Capitolo 2

1) Quali sono le differenze tra il paradigma “Client-Server” e il “peer to peer”?

Il paradigma peer-to-peer è molto utilizzato, tutti conoscono i programmi per la condivisione dei file (File sharing).

File sharing: ciascun peer può distribuire qualsiasi porzione del file abbia ricevuto aiutanto in questo modo il server nel proceso di distribuzione, ciò vuol dire che un peer potrà riferirsi ad un altro peer per avere un a certa porzione del file e non dover chiedere per forza al server.

La differenza sostanziale con il paradigma client-server e che non è chiarito a priori chi è il client e chi è il server, quindi in una rete P2P tutti gli host sono client e server nello stesso momento.

Grazie al modello P2P, i computer possono comunicare e condividere i file e altre risorse, invece di passare attraverso un server centralizzato. Ciascun computer (nodo) è responsabile del passaggio dei dati alle altre macchine.

Le connessioni non nascono spontaneamente, ma devono essere richieste da una delle parti in causa.

A differenza dell'approccio client-server, chi richiede la connessione ***non*** è ad un livello gerarchico inferiore; infatti entrambi i partecipanti sono alla pari (peer).

In una architettura peer to peer, i computer che sono stati tradizionalmente usati solo come client, comunicano tra di loro e possono agire sia da client che da server, assumendo un ruolo che è molto efficiente per la rete. Questo riduce notevolmente il carico sui server.

Dopo la connessione tra i peer, ha inizio la comunicazione tra i nodi che avviene tramite uno scambio di messaggi.

I messaggi servono a:

- segnalare la propria presenza sulla rete
- chiedere una o più risorse risorsa
- servire la richiesta di una o più risorse
- trasferire le risorse

2) Quali sono le principali differenze tra il protocollo HTTP e il protocollo SMTP?

Sia SMTP che HTTP utilizzano connessioni persistenti TCP su porte ovviamente diverse.

SMTP mette tutti gli oggetti in un unico messaggio, mentre HTTP incapsula ogni oggetto in un messaggio.

SMTP trasferisce solo oggetti di tipo posta elettronica, mentre HTTP oggetti in generale.

HTTP viene utilizzato per scaricare oggetti caricati su un server (perché è un pull protocol), mentre SMTP serve a caricare oggetti su un server (dal server di posta di invio al server di posta di ricezione, dallo user agent al server di posta di invio essendo SMTP un push protocol).

3) Quali sono le differenze tra HTTP e SMTP nella gestione di un documento che contiene sia testo che immagini?

SMTP converte tutto in ASCI a 7 bit e sfrutta il protocollo MIME per il file multimediale e poi incapsula in pacchetti TCP, mentre HTTP incapsula il tutto in pacchetti TCP senza effettuare nessuna conversione.

4) Per quale motivo in HTTP sono necessari i cookie? Descrivere cosa sono e come si utilizzano attraverso un semplice esempio.

HTTP è un protocollo STATELESS quindi i Server non devono occuparsi di mantenere lo stato di ciascuna connessione corrente, *tuttavia* potrebbe risultare utile per il Web Server, in diverse situazioni, mantenere delle informazioni e quindi autenticare gli utenti.

Autenticare gli utenti nella fase di richiesta HTTP è possibile mediante uno strumento chiamato ***Cookie*** molto utilizzato dai siti odierni.

Il Cookie è un identificativo, puntatore di informazioni che conserva lo stato, è implementato a livello applicativo e viene immagazzinato nel Client. Esso consente al Server di tenere traccia degli utenti.

Per Esempio se un utente visita per la prima volta un sito che utilizza cookies, l’utente può fornire un’identificazione. Sucessivamente il browser passa un’intestazione di cookie al

Server durante le successivi visite al sito, identificando quindi l'utente sul server.

La tecnologia dei cookie [RFC2965] presenta diverse componenti:

- Una riga di intestazione nel messaggio di risposta HTTP.
- Una riga di intestazione nel messaggio di richiesta HTTP
- Un file cookie mantenuto sul sistema terminale dell'utente e gestito dal browser dell'utente.
- Un database sul sito.

5) Per le applicazioni P2P, siete d'accordo con l'affermazione "Non esiste la nozione di lato client e server in una sessione di comunicazione"? Perche?

Si, nell'architettura P2P non vi è una differenza tra client e server, i peer possono fungere sia da client che da server. In questo modo si riduce notevolmente il carico sui server.

6) Supponete di voler fare una transazione da un client remoto a un server il più velocemente possibile. Userete TCP o UDP?

UDP perché non ha bisogno di stabilire una connessione con handshaking a tre vie, che infatti comporta un aumento del tempo di trasferimento di un file.

7) Quali informazioni usa un processo in esecuzione su un host per identificare un processo di un altro host?

L'indirizzo IP dell'host con cui vuole comunicare e la socket dell'host (porta).

8) Che cosa si intende per protocollo che fa uso di handshaking?

Protocollo che offre un servizio orientato alla connessione in cui, come per esempio il protocollo TCP, sia Client e Server possono scambiarsi messaggi di controllo a livello di trasporto in modo da poter poi stabilire una connessione (per esempio TCP) per poter comunicare poi in base alle richieste fatte dal client.

9)Perchè HTTP,FTP,SMTP,POP3 fanno uso di TCP e non di UDP?

Questo perché TCP offre un servizio affidabile, cioè che il trasporto dei dati avverrà in maniera corretta senza perdite o diversi ordini di arrivo. UDP non offre nessuna garanzia.

10) Perchè di dice che FTP invia informazioni di controllo "fuori banda"?

FTP usa due tipi di connessioni TCP parallele: connessione controllo, connessione dati. E dato che ha una connessione di controllo separata si dice che invia informazioni fuori banda.

11) Supponete che Alice, con un account di posta elettronica basato sul Web (quale hotmail o gmail) invi messaggio a Bob che accede alla propria casella sul proprio server usano POP3. Descrivere come il messaggio giunge dall'host di Alice a quello di Bob. Assicuratevi di elencare la serie di protocolli a livello di applicazione usati per trasferire il messaggio tra i due host.

I messaggi di posta elettronica vengono scambiati tramite protocollo di applicazione SMTP che utilizza il protocollo dello strato di trasporto TCP.

Vi sono 2 componenti fondamentali nei nei messaggi di posta elettronica: *User Agent* (colui che si occupa dell'inoltro dei messaggi, della lettura e scrittura di essi da parte dell'utente) e *Mail server* (colui che prende in coda i messaggi da inoltrare al Mail server di un altro utente oppure che prende in coda i messaggi arrivati da un Mail server di un altro utente).

Alice è il mittente e tramite il suo User Agent invia il messaggio che vuole far ricevere a Bob, questo messaggio andrà in coda al Mail server di Alice, il quale utilizza il protocollo SMTP per trasferire il messaggio al Mail server di Bob.

Ora che il messaggio è arrivato nel Mail server di Bob, Bob deve poterlo leggere; Bob però non può utilizzare il protocollo SMTP per prendere il messaggio perché il protocollo SMTP è un protocollo di tipo PUSH e non PULL, ecco perché utilizza POP3.

Tramite il protocollo POP3 può prendere il messaggio inviato da Alice:

1) User Agent di Bob dovrà fornire username e password per poter accedere ai messaggi del Mail server di Bob.

2) User Agent reuperà i messaggi
3) Messaggio di quit e quindi conclusione della sessione.

12) Dal punto di vista dell'utente, qual è la differenza tra le modalità "scarica e cancella" e "scarica e mantieni" in POP3?

Nella modalità "scarica e cancella" un utente non potrà andare a rileggere un messaggio ricevuto con un altro dispositivo che non sia quello che ha utilizzato per leggere il messaggio la prima volta; invece con la modalità "scarica e mantieni" è possibile.
Per esempio se Bob vuole leggere messaggio che gli ha inviato Alice da tablet invece che dal suo pc, con la modalità "scarica e mantieni" può tranquillamente farlo.

13) É possibile per il web server e per il mail server di un'azienda avere esattamente lo stesso sinonimo di un hostname (per esempio, foo.com)? Quale sarebbe il tipo di RR che contiene l'hostname del server di risposta?

Si, può essere fatto utilizzando il type = MX che consente agli hostname dei mail server di avere sinonimi semplici (per esempio: foo.com,mail.bar.foo.com,MX)

14) Supponete che Alice fornisca in BitTorrent chunk di un file a Bob durante tutto un intervallo di 30 secondi. Bob ricambierà necessariamente il favore inviando ad Alice dei chunk nello stesso intervallo?

Se Alice sta inviando a Bob dei chunk rari allora Bob invierà potrebbe con maggiori probabilità inviare dei chunk ad Alice, dato che ogni 10 sec sceglie 4 peer a cui mandare chunk. Ma potrebbe sempre capitare che Alice non faccia parte di quei 4 peer e quindi nei 30 non riceva nessun chunk da Bob.

15) Considerate un nuovo peer, Alice, che entra a far parte di BitTorrent senza aver nessun chunk. Senza chunk, non può diventare uno dei 4 peer unchoked di qualcuno degli altri peer, in quanto non ha nulla da inviare. Come fa Alice a ottenere il suo primo chunk?

Alice potrà ricevere il suo primo chunk solamente da quei peer che ormai hanno completato l'intero file e che hanno deciso di non lasciare il torrent, ma di rimanere inviando chunk agli altri peer. Il peer che ha completato il file invierà tranquillamente ad Alice perché non ha il vincolo che Alice debba inviargli un chunk in quanto lui ha finito di richiederne.

16) Che cos'è una rete di overlay? Include i router? Quali sono i collegamenti della rete di overlay?

Una rete di overlay è una rete di calcolatori costuita su un'altra rete. Un esempio di architettura che usa rete di overlay è P2P, in particolare nel DHT (Distribuited Hash Table). I collegamenti non sono connessioni fisiche ma semplici relazioni virtuali tra coppie di peer, senza includere i router.

17) Si consideri una DHT con una topologia di copertura mesh, in cui quindi ogni peer traccia tutti gli altri nel sistema. Quali sono i vantaggi e gli svantaggi di questa progettazione? Quali sono i vantaggi e gli svantaggi di una DHT circolare (senza scorciatoie)?

Una DHT con topologia di copertura mesh ha il vantaggio che, se un peer volesse sapere chi è il peer più vicino alla chiave che cerca, lo saprebbe immediatamente dato che si tiene traccia di tutti i peer presenti nel sistema, ma ha lo svantaggio che questo approccio è impraticabile per i sistemi su larga scala perché tenere traccia di tutti i peer del sistema implicherebbe una quantità di spazio enorme e in continua espansione.

In una DHT circolare (senza scorciatoie) i peer conoscono il proprio successore e il proprio predecessore quindi se si vuole cercare il peer che contiene quella chiave basta chiedere al successivo, se il successivo non ha quella chiave esso andrà a chiederla al suo successore e cosi via fin quando non viene trovato il peer avente quella chiave.

Il vantaggio di questo approccio sta nel non dover tenere traccia di tutti i peer del sistema ma lo svantaggio è che si mandano al più N/2 messaggi, dove N è numero di peer, quindi il costo nel caso peggiore è $\Theta(n)$.

18) Il server UDP richiedeve una sola socket, mentre il server TCP ne richiedeva due. Perchè? Se il server TCP dovesse supportare *n*

connessioni temporanee, ciascuna provieniente da un diverso client, di quante socket avrebbe bisogno?

UDP non è un protocollo orientato alla connessione, quindi non esiste l'handshaking, questo vuol dire che la comunicazione con diversi client si svolge sulla stessa interfaccia.

Invece nel protocollo TCP ,che è un protocollo che offre un servizio orientato alla connessione, il server si metterà in ascolto su una porta,il client effettuerà una richiesta di connessione verso il server.

Da notare che possiamo avere due numeri di porta diversi, perchè una potrebbe essere dedicata solo al traffico in uscita, l'altra solo in entrata; questo dipende dalla configurazione dell'host.

In sostanza, non è detto che la porta locale del client coincida con quella remota del server. Il server riceve la richiesta e crea una nuova connessione.

In questo modo client e server comunicano attraverso un canale virtuale, tra la socket del client e nuova socket del server, creata appositamente per il flusso dati di questa connessione: *Data Socket.*

Questo vuol dire che se ho *n* connessioni avrò *n+1* socket, cioè n data socket (una per ogni client quando viene accettata la richiesta di connessione) e poi una socket comune a tutti che è quella che rimane in ascolto per ricevere nuove richieste di connessione.

19) Nell'applicazione client/server su TCP perché il programma server deve essere mandato in esecuzione prima del client? E, nel caso dell'applicazione client/server su UDP, perché il programma client può essere mandato in esecuzione prima del server?

Nel protocollo TCP deve stabilirsi una connessione tra client e server e quindi il server deve mettersi in ascolto per ricevere le varie richieste di connessione da parte dei client.

Invece nel protocollo UDP questa connessione non ci sta dato che è offerto un servizio senza connessione e quindi può iniziare prima il client del server.

20) Vi viene chiesto di realizzare un sistema Peer to Peer che minimizzi i tempi di ricerca dei contenuti, che tipo di infrastruttura proporreste? Motivare la risposta

L’infrastruttura che andrebbe utilizzata è il DHT con le scorciatoie.

La DHT – Distribuited Hash Table assume che la chiave sia identificativo del file/contenuto e il valore la posizione di quest’informazione (indirizzo IP). Ogni elemento della tabella è indirizzabile ed ogni indirizzo a cui è assegnata la chiave è riferito al peer successivo.

Ogni peer conosce il suo successore e il suo predecessore e un numero relativamente piccolo di peer disseminati nel sistema a lui uniti tramite scorciatoie, in questo modo quando un peer vuole trovare il peer più vicino a quella chiave che cerca, non solo interroga il suo successivo ma vede anche se tra i peer che lui conosce ci sta quello più vicino a quella chiave. In questo modo ho che il numero di messaggi scambiati nel caso peggiore è $\Theta(\log n)$.

21) Si consideri il protocollo HTTP.

1. **Il protocollo permette la possibilità di memorizzare lo stato della sessione?**
2. **Come è possibile permettere l’accesso dei client solo attraversa autenticazione?**
3. **Si illustri una modalità per tenere traccia delle pagine web visitate da un client.**

1) Il protocollo HTTP è stateless, ciò vuol dire che non mantiene lo stato del processo ne tanto meno lo stato della connessione.

2) Si possono utilizzare i cookies per autenticare i client, il cookie è un identificativo, un puntatore alle informazioni che conserva lo stato di quel client che viene implementato a livello applicativo e viene immagazzinato dal client.

3) La prima volta che un utente accede ad un sito, (per esempio www.amazon.com),il sito crea un identificativo unico e una voce nel proprio database, indicizzata dal numero identificativo. Il server risponde al client, includendo nella risposta HTTP l’intestazione *Set – cookie:* che contiene il numero identificativo, (*Set – cookie: 1678*).

Quando client riceve il messaggio di risposta di HTTP, vede intestazione *Set- cookie:* e aggiunge una riga al file dei cookie che gestisce. La riga include il nome dell’host server e il numero identificativo nell’intestazione *Set- cookie: .*

Se client ritorna nuovamente su quel sito , ciascuna delle richieste HTTP che farà al server del sito includerà la riga di intestazione *Cookie:* numero identificativo (per esempio *Cookie: 1678*)

In questo modo è possibile monitorare l'attività di quell'utente nel sito. Sebbene il sito non conosca il nome, sa esattamente quali pagine sono state visitate da quell'utente in ordine e a quali orari.

22) Il tipo di un record DNS può assumere i seguenti valori: A,NS,CNAME e MX. Spiegare il significato di ciascuno di essi e discutere le limitazioni che si avrebbero se i tipi possibili fossero solo A,NS,CNAME. Per quale motivo si può affermare che il DNS è un protocollo a livello applicazione?

Il record di risorsa RR contiene diversi campi: (Name,Value,Type,TTL).

Il TTL è il time to live, cioè il tempo di vita di un record e determina quando una risorsa vada rimossa dalla cache.

Il Name e Value dipendono dal Type, che può essere di 4 tipi.

Se *Type =A* allora *Name* è il nome dell'host e *Value* il suo indirizzo IP. Quindi il record di tipo A fornisce la corrispondenza tra hostname e il suo indirizzo IP

Se *Type=NS* allora *Name* è un dominio e *Value* è hostname del DNS server autoritativo che sa come ottenere gli indirizzi IP degli host del dominio. Questo record è utilizzato per instradare le richieste DNS successiva alla prima concatenazione delle query.

Se *Type=CNAME* allora *Value* rappresenta il nome canonico dell'host per il sinonimo *Name*. Questo record può fornire agli host richiedenti il nome canonico relativo ad un hostname.

Se *Type=MX* allora *Value* è il nome canonico di un mail server che ha il sinonimo *Name.* Questo record consente agli hostname dei mail server di avere sinonimi semplici.

Se non ci fosse il Type=MX , per esempio una società, ***non potrebbe avere*** gli stessi sinonimi per il proprio server di posta e per uno dei propri altri server.

Il DNS è un protocollo del livello di applicazione perché sfrutta il paradigma client/server a cui gli utenti non interagiscono direttamente, inoltre utilizza il protocollo di trasporto UDP per tradurre i nomi degli host nei loro indirizzi IP.

23) Per load balancing si intende la capacità di bilanciare il carico tra più server che erogano il medesimo servizio. Com'è possibile realizzare un meccanismo di load balancing usando il protocollo DNS? Fare un diagramma.

Il protocollo DNS si basa, come altri protocolli a livello di applicazione, sulla logica client-server.
I server DNS elaborano le varie richieste di traduzione nomi in indirizzi IP ai confini della rete, (la complessità ai confini della rete). Essendo però le richieste numerose e soprattutto provenienti da diverse parti del mondo, un sistema DNS basato su logica centralizzata non sarebbe praticabile.
Ecco allora che si adottano meccanismi di load balancing distribuendo il carico di lavoro sui vari server DNS instaurando una gerarchia tra i vari server. L'attuale gerarchia DNS prevede una suddivisione di essi in tre categorie:

1) Root server
2) Server top-level domain – TLD DNS
3) Server di competenza (authoritative server) – DNS server autoritativo
4) LDNS, name server locali