DOMANDE ESAME

**Capitolo 1:**

**1-Descrivi cause di ritardo nelle reti a commutazione di pacchetto**

**Capitolo 2:**

**2-Descrivi protocollo HTTP e differenze tra HTTP 1.0 e 1.1**

**2.1-Descrivi protocollo HTTP 1.0**

**2.2-Descrivi protocollo HTTP 1.1**

**3-Descrivi il caching nei sistemi Web**

**4-Descrivi Posta elettronica e i protocolli di livello applicativo usati**

**5-Descrivi DNS**

**5.1-Descrivi gerarchia dei server DNS e differenze query iterative e ricorsive**

**Capitolo 3:**

**6-Descrivi comunicazione affidabile TCP**

**7-Descrivi demultiplexing in TCP**

**8-Descrivi ACK ritardati in TCP**

**9-Descrivi calcolo del timer di ritrasmissione in TCP**

**10-Descrivi controllo di flusso in TCP**

**11-Descrivi la ritrasmissione rapida in TCP**

**12-Descrivi controllo di congestione in TCP**

**Capitolo 4:**

**13-Descrivi BGP**

**14-Descrivi router NAT e connessione tra client dietro dei NAT**

**15-Descrivi RIP**

**15.1-Descrivi routing inter-AS e differenze tra routing intra-AS e inter-AS**

**16-Descrivi algoritmi di routing distance vector e l’avvelenamento percorso**

**16.1- Descrivi algoritmi di routing link state e Dijkstra**

**16.2- Descrivi algoritmi di routing link state e OSPF**

**Capitolo 5:**

**17-Descrivi protocollo MAC in Ethernet**

**18-Descrivi protocollo ALOHA**

**19-Descrivi protocollo CSMA/CD**

**20-Descrivi ARP**

**21-Descrivi rete Ethernet**

**22-Descrivi Switch Ethernet e differenze con hub e perché non ho cicli**

**Capitolo 8:**

**23-Descrivi crittografia a chiave simmetrica**

**24-Descrivi crittografia a chiave pubblica**

**25-Descrivi firma digitale**

**26-Descrivi sicurezza e-mail**

**27-Descrivi SSL**

**28-Descrivi IPsec**

**29-Descrivi Firewalls**

**1-Cause di ritardo nelle reti a commutazione di pacchetto**

Quando un pacchetto viaggia da una sorgente ad una destinazione, subisce diversi tipi di ritardo a ciascun nodo. In particolare, quando un pacchetto arriva al router A, esso ne esamina l’intestazione per capirne il collegamento di uscita e quindi invia il pacchetto su questo collegamento solo se non ci sono altri pacchetti in fase di trasmissione, altrimenti dovrà mettersi in coda ed attendere. Tra i diversi tipo di ritardo abbiamo: il ritardo di elaborazione, il ritardo di accodamento, il ritardo di ritrasmissione e il ritardo di propagazione. Questi, sommati tra loro, formano il ritardo totale di nodo. Il ritardo di elaborazione è il tempo richiesto per controllare l’intestazione del pacchetto e poterne determinare il collegamento di uscita. Il tempo di accodamento è il tempo che il pacchetto deve attendere mentre sta in coda in attesa di essere trasmesso e questo dipende dal numero di pacchetti in coda che lo precedono. Il ritardo di trasmissione è il tempo necessario per tramettere tutti i suoi bit e corrisponde a L/R con L la lunghezza del pacchetto in bit e R la velocità di trasmissione del collegamento. Infine, il ritardo di propagazione è il tempo necessario per propagare un bit da un router al successivo e dipende dalla velocità della luce.

**2-Descrivi protocollo HTTP e differenze tra HTTP 1.0 e 1.1**

È un protocollo a livello di applicazione del Web che costituisce il cuore del Web. È implementato in due programmi, client e server, in esecuzione su sistemi periferici diversi che comunicano tra loro scambiandosi messaggi. Un browser web implementa il lato client di HTTP, facendo richiesta di oggetti e ottenendoli dal server, mentre un server web implementa il lato server di HTTP ospitando gli oggetti web indirizzabili tramite URL. HTTP definisce, dunque, come i clients web richiedono le pagine web ai web servers e come i server trasferiscono le pagine web ai clients. Quando un utente richiede la pagina Web, il browser invia al server messaggi di richiesta HTTP per gli oggetti nella pagina. Il server riceve la richiesta e risponde con un messaggio di risposta HTTP contenente gli oggetti.

Il protocollo HTTP utilizza il protocollo TCP come il suo protocollo di livello inferiore. Infatti, il client HTTP instaura una connessione TCP con il server nella porta 80 e una volta che è stabilita, client e server si possono scambiare messaggi che seguono il protocollo HTTP. Una volta terminato lo scambio di messaggi si chiude la connessione. Inoltre, HTTP è detto stateless, ossia i server non mantengono informazione sulle richieste passate quindi non ha memoria. Infatti, se un client chiede in un periodo di tempo ravvicinato lo stesso file, il server non risponde dicendo che l'oggetto è già stato servito, ma risponde come se fosse stata la prima connessione. In molte applicazioni Internet, client e server comunicano per un lungo periodo di tempo, con il client che inoltra una serie di richieste e il server che risponde a ciascuna di esse. A seconda del tipo di interazione fra i due, il progettista dell'applicazione deve scegliere quale tipo di connessione adottare: connessioni HTTP persistenti, connessioni HTTP non persistenti.

Nelle connessioni non persistenti, quando un client web collegato con un server web deve fare molteplici richieste per oggetti diversi, per ogni oggetto bisogna usare una connessione TCP diversa, quindi non rimane la stessa connessione tra diversi oggetti. I difetti di questo tipo di connessione sono i seguenti: per ogni oggetto richiesto occorre aspettare 2\*RTT; si ha un consumo di risorse elevato; anche considerando che un browser può avviare più connessioni in paralleli, esse gravano sul server che ne può gestire di meno.

Nelle connessioni persistenti, invece, il server lascia la connessione TCP aperto dopo aver inviato la risposta. Le successive richieste e risposte tra i due possono essere inviati lungo la stessa connessione. Questo permette di inviare una intera pagina Web su una singola persistente connessione TCP e, in alcuni casi, inviare molteplici pagine Web contemporaneamente. Le connessioni persistenti si differenziano in due tipologie:

connessioni persistenti senza pipeling, le quali non consentono l'invio incontrollato di richieste e quindi di risposte tra client e server. In particolare, il client deve aspettare la risposta del server per effettuare la nuova richiesta mantenendo, però, la stessa connessione; connessioni persistenti con pipeling: questi tipi di connessione consentono l'invio e la ricezione di richieste e risposte HTTP in contemporanea, senza quindi dover aspettare la risposta del server.

**2.1 Descrivi HTTP 1.0 (non persistenti)**

È un protocollo a livello di applicazione del Web che costituisce il cuore del Web. È implementato in due programmi, client e server, in esecuzione su sistemi periferici diversi che comunicano tra loro scambiandosi messaggi. Un browser web implementa il lato client di HTTP, facendo richiesta di oggetti e ottenendoli dal server, mentre un server web implementa il lato server di HTTP ospitando gli oggetti web indirizzabili tramite URL. HTTP definisce, dunque, come i clients web richiedono le pagine web ai web servers e come i server trasferiscono le pagine web ai clients. Quando un utente richiede la pagina Web, il browser invia al server messaggi di richiesta HTTP per gli oggetti nella pagina. Il server riceve la richiesta e risponde con un messaggio di risposta HTTP contenente gli oggetti.

Il protocollo HTTP utilizza il protocollo TCP come il suo protocollo di livello inferiore. Infatti, il client HTTP instaura una connessione TCP con il server nella porta 80 e una volta che è stabilita, client e server si possono scambiare messaggi che seguono il protocollo HTTP. Una volta terminato lo scambio di messaggi si chiude la connessione. Inoltre, HTTP è detto stateless, ossia i server non mantengono informazione sulle richieste passate quindi non ha memoria. A seconda del tipo di interazione fra i due, il progettista dell'applicazione deve scegliere quale tipo di connessione adottare: connessioni HTTP persistenti, connessioni HTTP non persistenti.

Nelle connessioni non persistenti, quando un client web collegato con un server web deve fare molteplici richieste per oggetti diversi, per ogni oggetto bisogna usare una connessione TCP diversa, quindi non rimane la stessa connessione tra diversi oggetti. Analizziamo nel dettaglio i passi di trasferimento di una pagina Web dal server al client nel caso di connessione non persistente:

1.Il processo client HTTP inizia una connessione TCP al server sulla porta 80 di default. Associata a questa connessione TCP, dopo che il server ha accettato la connessione, ci sarà un socket al client ed uno al server;

2.Il client HTTP invia una richiesta HTTP al server tramite il suo socket, questa richiesta contiene l'oggetto che richiede;

3.Il processo HTTP del server riceve la richiesta di messaggio nella sua socket e restituisce l'oggetto dal suo archivio, la incapsula in un messaggio di risposta HTTP e la invia al client dalla sua socket;

4.Il processo del server HTTP chiude la connessione;

5.Il client HTTP riceve il messaggio di risposta e la connessione TCP termina. La risposta contiene il file HTML e nel caso in cui vi sono delle referenze ad altri oggetti le esamina;

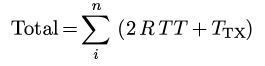
6.Per ogni referenza all'interno del file HTML ripete i passi 1-5.

*Quindi nella connessione non persistente, il client deve avviare una connessione ogni qual volta deve richiedere un oggetto al server avviando una connessione TCP, inviando una richiesta HTTP, aspettando la sua risposta e nel caso ripetere.*

*In questa ottica, definiamo RTT (Round Trip-Time) il tempo necessario affinché un piccolo pacchetto viaggi dal client al server e viceversa (andata e ritorno). Questo parametro include:*

*ritardo di propagazione, ritardo di accodamento, ritardo di processamento, ritardo di trasmissione. Per ogni oggetto quindi avremmo un tempo totale di:*

*(Ttx=tempo di trasmissione dell’oggetto)*

*Per n oggetti:*

I difetti di questo tipo di connessione sono i seguenti:

• Per ogni oggetto richiesto occorre aspettare 2\*RTT;

• Si ha un consumo di risorse elevato;

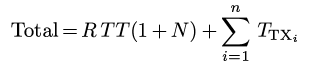
• Anche considerando che un browser può avviare più connessioni in paralleli, esse gravano sul server che ne può gestire di meno.

**2.2 Descrivi HTTP 1.1 (persistenti)**

È un protocollo a livello di applicazione del Web che costituisce il cuore del Web. È implementato in due programmi, client e server, in esecuzione su sistemi periferici diversi che comunicano tra loro scambiandosi messaggi. Un browser web implementa il lato client di HTTP, facendo richiesta di oggetti e ottenendoli dal server, mentre un server web implementa il lato server di HTTP ospitando gli oggetti web indirizzabili tramite URL. HTTP definisce, dunque, come i clients web richiedono le pagine web ai web servers e come i server trasferiscono le pagine web ai clients. Quando un utente richiede la pagina Web, il browser invia al server messaggi di richiesta HTTP per gli oggetti nella pagina. Il server riceve la richiesta e risponde con un messaggio di risposta HTTP contenente gli oggetti.

Il protocollo HTTP utilizza il protocollo TCP come il suo protocollo di livello inferiore. Infatti, il client HTTP instaura una connessione TCP con il server nella porta 80 e una volta che è stabilita, client e server si possono scambiare messaggi che seguono il protocollo HTTP. Una volta terminato lo scambio di messaggi si chiude la connessione. Inoltre, HTTP è detto stateless, ossia i server non mantengono informazione sulle richieste passate quindi non ha memoria. A seconda del tipo di interazione fra i due, il progettista dell'applicazione deve scegliere quale tipo di connessione adottare: connessioni HTTP persistenti, connessioni HTTP non persistenti.

Nelle connessioni persistenti, il server lascia la connessione TCP aperto dopo aver inviato la risposta. Le successive richieste e risposte tra i due possono essere inviati lungo la stessa connessione. Questo permette di inviare una intera pagina Web su una singola persistente connessione TCP e, in alcuni casi, inviare molteplici pagine Web contemporaneamente. Le connessioni persistenti si differenziano in due tipologie.

**Connessioni persistenti senza pipeling: questi tipi di connessioni non consentono l'invio incontrollato di richieste e quindi di risposte tra client e server. In particolare, il client deve aspettare la risposta del server per effettuare la nuova richiesta mantenendo, però, la stessa connessione. *Tramite questo tipo di connessione, l'RTT viene ridotto a 1 RTT ed in particolare, nel caso di invio di richieste e risposte di n oggetti:*

Connessioni persistenti con pipeling: questi tipi di connessione consentono l'invio e la ricezione di richieste e risposte HTTP in contemporanea, senza quindi dover aspettare la risposta del server. Quindi il tempo che realmente viene speso è quello per l'inizializzazione della connessione TCP e per l'invio del primo oggetto. *Il tempo totale per n oggetti vale:*



**3-Descrivi il caching nei sistemi Web**

Una Web Cache o proxy server è un'entità di rete che soddisfa le richieste HTTP al posto del Web server originario, ossia fa da intermediario per un insieme di utenti. Esso ha il proprio dispositivo di archiviazione e mantiene le copie degli oggetti recentemente richiesti e in particolare il browser di un'utente può essere configurato in modo tale che tutte le richieste HTTP dell'utente vengano ridirette nella Web Cache.

Un proxy server funziona nel seguente modo:

1.Il browser stabilisce una connessione TCP al Web Cache e gli invia una richiesta HTTP per l'oggetto;

2.La Web cache controlla se la copia dell'oggetto è salvata localmente. Se lo è, il proxy server ritorna l'oggetto all'interno di un messaggio di risposta HTTP al browser del client;

3.Se il proxy server non possiede l'oggetto, la web cache apre una connessione TCP al server originario. A questo punto, il proxy server invia una richiesta di connessione TCP con il server originario ed invia la richiesta dell'oggetto;

4.Quando la Web cache riceve l'oggetto, si salva una copia localmente e ne manda un'altra, all'interno di un messaggio di risposta HTTP, al browser del client.

Il vantaggio avviene quando ci sono tanti client che fanno la stessa richiesta per cui la richiesta non viene inviata al server di origine ma al proxy server che, avendo già l’oggetto richiesto, lo restituisce direttamente all’utente stesso. Se non ce l’ha, lo richiede al server di origine, lo ottiene, lo salva e lo invia al client.

Come possiamo notare, la cache è sia server che client durante l'intero processo. Quando riceve la richiesta da un browser e gli invia risposte allora è un server. Quando invia la richiesta e riceve la risposta dal server di origine è un client.

La Web Caching è utile per due motivazioni:

• riduce il tempo di risposta per una richiesta del client.

• permette di migliorare il traffico sull'intero internet e migliore le prestazioni di tutte le applicazioni.

• in ambito aziendale permette di capire se l’uso della rete è effettuato in maniera corretta ed eventualmente eliminare determinati contenuti.

• riduce il traffico di un accesso istituzionale ad internet e quindi non si deve spendere per un miglioramento della rete.

**4-Descrivi Posta elettronica e i protocolli di livello applicativo usati**

Il servizio di posta elettronica è un altro esempio di protocollo applicativo. In particolare, è un protocollo client server, tuttavia, ha l’obiettivo di permettere lo scambio di messaggi tra utenti ma gli utenti non interagisco mai tra loro ma solo tramite il servizio di posta gestito dai server di posta che interagiscono tra loro. È formato da tre componenti:

• User Agent: permettono all'utente di leggere, rispondere, e comporre i messaggi.

• Mail Servers: rappresentano il nucleo dell'infrastruttura delle mail ed in ognuna di queste, per ogni utente, vi è una mailbox per mantenere e gestire i messaggi che gli arrivano. Inoltre, devono gestire le code dei messaggi formatasi quando il tentativo di invio dei messaggi non è andato a buon fine;

• Simple Mail Transfer Protocol (SMTP): è un protocollo client/server in cui gli utenti non interagiscono tra di loro ma con il server di posta elettronica comunicando messaggi tra di loro tramite i Mail Server.

Il protocollo **SMTP** è il principale protocollo di livello applicativo per le e-mail di Internet. Questo protocollo utilizza un servizio di trasferimento dati di TCP per inviare dal server le e-mail del mittente alla mailbox del destinatario. La porta utilizzata è la porta 25 ed i messaggi che si inviano sono in 7-bit ASCII. Inoltre, è formato da lato client (quando il mail server invia e-mail ad un altro mail server) e lato server (eseguito sulla mailbox del server). In particolare, si alternano tre fasi: handshaking, trasferimento di messaggi e chiusura della connessione. Una volta che la connessione è stabilita, il server e il client attuano l'handshaking trasferendosi informazioni, tra cui l'indirizzo e-mail del mittente e del destinatario. Dopo questa fase, si invia il messaggio. Quando un'e-mail viene inviata, l'User Agent mittente utilizza SMTP per inviare il messaggio nel proprio mail server ed il mail server utilizza nuovamente SMTP per consegnare l'e-mail all'User Agent destinatario, il quale non può utilizzare il protocollo SMTP per ottenere il messaggio poiché questa operazione è un'operazione pull, mentre SMTP è un protocollo push. A tale scopo si introducono tre protocolli di accesso alle e-mail: POP3, Access Protocol (IMAP) e HTTP.

Il protocollo **POP3** è un protocollo molto semplice, ma dalle funzionalità limitate. Inizia quando l'User agente apre una connessione al mail server sulla porta 110. Quando la connessione TCP viene stabilita, POP3 segue 3 fasi:

-Autorizzazione: l'user agent invia, in chiaro, username e password per essere autenticato.

-Transazione: l'user agent preleva i messaggi e seleziona quelli da eliminare e ottiene le statistiche di e-mail; ad ognuno di questi comandi il server risponde con: +OK: se il precedente comando è andato a buon fine oppure -ERR: se qualcosa è andato storto. Inoltre, questa fase può essere configurata per download and delete o download and keep: nel primo l'user agent una volta ricevuto il comando quit, eliminerà tutti i messaggi; nel secondo l'user agente salva i messaggi.

-Update: Il mail server elimina i messaggi segnati per la cancellazione;

Questo protocollo non mantiene informazioni sullo stato dell'user agent.

**IMAP** è un protocollo di accesso mail, possiede più funzionalità di POP3, ma molto più complesso. In IMAP il server mantiene le caselle di posta in arrivo e tutti i folder degli utenti all’interno del server stesso. Un server IMAP associa ad ogni messaggio una cartella, quando un messaggio arriva al server viene associato nella cartella INBOX. Talvolta, la mailbox può spostare l'e-mail in una cartella creata dall'utente, leggerla, eliminarla. Il protocollo IMAP fornisce dei comandi che permettono all'utente di creare cartelle e muovere i messaggi tra le cartelle, di cercare messaggi specifici e, inoltre, si mantengono informazioni sullo stato dell'utente lungo la sessione.

*Quando un utente vuole accedere alla propria mailbox, i messaggi vengono spediti dal mail server al proprio browser utilizzando il protocollo* ***HTTP****. Se volessimo inviare un messaggio, quest'ultimo è inviato dal browser al mail server, dopo di che si utilizza il protocollo SMTP come visto fino ad ora.*

**5-Descrivi DNS**

Il DNS nasce dall'esigenza di identificare gli host e concilia l’approccio identificativo effettuato con il nome e con l’indirizzo IP: infatti, il DNS permette di tradurre i nomi degli host nei loro indirizzi IP. Per avere un mapping tra hostname ed indirizzo IP, occorre un servizio di directory che traduce gli hostname in indirizzi IP. La soluzione a questo problema è data dal domain name system DNS. Il DNS è un database distribuito (e non centralizzato), che utilizza il protocollo UDP, in una gerarchia di DNS servers e un protocollo di livello applicativo che permette agli host di eseguire query al database distribuito*.* La traduzione viene effettuata nel seguente modo:

• L'utente esegue il lato client dell'applicazione DNS;

• Il browser estrae l'hostname dall'URL e lo passa alla parte cliente dell'applicazione DNS;

• Il client DNS invia una query contenente l'hostname al DNS server;

• Il DNS client riceve una risposta che include l'indirizzo IP per l'hostname;

• Una volta che il browser riceve l'indirizzo IP dal DNS, può iniziare una connessione TCP al processo server HTTP localizzato sulla porta 80 dell'indirizzo IP.

Il protocollo DNS fornisce i seguenti servizi:

• Host aliasing: un host con un nome complicato può avere uno o più nomi che riferiscono allo stesso indirizzo IP;

• Mail server aliasing: Il protocollo DNS può essere invocato da un'applicazione mail per ottenere dall'hostname l'indirizzo IP;

• Load distribution: DNS è utilizzato anche per implementare una distribuzione di carico lungo i server replicati.

Un'implementazione effettiva di DNS potrebbe essere quella di centralizzare tutte le traduzioni degli indirizzi su un unico DNS server. Tuttavia, ciò è irrealizzabile. Infatti, le problematiche sono le seguenti:

• Single point failure: Se l'unico server DNS va in crash, lo fa anche tutto Internet;

• Traffic Volume: Un unico server DNS dovrebbe gestire tutte le query, per tutte le richieste HTTP, per i messaggi etc..

• Distant centralizer database: Un singolo server DNS non può essere vicino a tutti i client che inviano query DNS;

• Manutenzione: L'unico server DNS dovrebbe tenere traccia di tutti i record di tutti gli host internet. Oltre ad essere di grandi dimensioni, dovrebbe aggiornare frequentemente ogni host.

La soluzione a cui si è arrivati, è la realizzazione di un database distribuito gerarchicamente, ossia il DNS utilizza un grande numero di server (nessuno dei quali possiede la mappa di tutti gli indirizzi IP), organizzati in maniera gerarchica distribuiti lungo tutto il mondo.

**5.1-Descrivi gerarchia dei server DNS e differenze query iterative e ricorsive**

Il DNS nasce dall'esigenza di identificare gli host e concilia l’approccio identificativo effettuato con il nome e con l’indirizzo IP: infatti, il DNS permette di tradurre i nomi degli host nei loro indirizzi IP. Per avere un mapping tra hostname ed indirizzo IP, occorre un servizio di directory che traduce gli hostname in indirizzi IP. La soluzione a questo problema è data dal domain name system DNS. Il DNS è un database distribuito (e non centralizzato), che utilizza il protocollo UDP, in una gerarchia di DNS servers e un protocollo di livello applicativo che permette agli host di eseguire query al database distribuito*.*

Supponiamo che l'host desidera l'indirizzo IP di una pagina. Ci sono due modi per agire: Query ricorsiva e Query iterativa.

1. Nella prima l’host contata il suo DNS locale, che contatta il root server se necessario che contatta il server TPD se necessario il quale contatta il DNS server autoritario se necessario ed infinite ritorna l'indirizzo IP all'host richiedente (nota che utilizzando il caching il traffico viene molto diminuito). In generale, non potremmo conoscere il nome del server autoritario, ma possiamo conoscere il server intermediario che deve poter risalire al nome del server autoritario. Dunque, in quella ricorsiva si richiede la risoluzione del nome contattando i diversi server e affidando loro ogni volta il compito di risolvere la corrispondenza e ciò può creare alto carico nei nodi interpellati.
2. Nella seconda l’host contatta il suo DNS locale che contatta il root server e qui cambia come il root server gestisce la richiesta. Infatti, stavolta restituisce l’indirizzo del TLD server da contattare e poi è il DNS server che deve contattare il TLD e se necessario poi contatterà lui direttamente il DNS autoritativo. In altre parole, la query dall'host richiedente al server DNS è ricorsiva e le rimanenti query sono iterative ed è il DNS che si occupa di gestire la corrispondenza.

DNS sfrutta intensivamente il DNS caching per migliorare le performance sul ritardo e ridurre il numero di messaggi DNS in Internet. L'idea è molto semplice: in una query a catena, quando un server DNS riceve una risposta può salvarsi quel mapping tra hostname ed indirizzo in una propria cache.Se l'hostname/Indirizzo IP è in cache nel server DNS ed arriva un'altra query per lo stesso hostname, il server DNS può fornire l'indirizzo IP desiderato anche se non è autoritario.Dal momento che gli hostname e gli indirizzi IP mediamente non sono permanenti, i server DNS scartano le informazioni in cash dopo un certo periodo di tempo.

**6-Comunicazione affidabile TCP**

Il servizio del livello di rete di Internet è inaffidabile: IP non garantisce la consegna del datagramma, la consegna in ordine dei datagrammi e l'integrità del dato nei datagrammi. TCP, invece, crea un servizio affidabile di dati al di sopra del protocollo IP, assicurandosi che il flusso dei dati che fuoriesce da TCP non sia corrotto, non presenti mancanze, non vi siano duplicazione e che sia in sequenza: in pratica il flusso di byte in uscita deve essere uguale a quello in entrata dall'altra parte della connessione.

In particolare, possiamo vedere come TCP risponde a tre tipi di eventi relativi alla trasmissione e alla ritrasmissione dei dati:

1.Quando TCP riceve un dato dall'applicazione, lo incapsula in un segmento e lo passa al protocollo IP. Se il Timer non è in esecuzione per qualche altro segmento, TCP inizializza il timer quando lo passa ad IP. La scadenza del timer è il TimeoutInterval, che viene calcolato come:

TimeoutInterval=EstimatedRTT+4\*DevRTT

2.Quando TCP risponde ad un evento di timeout, il protocollo ritrasmette il segmento e riavvia il timer;

3.Quando TCP riceve un ACK dal destinatario, TCP confronta il valore dell'ACK y con SendBase. La variabile di stato TCP SendBase è il numero di sequenza del più vecchio byte non riconosciuto. Poiché gli ACK in TCP sono cumulativi, y riconosce come ricevuti tutti i segmenti prima del numero di byte y. In caso contrario i segmenti non sono stati riconosciuti. A questo punto si aggiorna la variabile SendBase e si riavvia il timer se ci sono segmenti non ancora riconosciuti.

Ogni volta che avviene un timeout TCP ritrasmette il segmento con più basso numero di sequenza che non ha ricevuto l’ACK ma il successivo intervallo di timeout viene impostato al doppio del valore precedente e ciò permette un minimo di controllo di congestione. Un problema correlato alle ritrasmissioni è il periodo troppo lungo di timeout, il quale impone al mittente di ritardare il nuovo invio del pacchetto perso. Spesso, però, il mittente riesce a rilevare la perdita di un pacchetto grazie agli ACK duplicati relativi ad un segmento il cui ACK è stato già ricevuto da parte del mittente. Nel caso di tre ACK duplicati, il mittente effettua la cosiddetta ritrasmissione rapida, rinviando il segmento perso prima che termini il timer.

**7-Descrivi demultiplexing in TCP**

Le estensioni offerte dal livello di trasporto sono il multiplexing e il demultiplexing che estendono il servizio di consegna host-to-host fornito dal livello di rete in un servizio di consegna process-to-process per le applicazioni in esecuzione sugli host.

In particolare, nell'host di destinazione, il livello di trasporto riceve i segmenti dal livello di rete sottostante ed ha il compito di consegnare il dato contenuto in questi segmenti al giusto processo d'applicazione in esecuzione nell'host. Il livello di trasporto non trasporta il segmento direttamente al processo, ma al socket intermediario. Inoltre, poiché in qualsiasi istante di tempo vi possono essere più di un socket nell'host ricevente, ogni socket ha un identificativo unico.

A questo punto quello che vogliamo capire è come un host ricevente direziona un segmento alla giusta socket. A tale scopo, il segmento è dotato di campi utili al compimento di questa funzione ed il livello di trasporto controlla questi campi per identificare la socket ricevente e la direzione su di essa. Questo processo viene detto DEMULTIPLEXING.

Al contrario, il processo che fa si che pacchetti uscenti dalla stessa macchina abbiano informazioni che permettono di capire quale processo agirà su quei pacchetti, è detto MULTIPLEXING.

Demultiplexing senza connessione (protocollo UDP)

Quando una socket UDP viene creata, contiene l’indirizzo IP locale e un numero di porta non attualmente in uso nell'host. A questo punto supponiamo che un processo nell'host A con un suo numero di porta UDP, vuole inviare una porzione di un dato di un'applicazione ad un processo nell'host B con un certo numero di porta.

L'host A per fare ciò deve effettuare la multiplazione: crea un segmento di livello di trasporto che include i dati dell'applicazione, il numero di porta sorgente, il numero di porta destinatario e due altri valori. Il livello di trasporto a questo punto invia il risultante segmento al livello di rete che lo incapsula in un datagramma IP ed attua un tentativo “best-effort” (come tipico dal protocollo IP) per consegnare il segmento all'host ricevente. Appena il segmento arriva all'host, quest'ultimo lo demultipla alla corretta socket esaminando il numero di porta destinazione. Se due segmenti UDP hanno differente indirizzo IP sorgente, ma hanno lo stesso indirizzo IP destinazione, allora i due segmenti saranno inviati allo stesso processo destinazione attraverso la stessa socket.

Demultiplexing con connessione (protocollo TCP)

Quando una socket TCP viene creata, a differenza di una socket UDP che contiene l’indirizzo IP sorgente e il numero di porta, contiene l’IP sorgente, l’IP destinazione, il numero di porta sergente e il numero di porta destinazione. Inoltre, quando un segmento TCP arriva dalla rete ad un host, l'host utilizza tutti e quattro i valori per demultiplare il segmento alla socket corretta. In TCP, un singolo host può gestire più connessioni in parallelo e per ogni connessione si usa una socket diversa. Quindi, al contrario di UDP, due segmenti con differenti indirizzi IP sorgenti o numeri di porta sorgenti saranno demultiplati in due differenti socket.

**8-Descrivi ACK ritardati in TCP**

Il protocollo TCP utilizza il meccanismo degli ACK ritardati per offrire un servizio affidabile di trasporto dati. Quando il destinatario TCP riceve un segmento con numero di sequenza maggiore del prossimo numero di sequenza atteso, allora rileva un segmento mancante. Il ricevitore, quando riceve un segmento, non invia subito l’ACK ma aspetta un tempo per inviare l’ACK insieme ad un altro messaggio. Questo meccanismo è detto piggybacking e ogni volta che viene ricevuto un segmento parte un timer e se in questo lasso di tempo il ricevitore deve inviare dei dati al mittente, ingloba l’ACK nel segmento. Se per esempio si riceve un segmento, parte il timer e ne arriva un altro, allora non faccio finire il timer e mando l’ACK. IN particolare, se la finestra è molto grande allora TCP manda un ACK ogni 2 segmenti. Se, invece, ho dei pacchetti in disordine, il ricevente non fa partire il timer dato che TCP utilizza ACK cumulativi. Perciò, invia lo stesso ACK dei segmenti che sono stati ricevuti in ordine e quindi vengono riscontrati molti ACK duplicati e viene applicato il meccanismo del fast retrasmit.

**9-Descrivi calcolo del timer di ritrasmissione in TCP**

TCP utilizza un meccanismo di timeout/ritrasmissione per recuperare la perdita dei segmenti. Il timeout dovrebbe essere grande quanto l'RTT da quando il segmento viene inviato finché non viene riconosciuto (ACK). Se il timer è troppo piccolo (più piccolo del RTT) c’è rischio di non aver ricevuto un riscontro e quindi potrei rinviare un segmento anche se non c’è bisogno. Se il timer è troppo grande causa una reazione molto lenta quando viene perso un pacchetto.

Iniziamo lo studio della gestione dei timer TCP considerando come TCP stima l'RTT.

Il round-trip-time SampleRTT per un segmento è il tempo che intercorre dall'invio del segmento alla ricezione dell'ACK. Invece di misurare un SampleRTT per ogni segmento trasmesso, lo si calcola istantaneamente per un pacchetto a cui non corrisponde ancora un ACK. Questo valore varia in base al segmento a causa della congestione nei router e dal carico variabile degli end-system, dunque, si vuole considerare la media di questo valore detto EstimatedRTT. Si ottiene nel seguente modo (filtro passa basso):



Questo valore è una media pesata di SampleRTT, in cui si mette maggiore peso sui campioni recenti piuttosto che quelli vecchi. Questo perché i campioni recenti rappresentano maggiormente la congestione attuale della rete (alpha=0.125).

Inoltre, è utile anche introdurre un parametro che considera la variabilità dell'RTT detto DevRTT. Questo valore, in particolare, rappresenta di quando SampleRTT si discosta da EstimatedRTT:



Ossia quanto dista in valore assoluto il valore osservato rispetto alla stima della sua media (beta=0.25).

Avendo ora una stima del valore medio (EstimatedRTT) e una stima di quanto sia la variabilità dei ritardi (DevRTT), TCP imposta come valore del timeout:



**10-Descrivi controllo di flusso in TCP**

Ricordando che l'host in ogni lato della connessione TCP possiede un buffer per la connessione, quando la connessione TCP riceve dei bytes che sono corretti e in sequenza, li posiziona nel buffer di ricezione. Poiché il buffer è di dimensione finita, se l'applicazione è lenta a leggere i dati, il mittente può facilmente portare il buffer del destinatario in overflow. A questo scopo TCP fornisce un servizio di controllo di flusso alla sua applicazione per eliminare la possibilità del mittente di colmare il buffer del destinatario. Nella pratica consiste nel far combaciare la velocità con cui si inviano i dati a quella a cui vengono letti.

Quindi, TCP fornisce flow control assegnando al mittente una variabile detta receiver window: viene usata per dare al mittente un'idea di quando spazio libero è disponibile al destinatario. Dato che TCP è full-duplex, il mittente ad ogni lato della connessione mantiene un distinto receiver window. In questo modo si evita che il mittente invii dati al ricevente quando non ha posto in cui metterli.

Supponiamo che un host A sta inviando un file di grandi dimensioni ad un Host B su una connessione TCP. L'host B alloca un buffer di ricezione nella sua connessione, definendo la dimensione RcvBuffer. In qualche istante di tempo, il processo dell'applicazione nel destinatario legge dal buffer. Definiamo le seguenti variabili:

• LastByteRead: è il numero dell'ultimo byte nel flusso dati letti dal buffer da parte del processo dell'applicazione B;

• LastByteRcvd: è il numero dell'ultimo byte nel flusso dati che è arrivata dalla rete ed è stata salvata nel buffer di ricezione;

Dato che TCP non permette l'overflow del buffer allocato, dobbiamo avere:



Definiamo la receive window come la dimensione rimanente nel buffer:

Questa variabile è dinamica nel tempo. L'host B dice all'host A quanta spare room possiede nel buffer della connessione salvando in ogni segmento la variabile rwnd.

Inizialmente, l'host B imposta:

rwnd=RcvBuffer

HostA segue due variabili, LastByteSent e LastByteAcked. Mantenendo la quantità dei dati non ancora riconosciuti minore di rwnd, l'host A si assicura che non sta “overflowando” il buffer di destinazione nell'host B. In particolare, imposta il suo thorughput come segue:



**11-Descrivi la ritrasmissione rapida in TCP**

Uno dei problemi delle ritrasmissioni è che il periodo di timeout potrebbe essere molto lungo. Quando un segmento è perso, questo lungo periodo di timeout forza il mittente a ritardare il rinvio del pacchetto perso, aumentando conseguentemente il ritardo. Il mittente spesso può rilevare la perdita del pacchetto prima che l'evento di timeout avviene tramite gli ACK duplicati. L'ACK duplicato è un ACK di un segmento che è stato già ricevuto dal mittente.

Quando il destinatario riceve un segmento con un numero di sequenza che è maggiore del prossimo numero di sequenza atteso in ordine, rileva una mancanza nel flusso dei dati: rileva il segmento mancante. Questa mancanza potrebbe essere il risultato di una perdita o un riordino dei segmenti all'interno della rete. Inoltre, ricordiamo che TCP non utilizza NACK, il destinatario non può inviare un NACK esplicito al mittente; tuttavia, manda un ACK duplicato che corrisponde all'ultimo byte ricevuto in ordine. Poiché il mittente invia spesso un grande numero di segmenti, se uno di questi viene perso, ci saranno molti ACK duplicati. Se il mittente TCP riceve 3 ACK duplicati per lo stesso dato, considera come se il segmento seguente a quello di cui abbiamo riscontrato 3 ACK duplicato è perso. In questo casto, il mittente TCP effettua il cosiddetto fast retrasmit: ritrasmette il segmento mancante prima che il timer scada.

**12-Controllo di congestione in TCP**

Il problema del controllo della congestione consiste nell’avere troppi pacchetti nella rete che non riesco a gestire. Inoltre, la ritrasmissione dei pacchetti persi contribuisce ancora di più ad aumentare la congestione causando così il collasso della rete. Per evitare ciò esistono diversi approcci che differiscono per il ruolo che ha la rete nel supportare il controllo di congestione:

1. Controllo di congestione assistito dalla rete:

Il controllo è assistito o gestito dalla rete ossia sono i router che forniscono informazioni per poter garantire che gli utenti inviino una quantità di traffico compatibile con la capacità della rete di inviare traffico in quel momento. Queste informazioni possono essere un singolo bit che dice se c’è o non c’è congestione (TCP/IP, ATM) oppure direttamente la frequenza di trasmissione che il router supporta in quel momento (ATM ABR). ATM ABR è un servizio di tipo elastico per il trasferimento dati ossia se la rete è sottoutilizzata allora ABR usa tutta la larghezza di banda disponibile mentre se è congestionata ABR diminuisce la velocità di trasmissione.

2.Controllo di congestione end-to-end:

Le entità di trasporto di livello 4 agli estremi della rete devono gestire il controllo da estremo ad estremo. Poiché la rete non mi dice quando c’è congestione, capisco che essa avviene quando perdo segmenti e di conseguenza TCP diminuisce l’ampiezza della propria finestra.

L’approccio consiste nell’imporre a ciascun mittente un limite alla velocità di invio di dati, ossia manipolando la dimensione della receiver window.

Per capire se c’è congestione vedo se ci sono stati perdite di segmenti tramite o un timeout o 3 ACK duplicati: in questo caso il TCP riduce la velocità di trasmissione (rate) riducendo la dimensione della finestra di congestione. L’algoritmo per gestire la congestione presenta 3 fasi:

1. Slow Start: quando una connessione TCP comincia, la dimensione della finestra di congestione è minima (1MSS) e ciò implica avere un rate bassissimo. Inizia, dunque, molto lentamente ma il rate aumenta in modo esponenziale poiché ogni volta che invio un segmento e ricevo il suo ACK, aumento la dimensione della finestra di 1, e dunque la finestra raddoppia ogni RTT. Questo approccio conservativo è dato dal fatto che una nuova connessione perturba lo stato della rete quindi meglio iniziare a trasmettere lentamente.
2. Congestion Avoidance: caratterizza lo stato di una connessione TCP a regime. Dopo una fase iniziale, TCP ha un andamento della finestra che cresce linearmente nel tempo: la dimensione aumenta di 1 ogni RTT se tutto va bene ma quando TCP si accorge di una perdita tramite 3 ACK duplicati allora dimezza la dimensione della finestra (comportamento a dente di sega). TCP ha un comportamento conservativo, invece, quando la perdita viene rilevata con un timeout che scade: TCP prende la finestra e la pone uguale a 1 e ricomincia con la fase di Slow Start. Questo perché ricevere 3 ACK duplicati vuol dire che ho perso qualcosa ma i segmenti successivi sono arrivati mentre se c’è un timeout vuol dire che non ci sono i 3 ACK e quindi ho perso tanti segmenti.

Il passaggio dalla velocità esponenziale alla velocità lineare è indicato dalla variabile di Threshold. All’inizio, quando non ho ancora perso niente, il Threshold è impostato dal protocollo, in generale il Threshold è pari alla metà della dimensione della finestra prima della perdita precedente.

1. Fast recovery: se la perdita è tramite 3 ACK, in realtà TCP pone la dimensione della finestra pari a Threshold+3 ed entro nella fase Fast recovery per recuperare velocemente e tornare nella fase di Congestion-Avoidance.

*Quindi quando CongWin è minore del Threshold, il mittente è nella fase di Slow Start e la finestra cresce esponenzialmente.*

*Quindi quando CongWin è maggiore del Threshold, il mittente è nella fase di Congestion-Avoidance e la finestra cresce linearmente.*

*Se ho delle perdite tramite 3 ACK, il Threshold è impostato a CongWin/2 e la finestra impostata al valore del Threshold+3, se ho delle perdite tramite un timeout, il Threshold è impostato a CongWin/2 e la finestra impostata a 1.*

**13-Descrivi BGP**

È un protocollo di tipo path-vector basato su prefissi e policy ed è l’attuale standard de facto dei protocolli di instradamento tra sistemi autonomi. Questo protocollo permette di ottenere informazioni sulla raggiungibilità dei sistemi autonomi vicini, propagare informazioni di raggiungibilità a tutti i router interni di un sistema autonomo, comunicare ad ogni sottorete la propria esistenza ed infine determinare delle “buone” rotte verso le sottoreti.

In BGP, coppie di router si scambiano informazioni di instradamento tramite connessioni TCP semipermanenti, che possono essere instaurate tra due router di diversi sistemi autonomi (sessione BGP esterna) o dello stesso sistema autonomo (sessione BGP interna). I router ai capi di una connessione TCP sono detti BGP peer e la connessione TCP è detta sessione BGP. BGP permette di conoscere le destinazioni raggiungibili che sono dei prefissi che rappresentano una o più sottoreti. Le operazioni effettuate sono: stabilire una connessione tra router sulla porta 179, scambiare tutte le rotte conosciute, scambiarsi periodicamente degli aggiornamenti. I tipi di messaggi presenti sono: OPEN: viene inizializzata una connessione TCP, UPDATE: annunciare degli aggiornamenti, KEEP ALIVE: aggiornare il fatto che una rete è presente, NOTIFICATION: messaggi di errori.

Il protocollo è di tipo incrementale e nel tempo imparerà l’esistenza di certi percorsi verso certe destinazioni. Memorizza i percorsi in una tabella, applica delle policy per selezionarne una e valuta se comunicare le rotte imparate ai vicini. I messaggi scambiati sono messaggi di annuncio (seleziono una nuova rotta e la comunico) o di withdrawal (dico che una rotta non c’è più). Una rotta è l’insieme di un prefisso destinazione e un insieme di attributi che includono diverse informazioni tra cui:

-AS path: la sequenza dei numeri di sistemi autonomi che sono coinvolti nel percorso

-Next-Hop: viene usato dai router per configurare le loro tabelle di inoltro in modo corretto. In esso è riportata l’interfaccia del router che inizia l’AS-path. Per scegliere tra diverse rotte di un sistema autonomo, BGP sceglie l’AS path più breve ossia il percorso che attraversa meno sistemi autonomi.

BGP non si limita solo a questo routing ma si basa anche su altre policy. Infatti, quando un router riceve un annuncio di rotta, usa le proprie policy per decidere se accettare o filtrare la rotta e per decidere quale scegliere si applicano regole di eliminazione:

1-guardo la local preference ossia quella che preferisco di più in base a delle policy.

2-a parità di preferenza, scelgo quella migliore ossia quella più corta.

3-a parità di lunghezza, scelgo quella con il next hop più vicino ossia quello con il percorso con costo minore (hot potato routing).

4-altrimenti si usano altri criteri addizionali.

Riguardo le policy, una volta che importo una rotta posso decidere di scartarla o modificarne gli attributi mentre quando ne esporto una posso decidere di scartarla se non la voglio comunicare ai miei vicini o posso modificarne gli attributi.

**14-Descrivi router NAT e connessione tra client dietro dei NAT**

Per gestire l’allocazione di indirizzi sono state introdotte le NAT secondo le quali si può riusare lo stesso intervallo di indirizzi e dunque posso avere più nodi che hanno lo stesso indirizzo. In particolare, se un router è di tipo NAT fa un’operazione particolare: quando riceve per un nodo interno un pacchetto da trasmettere verso l’esterno, modifica il campo sorgente di tutti i pacchetti, sostituendo al campo sorgente l’indirizzo IP della propria interfaccia e assegna un nuovo numero di porte. Il motivo del NAT è che in questo modo la rete locale usa un solo indirizzo IP poiché gli altri sono indirizzi all’interno della rete e dunque privati. Inoltre, poiché i nodi interni hanno indirizzi non visibili, i dispositivi interni alla rete non sono indirizzabili in modo esplicito e questo avrà un impatto sulla sicurezza. Infine, i nodi all’interno della rete possono comunicare tra loro senza problemi.

Per quanto riguarda il funzionamento di un router NAT, suppongo che un nodo voglia mandare un pacchetto verso l’esterno, lo inoltra al router NAT il quale associa nella propria tabella all’indirizzo IP interno il numero di porta e all’indirizzo IP esterno un altro numero di porta e modifica la parte sorgente dell’indirizzo e del numero di porta. Il pacchetto viene inviato all’esterno e il server ricevente invia la risposta al router (poiché non sa quale nodo glielo ha inviato), il quale dalla tabella sostituisce i valori di indirizzo e porta del nodo interno e lo inoltra alla destinazione corretta. In tutto ciò, il client e il server non si accorgono che c’è un router che fa da NAT.

Osservo che se dall’esterno arriva un pacchetto che non trovo nella tabella NAT, quel pacchetto viene scartato per una questione di sicurezza poiché non ho trovato nessuna corrispondenza nella tabella. Inoltre, poiché devo assegnare un numero di porta diverso, ciò limita il numero di connessioni. Infine, il router in generale opera fino al livello 3 ma un router NAT deve modificare gli indirizzi, quindi, deve per forza operare fino al livello 4.

Un problema che riguarda i router NAT è il cosiddetto “attraversamento della NAT”. Infatti, se un nodo nella rete genera pacchetti verso l’esterno tutto funziona, ma se sono un utente che vuole parlare con un server dietro una NAT il tutto diventa complesso, poiché il server ha un indirizzo IP privato e dunque nessuno può raggiungerlo dall’esterno. Una soluzione è per esempio configurare in modo statico all’interno della tabella la corrispondenza indirizzo esterno/porta 80 e indirizzo interno/porta 80 così tutto il traffico diretto a questo indirizzo IP/porta 80 viene instradato al server corretto. Questa soluzione funziona ma per un solo server all’interno della rete. Una seconda soluzione consiste nel protocollo IGD che permette di configurare in maniera automatica le corrispondenze all’interno del router NAT. Un’ultima soluzione per far comunicare i client tra loro è illustrata nel funzionamento di Skype. C’è un client che vuole contattare un altro client che sta dietro un router NAT. Si usa un meccanismo per il quale ogni client, sia interno che esterno alla rete, deve stabilire una connessione con un server che fa da relay (server esterno skype) così lo scambio di traffico P2P tra i due client avviene tramite questo server esterno. Quindi un’applicazione P2P a causa delle NAT si ritrasforma in un’applicazione client-server.

**15-Descrivi RIP**

Il routing su internet è un discorso molto ampio che si basa sul concetto di sistema autonomo, ossia un sistema composto da regioni (gruppi di router) sotto lo stesso controllo amministrativo e la rete è composta da un insieme di sistemi autonomi tra loro interconnessi. CI sono due livelli di routing, quello inter-sistema autonomo, che permette di far comunicare tra loro tutti i sistemi autonomi e quello intra-sistema autonomo, il cui responsabile è il sistema autonomo stesso, in particolare su quale tipo di algoritmo usare. I protocolli intra-sistemi autonomi sono anche detti Interior Gateway Protocols (IGP) e i più usati sono: RIP, OSPF e IGRP.

Il RIP è un algoritmo Distance Vector, la cui metrica è il numero di hops (numero di sottoreti attraversate lungo il percorso minimo dal router sorgente al router destinazione) ossia tutti i collegamenti hanno costo unitario. In particolare, i costi sono calcolati tra i router sorgente e la sottorete destinazione. Il costo massimo di un percorso è limitato a 15 per cui l’uso di RIP è limitato a sistemi autonomi con diametro^2 inferiore a 15 hop. In RIP, a differenza dei protocolli DV dove i router adiacenti si scambiavano i vettori delle distanze, i router adiacenti si scambiano gli aggiornamenti di routing ogni 30 secondi circa usando un messaggio RIP advertisements, il quale contiene un elenco che comprende la distanza del mittente rispetto ad un massimo di 25 sottoreti.

Un esempio che illustra il funzionamento del RIP è il seguente:

ho un sistema autonomo con dei router con delle sottoreti, ciascun router mantiene una tabella di instradamento che contiene il vettore delle distanze e la tabella di inoltro. La prima colonna contiene le sottoreti di destinazione, la seconda il router successivo lungo il percorso più beve verso la destinazione e la terza indica il numero di hop per raggiungere la sottorete di destinazione lungo il percorso più breve. Ogni 30 secondi vengono aggiornate le tabelle, perciò, potranno cambiare i valori in base agli advertisements ricevuti. Può succedere che si perdono i messaggi oppure che un router non funzioni più. Per capire se qualcosa non funziona, si aspettano 3 minuti e se non si ricevono advertisements dal vicino vuol dire che quel router o quel link non funziona più, quindi non considero più validi i percorsi per quel nodo e si aggiornano le tabelle.

I messaggi dei RIP hanno un formato standard con un’intestazione e una sequenza di 20 byte per ogni possibile destinazione. 8 bit servono per indicare il tipo di comando, 8 bit per la versione. Ogni destinazione contiene il tipo di famiglia di indirizzi, l’indirizzo IP della rete e la metrica. La lunghezza massima di un datagramma è di 512 bytes.

**15.1-Descrivi inter-AS e differenze con intra-AS**

Il routing su internet è un discorso molto ampio che si basa sul concetto di sistema autonomo, ossia un sistema composto da regioni (gruppi di router) sotto lo stesso controllo amministrativo che usano lo stesso protocollo di instradamento, come DV e LS, detto protocollo intra-sistema autonomo. Perciò diversi sistemi autonomi posso usare diversi protocolli intra-sistemi autonomi. Per far funzionare l’intera rete come una cosa sola, vengono introdotti dei router speciali “gateway routers” che stanno al confine tra sistemi autonomi, i quali eseguono il protocollo intra-sistema autonomo nella rete ed inoltre eseguono protocolli diversi che hanno il compito di permettere l’instradamento tra sistemi autonomi (protocolli inter-sistemi autonomi). Quindi le tabelle di routing dei nodi sono popolate con le informazioni sia interne che esterne al sistema autonomo. La rete è composta da un insieme di sistemi autonomi tra loro interconnessi. Essi possono essere di diverso tipo in base al ruolo del sistema autonomo all’interno della rete:

-Stub AS: è un AS collegato ad altri sistemi autonomi tramite un altro sistema autonomo che fa da intermediario.

-Multihomed AS: è un AS collegato a più sistemi autonomi.

-Transit AS: fornisce connessione ad altri sistemi autonomi ma non genera traffico né ne è il destinatario.

Il protocollo inter-sistema autonomo più utilizzato è il BGP, mentre i protocolli intra-sistemi autonomi, anche detti Interior Gateway Protocols (IGP), più usati sono RIP, OSPF e IGRP.

OSPF è un protocollo link state che utilizza il flooding di informazioni riguardo lo stato dei collegamenti e l’algoritmo di Dijkstra per trovare il percorso a costo minimo. Un router costruisce un grafo del sistema autonomo e manda in esecuzione Dijkstra per trovare un albero di percorsi minimi verso tutte le sottoreti. Ogni volta che c’è un cambiamento nello stato di un collegamento il router manda informazioni di instradamento via broadcast a tutti i router del sistema autonomo e periodicamente invia lo stato dei collegamenti anche se non ci sono variazioni. I vantaggi di OSPF sono:

-Sicurezza: gli scambi tra router sono autentificati quindi solo router fidati possono far parte di questo protocollo.

-TOS: pacchetti diversi vengono gestiti in modo diverso.

-Percorsi con stesso costo: se più percorsi vanno verso una destinazione con lo stesso costo, OSPF consente di usarli senza doverne scegliere uno.

-Supporto per routing unicast e multicast

In BGP, coppie di router si scambiano informazioni di instradamento tramite connessioni TCP semipermanenti, che possono essere instaurate tra due router di diversi sistemi autonomi (sessione BGP esterna) o dello stesso sistema autonomo (sessione BGP interna). I router ai capi di una connessione TCP sono detti BGP peer e la connessione TCP è detta sessione BGP. BGP permette di conoscere le destinazioni raggiungibili che sono dei prefissi che rappresentano una o più sottoreti. Le operazioni effettuate sono: stabilire una connessione tra router sulla porta 179, scambiare tutte le rotte conosciute, scambiarsi periodicamente degli aggiornamenti.

**16-Descrivi algoritmi di routing distance vector e avvelenamento percorso**

Il problema di instradare un pacchetto tra sorgente e destinazione è riconducibile al problema di instradare il pacchetto tra questi due router e lo scopo di un algoritmo di instradamento è quello di trovare il percorso con costo minore. Per formulare i problemi di instradamento si usa un grafo G=(N,E) ossia un insieme di N nodi, che rappresentano i router, e un insieme di E archi, che rappresentano i collegamenti fisici tra i router e hanno un etichetta con un costo*.* Gli algoritmi di instradamento sono classificabili come:

Globali: tutti i router hanno completa informazione sulla tipologia della rete e sui costi dei link. Questo tipo di algoritmo viene detto di tipo link state.

Decentralizzati: non si hanno informazioni complete sulla tipologia della rete o sui costi dei link ma inizialmente i nodi conoscono solo i costi dei collegamenti a loro adiacenti. Successivamente attraverso un processo iterativo e lo scambio di informazioni con i nodi adiacenti, un nodo calcola il percorso a costo minimo verso una destinazione. Questo tipo di algoritmo viene detto di tipo distance vector poiché ogni nodo elabora un vettore di stima dei costi verso gli altri nodi della rete.

L’algoritmo DV è distribuito, in quanto ciascun nodo riceve parte dell’informazione da uno o più dei suoi vicini direttamente connessi a cui successivamente restituisce i risultati del calcolo iterativo, in quanto questo processo si ripete fino a che non c’è più scambio di informazioni tra vicini e asincrono, in quanto non richiede che tutti i nodi operino al passo con gli altri.

Questi algoritmi si basano sull’equazione di programmazione dinamica di Bellman-Ford. Sia dx(y) il costo del percorso a costo minimo dal nodo x al nodo y, sia minv il minimo tra tutti i vicini di x, allora i costi minimi si basano su questa equazione:

Ossia il costo del percorso di costo minimo è pari al minimo preso tra tutti i vicini del costo con cui la x va ai vicini più il costo del percorso di costo minimo tra il generico vicino e la destinazione. L’idea è usare l’equazione in modo iterativo. Definisco Dx(y) la stima del costo del percorso di costo minimo da x a y, Dx= [Dx(y):y in N] l’insieme della stima presso x del costo verso tutte le destinazione y in N. Ogni nodo mantiene i seguenti dati: il costo c(x,v), il vettore delle istanze Dx ed i vettori delle istanze di ciascuno dei suoi vicini Dv. Lavorando in modo asincrono, quando un nodo riceve un vettore delle istanze da un vicino, aggiorno la stima delle istanze usando l’equazione di BF ma al posto di “d” c’è “D”.

Nell’algoritmo, dunque, ogni nodo è in attesa di avere informazioni dai vicini o che cambi qualcosa su un link locale; in entrambi i casi ricalcolo le mie stime (c(x,v) oppure Dv(y)); se il vettore dei costi cambia allora lo comunica ai miei vicini, altrimenti mi metto in attesa;

In una rete può succedere che vengono cambiati i pesi e se il costo aumenta l’algoritmo DV ci mette tantissimo a convergere e la rete entra in loop. Questo perché gli algoritmi DV conoscono il costo di un percorso ma non conosco il percorso per dove passa quindi si comportano malissimo quando c’è un aumento del costo di un link. Per evitare che questo avvenga avviene quello che viene chiamato “avvelenamento del percorso inverso”: se un nodo z fa passare il traffico attraverso y per raggiungere x, quando z invia il vettore delle distanze a y gli dice che verso la destinazione x, che raggiunge passando per y, non ci sa arrivare (infinito). Questo fa sì che si riduce il problema ma non si risolve del tutto.

***Confronto LS e DV***

*Nel calcolare l’instradamento i due algoritmi sono differenti. Nel LS, ciascun nodo comunica con tutti gli altri nodi via broadcast ma li informa solo dei costi dei link direttamente connessi. Nel DV ciascun nodo dialoga solo con i vicini direttamente connessi, dando informazioni sulle stime a costo minimo da sé stesso a tutti i nodi nella rete. Altre differenze sostanziali sono:*

*Complessità messaggi:*

*LS: ogni nodo conosce il costo di ogni collegamento nella rete e necessito quindi di O(nE) messaggi inviati.*

*DV: ogni nodo comunica solo con i vicini.*

*Velocità convergenza:*

*LS: O(n^2).*

*DV: il tempo di convergenza può variare. Posso avere dei loop e c’è il problema del conteggio all’infinito.*

*Robustezza (in caso di malfunzionamenti):*

*LS: ogni nodo può comunicare un’informazione scorretta tramite un link e ogni nodo si calcola la propria tabella. Gli errori sono quindi limitati all’area dove c’è il nodo difettoso e ciò fornisce un certo grado di robustezza.*

*DV: manda messaggi relativi a percorsi sull’intera rete e quindi ci sono errori che si propagano su tutta la rete.*

**16.1-Descrivi algoritmi di routing link state e Dijkstra**

Il problema di instradare un pacchetto tra sorgente e destinazione è riconducibile al problema di instradare il pacchetto tra questi due router e lo scopo di un algoritmo di instradamento è quello di trovare il percorso con costo minore. Per formulare i problemi di instradamento si usa un grafo G=(N,E) ossia un insieme di N nodi, che rappresentano i router, e un insieme di E archi, che rappresentano i collegamenti fisici tra i router e hanno un etichetta con un costo*.* Gli algoritmi di instradamento sono classificabili come:

Globali: tutti i router hanno completa informazione sulla tipologia della rete e sui costi dei link. Questo tipo di algoritmo viene detto di tipo link state.

Decentralizzati: non si hanno informazioni complete sulla tipologia della rete o sui costi dei link ma inizialmente i nodi conoscono solo i costi dei collegamenti a loro adiacenti. Successivamente attraverso un processo iterativo e lo scambio di informazioni con i nodi adiacenti, un nodo calcola il percorso a costo minimo verso una destinazione. Questo tipo di algoritmo viene detto di tipo distance vector poiché ogni nodo elabora un vettore di stima dei costi verso gli altri nodi della rete.

Un algoritmo di instradamento link state è per esempio quello di Dijkstra. Si assume che sia nota l’insieme dei nodi, l’insieme degli archi e i relativi costi. Ogni nodo periodicamente invia agli altri nodi lo stato di tutti i link a cui è collegato in modo che dopo un intervallo di tempo tutti i nodi hanno queste informazioni. Una volta ottenute le informazioni, ciascun nodo si calcola il percorso di costo minore da ogni nodo ad ogni altro nodo. Così si può costruire la tabella di instradamento. L’algoritmo è iterativo. La notazione utilizzata è:

-c(x,y) è il costo del link dal nodo x a y, assumendo che sia infinito se i nodi non sono vicini.

-D(v) è il valore corrente del costo di un percorso verso una destinazione v.

-p(v) è il predecessore lungo un percorso che va fino a v.

-N’ è l’insieme dei nodi per cui ad un certo passo k ho già calcolato il modo esatto per cui l’insieme dei nodi è di costo minimo.

L’algoritmo consiste in un passo di inizializzazione e un ciclo che viene eseguito una volta per ogni nodo del grafo:

Inizializzazione: Il nodo u esegue l’algoritmo. Mi chiedo quale è il percorso di costo minimo verso tutte le possibili destinazioni. N’=u mentre per tutti gli altri nodi se v è adiacente a u allora inizializzo il costo verso v con il costo dell’arco stesso D(v)=c(u,v) altrimenti imposto il costo a infinito.

Loop: Prende il nodo w che non è in N’ tale che D(w) è minimo; aggiunge w a N’; aggiorna D(v) per tutti i nodi adiacenti a w che non sono in N’ in modo che D(v)=min(D(v),D(w)+c(w,v)); itero fino a che tutti i nodi sono in N’.

L’algoritmo di dijsktra permette di calcolare in n passi il costo e il percorso di costo minimo verso tutte le destinazioni partendo da un nodo. L’algoritmo è dell’ordine di O(n^2) anche se si può ottimizzare fino ad avere O(nlogn) ed in generale è un algoritmo facile da implementare. Nel caso in cui il costo dei link è rappresentato dal carico attraverso di essi per rappresentare la congestione del link si verificano delle oscillazioni del cammino poiché nell'esecuzione dell'algoritmo verranno trovati path differenti in base al traffico di rete. Per prevenire queste oscillazioni una soluzione potrebbe essere quella di evitare di assegnare il costo del link sull'ammontare di traffico oppure è quella di assicurarci che non tutti i router eseguono l'algoritmo LS nello stesso tempo.

***Confronto LS e DV***

*Nel calcolare l’instradamento i due algoritmi sono differenti. Nel LS, ciascun nodo comunica con tutti gli altri nodi via broadcast ma li informa solo dei costi dei link direttamente connessi. Nel DV ciascun nodo dialoga solo con i vicini direttamente connessi, dando informazioni sulle stime a costo minimo da sé stesso a tutti i nodi nella rete. Altre differenze sostanziali sono:*

*Complessità messaggi:*

*LS: ogni nodo conosce il costo di ogni collegamento nella rete e necessito quindi di O(nE) messaggi inviati.*

*DV: ogni nodo comunica solo con i vicini.*

*Velocità convergenza:*

*LS: O(n^2).*

*DV: il tempo di convergenza può variare. Posso avere dei loop e c’è il problema del conteggio all’infinito.*

*Robustezza (in caso di malfunzionamenti):*

*LS: ogni nodo può comunicare un’informazione scorretta tramite un link e ogni nodo si calcola la propria tabella. Gli errori sono quindi limitati all’area dove c’è il nodo difettoso e ciò fornisce un certo grado di robustezza.*

*DV: manda messaggi relativi a percorsi sull’intera rete e quindi ci sono errori che si propagano su tutta la rete.*

**16.2-Descrivi algoritmi di routing link state e OSPF**

Il problema di instradare un pacchetto tra sorgente e destinazione è riconducibile al problema di instradare il pacchetto tra questi due router e lo scopo di un algoritmo di instradamento è quello di trovare il percorso con costo minore. Per formulare i problemi di instradamento si usa un grafo G=(N,E) ossia un insieme di N nodi, che rappresentano i router, e un insieme di E archi, che rappresentano i collegamenti fisici tra i router e hanno un etichetta con un costo*.* Gli algoritmi di instradamento sono classificabili come:

Globali: tutti i router hanno completa informazione sulla tipologia della rete e sui costi dei link. Questo tipo di algoritmo viene detto di tipo link state.

Decentralizzati: non si hanno informazioni complete sulla tipologia della rete o sui costi dei link ma inizialmente i nodi conoscono solo i costi dei collegamenti a loro adiacenti. Successivamente attraverso un processo iterativo e lo scambio di informazioni con i nodi adiacenti, un nodo calcola il percorso a costo minimo verso una destinazione. Questo tipo di algoritmo viene detto di tipo distance vector poiché ogni nodo elabora un vettore di stima dei costi verso gli altri nodi della rete.

OSPF è un protocollo link state che utilizza il flooding di informazioni riguardo lo stato dei collegamenti e l’algoritmo di Dijkstra per trovare il percorso a costo minimo. Un router costruisce un grafo del sistema autonomo e manda in esecuzione Dijkstra per trovare un albero di percorsi minimi verso tutte le sottoreti. I costi dei collegamenti vengono impostati a 1 oppure in modo inversamente proporzionale alla capacità del collegamento. Ogni volta che c’è un cambiamento nello stato di un collegamento il router manda informazioni di instradamento via broadcast a tutti i router del sistema autonomo e periodicamente invia lo stato dei collegamenti anche se non ci sono variazioni. I vantaggi di OSPF sono:

-Sicurezza: gli scambi tra router sono autentificati quindi solo router fidati possono far parte di questo protocollo.

-TOS: pacchetti diversi vengono gestiti in modo diverso.

-Percorsi con stesso costo: se più percorsi vanno verso una destinazione con lo stesso costo, OSPF consente di usarli senza doverne scegliere uno.

-Supporto per routing unicast e multicast

-può essere organizzata come una rete gerarchica, ossia in due livelli: livello di area e livello di backbone e gli advertisement sono limitati ad ogni area ossia i router conoscono la topologia solo della loro area. I router di bordo area, ossia quelli che fanno parte sia dell’area sia del livello backbone, hanno il ruolo di riassumere le distanze e annunciarle agli altri router di bordo. L’area di backbone ha il ruolo di instradare il traffico tra le altre aree del sistema.

L’insieme dei messaggi (advertisements) è detto link state database e ogni router ha lo stesso link state database. Ogni volta che invio un messaggio devo ricevere un ACK. A differenza di RIP dove i messaggi viaggiano in UDP, in OSPF i messaggi viaggiano in pacchetti IP e hanno un’intestazione e un corpo con dati specifichi di quel messaggio e advertisement.

**17-Descrivi protocollo MAC in Ethernet**

Esistono due tipi di collegamento di rete: punto a punto e broadcast. Il primo è costituito da un trasmittente e da un ricevente mentre il secondo può avere più nodi trasmittenti e riceventi e quando un nodo trasmette un frame lo inoltra a tutto i nodi. Quando due o più nodi trasmettono un frame, i nodi riceventi ricevono contemporaneamente più frame creando delle collisioni e per risolvere occorre coordinare le trasmissioni dei nodi trasmittenti e questo è il compito dei protocolli ad accesso multiplo. I protocolli ad accesso multiplo posso essere divisi in protocolli a suddivisione del canale, che si basa sull’idea di dividere la risorsa in parti più piccole e allocare le porzioni ai singoli nodi, protocolli ad accesso casuale, che si basa sull’idea che la risorsa non è condivisa e permetto ai nodi di comunicare quando vogliono accettando le collisioni, ed infine protocolli a rotazione, che si basa sull’idea che vengono assegnati dei turni ai nodi per trasmettere eliminando le collisioni.

-Protocolli a suddivisione del canale: la suddivisione della risorsa in parti più piccole può essere a divisione di tempo (TDMA) e a divisione di frequenza (FDMA). Nel primo viene suddiviso il canale in intervalli di tempo, ciascuno suddiviso in N slot temporali. Ogni slot è quindi assegnato ad uno degli N nodi e ogni volta che un nodo deve inviare un pacchetto, trasmette i bit durante il suo slot di tempo che gli viene assegnato. Nel secondo viene suddiviso il canale in frequenze differenti e assegna ogni frequenza ad un nodo.

-Protocolli ad accesso casuale: nell’accesso casuale, un nodo trasmette sempre alla massima velocità R bps e quando si verifica una collisione, i nodi coinvolti ritrasmettono i loro frame fino a che non raggiungono la destinazione senza collisioni. La ritrasmissione non è immediata ma avviene dopo un random delay e ogni nodo seleziona un ritardo casuale indipendente da quello degli altri nodi. Esempi di protocolli ad accesso casuale solo lo slotted ALOHA, ALOHA e CSMA.

-Protocolli a rotazione: sono stati creati poiché ALOHA e CSMA non funzionano bene quando sono attivi M nodi e i più importanti sono il protocollo polling e il protocollo token-passing.

Nel primo protocollo uno dei nodi è detto Master e interpella a turno gli altri, inviando un messaggio al nodo 1 dicendogli che può trasmettere fino ad un numero massimo di frame, successivamente inviando un messaggio al nodo 2 dicendogli che può trasmettere fino ad un numero massimo di frame e così via per tutto gli altri nodi. In questo modo vengono eliminate le collisioni e gli slot vuoti ma si presentano alcuni svantaggi come l’introduzione del ritardo di polling, ossia il tempo necessario per avvisare un nodo che può trasmettere, e il fatto che se il Master si guasta, l’intero canale diventa inattivo.

Nel secondo protocollo non esiste un Master ma un messaggio di controllo detto token che viaggia tra i nodi con un ordine prefissato. In pratica il nodo con il token trasmette un numero massimo di frame consentito e poi passa il token al nodo successivo che farà la stessa cosa. Se un nodo non ha frame da trasmettere, passa direttamente il token al nodo successivo. Questo protocollo è decentralizzato ed efficiente ma, nel caso in cui un nodo si guasti, l’intero canale si blocca oppure se un nodo dimentica di inoltrare il token bisognerà recuperarlo.

**18-Descrivi protocollo ALOHA**

Esistono due tipi di collegamento di rete: punto a punto e broadcast. Il primo è costituito da un trasmittente e da un ricevente mentre il secondo può avere più nodi trasmittenti e riceventi e quando un nodo trasmette un frame lo inoltra a tutto i nodi. Quando due o più nodi trasmettono un frame, i nodi riceventi ricevono contemporaneamente più frame creando delle collisioni e per risolvere occorre coordinare le trasmissioni dei nodi trasmittenti e questo è il compito dei protocolli ad accesso multiplo. Un protocollo di accesso multiplo per un canale broadcast con velocità R bit al secondo dovrebbe avere le seguenti caratteristiche:

-quando un solo nodo deve inviare dei dati, questo ha un throughput pari a Rbps.

-quando M nodi devono inviare dati, questi hanno un throughput pari a R/M bps.

-il protocollo è decentralizzato ossia non ci sono nodi principali

-il protocollo è semplice.

I protocolli ad accesso multiplo posso essere divisi in protocolli a suddivisione del canale, che si basa sull’idea di dividere la risorsa in parti più piccole e allocare le porzioni ai singoli nodi, protocolli ad accesso casuale, che si basa sull’idea che la risorsa non è condivisa e permetto ai nodi di comunicare quando vogliono accettando le collisioni, ed infine protocolli a rotazione, che si basa sull’idea che vengono assegnati dei turni ai nodi per trasmettere eliminando le collisioni.

Nell’accesso casuale, un nodo trasmette sempre alla massima velocità R bps e quando si verifica una collisione, i nodi coinvolti ritrasmettono i loro frame fino a che non raggiungono la destinazione senza collisioni. La ritrasmissione non è immediata ma avviene dopo un random delay e ogni nodo seleziona un ritardo casuale indipendente da quello degli altri nodi. Esempi di protocolli ad accesso casuale solo lo slotted ALOHA, ALOHA e CSMA.

Il primo protocollo ALOHA era privo di slot e decentralizzato e appena arrivava un frame il nodo lo trasmetteva subito al canale broadcast. Se c’era collisione il nodo ritrasmetteva subito con probabilità p. Quindi per determinare l’efficienza di ALOHA analizzo un singolo nodo e come unità di tempo prendo il tempo di trasmissione di un frame. La probabilità che un nodo stia trasmettendo è p e, supponendo che la trasmissione inizi a t0, nessuno deve cominciare un’altra trasmissione nell’intervallo (t0-1,t0) affinché vada a buon finela trasmissione. La probabilità che nessuno cominci un’altra trasmissione è (1-p)^N-1 e analogamente la probabilità che nessuno trasmetti mentre il nodi i sta trasmettendo è sempre (1-p)^N-1. Quindi la probabilità che un nodo trasmetta con successo è p(1-p)^2(N-1) e facendo il limite trovo che l’efficienza massima è la metà di quella di slotted ALOHA ossia 0.18 ossia il 18% degli slot nel caso migliore compie lavoro utile.

**19-Descrivi protocollo CSMA/CD**

Esistono due tipi di collegamento di rete: punto a punto e broadcast. Il primo è costituito da un trasmittente e da un ricevente mentre il secondo può avere più nodi trasmittenti e riceventi e quando un nodo trasmette un frame lo inoltra a tutto i nodi. Quando due o più nodi trasmettono un frame, i nodi riceventi ricevono contemporaneamente più frame creando delle collisioni e per risolvere occorre coordinare le trasmissioni dei nodi trasmittenti e questo è il compito dei protocolli ad accesso multiplo. Un protocollo di accesso multiplo per un canale broadcast con velocità R bit al secondo dovrebbe avere le seguenti caratteristiche:

-quando un solo nodo deve inviare dei dati, questo ha un throughput pari a Rbps.

-quando M nodi devono inviare dati, questi hanno un throughput pari a R/M bps.

-il protocollo è decentralizzato ossia non ci sono nodi principali

-il protocollo è semplice.

I protocolli ad accesso multiplo posso essere divisi in protocolli a suddivisione del canale, che si basa sull’idea di dividere la risorsa in parti più piccole e allocare le porzioni ai singoli nodi, protocolli ad accesso casuale, che si basa sull’idea che la risorsa non è condivisa e permetto ai nodi di comunicare quando vogliono accettando le collisioni, ed infine protocolli a rotazione, che si basa sull’idea che vengono assegnati dei turni ai nodi per trasmettere eliminando le collisioni.

Nell’accesso casuale, un nodo trasmette sempre alla massima velocità R bps e quando si verifica una collisione, i nodi coinvolti ritrasmettono i loro frame fino a che non raggiungono la destinazione senza collisioni. La ritrasmissione non è immediata ma avviene dopo un random delay e ogni nodo seleziona un ritardo casuale indipendente da quello degli altri nodi. Esempi di protocolli ad accesso casuale solo lo slotted ALOHA, ALOHA e CSMA.

In ALOHA i nodi trasmettono un frame a prescindere dal fatto che un altro nodo potrebbe star trasmettendo un altro frame. CSMA si comporta in maniera più “educata”, in particolare ascolta il canale prima di trasmettere (carrier sensing) e se il canale sta già trasmettendo, il nodo aspetta finché non si è liberato. Inoltre, rimane sempre in ascolta del canale e se un altro nodo sta trasmettendo un frame che interferisce con il proprio, si ferma, attende un intervallo di tempo e poi riprova (collision detection).

Nonostante ciò, è possibile che avvengano delle collisioni. Infatti, se per esempio un nodo A ad un certo istante vede il canale libero, comincia la trasmissione e i suoi bit iniziano a propagarsi. Se ad un instante successivo un nodo B deve trasmettere un frame ma i bit del nodo A non lo hanno ancora raggiunto, il nodo B pensa che il canale sia libero e comincia a trasmettere e si creerà una collisione. Quindi, il ritardo di propagazione ha un ruolo fondamentale per determinare l’efficienza del protocollo e maggiore è il ritardo, maggiore è la probabilità che un nodo non si accorga che in realtà qualcuno prima di lui ha già cominciato una trasmissione. Per questo motivo viene aggiunta la collision detection. Grazie al CSMA/CD i nodi, se si accorgono di una collisione, smettono di trasmettere e dunque grazie al sensing del canale capiscono se avvengono delle collisioni. L’efficienza ottenuta con questo protocollo CSMA/CD è di: 1/(1+5tprop/ttrans)

**20-Descrivi ARP**

Tutti i nodi hanno un indirizzo IP a 32 bit che è l’indirizzo di livello di rete, usato per gestire i datagrammi. L’indirizzo che invece caratterizza le singole schede è detto indirizzo MAC, formato da 48 bit, il cui ruolo è quello di garantire l’indirizzamento all’interno di una rete di livello due. L’indirizzo MAC è di tipo flag ossia non è legato alla posizione del nodo nella rete ma individua quella scheda all’interno della rete mentre gli indirizzi IP devono essere aggiornati in base alla posizione del nodo nella rete. Il protocollo ARP ha il compito di permettere la risoluzione dei due indirizzi: dato un indirizzo IP voglio conoscere l’indirizzo MAC. Per fari ciò, ogni nodo IP ha una tabella ARP che contiene le informazioni dei mapping tra indirizzi IP e MAC e in aggiunta contiene un valore detto TTL (time to live) che indica quando bisogna eliminare una data informazione della tabella (tipicamente 20 minuti).

-Per capire il funzionamento del protocollo ARP, suppongo che A voglia inviare un datagramma a B, il quale sta nella stessa sottorete, e l’indirizzo MAC di B non è nella tabella ARP di A. Per cui A chiede, tramite un messaggio broadcast, l’indirizzo MAC del nodo B e B risponderà con l’informazione richiesta. Una volta ricevuta la risposta potrò comunicare con il nodo B ed inoltre mi salvo questa informazione nella tabella ARP. ARP è quindi un protocollo plug and play, ossia un nodo si collega alla rete e se vuole un indirizzo MAC di un nodo basta che lo chieda.

-Il messaggio ARP è formato da 28 byte che contiene le informazioni per fare la traduzione da indirizzo IP e indirizzo MAC. Nel messaggio di richiesta, il nodo richiedente invia le informazioni del suo indirizzo MAC in modo che tutti sanno chi ha fatto la richiesta e il suo indirizzo MAC.

-È un protocollo particolare perché funziona solo nell’ambito della stessa rete LAN e non posso chiedere indirizzi MAC di nodi all’esterno della mia rete ed inoltre un messaggio ARP si limita solo a chiedere l’indirizzo MAC associato ad un preciso indirizzo IP. Per questi motivi lo si può pensare come un protocollo di livello due ma in realtà si considera come protocollo intermedio, ossia un protocollo di livello tre perché risolve indirizzi tre in indirizzi due ma limitato all’interno di una sottorete.

**21-Descrivi rete Ethernet**

La maggior parte delle reti LAN sono reti ethernet, sia perché funzionano bene sia perché sono economiche. La struttura di una rete ethernet è quella di una “stella”, ossia un componente centrale detto switch (inizialmente era un hub) a cui sono collegati tutti i nodi. Lo switch è privo di collisioni perché c’è un cavo che permette le trasmissioni da computer a switch e un altro che permette le trasmissioni da switch a computer.

La struttura di un frame ethernet è fatta da un’intestazione e un’informazione di coda: è presente un preambolo, un campo di informazioni di destinazione e sorgente, un campo che indica il tipo di livello tre è presente, un campo dati e un campo di controllo di errore CRC.

Il servizio è di tipo connectionless e di tipo inaffidabile, infatti, a livello due non saprò mai se un frame arriva o no inoltre il tipo di protocollo MAC usato è CSMA/CD.

Per la ritrasmissione in ethernet si utilizza il protocollo NIC, il quale aggiusta il meccanismo di ritrasmissione in base al livello di congestione percepito nella rete. Quando una scheda ethernet riceve un datagramma da trasmettere, genera il frame corrispondente, vedo se il canale è libero e mi metto in attesa se è occupato, inizio la trasmissione e se non rilevo collisioni la trasmissione va a buon fine. Se c’è una trasmissione in contemporanea, interrompo la trasmissione e si invia un segnale di collisione (jam signal). A questo punto il nodo entra nello stato di exponential backoff che mi dice il tempo di attesa prima di poter riprendere la trasmissione. Questo tempo si misura in tempi multipli di quanto ci mette una scheda ad inviare blocchi di 512 bit ossia si attende un tempo di K\*512 unità di tempo, con K che aumenta in base al numero di collisioni presenti. L’efficienza ottenuta con questo protocollo ethernet CSMA/CD è di: 1/(1+5tprop/ttrans). Esistono diverse versioni di ethernet a velocità diverse. La parte che cambia è la parte fisica che dipende dal tipo di cavo utilizzato.

Usata nella versione 10BaseT, la codifica **Manchester encoding** è fatta in modo che ogni bit da trasmettere sia codificato in una forma d’onda che abbia una transizione dall’alto al basso (caso 1) o dal basso all’alto (caso 0). Il fatto che ogni bit ha una transizione fa sì che la forma d’onda che invio ha media zero cioè il segnale elettrico che invio ha componenti solo in corrente alternata e questa forma d’onda corrisponde al clock del mittente.

Ethernet ha avuto diverse denominazioni come 10BaseT, 10Base2 e 100BaseT, dove il primo numero indicava la velocitò dello standard in Mbps, Base indicava che il segnale veniva inviato in banda base e la parte finale rimandava al mezzo fisico. Segmenti più lunghi si potevano ottenere usando un repeater. Successivamente si è passati a reti più affidabili utilizzando una topologia ad hub. Un hub è una scatola, a cui si connettono i vari calcolatori, che funge da ripetitore con lo svantaggio che se due nodi trasmettono in contemporanea, avviene una collisione.

**22-Descrivi Switch Ethernet e differenze con hub**

Nella fine degli anni 80/90 furono realizzate delle reti in modo più affidabile grazie agli Hub. Un Hub è una scatola, a cui si connettono i vari calcolatori, che funge da ripetitore con tantissime porte, con lo svantaggio che se due nodi trasmettono in contemporanea, avviene una collisione e dunque si ottiene una rete con un unico dominio di collisione. Gli hub erano il primo modo di interconnettere tra loro le varie porzioni di rete, successivamente si utilizzarono i bridges, ossia degli switch più lenti e con poche porte, ed infine si usarono gli switch, che erano veloci e con molte porte.

Il modo più semplice per interconnettere le varie porzioni di rete era collegare diversi hub tra loro tramite un altro **hub**, tuttavia, tutti collidevano con tutti e la capacità della rete rimaneva sempre uguale. Così si cercò di mantenere separati i domini di collisione. Un miglioramento rispetto agli hub furono i **bridges**, considerati come gli antenati degli switches. I bridges hanno il compito di separare i domini di collisione riuscendo a realizzare un’unica rete LAN fatta da diversi segmenti i cui nodi possono comunicare senza collidere con i nodi di un altro segmento. Dunque, un bridges, a differenza di un ripetitore, memorizza e inoltra frame in modo selettivo, è trasparente ossia i nodi non si accorgono della sua presenza ed inoltre è plug and play (funziona da solo) e self-learning (impara da solo il modo corretto di operare). Infatti, un bridge ha una tabella le cui informazioni vengono imparate automaticamente dal bridge, il quale memorizza le informazioni riguardo l’indirizzo LAN di un nodo, l’interfaccia del bridge e il tempo per cui l’informazione è ritenuta valida.

Quando un bridge riceve un frame, deve decidere se inoltrare il frame (forwarding) o scartarlo (filtering). Per prima cosa appena riceve un frame memorizza o aggiorna la propria tabella, successivamente cerca nella tabella l’indirizzo MAC di destinazione e una volta trovato, se l’indirizzo MAC del destinatario è sullo stesso segmento da dove mi è arrivato il frame, allora lo scarto, altrimenti lo inoltro sull’interfaccia corretta; se l’entry, invece, non è nella tabella allora inoltro il frame su tutte le altre interfacce. Nelle reti più hub sono collegati a più bridge e questo causa la creazione di cicli e ciò crea dei problemi perché i messaggi broadcast vengono rimbalzati da tutti i bridge su tutte le altre interfacce (problema del broadcast storm). Per questo motivo una rete che usa bridge non può avere dei cicli.

Infine, ci fu un ulteriore sviluppo arrivando così agli **Switch**, che sono dei bridge con molte interfacce. Tipicamente ad uno switch è collegato un solo nodo e non un’intera rete e dunque una rete con uno switch è formato da 2\*N nodi perché ogni nodo ha due collegamenti per evitare le collisioni, ossia lo switch separa i domini di collisione. Anche lo switch, come il bridge, impara quale nodo è raggiungibile con una precisa interfaccia imparando tramite una switch table. Inoltre, gli switches possono essere collegati tra di loro.

**23-Descrivi crittografia a chiave simmetrica**

Suppongo che un mittente voglia inviare un messaggio ad un destinatario. Il messaggio originario è detto plaintext (testo in chiaro) e successivamente viene trasformato da un algoritmo di cifratura, tramite una precisa chiave del mittente, in un messaggio ciphertext (messaggio cifrato), il quale è incomprensibile ad altri utenti. Infine, il destinatario trasforma il ciphertext in plaintext grazie ad un algoritmo di decifratura, tramite una sua chiave precisa. Ci sono due meccanismi per realizzare la crittografia: Crittografia a chiave simmetrica e a chiave pubblica.

La crittografia a chiave simmetrica si basa sull’esistenza di una chiave, che cambia a seconda del meccanismo usato per la cifratura, comune tra A e B usata sia per cifrare che per decifrare il messaggio. Esistono più tipi di algoritmi:

-il cifrario di Cesare, è un algoritmo che si basa sulla sostituzione di ciascuna lettera del messaggio originare con un’altra lettera sfasata rispetto alla prima di un numero k di posizioni. La chiave corrisponde alla variabile k.

-il cifrario monoalfabetico, è un algoritmo che si basa sulla sostituzione di ciascuna lettera del messaggio originale con un’altra lettera senza uno schema regolare. La chiave corrisponde al cifrario utilizzato.

-metodo più associazioni: il problema di questi due tipi di algoritmi di crittografia è che dal testo io posso ricostruire lo schema con l’analisi statistica. Per rendere più complicata la ricostruzione del messaggio da parte di intrusi viene usato un metodo con più associazioni di lettere ossia con più cifrari invece di uno. In questo modo uso diversi cifrari seguendo un pattern preciso che mi indica l’ordine di utilizzo dei cifrari. La chiave corrisponde agli n cifrari e al pattern usato.

-cifrari a blocchi: si cerca di aumentare il tipo di relazione tra i bit di ingresso e i bit di uscita. Per far ciò si divide il messaggio da cifrare in blocchi di k bit e ogni blocco viene cifrato in modo indipendente, ottenendo blocchi di bit di testo cifrato. Infine, i diversi blocchi vengono riassemblati e mescolati, alterandone così l’ordine per poi ripetere la procedura più volte. Il problema di questo meccanismo è che, per quanto possa essere articolata la cifratura dei messaggi, se viene dato in ingresso lo stesso messaggio in input, verrà restituito sempre lo stesso messaggio cifrato in output.

-cifrari a blocchi concatenati: per risolvere il problema dello stesso input, viene considerato lo schema a blocchi concatenati. Ogni messaggio, prima di cifrarlo, lo combino con un’operazione opportuna (xor) tra la sequenza di bit in ingresso (m(i)) e il codice i-1 esimo che ho cifrato per il blocco precedente (c(i-1)). Ogni codice cifrato, quindi, dipende da tutta la sequenza in ingresso.

Un esempio è il DES che usava chiavi a 56 bit e 64 bit di input ed era molto sicuro, considerando che a volte si usava il 3-DES e la tecnica dei cifrari a blocchi concatenati. Oggi non si usa il 3-DES ma una tecnica più avanzata detta AES, che usa una tecnica simile al DES ma lavora con 128 bit e ciò permette una maggiore sicurezza.

**24-Descrivi crittografia a chiave pubblica**

Suppongo che un mittente voglia inviare un messaggio ad un destinatario. Il messaggio originario è detto plaintext (testo in chiaro) e successivamente viene trasformato da un algoritmo di cifratura, tramite una precisa chiave del mittente, in un messaggio ciphertext (messaggio cifrato), il quale è incomprensibile ad altri utenti. Infine, il destinatario trasforma il ciphertext in plaintext grazie ad un algoritmo di decifratura, tramite una sua chiave precisa. Ci sono due meccanismi per realizzare la crittografia: Crittografia a chiave simmetrica e a chiave pubblica.

La crittografia a chiave pubblica permette di usare chiavi diverse e quindi non devono essere scambiate, il cui processo rappresenta il principale rischio di intercettazione. L’idea è che non c’è una chiave comune segreta bensì due chiavi per ciascuna parte: una chiave pubblica (K+) e una privata (K-) che non va condivisa e conoscere la chiave pubblica non da informazione sulla chiave privata, nonostante siano correlate. Applicando prima la chiave pubblica e poi la privata si ottiene il messaggio (K-(K+(m))=m). L’algoritmo principale a chiave pubblica è **l’RSA**.

RSA si basa su due punti principali: la scelta delle chiavi (pubblica e privata) e gli algoritmi di cifratura e decifratura. Per generare le chiavi vengono eseguiti i seguenti passi:

1. Vengono scelti due numeri primi *p* e *q* dell’ordine di 1024 bit e maggiore è il loro valore più sarà sicura RSA ma sarà più complicata la cifratura e la decifratura.
2. Viene calcolato *n=p\*q* e *z=(p-1)(q-1)*
3. Viene scelto un numero *e* minore di *n*, diverso da 1 e che non abbia divisori comuni con *z*.
4. Viene scelto un numero d tale che *e\*d*-1 sia divisibile per *z*
5. La coppia (*n, e*) è la chiave pubblica e la coppia (*n, d*) è la chiave privata.

Ora per cifrare un messaggio m vuol dire:

1. Calcolare c = memod n ossia il messaggio cifrato.
2. Calcolare m = cd mod n ossia il messaggio decifrato, utilizzando la chiave segreta.

Una proprietà importante del RSA è che se prendo un messaggio, lo cifro con la chiave pubblica e poi lo decifro con la chiave privata ottengo il messaggio ma vale anche l’opposto, ossia se prendo il messaggio, lo cifro con la chiave privata e poi lo decifro con la chiave pubblica ottengo il messaggio.

Inoltre, RSA è sicuro perché nota la chiave pubblica è difficile calcolare *d* e quindi la chiave privata.

L’RSA è un processo molto più lungo rispetto al DES, a causa dell’elevamento a potenza ma allo stesso tempo il DES è meno sicuro del RSA. Perciò si può utilizzare una combinazione dei due: uso un meccanismo a chiave pubblica per comunicare la chiave simmetrica da usare e poi uso la cifratura a chiave privata per lo scambio di informazioni.

**25-Descrivi firma digitale**

È una tecnica crittografica simile alla firma a mano. Chi invia un documento può firmarlo in modo da certificare la sua provenienza ed è verificabile e non può essere alterato. L’idea è che quando un mittente scrive un documento, solo lui lo può firmare usando la sua chiave privata e cifrarlo. Quando un destinatario riceve un messaggio firmato, per verificare che effettivamente è stato inviato e firmato da un certo mittente e non è stato firmato da nessun altro, applica la chiave pubblica del mittente e verifica l’autenticità. L’uso della chiave pubblica per le firme digitali ha il problema per cui la cifratura e la decifratura risultano essere complicati e onerosi ed è per questo che viene usato un approccio più efficiente che consiste nell’introduzione delle funzioni hash nella firma. In pratica, il mittente non firma l’intero documento ma soltanto una sua “impronta”, ottenuta applicando al messaggio una funzione hash che genera una sequenza di lunghezza minore. Se si combinano le due cose, un mittente invia un messaggio, si calcola con la funzione hash il digest H(m) e lo firma con la chiave privata, in modo che il destinatario riceva il messaggio originario e il digest criptato. A questo punto, il destinatario prende il messaggio, ci applica la funzione hash, calcola H(m), usa la chiave pubblica del mittente per decifrare la firma e confronta se questi due digest sono uguali.

**26-Descrivi sicurezza e-mail**

Applichiamo i principi di crittografia per migliorare la sicurezza della posta elettronica. Un’entità vuole inviare un’e-mail ad un destinatario. Si usa un meccanismo a chiave simmetrica dove la chiave viene scambiata usando la chiave pubblica del destinatario, la quale si ottiene dalla CA.

-Un esempio confidenziale, in cui il destinatario non sa nulla riguardo il mittente, è il seguente: il mittente scrive un messaggio, genera una chiave simmetrica e la usa per cifrare il messaggio (K­(m)). La chiave simmetrica deve essere comunicata al destinatario, viene cifrata usando la chiave pubblica del destinatario ed invio in rete il messaggio cifrato e la chiave simmetrica. Se qualcuno intercetta il messaggio non riesce a risalire alla chiave simmetrica. Il destinatario usa la sua chiave privata per estrarre la chiave simmetrica e la usa per decifrare il messaggio cifrato dal mittente.

-Un esempio che garantisce autenticazione e integrità è il seguente: il messaggio viene inviato in chiaro e viene calcolato la hash del messaggio e applica la firma digitale sul digest del messaggio. Entrambe le cose vengono inviate e il destinatario dal messaggio si ricalcola l’hash, usa la chiave pubblica del mittente (ottenuta dal certificato della CA) per estrarre il contenuto del messaggio cifrato (H(m)) e confronta se sono uguali per capire se il messaggio è quello originale e se gli è stato inviato proprio dal mittente che lui si aspetta.

-Un esempio che combina sia confidenzialità, autenticazione e integrità è il seguente: il mittente prende il messaggio, si calcola l’hash e applica la firma digitale sul digest del messaggio. Il mittente genera una chiave simmetrica, la cifra con la chiave pubblica del destinatario e invia un messaggio con la chiave simmetrica cifrata e un messaggio cifrato che contiene il messaggio originario con la firma digitale. Il destinatario usa la sua chiave privata per estrarre la chiave simmetrica, la usa per decifrare il messaggio, applica l’hash al messaggio, applica la chiave pubblica del mittente così autentica il messaggio per estrarre (H(m)) e fa il confronto. Così so che il messaggio è stato inviato proprio dal mittente che mi aspettavo, sono sicuro della sua identità e so che il messaggio non è stato modificato. Un protocollo che funziona come sopra riportato è il PGP, il quale permette la sicurezza della posta elettronica ed usa crittografia a chiave simmetrica, a chiave pubblica, funzione hash e firma digitale.

**27-Descrivi SSL**

La crittografia può essere usata per arricchire TCP dando luogo ad un protocollo detto SSL, una versione arricchita di TCP con servizi di sicurezza, tra cui la riservatezza, l’integrità dei dati e l’autenticazione del client e del server. Rispetto agli altri protocolli questo permise il commercio sul web grazie alla sua sicurezza e cercò di rimediare alle mancanze degli altri protocolli come la cifratura e l’autenticazione di un web-server. Inoltre, era di facile applicazione ed era utilizzabile da qualsiasi applicazione che supportasse TCP. A differenza di un’applicazione normale, la quale interagisce tramite l’API con il protocollo TCP, nel caso di SSL, questo è un protocollo intermedio tra l’applicazione e TCP (protocollo intermedio di livello tra il 4 e il 5). SSL è implementato con delle librerie particolari ossia librerie software che permettono di invocare l’apertura e la chiusura di connessioni.

SSL è caratterizzato da diversi passi: l’handshake, in cui le due parti si scambiano il certificato e le chiavi per realizzare la comunicazione, la derivazione delle chiavi, in cui le parti usano una chiave comune per generare le altre chiavi, il trasferimento dati, in cui vengono trasferiti i dati, e la chiusura della connessione, in cui viene terminato lo scambio di dati.

Handshake: una volta inizializzata la connessione, nella fase di handshake il client invia un messaggio e il server risponde con il proprio certificato firmato con la chiave pubblica. Il client allora genera la chiave “master” e la invia, tramite la chiave pubblica, al server, il quale la decifra.

Derivazione delle chiavi: tramite la chiave master vengono generate 4 chiavi, due per la cifratura (una per i dati da client a server e un’altra per il viceversa) e due per la parte di integrità (MAC).

Trasferimento dati: il flusso di dati viene suddiviso in record, ne viene calcolato il MAC insieme alla chiave, ottenendo il messaggio e la firma. Tutto ciò, insieme ad un SLL number, viene cifrato con la chiave simmetrica, gli viene aggiunto il tipo di messaggio, la versione di SSL e la lunghezza e il tutto viene inviato.

Oltre a questi passi, c’è una fase di negoziazione, in cui viene deciso che tipo di algoritmo di cifratura usare.

**28-Descrivi IPsec**

L’idea è che l’entità cifrano il contenuto del payload e dunque Il traffico scambiato non è visibile. Un modo per vedere la sicurezza al livello IP è l’uso, da parte di molti istituti, del protocollo di sicurezza IPsec per creare delle VPN, la cui idea è quella di far passare il traffico tra uffici al di sopra dell’internet pubblica, creando così una rete privata di tipo virtuale, in cui il traffico viene cifrato. Questo perché la creazione di una vera e propria rete privata ha un costo di acquisto, di manutenzione e di installazione molto elevati.

I servizi offerti da IPsec sono la cifratura a livello di datagramma, l’integrità dei dati, l’autenticazione dell’origine, evitare attacchi di replay e confidenzialità dei dati. Esistono due principali protocolli: AH (authentication header) ed ESP (encapsulation security payload). Il primo fornisce solo l’autenticazione della sorgente e l’integrità dei dati mentre il secondo fornisce anche la riservatezza e per questo è più diffuso. Inoltre, IPsec ha due modalità di funzionamento: transport e tunnel. Nel primo solo il payload del datagramma è cifrato e autenticato, nel secondo l’intero datagramma viene cifrato e autenticato e questo viene incapsulato in un nuovo datagramma con un nuovo indirizzo IP, inviato successivamente verso la destinazione. Tra le 4 combinazioni di protocollo AH ed ESP con le due modalità di funzionamenti, quella con ESP in modalità tunnel è la più utilizzata e la più importante.

Prima di inviare i datagrammi, l’host sorgente e destinatario creano un canale logico a livello di rete detto associazione di sicurezza (SA). Essendo unidirezionali, se i due host vogliono scambiarsi datagrammi in modo sicuro devono esistere due associazioni di sicurezza. Le entità che si scambiano traffico devono mantenere le informazioni sulle SA e per garantire la comunicazione sicura ho bisogno di una coppia di SA per ogni coppia di entità che vogliono comunicare tra loro. Tutte queste informazioni riguardo le SA vengono salvate nel loro database di associazione di sicurezza (SAD). Queste informazioni, infatti, servono ogni volta che devo scambiare traffico perché per ogni pacchetto da inviare da un router, devo sapere se è soggetto a politiche IPsec e vedere quali SA è associata a quel pacchetto. Se ci sono n venditori ogni router dovrà gestire 2+2n SA.

Un datagramma è formato dal pacchetto originale, un trailer ESP che aggiusta, tramite il padding, la lunghezza in modo che sia multipla di quanto previsto dal protocollo e queste due parti vengono cifrate. Inoltre, c’è l’intestazione ESP che include l’SPI, che indica all’entità ricevente a quale SA il datagramma appartiene, e il numero di sequenza e questa parte, insieme alla parte cifrata, viene autenticata. Tutto questo viene messo all’interno di un nuovo pacchetto IP. Il sequence number per una nuova SA parte da 0 e ogni volta che devo scambiare su una SA un datagramma lo incremento e ciò permette di prevenire un replay attack.

Tutte queste informazioni vengono memorizzate dall’entità IPsec e da un SAD in una nuova struttura detta SPD (security policy database) e dunque per ogni datagramma le entità devono sapere come gestire quel pacchetto secondo IPsec. In particolare, l’SPD mantiene l’informazione su cosa fare quando arriva un pacchetto.

Dunque, un intruso non riesce ad accedere al contenuto dei pacchetti perché sono cifrati, non riesce a vedere le sorgenti e le destinazioni perché queste informazioni fanno parte del payload che viene cifrato, non può usare l’indirizzo IP come indirizzo IP sorgente per sostituirsi ad esso perché la cifratura usa una chiave nota solo ai router e, infine, non può fare un replay attack perché i datagrammi vengono numerati e la numerazione è protetta con l’integrità. Quindi un intruso può vedere il traffico ma non può estrarne informazioni.

**29-Descrivi Firewalls**

Il firewall è una combinazione di hardware e software che separa una rete privata da Internet e permette di controllare il traffico tra il mondo esterno e le risorse interne, scegliendo quali pacchetti far transitare e quali no. Gli obbiettivi di un firewall sono:

1. Impedire gli attacchi di SYN flooding, ossia un attacco di tipo denial of service nel quale un intruso invia una serie di richieste SYN-TCP verso il sistema oggetto dell’attacco, senza rispondere con un ACK e riempendo così la struttura dati del server in cui viene tenuta traccia delle connessioni instaurate.
2. Prevenire modifiche dei dati interni alla rete.
3. Impedire accessi dall’esterno della rete verso l’interno.

I firewall possono essere divisi in tre categorie:

1. Firewall stateless che fanno filtraggio di pacchetti, il cui compito è quello di analizzare il traffico in entrata ed in uscita e decidere se farlo passare o meno. Questa scelta viene effettuata in base alle informazioni di livello 3 o 4 come l’indirizzo IP sorgente e destinazione, i numeri di porta sorgente e destinazione, il tipo di messaggio ICMP e i flag TCP come i SYN e gli ACK. Per esempio, se non si vuole avere traffico Web uscente allora si scartano tutti i pacchetti con destinazione porta 80, oppure se non si vogliono avere connessioni TCP eccetto per quelle che riguardano un preciso Web server allora tutte le richieste SYN vengono scartate tranne quelle con un indirizzo stabilito e con porta destinazione 80.

Per effettuare questi controlli, ci sono delle regole che vengono implementate tramite delle liste di controllo di accesso (ACL): se un pacchetto non rispetta determinate regole allora non viene fatto passare.

1. Firewall con stato che fanno filtraggio di pacchetti: a differenza del precedente, in cui le decisioni del filtraggio venivano prese su ciascun pacchetto in modo indipendente senza guardare la storia precedente, in questi firewall viene tenuto traccia delle connessioni TCP in modo da vedere se determinati pacchetti hanno senso in quel momento. In questo caso, le ACL hanno un controllo sullo stato della connessione in quel momento.
2. Firewall di tipo application gateways: nel caso in cui si voglia fornire un servizio Telnet solo a determinati utenti, bisogna avere informazioni sull’identità degli utenti e queste informazioni non si trovano nelle intestazioni IP, TCP o UDP ma sono nel livello applicativo. Per far ciò, i firewall fanno il filtraggio dei pacchetti unito ad un gateway a livello applicativo, che prende decisioni in base ai dati applicativi. Ci sono, però, delle limitazioni perché i nodi possono cambiare il proprio indirizzo IP sorgente, ogni applicazione ha bisogno del proprio application gateway e ci sono dei costi in quanto i dati devono passare attraverso il gateway.