**Capitolo: Il Livello Applicazione nel Modello OSI**

**1. Introduzione al Livello Applicazione**

Il **livello applicazione** è il settimo e ultimo livello del modello OSI, responsabile dell'interazione diretta con l'utente e della fornitura dei servizi necessari per applicazioni di rete. È il livello più vicino all'utente finale e si basa su connessioni logiche tra applicazioni.

**Obiettivi principali:**

1. Fornire un'interfaccia tra l'utente e la rete.
2. Gestire la comunicazione tra applicazioni distribuite.
3. Utilizzare i protocolli per scambiare dati e fornire servizi richiesti dall'utente.

**Funzionamento:**

* Utilizza i servizi del livello di trasporto per la trasmissione affidabile dei dati.
* Supporta protocolli standard e non standard per soddisfare diverse esigenze.

**Protocolli principali:**

* **Standard:** HTTP (web), FTP (trasferimento file), SMTP (posta elettronica).
* **Non standard:** Protocolli personalizzati per applicazioni specifiche.

**2. Paradigmi del Livello Applicazione**

1. **Client/Server:**
   * **Definizione:** Modello di comunicazione in cui un dispositivo (client) invia richieste a un altro dispositivo (server) che elabora e restituisce una risposta.
   * **Esempi:** Navigazione web, sistemi di gestione database.
   * **Vantaggi:**
     + Centralizzazione dei dati.
     + Facilità di manutenzione e controllo.
   * **Svantaggi:**
     + Possibile sovraccarico del server.
     + Dipendenza dal server centrale.
2. **Peer-to-Peer (P2P):**
   * **Definizione:** Ogni dispositivo può agire sia come client che come server.
   * **Esempi:** BitTorrent, reti di file sharing.
   * **Vantaggi:**
     + Distribuzione del carico.
     + Riduzione della dipendenza da un server centrale.
   * **Svantaggi:**
     + Sicurezza più complessa.
     + Gestione e controllo decentralizzati.
3. **Paradigma Misto:**
   * **Definizione:** Combina le caratteristiche di client/server e P2P.
   * **Esempi:** Skype (usa server centrali per il login, ma connessioni P2P per le chiamate).

**3. API (Application Programming Interface)**

* **Definizione:** Interfacce che permettono a programmi diversi di interagire con il sistema operativo o altri software.
* **Esempi comuni:**
  + API di Google Maps.
  + API per database relazionali.

**4. Comunicazione e Protocolli**

1. **Socket e Comunicazione:**
   * **Definizione:** Interfaccia software che permette la comunicazione tra processi.
   * **Componenti principali:**
     + **Indirizzo IP:** Identifica univocamente un dispositivo nella rete.
     + **Porta:** Specifica un servizio o applicazione.
     + **Socket Address:** Combinazione di indirizzo IP e porta, utilizzata per identificare un endpoint.
2. **Protocollo HTTP e HTTPS:**
   * **HTTP:**
     + Protocollo stateless utilizzato per trasferire pagine web (porta 80).
     + **Metodi principali:**
       - GET: Richiesta di dati.
       - POST: Invio di dati.
       - PUT: Aggiornamento di dati.
       - DELETE: Rimozione di risorse.
     + Connessioni persistenti introdotte in HTTP/1.1.
   * **HTTPS:**
     + Versione sicura di HTTP che utilizza crittografia SSL/TLS (porta 443).
3. **TCP vs UDP:**
   * **TCP:** Connessione affidabile, orientata alla connessione (es. email, HTTP).
   * **UDP:** Protocollo più leggero e veloce, ma meno affidabile (es. streaming, VoIP).

**5. Servizi Applicativi**

1. **HTTP e Risposte del Server:**
   * **Codici di stato:**
     + 2xx: Operazione riuscita (es. 200 OK).
     + 4xx: Errore lato client (es. 404 Not Found).
     + 5xx: Errore lato server (es. 500 Internal Server Error).
   * **Intestazioni comuni:**
     + Content-Length: Dimensione del contenuto.
     + Set-Cookie: Invio di cookie al client.
2. **Cookie:**
   * **Definizione:** File memorizzati nel browser per identificare o tracciare l'utente.
   * **Tipologie:**
     + Cookie di sessione: Eliminati alla chiusura del browser.
     + Cookie persistenti: Conservati per un tempo definito.
3. **Cache e Proxy:**
   * **Cache web:** Conserva copie delle risorse per ridurre i tempi di caricamento.
   * **Server proxy:** Agisce come intermediario tra client e server, migliorando sicurezza e prestazioni.
4. **FTP (File Transfer Protocol):**
   * **Funzionamento:** Utilizza due connessioni (una per i comandi, l'altra per i dati).
   * **Modalità:** Attiva (connessione iniziata dal server) e passiva (connessione gestita dal client).
5. **Posta Elettronica:**
   * **Protocolli:**
     + SMTP (invio email).
     + POP3 (scaricamento email).
     + IMAP4 (sincronizzazione email).
   * **Componenti:**
     + UA (User Agent): Client email come Outlook.
     + MTA (Message Transfer Agent): Gestisce la consegna tra server.

**6. DNS e Risoluzione dei Nomi**

1. **Domain Name System (DNS):**
   * **Definizione:** Sistema distribuito per tradurre nomi leggibili in indirizzi IP.
   * **Tipi di record:**
     + A: Indirizzo IPv4.
     + AAAA: Indirizzo IPv6.
     + MX: Server di posta.
     + CNAME: Alias.
2. **Modalità di Risoluzione:**
   * **Ricorsiva:** Il server DNS completa tutto il processo per il client.
   * **Iterativa:** Il client interroga più server fino a ottenere una risposta definitiva.

**7. TELNET e SSH**

1. **TELNET:**
   * Protocollo per l'accesso remoto a sistemi, non sicuro (trasmette dati in chiaro).
2. **SSH (Secure Shell):**
   * Alternativa sicura a TELNET, utilizzata per l'accesso remoto con crittografia.

**8. Conclusioni**

Il livello applicazione è il punto di contatto tra gli utenti e i servizi di rete, responsabile di fornire protocolli standardizzati per la comunicazione e l'elaborazione dei dati. Paradigmi come client/server e P2P hanno rivoluzionato il modo in cui le applicazioni funzionano, rendendo il livello applicazione fondamentale per le moderne infrastrutture di rete. La comprensione dei protocolli e dei meccanismi alla base di questo livello è essenziale per la progettazione e l'ottimizzazione dei sistemi distribuiti.