­­­­­ **LIVELLO APPLICATIVO**

**ICP**: inter process comunication, tecnologie software il cui scopo è consentire ai vari processi in esecuzione di comunicare, anche se risiedono su host diversi.

Vengono utilizzati protocolli standardizzati da **IEFT** (Internet Engineering Task Force).

Lo scambio di messaggi fra processi utilizza i servizi dei livelli inferiori attraverso i SAP (primitive da richiedere al livello inferiore, servizi che possono essere offerti al livello superiore; fa parte del concetto di interfaccia). Ogni processo è associato a un **SAP** (Service Access Point) attraverso un numero (numero di porta).

I livelli inferiori dal trasporto in poi sono controllati dal sistema operativo.

Indirizzare un processo applicativo richiede una coppia di numeri: l’indirizzo dell’host (IP) e il numero di porta. Questa coppia di numeri caratterizza la **socket** (porta di comunicazione). Nella comunicazione in rete ci sarà una coppia di socket, quelle dei due comunicanti.

Il servizio di trasporto del sistema operativo richiede una socket per conto dell’applicazione.

Requisiti delle applicazioni: affidabilità, ritardo, banda.

Protocollo di scambio dati (tipo, sintassi, semantica, tempistiche e modalità).

Architetture applicative:

Server:

* Host sempre attivo
* IP permanente
* Utilizzo di macchine in cluster

Client:

* Connesso in modo discontinuo
* IP variabile
* Non comunicano con altri client

P2P:

* Fortemente scalabile ma difficile da gestire
* Terminali comunicano direttamente
* Non ci sono server sempre connessi e cambiano indirizzo IP

**IL SERVIZIO DI WEB BROWSING - HTTP**

Le pagine web sono costruite a oggetti, hanno un file html base che chiama gli altri oggetti. Ogni oggetto è indirizzato da una **URL** (Uniform Resource Locator).

Il protocollo http trasferisce oggetti generici identificati da una URL.

**Protocollo** **stateless**: nessuna memoria delle richieste viene mantenuta nei server. Non mantiene lo stato della transazione, tratta richiesta per richiesta.

http si appoggia a TCP a livello di trasporto attraverso la porta 80:

1. Client inizia connessione verso il server
2. Server accetta connessione (+ notifica di risposta)
3. Scambio di informazioni (pagine web + messaggi di controllo)
4. Chiusura connessione

Connessione non persistente: Inviato l’oggetto, il server chiude la connessione TCP. Possono essere aperte più connessioni TCP in parallelo per minimizzare il ritardo.

Connessione persistente: rimane aperta per trasferire più oggetti o più pagine web.

* **Without pipelining**: richieste http inviate in serie.
* **With pipelining**: richieste inviate in parallelo, è il default in http v1.1.

**MESSAGGI HTTP:**

Header HTTP: contengono informazioni sulle capacità del browser. Contiene informazioni di servizio come:

* Cache-control: informazioni sulla cache.
* Accept: formati accettati.
* Accept-language: linguaggio accettato / codifica.
* Authorization: permessi del client.
* If-modified-since.
* Tipo di user-agent (il programma-client).
* Connection.

Nei messaggi di risposta non compare l’URL e si aggiunge invece la status line.

**COOKIES**

http è un protocollo stateless, tra i modi per mantenere lo stato ci sono i cookies.

Essi sono delle stringhe alfanumeriche che vengono inseriti negli header dei messaggi di richiesta/risposta http; vengono mantenuti in locale sull’host e sul server.

La prima volta che un client visita un sito con cookies, il server risponde con il messaggio set-cookie.

Possono essere prodotti da:

* Server a cui si richiede il servizio applicativo
* Terze parti (devono essere accettati dall’utente)

Possono essere usati per:

* Identificazione
* Profilazione (devono essere accettati dall’utente)

**PROXY HTTP**

Sono dei dispositivi che instradano a livello applicativo (router http/application gateway). Nascono per rendere più veloce la navigazione, spostano le pagine web dal server originale a un server vicino all’utente che la richiede (proattivi). Essi memorizzano delle parti di pagina web più vicino all’utente, per questo si chiamano cache di rete.

Tutti i client http inoltrano le proprie richieste non al server originale ma tutte verso il proxy, che se contiene la pagina web nella cache la restituisce, sennò la chiede al server. La richiesta al proxy avviene con If-modified-since.

Il server vede le richieste arrivare tutte dal proxy, maschera l’utente (non trasparenza).

Vengono spesso usati per pratiche di DPI (Deep Packet Inspection).

**DIFFERENZE TRA LE DIVERSE VERSIONI DI HTTP**

**HTTP 1.0**: connessioni non persistenti.

**HTTP 1.1**: connessioni persistenti ma seriali, ma un solo oggetto può rallentare tutto perché non sono connessioni parallele.

**HTTP 2.0:**

* Non è testuale ma binario, trasferisce “frame”.
* Multiplazione: una connessione TCP per stream multipli.
* Compressione degli header.
* Servizio di server push.
* Controllo di flusso.
* Due livelli di sicurezza tra l’applicazione e il trasporto.

Tipi di frame:

* DATA: parte di uno stream.
* HEADERS: usata per aprire uno stream.
* PRIORITY: specifica la priorità di uno stream.
* RST\_STREAM: termina uno stream.
* SETTINGS: trasporta parametri di configurazione.
* PUSH PROMISE: gestisce il servizio di push.
* PING, GOAWAY, WINDOW\_UPDATE, CONTINUATION

**HPACK (HTTP 2)**

Gli header possono contenere molti cookies e molte header line per autenticazione oltre a informazioni ridondanti.

* Codifica di Huffman: assegno stringhe binarie a simboli più comuni.
* Indexing: assegno indice a header line più comuni ed invio solo l’indice.
* Codifica differenziale: l’header delle richieste successive riporta solo la differenza dalle richieste precedenti.

**MULTIPLAZIONE (HTTP 2)**

Lo scambio di frame tra client e server è organizzato in stream, cioè sequenze logiche di frame, ognuno con una certa priorità impostata dal browser.

**HTTPS (HTTP 2)**

**SSL** (Secure Socket Layer) e **TLS** (Transport Layer Security) sono a un livello intermedio tra applicazione e servizio TCP.

* Handshake: ci si accorda sulla cifratura. Questa fase comprende lo scambio di certificato tra server e client per verificare l’identità di entrambi. Questo certificato è generato da una Certification Authority e contiene la firma digitale della CA.

Vengono scambiate chiavi simmetriche per le cifrature dei dati, queste chiavi vengono scambiate su una connessione cifrata con chiavi asimmetriche.

C: ClientHello

S: ServerHello – Certificate – ServerHelloDone

C: ClientKeyExchange – ChangeCipherSpec - Finished

S: ChangeCipherSpec - Finished

* Trasferimento dati: i dati sono suddivisi in record (PDU) cifrati secondo le specifiche definite nella fase precedente.
* Chiusura connessione (C: Alert: warning, close\_notify).

**SERVIZIO DI POSTA ELETTRONICA**

Mail Server (router di mail):

* Hanno una coda di email in ingresso (mailbox).
* Una coda di email in uscita.
* Ricevono da altri mail server le mail destinate ai client
* Usano protocollo:

1. **SMTP**: con altri server e con i client d’utente in uplink.
2. **POP3**/**IMAP** con i client d’utente in downlink.

**SMTP**

* È un protocollo applicativo Client-Server.
* Quando server riceve messaggio da un client di cui si occupa:

1. Mette messaggio in coda.
2. Apre una connessione TCP con la porta 25 del server destinatario.
3. Trasferisce mail.
4. Chiude connessione TCP.

* Comandi e risposte sono testuali, richiede che il corpo dei messaggi sia in ASCII. I documenti binari devono essere convertiti in ASCII 7 bit, come specificato in **RFC 822**. È un problema per la sicurezza e occupa molto spazio.

Questo standard richiede anche l’header della mail così impostato:

To:

From:

Subject:

* Similitudini con HTTP: Client-Server, TCP, header testuale.
* Differenze con HTTP: Porta diversa (25, 80), SMTP è tutto testuale, SMTP ha la funzione push.

**MIME (Multipurpose Internet Mail Extensions) RFC 2045 – 2046**

Estende il formato 822 dei messaggi mail per supportare contenuti multimediali. Definisce header per specificare il contenuto (Content-Type) ed il tipo di codifica (Base64).

Consente di trasferire più oggetti come parti di uno stesso messaggio.

**DNS: Domain Name System**

Gli indirizzi IP sono poco adatti ad essere usati dagli applicativi, per questo si utilizzano indirizzi simbolici. Occorre una mappatura fra indirizzi IP e nomi simbolici. Questo servizio è offerto dal protocollo DNS, basato su UDP e operante sulla porta 53.

I server che si occupano del servizio DNS si chiamano name server, hanno organizzazione gerarchica.

Servizi aggiuntivi:

* Host aliasing: permette indirizzi sinonimi per i domini. Esiste il nome “canonico” che è quello principale.
* Mail server aliasing: associa i server delle aziende ai mail server corrispondenti.
* Load distribution: distribuisce le richieste a diversi server replicati che gestiscono i siti più richiesti.

**IL DATABASE DEI NAME SERVERS**

Ogni livello nella gerarchia ha una diversa profondità di informazione.

I Name Server possono essere:

1. Root DNS Server: contengono gli indirizzi dei Top-Level-Domain, sono gestiti da 13 organizzazioni.
2. TLD: (.com)
3. *Generico*: (google.com)
4. Local Name Server: sono direttamente collegati agli host e gestiti dai ISP. Ognuno di essi è collegato ai root.

Authoritative Name Servers (tipologia logica): server responsabile di un particolare hostname. Si definisce autoritativo quando ha la risposta alla richiesta. Contengono record di tipo A dei siti per cui sono autoritativi e record NS per gli altri (ognuno associato a un record A per instradare la richiesta a un altro server).

La richiesta può avvenire in modalità iterativa (ogni passaggio vede l’intervento del LNS) o ricorsiva (LNS coinvolto solo all’inizio e alla fine).

I server hanno anche funzione di cache per dei domini su cui non sono autoritativi. Il TTL è deciso dal server autoritativo ed è un indice di quanto stabile nel tempo è l’informazione relativa, viene usato per decidere il time out. È per questa funzione di cache che di solito si usa il metodo ricorsivo. Di solito i TLD sono memorizzati nel LNS.

**RECORD DNS**

Formato di un record di risorsa (RR) di un database DNS:

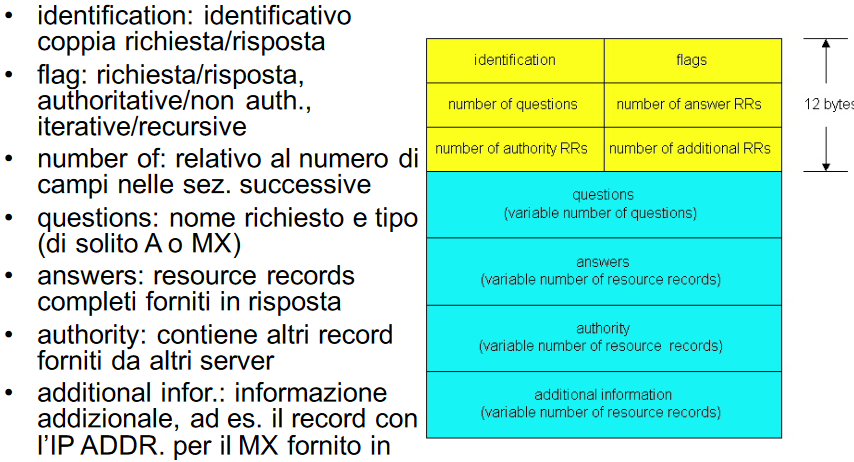
Name | Value | Type | TTL

Ogni messaggio di risposta DNS contiene uno o più RR.

I tipi (che determinano il valore di Name e Value) possono essere:

1. **A**: Name è il nome dell’host e Value è il suo indirizzo IP.
2. **NS**: Name è un dominio (quale foo.com) e Value è l’hostname del DNS server autoritativo che sa come ottenere gli indirizzi IP degli host nel dominio.
3. **CNAME**: Name è un nome alternativo per un host il cui nome canonico è Value.
4. **MX**: Name è dominio di mail e Value il nome del mail server.

**MESSAGGI DNS**



**CDN – Content Distribution Network**

Rete di server geograficamente distribuita che ospita “copie” dei contenuti richiesti.

La società proprietaria del contenuto di solito gestisce anche la rete e di conseguenza le richieste, che possono essere instradate secondo 3 principi:

1. Server più vicino.
2. Percorso più corto.
3. *Personalizzato* (dall’utente).

**ARCHITETTURA PEER-TO-PEER**

I client si connettono in modo intermittente a internet assumendo ogni volta IP diversi.

Viene usato il protocollo HTTP per il download dei file.

Nell’architettura a “directory centralizzata” la ricerca dei contenuti è fortemente centralizzata: il server connette i due peer. Con questo modello il server è un collo di bottiglia per il sistema, e se si tratta di operazioni illegali è facile risalire al gestore della rete.

Nell’architettura distribuita il protocollo è di dominio pubblico, e fa da base a software applicativi diversi. Si basa su un grafo di overlay: i peer si collegano a un numero di peer vicini, e la ricerca viene distribuita.

* Il richiedente per accedere alla rete deve connettersi almeno a un altro peer, quindi fa richieste basate su liste note.
* Il peer connesso inoltra la richiesta (ping) e crea la rete di overlay
* In base alle risposte (pong) la rete si ramifica.

**BITTORRENT**

L’insieme dei peer che partecipano alla distribuzione di un file è chiamato torrent, questa lista viene gestita da un server tracker (a cui tutti devono segnalare la propria attività per aggiornare la lista di torrent attivi). Il peer entrante riceve dal server un Torrent contenente indirizzi IP, quindi stabilisce connessioni TCP con un sottoinsieme dei peer nella lista (neighboring peers), ognuno dei quali fornisce una lista dei chunk posseduti.

I file vengono suddivisi in chunk da 256kbyte.

Rarest First: tra tutti i chunk mancanti, il richiedente scarica prima i più rari nelle liste di chunk ricevute da tutti i peers.

Tit-For-Tat: Il peer entrante risponde a richieste che provengono dagli x-peer che inviano chunk al massimo rate, tutti gli altri sono strozzati. I migliori vengono aggiornati ogni 10 secondi. Ogni 30 secondi viene scelto casualmente un peer per l’invio di chunk (optimistic unchoking).