funzionare" hanno bisogno di avere una garanzia sulla larghezza di banda minima disponibile, cioè possono richiedere un throughput garantito di R bit/s: si pensi alla trasmissione di un evento in diretta in una Web-TV. Altre applicazioni, invece, non hanno questo bisogno come prioritario, ma utilizzano in modo elastico l'ampiezza di banda che si rende disponibile: tipici esempi sono la posta elettronica o i sistemi FTP.

> Sappiamo che Internet ha una natura eterogenea e quindi i protocolli di trasporto non sono in grado di garantire la presenza di una certa quantità di banda per tutta la durata necessaria per trasmettere un messaggio ma, come vedremo, mette a disposizione la possibilità di implementare un protocollo applicativo flessibile che realizza un circuito virtuale che fallisce o si adatta se la banda è insufficiente.

## Temporizzazione

Alcune applicazioni, come la telefonia VoIP, i giochi interattivi, gli ambienti virtuali, per essere "realistiche" ammettono solo piccoli ritardi per essere efficaci: lo strato di trasporto non è in grado di garantire i tempi di risposta perché le temporizzazioni presenti assicurano un certo ritardo end-to-end tra le applicazioni.
Il protocollo TCP garantisce la consegna del pacchetto, ma non il tempo che ci impiega e neppure il protocollo UDP, nonostante sia più veloce del TCP, è temporalmente affidabile.

La soluzione, come vedremo, è quella di utilizzare un protocollo di trasporto in tempo reale, come RTP (Real Time Protocol), che è in grado di studiare i ritardi di rete e calibrare gli apparati e i collegamenti per garantire di restare nei limiti di tempo prefissati, scegliendo alternativamente quando utilizzare UDP e quando TCP.

## Sicurezza

Una applicazione può richiedere allo strato di trasporto anche la cifratura di tutti i dati trasmessi in modo tale che anche se questi venissero intercettati da malintenzionati non si perda la riservatezza.
È quindi possibile che vengano richiesti dei servizi di sicurezza da applicare per garantire l'integrità dei dati e l'autenticazione end-to-end.

# …mappa finale