Esercizi Preparazione

Capitolo 1

1. **Definire il concetto di protocollo e poi spiegare come funziona un protocollo particolare a scelta. Perché gli standard sono importanti nei protocolli?**

Un protocollo è un insieme di regole che definisce il formato e l'ordine con cui i messaggi devono essere scambiati per fare in modo che i due dispositivi possano comunicare senza intoppi. Un esempio di protocollo è quello usato per il web ovvero HTTP (Hypertext Transfer Protocol) quest'ultimo prevede un formato per il messaggio di interrogazione (HTTP Request) e un formato per la risposta (HTTP Response). Gli standard sono importanti nei protocolli perchè nella rete se ogni dispositivo comunicasse con le sue regole non avremmo modo di gestire tutti i metodi, mentre avendo degli standard possiamo avere delle comunicazioni con un senso compiuto.

2. **Quale vantaggio presenta una rete a commutazione di circuito rispetto ad una a commutazione di pacchetto? Quali vantaggi ha TDM rispetto a FDM in una rete a commutazione di circuito?**

I due standard di comunicazione sono commutazione di pacchetto e commutazione di circuito, il primo metodo prevede di riservare la banda per tutta la durata della comunicazione tra i due dispositivi, questo fa sì che ci sia garantita una banda minima ed è ottimale per i sistemi real-time come le chiamate telefoniche perchè non abbiamo ritardi, ma riservare un canale è uno spreco di risorse dato che potrebbe essere usato da altri dispositivi quando non viaggiano dati nella comunicazione, la commutazione di circuito è la scelta più adoperata perchè è più flessibile, presenta delay e ritardi, ma allo stesso tempo permette di viaggiare su strade diverse ai pacchetti anche per la stessa destinazione. I due modi per riservare una larghezza di banda sono: TDM (Time Division Multiplexing) e FDM (Frequency Division Multiplexing), la prima prevede di riservare l'intera larghezza di banda per uno "slot" temporale, la seconda divide il flusso per

larghezze di banda, TDM permette di sfruttare l'intera larghezza per un certo periodo di tempo mentre FDM no.

3. **Considerate l'invio di un pacchetto da un host ad un altro lungo un percorso fisso. Elencate le componenti di ritardo complessivo, indicando quali sono costanti e quali variabili.**

L'invio di un pacchetto lungo un percorso fisso presenta elementi variabili ed elementi costanti, bisogna vedere quali sono le componenti dei ritardi ovvero:

- Ritardo di elaborazione: ovvero il tempo impiegato dal router a elaborare l'intestazione del pacchetto per capire in quale porta dovrà uscire, generalmente è considerabile costante.
- Ritardo di accodamento: ovvero il tempo che il pacchetto rimane in coda, viene calcolato l'intensità di traffico e questo è un valore variabile perchè non possiamo controllare e neanche prevedere quando ci sarà un arrivo considerevole o meno di pacchetti, quindi è variabile.
- Ritardo di trasferimento: il tempo di trasferimento dovuto dalla grandezza del pacchetto, non dipende dal percorso ma da quanto è grande il pacchetto, quindi è costante.
- Ritardo di propagazione: il tempo impiegato dai bit a esser propagati sulla linea, non dipende dal pacchetto in questione ma dal percorso, possiamo considerarlo una costante.

4. **Il tempo di propagazione dipende dalla lunghezza del pacchetto? Dipende dalla velocità di trasmissione?**

Il tempo di propagazione è dovuto dalla distanza tra un nodo e un altro, non è dovuto a quanto è grande il pacchetto. Fa direttamente riferimento alla distanza impiegata dal primo bit a essere inviato e arrivare a destinazione, viene calcolato come: D/V dove D è la distanza e V è la velocità della luce.

4. **Quali sono i cinque livelli della pila di protocolli Internet? Quali sono i loro principali compiti?**

Il modello seguito presenta 5 livelli fondamentali per l'invio e la ricezione di dati, ogni livello ha un compito a parte ma tutti hanno un compito generale ovvero quello dell'incapsulamento e relativa operazione inversa. Ogni livello aggiunge una

sua intestazione così quando raggiunge destinazione il livello corrispettivo a quella intestazione potrà gestire i dati:

- Livello applicazione: il livello dove vengono generati di dati, si interfaccia con l'utente e presenta protocolli come HTTP per il web, DNS per la risoluzione di nomi di dominio, SMTP per inviare mail.
- Livello trasporto: il livello trasporto si occupa di scegliere in base all'applicazione, uno dei due protocolli trasporto ovvero TCP o UDP, il primo crea dei segmenti il secondo dei datagrammi, vengono aggiunte le porte fondamentali per la comunicazione.
- Livello rete: il livello rete si occupa di creare pacchetti derivanti dal livello trasporto, in questo livello grazie al protocollo IP si aggiungono gli indirizzi di sorgente e destinazione.
- Livello collegamento: il livello collegamento serve per creare frame dei pacchetti e in base al tipo di collegamento, Ethernet ad esempio, vengono inviati al mezzo trasmissivo.
- Livello fisico: il livello dove avviene la effettiva trasmissione dei bit, viaggiano su questo canale fino a destinazione, una volta arrivati risaliranno dal basso i livelli fino all'applicazione.

5. **Spiegare i campi del datagram IP.**

Il pacchetto IP è composto da:

- Version: indica la versione del protocollo IP
- Type of Service: indica la priorità del pacchetto
- Head Length: indica la lunghezza dell'header dato che con il campo options diventa variabile da 20 bytes fino a 24 bytes
- Packet Length: indica la dimensione totale di header + payload
- ID: indica l'ID del pacchetto e serve per frammentazione e assemblamento
- Flags: 3 bit che indicano: Bit0 se è reserved, Bit1 DF (Don't Fragment) ovvero che il pacchetto non deve essere frammentato dal link e Bit2 MF (More Fragments) ovvero che ci sono ancora frammenti, se è l'ultimo sarà a 0.
- Offset: indica di quanti byte siamo spostati nel pacchetto originale e intero

- TTL: indica il numero di hop che può ancora attraversare, particolarmente usato per traceroute
- Next Header: indica che tipo di protocollo di trasporto si è usato, TCP, UDP o ICMP.
- Checksum: serve per calcolare l'integrità dei dati
- Options: opzioni di lunghezza variabile
- Source Address: indirizzo IP di sorgente
- Destination Address: indirizzo IP di destinazione

6. **Come è composto l'header di un pacchetto TCP e che compito svolgono i campi?**

Il segmento TCP è composto nell'header da:

- Source Port: indica la porta di sorgente
- Destination Port: indica la porta di destinazione
- ACK Number: indica il numero di ACK dal destinatario per il mittente
- Sequence Number: indica il numero di sequence del pacchetto del mittente per destinatario
- Header Length: quanto è lunga l'header dato il campo options variabile
- Not Used: sono 3 bit sempre a 0 per indicare se il pacchetto è riservato
- Flags: sono 6 bit ovvero U (Urgent), A (ACK), P (Push), S (Syn), R (Reset) e F (Fin)
- rwnd: indica la dimensione che ha a disposizione il destinatario nel buffer TCP per ospitare dati
- Checksum: calcolo sul pacchetto per determinare la sua integrità
- Urgent Data Pointer: puntatore a dati urgenti che devono essere trattati terminano nel pacchetto.

Capitolo 2

1. **Discutere, fare esempi e confrontare le architetture client-server e P2P**

Le due principali architetture sono client-server e peer-to-peer, la prima si basa su un dispositivo (il server) che è sempre attivo e pronto a rispondere alle richieste di altri dispositivi (i client). In questa architettura gli host non comunicano direttamente tra loro ma passano le richieste al server. Vediamo come si comporta il server nella condivisione di un file a N host:

$$D = max(\frac{N * F}{u_s}, \frac{F}{d_{min}})$$

In questa formula possiamo vedere due valori distinti che sono la velocità di upload del server per N copie di un file grande F e la velocità di download di un client del file grande F. Con il download dobbiamo prendere il valore minimo per il caso peggiore.

L'architettura peer-to-peer invece si basa su più dispositivi (peer) che sono connessi reciprocamente e scambiano dati in questa rete "chiusa" pezzi che compongono il file originale (chunk), un peer ha una duplice funzione ovvero fare l'upload (server) ma anche il download (client), vediamo il comportamento con la distribuzione di un file:

$$D = max(\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{N * F}{u_s + \sum u_i})$$

In questa formula invece abbiamo tre valori che sono la velocità di upload di un chunk nella rete, la velocità di download di un chunk dalla rete e per finire la velocità di distribuzione per N peer nella rete tenendo conto delle velocità di upload dei singoli.

Quindi l'architettura client-server è poco scalabile rispetto a una architettura p2p, ma è molto più sicura, ci sono molte variabili che possono rendere instabile quest'ultima come ad esempio i free-riders i quali scaricano tutto il file senza aiutare altri con l'upload.

2. **Se una transizione tra client e server deve essere la più veloce possibile, cosa è meglio tra TCP e UDP (motivare la risposta)?**

Per una transizione di dati tra client e server veloce possiamo usare il protocollo UDP (User Datagram Protocol) il quale è per definizione veloce, semplice e non orientato alla connessione. L'ultima caratteristica è proprio quella che giustifica le

prime due, un pacchetto UDP è formato ha un header semplice: porta sorgente, porta destinazione, checksum e length ovvero 8 bytes. Essendo un protocollo non orientato alla connessione possiamo inviare il pacchetto senza avere una fase di setup, in oltre non ha un controllo del flusso e quindi possiamo inviare più velocemente i pacchetti, cosa che TCP ha e si basa sulla rwnd. Per finire non ha neanche un controllo della congestione e per questo possono essere inviati nella rete, se vengono persi non saranno ritrasmessi, UDP non è sicuro ma è veloce e viene impiegato per funzioni apposite: DNS ad esempio o VoIP.

3. **Che cosa si intende per handshaking e spiegare contesto/motivazioni del suo uso.**

La fase di handshake è fondamentale per garantire l'affidabilità nella connessione TCP, per affidabilità significa che il mittente non andrà mai a riempire i buffer del destinatario o che i pacchetti persi dovranno essere ritrasmessi per ottenere una sequenza ordinata. Nella fase di handshake si stabiliscono parametri come la grandezza della finestra di invio di pacchetti, lo scambio delle chiavi per la crittografia o il numero di sequenza iniziale.

Si imposta in questo modo, attraverso il 3-ways handshake, il client invia un pacchetto SYN, il destinatario risponde con SYN-ACK e per stabilire ufficialmente la connessione il client invia l'ultimo ACK.

4. **Si supponga che Alice debba mandare un messaggio di posta elettronica a Bob e che quest'ultimo scarichi la posta con il POP3. Descrivere il "viaggio" di questo messaggio dettagliando i protocolli coinvolti.**

Per prima cosa Alice dovrà scrivere la mail attraverso il suo User-Agent, che attraverso il protocollo SMTP apre una connessione TCP con il server mail, anche quest'ultimo implementa il protocollo SMTP (Simple Mail Transfer Protocol) il quale si occupa di tutto il viaggio dall'inizio fino alla consegna, appena premuto il tasto "Invio" il protocollo va a fare un'interrogazione DNS (DNS Query con type uguale a MX) per scoprire in quale mail server è situata la casella di posta del destinatario, una volta che riceve la DNS Reply potrà inviare direttamente il messaggio aprendo un socket TCP sulla porta 25, una volta caricato il messaggio sulla casella di posta del destinatario dentro il server mail, sono due i protocolli che agiscono, POP3 (Post Office Protocol) o IMAP (Internet Message Access Protocol) il primo prende le mail periodicamente e le elimina dalla casella di posta e le carica nello User

Agent del destinatario altrimenti abbiamo IMAP il quale scarica le mail senza eliminarle e permette le sincronizzazione su più dispositivi.

5. Che differenza c'è tra POP3 e IMAP?

POP3 (Post Office Protocol): protocollo usato per scaricare le mail dal server mail, una volta scaricata la mail viene rimossa del server, veniva usato molto tempo fa perchè lo spazio sui server era limitato e si tendeva alla ottimizzazione di molti sistemi.

IMAP (Internet Message Access Protocol): protocollo usato per sincronizzare su più dispositivi le mail, quando vengono scaricate non vengono eliminate e questo permette di vederle su più dispositivi contemporaneamente e anche la organizzazione con cartelle.

6. **Descrivere la sequenza di system call sia per il client, sia per il server, per il protocollo TCP.**

Per il server abbiamo:

- Socket: crea un socket che verrà associato a un file descriptor
- Bind: associa IP e porta a quel socket
- Listen: aspetta comunicazioni in entrata
- Accept: stabilisce la connessione (qua avviene il 3-ways handshake)
- Read: legge i dati da un buffer
- Write: scrive i dai su un buffer
- Close: chiude la comunicazione

Per il client abbiamo:

- Socket: crea un socket che verrà associato a un file descriptor
- Connect: si connette a un socket di un server fornendo IP e porta (qua avviene l'altra parte del 3-ways handshake)
- Write: scrive i dai su un buffer
- Read: legge i dati da un buffer
- Close: chiude la comunicazione

7. **Descrivere la sequenza di system call sia per il client, sia per il server, per il protocollo UDP.**

Per il server abbiamo:

- Socket: crea un socket che verrà associato a un file descriptor
- Bind: associa IP e porta a quel socket
- Recvfrom: riceve dati da fonti che inviano pacchetti UDP senza stabilire una connessione
- Sendto: invia pacchetti, bisogna specificare ogni volta IP e porta di destinazione
- Close: chiude la comunicazione

Per il client abbiamo:

- Socket: crea un socket che verrà associato a un file descriptor
- Sendto: invia pacchetti, bisogna specificare ogni volta IP e porta di destinazione
- Recvfrom: riceve dati da fonti che inviano pacchetti UDP senza stabilire una connessione
- Close: chiude la comunicazione

A differenza di TCP, UDP non implementa le chiamate listen, accept e connect, proprio perchè non è orientato alla connessione.

8. **Spiegare come funziona la select per il multiplexing**

La select è una system call che serve per gestire il multiplexing mediante un solo canale di comunicazione TCP, possiamo gestire un set di sockets in lettura o in scrittura e possiamo eseguire un controllo attraverso un timer. Questa scelta di gestione permette di ridurre il carico di lavoro della CPU rispetto alla controparte che crea a ogni canale un processo dedicato. Per gestire il set abbiamo 4 macro che sono le seguenti:

- FD_ZERO: permette di inizializzare un set
- FD_SET: permette di aggiungere un file descriptor
- FD_CLR: permette di rimuovere dal set un file descriptor specifico

- FD_ISSET: permette di visualizzare se ci sono stati eventi su un file descriptor specifico

Con questo approccio possiamo impostare un timer, la select per tutta la durata del timer aspetta un evento di I/O e alla fine controlla con FD_ISSET se i socket sono pronti o meno ad eseguire I/O. FD_ISSET ritorna 1 se nel file descriptor c'è un evento altrimenti 0.

9. **Spiegare i parametri e come funzionano le system call read/write**

Entrambe hanno lo stesso numero di parametri che sono:

- File descriptor: ovvero il numero del file descriptor su cui dovrà essere effettuata l'operazione
- Buffer: un buffer con dentro il contenuto che si intende leggere o scrivere
- Count: quanto è grande il buffer, ovvero la sua lunghezza, solitamente impostata con strlen o sizeof.

10. **Quali tecniche si possono usare per rendere una applicazione di rete tollerante ai guasti?**

Per rendere una applicazione di rete tollerante ai guasti si possono prendere accorgimenti e strategie ad hoc per prevenire o curare il problema:

- Ridondanza hardware: in sistemi con client-server è importante che colui che risponde alle richieste ovvero il server sia sempre attivo, in oltre deve anche gestire un alto tasso di traffico e questo alla lunga può causare compromissioni, per questo bisogna avere sempre sistemi sostitutivi nel caso di rottura del principale.
- Failover: nel caso un componente fallisca bisogna avere una sostituzione rapida con i componenti di rimpiazzo.
- Replica dei dati: i dati non devono essere su un singolo componente perchè in caso di rottura avremmo perso per sempre quei dati, si introducono sistemi come il RAID dove abbiamo una ridondanza di dati distribuiti sui server.
- Load balancing: nel caso di rottura di un componente dobbiamo avere il traffico dati ridirezionato verso il componente sostitutivo.

- Utilizzo della virtualizzazione: con la virtualizzazione è semplice avere un sistema su un componente e poterlo replicare su altri senza impiegarci troppo.
- Backup dei dati: bisogna tenere costantemente dei backup in caso di corruzione o perdite, attraverso policy decisionali sul ogni quanto farne.
- Sistemi di monitoraggio: bisogna avere dei sistemi che rilevino dati come: intensità di traffico, stato dei componenti o temperature in modo da avere una visione sul proprio sistema.
- Disaster Recovery: nel caso di eventi naturali come interruzioni di corrente bisogna avere sistemi che in automatico forniscono alimentazione alla rete in modo che non vi siano interruzioni, i più grandi data center come Aruba hanno sistemi appositi che in caso di mancanza di corrente forniscono elettricità autoprodotta.

11. **Come funziona l'applicativo traceroute? Che valori restituisce?**

Traceroute è un applicativo che serve per poter verificare il delay da una sorgente a una destinazione, sfrutta il pacchetto ICMP (ICMP Echo Request) con un TTL incrementale che ogni volta che arriva a 0 torna indietro (ICMP Echo Reply con messaggio "Time Exceeded") e così abbiamo dati statistici come il RTT medio di quel preciso router, infatti per router vengono fatte 3 chiamate per avere una media di quanto tempo ci impiega ad andare e tornare, in questo modo possiamo avere una visione del percorso di un pacchetto e soprattutto quali router fanno da "collo di bottiglia", viene usato anche per strumento di diagnostica. Nel terminale dobbiamo digitare: traceroute nome_dominio.

12. Commenta il seguente codice:

```
rset=allset;
if(ready=select(maxd+1,&rset,NULL,NULL,NULL)<0) perror("select p
if(FD_ISSET(sockfd,&rset))  {
    sin_size = sizeof(their_addr);
    new_fd = accept(sockfd, (struct sockaddr *)&their_addr, &sin
    if(write(new_fd,"Riprovare più tardi\n",strlen("Riprovare pi
        perror("write");
}
```

Questo codice serve per mostrare l'utilizzo della funzione select() in quanto system call che monitora un set di file descriptor, in questo caso abbiamo una inizializzazione di rset che viene poi passato alla select per essere monitorato, successivamente se non ci sono stati errori si verifica se sockfd è presente nel set, presumibilmente dalla nomenclatura è un socket che deve fare lettura e quindi sta aspettando dati, nel caso sia presente un evento nel socket apre una nuova connessione con accept e scrive "Riprovare più tardi" al nuovo socket.

Capitolo 3

1. **Supponete che un server web sia in esecuzione sull'host C sulla porta 80. Supponete che questo web server usi le connessioni persistenti e stia al momento ricevendo richieste da diversi host, A e B. Tutte le richieste vengono inviate attraverso la stessa socket? Se vengono fatte passare attraverso socket diverse, entrambe hanno la porta 80? Discutete e spiegate questa soluzione.**

Le richieste dei socket A e B arriveranno nello stesso socket del server, ma saranno aperti dei socket dopo l'handshake con i singoli, questo per poter avere il socket pair che identifica univocamente la connessione. La connessione per persistente si intende che rimane aperta per un lasso di tempo definito senza ogni volta esser aperto e chiuso per ogni oggetto richiesto, questo rappresenta un miglioramento di performance con tutte le richieste che un web server deve gestire. Entrambi gli host avranno come porta di destinazione la porta 80 ma le loro porte di partenza potrebbero essere diverse, se sono uguali non c'è problema perchè avranno due IP di sorgente diverso, la coppia di socket è quella che identifica la connessione perciò non avremo problemi.