

Francesco Chiti

Internet: Evoluzione, Architetture e Applicazioni

Indice

1	Introduzione	7
1.1	Motivazioni	7
1.2	Vantaggi connessi alle reti	7
1.3	Evoluzione storica di Internet	7
1.4	Attori e Interfacce di rete	7
1.4.1	Componenti	7
1.4.2	Collegamenti	7
1.4.3	Classificazione delle reti	7
1.4.4	Interfacce	7
1.5	Programmazione delle socket	7
1.6	Modello funzionale a livelli	7
1.6.1	Principi fondativi	7
1.6.2	Vantaggi e svantaggi	7
1.6.3	Principio cross-layer	7
1.7	Standard ISO-OSI	7
1.8	Standard de facto TCP/IP	7
1.8.1	Elementi di rete	7
2	Livello Applicativo	9
2.1	Funzionalità	10
2.2	Livelli Sessione e Presentazione	12
2.3	Evoluzione delle applicazioni Internet e Web	14
2.4	Servizio DNS	20
2.5	Servizio Telnet	20
2.6	Servizio SMTP e POP/IMPAP	20
2.7	Servizio HTTP	20
2.7.1	Standard HTTP/1.0	20
2.7.2	Standard HTTP/1.1	20
2.7.3	Cache e Web Proxy	20
2.7.4	Standard HTTP/2.0	20
2.8	Paradigma Client-Server	20
2.8.1	Cloud, Fog e Edge Computing	20
2.8.2	Content Delivery Networks	20
2.9	Paradigma Peer-to-Peer	20
2.9.1	Protocollo BitTorrent	20
2.10	Social Networks	20
2.11	Blockchain	20

3	Tecnologie Web	21
3.1	Linguaggio HTML	21
3.1.1	Evoluzione storica	21
3.1.2	Principali tag	21
3.1.3	CSS	21
3.1.4	Form	21
3.1.5	XHTML	21
3.1.6	HTML5	21
3.2	Linguaggio Javascript	21
3.3	Linguaggio PHP	21
3.4	Motori di ricerca Web	21
3.5	Web2.0	21
3.5.1	Paradigmi REST e SOAP	21
4	Livello Trasporto	23
4.1	Funzionalità	24
4.2	Socket API	24
4.3	Protocollo UDP	24
4.4	Protocollo TCP	24
4.4.1	Apertura di una sessione	24
4.4.2	Chiusura di una sessione	24
4.4.3	Protocollo Sliding Window	24
4.4.4	Protocolli di ritrasmissione	24
4.4.5	Formato del segmento	24
4.4.6	Adattamento del timeout	24
4.5	Controllo di congestione	24
4.5.1	Motivazione	24
4.5.2	Approccio AIMD	24
4.6	TCP Tahoe	24
4.6.1	ACK clocking	24
4.6.2	Slow-start	24
4.6.3	Congestion Control	24
4.7	TCP Reno	24
4.7.1	Fast Retransmission/Fast Recovery	24
4.8	TCPNewReno e SACK	24
4.9	TCP ECN	24
4.10	TCP Vegas	24
4.11	TCP Westwood	24
4.12	Protocollo RTP	24
4.13	Protocollo RTCP	24
4.14	Protocollo SIP	24
4.15	Mobile IP	24

Prefazione

Introduzione

1.1 Motivazioni

1.2 Vantaggi connessi alle reti

1.3 Evoluzione storica di Internet

1.4 Attori e Interfacce di rete

1.4.1 Componenti

1.4.2 Collegamenti

1.4.3 Classificazione delle reti

1.4.4 Interfacce

1.5 Programmazione delle socket

1.6 Modello funzionale a livelli

1.6.1 Principi fondativi

1.6.2 Vantaggi e svantaggi

1.6.3 Principio cross-layer

1.7 Standard ISO-OSI

1.8 Standard de facto TCP/IP

1.8.1 Elementi di rete

Livello Applicativo

Introduzione

In questo capitolo verranno analizzati i livelli funzionali che caratterizzano entrambi i modelli standard di una rete informatica: il modello *ISO/OSI* ed il modello *TCP/IP*.

Benché lo standard internazionale per la comunicazione sia rappresentato dal modello *ISO/OSI*, esso è utilizzato più come un modello astratto di riferimento. Nella pratica comune lo standard di riferimento è il modello *TCP/IP*, caratterizzato da un numero inferiore di livelli e da una elevata praticità dovuta ad una maggiore facilità nella manutenzione, nell'aggiornamento e più in generale nella gestione dei livelli. I livelli *ISO/OSI* coinvolti nel processo di riduzione sono quel-

Applicazione	Applicazione
Presentazione	
----- Sessione -----	
Trasporto	Trasporto
Rete	Rete
Collegamento Dati	Accesso alla rete (Fisico)
Fisico	

Figura 2.1: Richiamo alla struttura dei modelli *ISO/OSI* (sinistra) e *TCP/IP* (destra).

li che andremo ad analizzare in questo capitolo, definendoli ed approfondendone i principali servizi offerti.

Il *livello Applicativo* (o Applicazione) è il settimo e più esterno livello nel modello

ISO/OSI ed il quarto (o quinto) livello nel TCP/IP. Le diverse accezioni che assume tale strato nei due standard sarà motivo di studio nelle sezioni successive. Prima di andare ad approfondire i servizi offerti dai livelli dei due standard, si tratterà di come si sono sviluppate le *applicazioni Web*, a partire dalla nascita di Internet fino ad arrivare ad oggi, in modo da avere un quadro generale dello sviluppo delle tecnologie che hanno permesso e permettono tutt'ora l'evoluzione della Rete.

2.1 Funzionalità

Il livello Applicazione è un livello astratto che ha la funzione di specificare nel punto logico di accesso a una rete le interfacce ed i protocolli di comunicazione utilizzati e più in generale, di fornire servizi ai processi distribuiti in rete al fine di garantire uno scambio di informazioni.

Quando due programmi devono comunicare tra loro è necessario prima di tutto che venga stabilito un modello di rete, e che esista una connessione logica tra le due entità (mittente e destinatario). Tramite essa si possono gestire congiuntamente i livelli dello *stack TCP/IP* per mezzo di un insieme di istruzioni (o funzioni) denominate *API (Application Program Interface)* che ci permettono di aprire e chiudere connessioni ed inviare e ricevere dati.

Durante una comunicazione ogni strato dello stack aggiunge un header al pacchetto dati che identifica il messaggio da trasmettere e le operazioni che il livello analogo deve svolgere. Questa operazione è chiamata **incapsulamento**. Per mezzo di essa è possibile identificare il dato effettivo e l'header aggiunto dallo strato attuale come un unico pacchetto dati che verrà successivamente passato allo strato sottostante. Il pacchetto risultante dal passaggio dell'intero stack verrà poi trasmesso in rete e successivamente ricevuto. Il computer o l'applicazione ricevente dovrà poi effettuare l'operazione inversa, estraendo ad ogni livello l'header necessario a pilotare le operazioni dei vari strati.

Il livello gerarchico comunemente più alto dello stack, come anticipato, è il livello Applicativo, il quale ha lo scopo di *standardizzare la comunicazione*. Nel TCP/IP infatti, il livello applicativo contiene i protocolli e le interfacce di comunicazione usati nelle trasmissioni processo - processo per completare l'esecuzione dei programmi all'interno di una rete, in cui è adottato il protocollo Internet (IP). Tra i protocolli di questo strato più comunemente usati troviamo HyperText Transfer Protocol (HTTP), File Transfer Protocol (FTP), Domain Name System (DNS) e Simple Mail Transfer Protocol (SMTP). Un esempio di concreto utilizzo si ha ogni volta che viene richiesto al nostro Browser di accedere ad una risorsa Web tramite il processo Browser. Il livello applicativo per completare la richiesta, chiamerà in causa il protocollo HTTP.

Per far arrivare al Client i dati richiesti secondo lo standard TCP/IP, il livello Applicazione deve passare i dati al livello Trasporto attraverso strutture dati logiche chiamate *porte*. L'utilizzo delle porte rende molto più semplice al livello Trasporto, capire che tipo di dato gli sta venendo passato. Se ad esempio venisse

inviato un dato sulla porta 25, significherebbe che il dato riguarda una richiesta e-mail. Se invece fosse stato mandato sulla porta 21 o 22, sarebbe stato un servizio FTP. Poiché sui sistemi dialoganti ci possono essere numerosi processi attivi, l'associazione alla porta giusta è un'operazione critica. Tale funzione è svolta tramite **socket**. Un socket è una *libreria standard* la cui funzione è quella di garantire l'invio e la ricezione dei dati tra Host remoti (se in una rete) o locali (*Inter-Process-Communication*). Nello specifico, nello standard TCP/IP, fungono da intermediari tra il livello Applicazione ed il livello Trasporto, fornendo tra l'altro degli *indirizzi di socket* agli interlocutori di ogni singola comunicazione, contenenti l'indirizzo e la porta della controparte.

Si possono distinguere tre diversi tipi di socket in base al tipo di connessione. Gli **Stream Socket** sono basati su protocolli come il TCP e garantiscono una comunicazione affidabile di tipo full-duplex, orientata alla connessione (*connection oriented*) e caratterizzata da un flusso di byte di lunghezza variabile. I **Data-gram Socket** sono basati su *User Datagram Protocol (UDP)*, un protocollo alternativo al TCP/IP che garantisce comunicazioni a bassa latenza (comunemente utilizzate per videochat) e prive di controllo del flusso. I **Raw Socket** (raw IP) hanno la caratteristica di "saltare" il livello di Trasporto, e rendere l'header accessibile direttamente al livello Applicativo. Tali indirizzi di socket sono infatti privi di una porta associata, ma contengono solo l'indirizzo dell'interlocutore.

Il livello Applicazione nel TCP/IP non specifica nessun tipo di regola o formato di dati accettabili, che le applicazioni devono invece tenere in conto. Per questo motivo nella prima stesura dello standard TCP/IP venne fortemente raccomandato di seguire il **principio di robustezza** (o legge di Postel), che recita:

"Be conservative in what you do, be liberal in what you accept from others."

In altre parole, quando viene inviato un messaggio ad un qualsiasi destinatario risiedente in un Host differente da quello del mittente, questo deve essere perfettamente conforme alla specifica richiesta. In caso, invece, di messaggi in entrata non conformi, questi dovranno essere accettati qualora il loro significato fosse chiaro.

Nel modello ISO/OSI il livello Applicativo è costituito da un insieme di protocolli ed interfacce di comunicazione tra i quali HTTP, FTP, DNS e SMTP. Una sostanziale differenza, però, è nella struttura stessa del modello. Il livello Applicativo del TCP/IP comprende sia il livello Applicativo che i livelli Presentazione e Sessione del modello ISO/OSI. Questa maggiore modularità degli strati dello standard ISO/OSI è dovuta ad una più precisa suddivisione delle funzioni tra essi. Il livello Applicazione di tale modello, lavora infatti direttamente col software applicativo e si occupa "solo" di verificare l'esistenza di un Host per la comunicazione del flusso dati e associarlo ad un *processo di rete*. Una volta appurata la presenza e la disponibilità di risorse su un determinato Host, il pacchetto dati composto dal dato vero e proprio e dall'header aggiunto dal livello Applicazione, è passato al sottostante livello Presentazione.

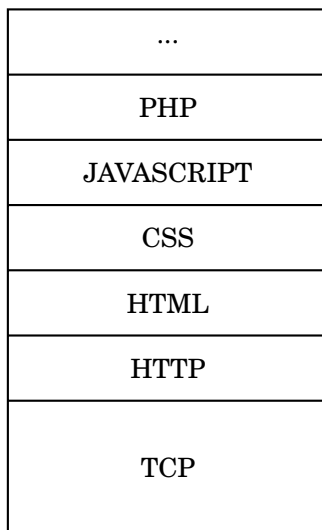
2.2 Livelli Sessione e Presentazione

Il livello Presentazione è il sesto livello del modello ISO/OSI ed è il primo livello ad occuparsi della semantica delle informazioni trasmesse. Ha la responsabilità di assicurare la compatibilità dei dati durante una comunicazione tra entità che utilizzano codifiche diverse. Questo avviene tramite una traduzione del dato in una forma universale e, quindi, comprensibile da altri Host. Nello specifico: una volta iniziata la comunicazione in linea di principio, cambia il formato dei dati da quello del dispositivo mittente (*sintassi locale*), a quello del dispositivo destinatario (*sintassi di trasferimento*)

Questo strato offre, inoltre, importanti servizi quali la formattazione, la codifica, la crittatura e la compressione di dati. Se, ad esempio, si volesse convertire un file di testo scritto in EBCDIC (Extended Binary Coded Decimal Interchange Code) in un file scritto in ASCII (American Standard Code for Information Interchange), tale conversione avverrebbe a questo livello. Un altro esempio si ha quando si invia un file di testo dal contenuto sensibile (come una password), ed è dunque necessario che il messaggio venga crittato. Tale operazione avviene proprio al livello Presentazione.

Una volta finita l'elaborazione del pacchetto dati questo viene passato allo strato inferiore, il livello Sessione.

Il livello Sessione è il quinto livello del modello OSI e fornisce i meccanismi per stabilire, gestire ed, infine, concludere una comunicazione (*o sessione*) tra una applicazione locale ed una remota, svolgendo inoltre un'operazione di verifica per assicurarsi della corretta ricezione del pacchetto dati. È solo a questo punto dello stack, infatti, dopo che il livello Applicativo e Presentazione hanno fornito i protocolli necessari ed adattato il pacchetto dati alla comunicazione che avviene la connessione, permettendo agli strati inferiori di attuare l'*effettivo* invio del pacchetto dati. In base alla durata della sessione è possibile classificare le diverse tipologie di comunicazione. Una trasmissione si definisce **simplex**, quando la sessione dura al massimo il tempo necessario ad inviare un solo messaggio in una direzione. Un'altra modalità è la **half-duplex**, e si ha quando la trasmissione è sufficientemente lunga per una comunicazione bidirezionale ma capace di gestire un solo messaggio alla volta. L'ultima e più recente modalità è la **full-duplex**, la quale permette una comunicazione bidirezionale e simultanea. Nel modello TCP/IP, tuttavia, questo livello non è presente e le modalità di instaurazione di una comunicazione tra due entità possono risultare più imprevedibili. Il livello Sessione è stato inglobato in parte nel livello Trasporto ed in parte nel livello Applicativo. Quando si va a richiedere una pagina Web tramite un Browser, nel TCP/IP i protocolli necessari vengono impilati direttamente sopra HTTP (**add-on**). Questa *decomposizione funzionale* è stata applicata per far fronte alla necessità di integrare nuovi protocolli sugli stessi sistemi, senza che essi debbano essere stravolti, rendendo invece i nuovi protocolli compatibili con quelli antecedenti. Se, ad esempio, dovessimo caricare una pagina Web per guardare un programma in streaming, i protocolli necessari alla visualizzazione di essa verrebbero impilati sopra al livello Applicativo, partendo dal più semplice



...
PHP
JAVASCRIPT
CSS
HTML
HTTP
TCP

Figura 2.2: Stratificazione del livello Applicativo in caso di comunicazione Real-Time.

ed anziano HTTP, fino ad arrivare ai più recenti Javascript e PHP. (Figura 2.2). A livello Sessione dell'ISO/OSI vengono gestite anche le funzioni di autenticazione e autorizzazione e viene fornito il servizio di *session restoration (checkpointing and recovery)*. Questo proverà, in caso di perdita di connessione, a ripristinarla riavviando se necessario la comunicazione.

Un esempio dei servizi offerti dal livello Sessione si ha durante una video chiamata, durante la quale è essenziale che il video e l'audio siano sincronizzati per evitare *problemi di lip synch* (problemi di sincronizzazione del movimento delle labbra con l'audio). In caso di perdita di tale proprietà, essa verrà opportunamente gestita.

Nel modello TCP/IP una sessione è gestita invece in tre fasi. La prima è quella di **Session Starting** (inizio sessione) nella quale il protocollo TCP adotta una *Three-Way-Handshake* (Figura 2.3) tra Client e Server per inizializzare la connessione. Più precisamente, come prima cosa il Client invia un messaggio *SYN* (Synchronize) al Server per richiedere una sessione, il quale risponderà con un messaggio *SYN ACK* (Synchronize Acknowledgment) per confermare la sincronizzazione e chiedere al Client se ha effettivamente richiesto una sessione. Infine il Client risponderà con un ultimo messaggio *ACK* per confermare la richiesta di connessione ed avviare così la fase di trasmissione dei dati: **Data Transmission**. L'ultima fase è quella di **Session Ending** (fine sessione), che avviene tramite una *Four-Way-Handshake* (Figura 2.3) che inizia ad esempio con un messaggio **FIN ACK** (Finished) da Server a Client per avvisare quest'ultimo che i dati richiesti sono stati completamente trasmessi. A questo punto il Client risponderà con un messaggio **ACK** per confermare la ricezione del messaggio precedente. Una volta completato il download dei dati, il Client invierà un messaggio **FIN**

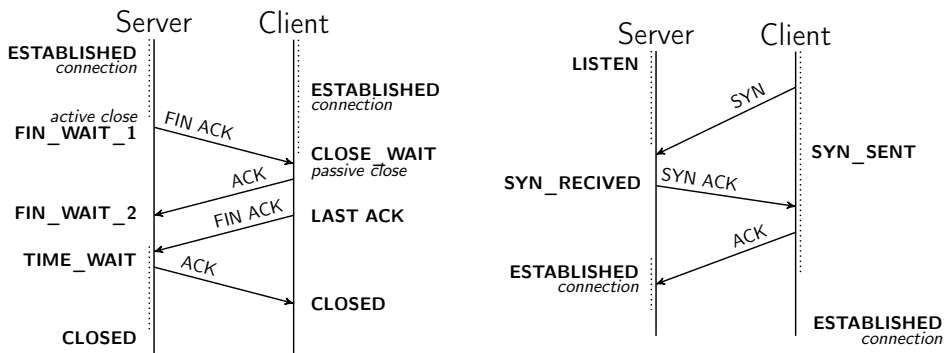


Figura 2.3: Three Way Handshake (destra) e Four Way Handshake (sinistra)

ACK per avvisare il Server di aver terminato, il quale, provvederà a chiudere finalmente la connessione con un ultimo messaggio **ACK** di conferma di ricezione. UDP, l'alternativa a TCP/IP, è anch'esso caratterizzato da una sessione (se così si può chiamare) molto semplice. Quando, infatti, un'applicazione utilizza il protocollo UDP per inviare dei dati, i pacchetti sono semplicemente inviati senza che vi sia effettuato nessun tipo di controllo di corretta ricezione o di disponibilità del destinatario. Se quest'ultimo perdesse qualche pacchetto, questi sarebbero difatti semplicemente persi ed il mittente non provvederebbe ad inviarli nuovamente. Il lato positivo di questa tecnologia si ha dal punto di vista della velocità di comunicazione, incrementata dalla totale mancanza di controlli e garanzie. UDP è infatti utilizzato là dove la velocità di comunicazione è fondamentale, mentre la presenza e correzione degli errori non è rilevante. Se ad esempio voi steste giocando ad un videogame online, è molto probabile che il protocollo a gestire il traffico dei pacchetti dati sia proprio UDP. Se quest'ultimo durante la trasmissione, per qualsivoglia motivo, non riuscisse a recapitare al vostro Client qualche pacchetto, comporterebbe un blocco momentaneo del gioco, al termine del quale il vostro personaggio potrebbe ritrovarsi teletrasportato all'improvviso in un diverso punto della mappa. Non essendoci, infatti, nessuna forma di *session restoration*, il flusso di dati continuerà semplicemente a scorrere, ignorando l'errore.

2.3 Evoluzione delle applicazioni Internet e Web

Dalla nascita di *Advanced Research Projects Agency Network (ARPANET)* a cavallo tra gli anni '60 - '70 si sviluppò un'architettura per le reti di telecomunicazioni, che sarebbe poi stata indicata come Internet. Nel primo periodo della sua vita, la Rete, era in una *fase di incubazione*, durante la quale veniva utilizzata solo dall'esercito con lo scopo di favorire l'efficienza nello sviluppo di nuove tecnologie di comunicazioni militari. A seguito di tale periodo e con il crescente interesse da parte della comunità scientifica, che si era occupata della sua progettazione e sviluppo, l'utilizzo della Rete venne esteso alle principali università

americane, le quali, tramite ad esempio l'innovativo servizio di *posta elettronica*, avevano la possibilità di scambiare rapidamente documenti. La Rete si trovava allora nella *fase accademica*. Negli stessi anni *Tim Berners-Lee* creò un'architettura (e un servizio) che semplificò drasticamente l'utilizzo della Rete, ormai rinominata Internet, e ne rendeva possibile lo sfruttamento a fini commerciali, avviando difatti la sua *fase commerciale*: il **World Wide Web (WWW)**. Nel 1991 lo stesso Tim Berners-Lee, creò il primo sito Web, avviando implicitamente una rivoluzione che oggi chiamiamo **Web 1.0**.

Il Web 1.0 si può classificare come l'Internet dei contenuti, caratterizzato da siti Web semplici e statici dai quali era possibile solo accedere a risorse senza poterle modificare o aggiungere. Le pagine Web erano scritte da una ristretta cerchia di persone e presentavano molti collegamenti ipertestuali statici (hyper link) ad altre pagine, in modo da dare all'utente una certa libertà di movimento all'interno della Rete (*surfing*). È col Web 1.0 che nacque il concetto di *Web application*, cioè una variante del paradigma di tipo Client-Server che viene eseguita dal Client tramite un *Web Browser*. Nel Web 1.0 il Client, rappresentato dal Browser, accedeva a pagine Web statiche rese accessibili da semplici Web Server. Infatti, come si nota dalla Figura 2.4, nella struttura della Rete non vi era nessun tipo di *Web Database*, né tanto meno qualsivoglia tipo di servizio I Browser utilizzati

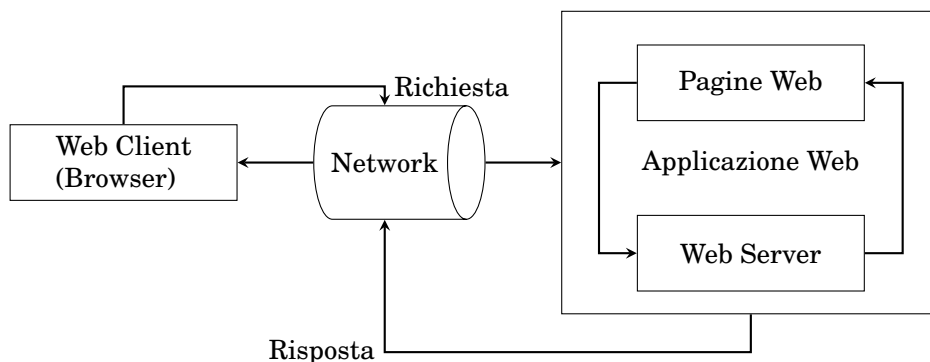


Figura 2.4: Modello semplificato della struttura del Web 1.0

per la navigazione erano molto semplici poiché l'unico linguaggio che dovevano interpretare era l'HTML e non era volutamente richiesta nessun tipo di compilazione. Nel Web 1.0 non era inoltre prevista in alcun modo, la separazione tra i dati e la loro rappresentazione a livello di codice HTML, rendendo problematica la gestione, la manutenzione e l'aggiornamento. Se, ad esempio, doveva essere cambiata anche solo il font, si doveva andare a modificare l'intera pagina.

Le sessioni erano gestite soltanto dal protocollo HTTP, tutt'oggi ancora utilizzato. Questo ha la caratteristica di essere un protocollo *privo di stato* (*state-less*) che, cioè, tratta ogni coppia richiesta-risposta tra Client e Server in modo indipendente dalle altre, causando di conseguenza la perdita di qualsiasi informazione scambiata all'interno di una ipotetica transazione. Ciò rendeva impossibile ad

esempio, qualsiasi applicazione commerciale. Tale problema venne risolto inizialmente con l'introduzione dei **cookies**. Questi sono piccole stringhe di testo che vengono inviate da un Web Server ad un Web Client (solitamente un Browser) alla loro prima interazione, per poi essere rimandate indietro dal Client al Server ogni volta che il Client accederà alla stessa porzione dello stesso dominio Web. I cookies sono tutt'oggi molto utilizzati, essendo essi stati resi compatibili con le nuove strutture della Rete. Questo fenomeno di integrazione causò che le nuove architetture dovessero rispettare il principio di *backward compatibility*. Tale principio prevede la possibilità di far cooperare tecnologie precedenti e successive, in modo che una eventuale innovazione non debba necessariamente portare ad una pesante manutenzione.

All'aumentare del numero dei servizi, degli utenti e dei loro requisiti, si rendeva sempre più necessaria la capacità di interagire con i contenuti stessi. Il mutamento fu veicolato da dei nuovi linguaggi di programmazione come *PHP*, per mezzo dei quali iniziarono a crearsi i primi *blog*, cioè siti Web sui quali era possibile inserire dei commenti. Questo significativo cambiamento comportò l'avvento delle prime *community* e venne identificato come **Web 1.5**.

Di lì a pochi anni la Rete si espanse in modo esponenziale con l'introduzione dei *Wiki* e dei *Social Network*, ponendo in primo piano l'interattività con l'utente ed aprendo la strada al cosiddetto **Web 2.0**.

Il livello applicativo, da un punto di vista funzionale, da statico divenne quindi dinamico, apportando modifiche architettrali principalmente lato Server, rendendolo cioè decisamente più complesso. Non si avevano più semplici pagine Web statiche, ma pagine Web più complesse e dinamiche, dotate di un elevato numero di servizi e script costantemente in comunicazione tra loro, per mezzo di specifici connettori, principalmente con *database* (Figura 2.5).

Due dei servizi più innovativi furono gli *RSS* ed i *Podcast*. L'*RSS* (Really Simple Syndication) è uno dei più popolari formati di distribuzione dei contenuti web basato su XML (eXtensible Markup Language) per mezzo del quale divenne possibile, tramite il meccanismo di feed-RSS, monitorare gli aggiornamenti di un sito senza visitarlo. Quest'ultima tecnologia rappresentò un vero punto di svolta nello sviluppo della Rete, in particolare per quanto riguarda la crescita dei *social network*. Per mezzo di essa, i siti Web guadagnarono la capacità di ricevere e pubblicare continui aggiornamenti, come ad esempio la visualizzazione dei commenti associati ad un post di *Facebook*. La funzionalità dei *Podcast* è molto simile ma, anziché monitorare ed aggiornare continuamente file di tipo testo o immagini, tramite tale meccanismo fu possibile espandere il servizio ai file audio e video.

Conseguentemente all'evoluzione del lato Server, i Browser divennero più complessi eliminando le incompatibilità che vi erano in passato con le Web Application ed offrendo una vasta gamma di servizi adesso imprescindibili. Un esempio è rappresentato dall'introduzione di *JavaScript*, per mezzo del quale si rese possibile eseguire codici direttamente sul Browser (*scripting*). Un altro importante

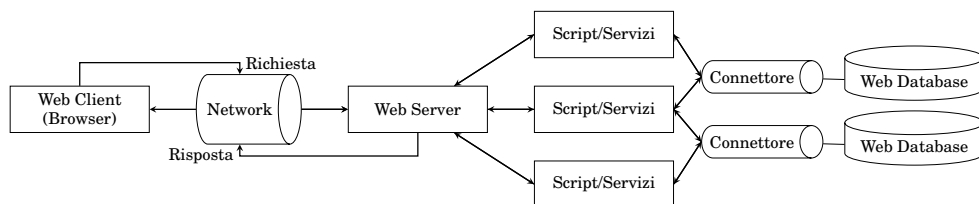


Figura 2.5: Modello semplificato della struttura del Web 2.0 e 3.0.

cambiamento fu quello portato da **HTTP/2**. Il protocollo HTTP non era infatti ancora mai stato cambiato dalla sua prima stesura (HTTP 1.1) del 1997, e col Web che si vedeva completamente rivoluzionato divenne una priorità avere un protocollo adatto alle nuove esigenze della Rete. HTTP/2 espanse il precedente protocollo senza cambiare le sue fondamenta, aggiungendo nuove funzioni e rendendo il processo di incapsulamento e trasferimento dei messaggi HTTP tra Client e Server, molto più veloce. In generale HTTP/2 non modificò la semantica di HTTP/1, ma modificò il modo in cui i dati venivano frammentati (*framed*) in modo da diminuire la latenza. Questo avvenne con l'aggiunta del nuovo livello di *binary framing*, posto tra l'interfaccia socket e l'API HTTP che, oltre a ridefinire il protocollo stesso da *testuale* (textual) a *binario* (binary), suddivise tutte le comunicazioni HTTP/2 in messaggi e frame più piccoli, ognuno dei quali codificato in formato binario.

Con HTTP/1 se un Client voleva eseguire più richieste parallelamente per incrementare la velocità, era obbligato ad utilizzare più *connessioni TCP* a causa della struttura stessa del modello di consegna di HTTP/1, il quale assicurava la consegna di una sola risposta alla volta per ogni connessione gestite tramite un'unica coda. Come conseguenza diretta si aveva un blocco della linea ed una inefficienza nell'uso della connessione TCP sottostante. Con l'arrivo di HTTP/2 ed il suo nuovo livello di framing queste limitazioni vennero superate e per mezzo della nuova tecnologia di *multiplexing* divenne possibile per Client e Server, dividere un messaggio HTTP in frame indipendenti, di intervellarli e di ricomporli all'altro capo.

Col Web 2.0 si registrò un fortissimo aumento degli utenti e dei contenuti pubblicati. Basti infatti pensare al fatto che ogni post pubblicato da qualcuno su un qualsiasi social network rappresenta effettivamente un contenuto reperibile da chiunque su Internet.

Passando dal Web 1.0 al Web 2.0 ed in seguito ad una crescita esponenziale di utenti e di contenuti (Figura 2.7), si è venuto a creare un problema: se prima su Internet scarseggiavano i contenuti, questi divennero sovrabbondanti e si pose il problema della ricerca semantica dei contenuti stessi. Nella prima parte di vita del Web, i motori di ricerca erano molto semplici, poiché semplici erano le richieste che li potevano essere sottoposte. Con la crescita della Rete e l'ampliamento della tipologia e della mole dei contenuti reperibili su di essa, le esigenze degli

utenti cambiarono, rendendo necessario un adeguamento dei motori di ricerca, mirato a restituire all'utente una corrispondenza sempre migliore. Fu così che gli algoritmi di ricerca sul Web fecero un primo passo verso l'*intelligenza artificiale*.

Il **Web 3.0**, il *Web of Things (WoT)*, è quello che utilizzeremo nel futuro immediato ed è basato sul paradigma dell'*intelligenza semantica* ad oggi già in evoluzione. Se vi è mai capitato di utilizzare un assistente virtuale, o un qualsiasi oggetto di uso comune dotato di una connessione ad Internet, siete già entrati in contatto col WoT. Il concetto generale è infatti quello di interconnettere utenti umani ed oggetti (**things**), in tutto il mondo, attraverso la Rete, caratterizzando ogni soggetto con uno specifico URL (o indirizzo di Rete). In questa fase, il Web è rappresentato da un ambiente in cui ogni file pubblicato (pagine HTML, immagini, video, testi ecc...) entra *concettualmente* a far parte di un enorme database, il *Web Database* nel quale ogni dato è interpretabile tramite *metadati* associati ad esso che ne specificano il contesto semantico in un formato adatto all'interrogazione, all'interpretazione e, più in generale, all'elaborazione automatica intelligente. L'*intelligenza* è infatti la caratteristica fondamentale di questa fase. Tramite algoritmi sempre più sofisticati che sfruttano l'intelligenza artificiale saranno sintetizzate a run time applicazioni capaci di interagire e collaborare tra loro e con gli utenti e di avere una vera e propria *coscienza del contesto*, sfruttando la potenza della semantica. Si pensi, ad esempio, a cosa potrebbe accadere se un'applicazione calendario ed una di posta elettronica fossero in grado di comunicare tra loro in completa autonomia. Ogni volta che venisse fissato un appuntamento tramite e-mail, il calendario potrebbe aggiornarsi da solo, tramite le informazioni comunicategli dall'applicazione di posta elettronica.

Per far sì che ciò sia possibile, il Web 3.0 si basa sul concetto che tutte le fonti siano codificate secondo gli stessi criteri e che quindi tutti i documenti condividano la lingua con cui sono scritti.

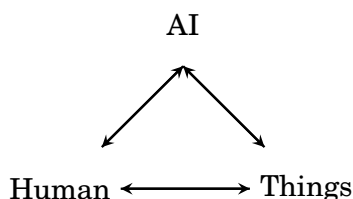


Figura 2.6: Interfacce ed interazioni tra esse nel Web 3.0

Tuttavia, essendo la Rete in continuo divenire, sullo sviluppo del Web 3.0 si possono fare solo congetture. Tim Berners-Lee vede il Web 3.0 come una rete raggiungibile da tutti, senza barriere, che tramite la semantica possa generare applicazioni Web nel complesso più efficaci di qualsiasi altra singola applicazione

mai creata. Questa potenzialità, oltre che dal punto di vista della semantica è intesa anche dal punto di vista grafico, dove la *grafica vettoriale scalabile* (SVG), secondo Tim Berners-Lee, prenderà il sopravvento. Per mezzo di essa è possibile esprimere figure interattive che possono essere ridefinite in qualsiasi momento ed in qualsiasi punto, senza perdere un grammo di qualità.

Sicuramente col progredire del WoT, l'intera struttura del Web tenderà a mutare, portandoci in un futuro dove la Rete sarà costituita da utenti, *things*, ed *intelligenze artificiali* (IA) sempre più simili ad utenti reali (Figura 2.6). In una tale visione del futuro le interazioni tra questi tre soggetti saranno del tutto inedite. Si pensi ad esempio ad una IA che riceve feedback ed invia direttive ad un oggetto connesso alla rete in modo del tutto autonomo ed intelligente, o che crea contenuti su Internet, ad esempio su un social network, in modo indistinguibile da un essere umano.

È interessante pensare in questa prospettiva, a come si evolveranno i social network. Con l'avvento di ulteriori soggetti su Internet oltre alle persone fisiche, non sarebbe fuori luogo pensare a social network composti prevalentemente da IA e things. Ci sono tuttavia persone che teorizzano si tenderà a conferire ai contenuti più gradi di libertà, non avendo più pagine ma **spazi** in cui poterci muovere tridimensionalmente per trovare ciò che cerchiamo.

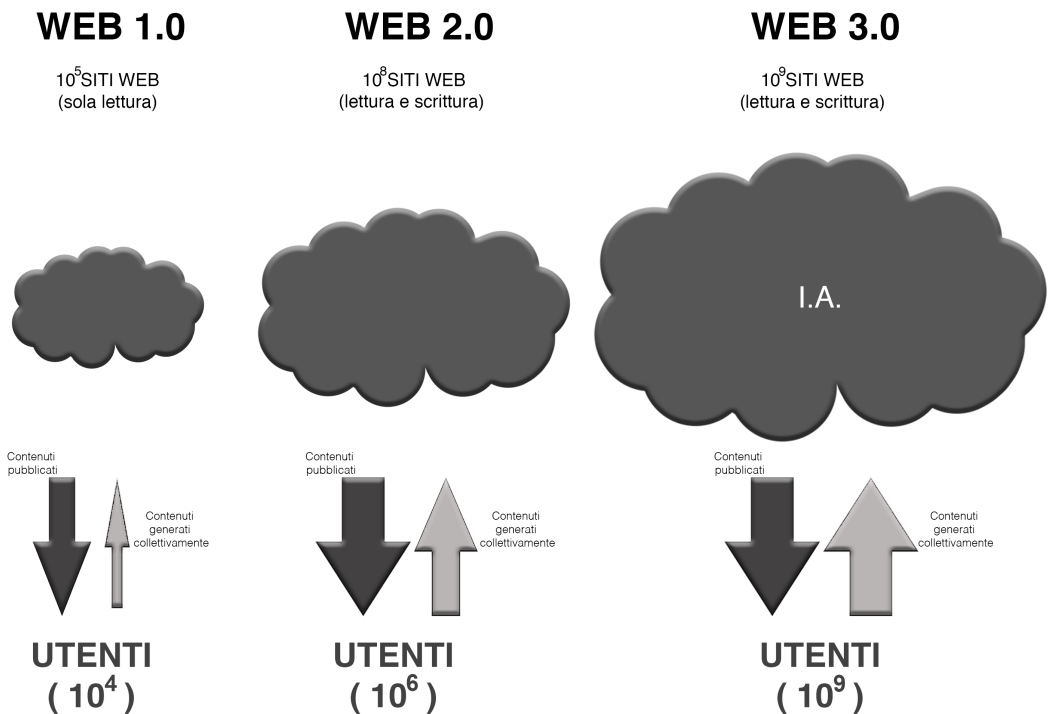


Figura 2.7: Crescita della rete Internet

2.4 Servizio DNS

2.5 Servizio Telnet

2.6 Servizio SMTP e POP/IMPAP

2.7 Servizio HTTP

2.7.1 Standard HTTP/1.0

2.7.2 Standard HTTP/1.1

2.7.3 Cache e Web Proxy

2.7.4 Standard HTTP/2.0

2.8 Paradigma Client-Server

2.8.1 Cloud, Fog e Edge Computing

2.8.2 Content Delivery Networks

2.9 Paradigma Peer-to-Peer

3

Tecnologie Web

3.1 Linguaggio HTML

3.1.1 Evoluzione storica

3.1.2 Principali tag

3.1.3 CSS

3.1.4 Form

3.1.5 XHTML

3.1.6 HTML5

3.2 Linguaggio Javascript

3.3 Linguaggio PHP

3.4 Motori di ricerca Web

3.5 Web2.0

3.5.1 Paradigmi REST e SOAP

4

Livello Trasporto

4.1 Funzionalità

4.2 Socket API

4.3 Protocollo UDP

4.4 Protocollo TCP

4.4.1 Apertura di una sessione

4.4.2 Chiusura di una sessione

4.4.3 Protocollo Sliding Window

4.4.4 Protocolli di ritrasmissione

4.4.5 Formato del segmento

4.4.6 Adattamento del timeout

4.5 Controllo di congestione

4.5.1 Motivazione

4.5.2 Approccio AIMD

4.6 TCP Tahoe

4.6.1 ACK clocking

4.6.2 Slow-start

4.6.3 Congestion Control

4.7 TCP Reno

4.7.1 Fast Retransmission/Fast Recovery

4.8 TCPNewReno e SACK