Appunti light quiz

**DNS**:

protocollo applicativo, utilizza il protocollo di trasporto UDP sulla porta 53 che ha il compito di tradurre il domino del sito in un indirizzo IP.

Utilizza una struttura gerarchica, con i vari server root, top-level-domain, server autoritativi in maniera iterativa e i server locali in maniera ricorsiva (host richiesta iterativa). è prevede il caching sui (server locali iterativi/client invia richieste ricorsive).

**FTP**: che si basa sul TCP utilizzando le porte 20(dati) e 21(controllo), esso opera a livello applicativo che si occupa del trasferimento di dati tra un client e server.

Esso può avere 2 modalità:

Passiva: il client chiede una connessione al server sulla 21, dicendogli su quale porta effimera comunicare i dati(20).

Attiva: il client chiede una connessione al server sulla 21 ma il server di sua iniziativa comunica sulla porta 20

Client: un socket viene utilizzato per l’accept della connessione dati.

Il secondo socket viene utilizzato per la vera e propria connessione dati.

Ed in fine il terzo viene utilizzato per la connessione di controllo.

Server: un socket viene utilizzato per l’accept della connessione dati.

Il secondo socket viene utilizzato per la vera e propria connessione dati.

Ed in fine il terzo viene utilizzato per la connessione di controllo

utilizzare Username e Password per accedere.

**HTTP**: protocollo applicativo, che si appoggia al TCP/80, può avere più connessioni contemporaneamente e utilizza i cookie per risolvere il problema della mancanza di stato da parte del server (usato come principale sistema per uno scambio di dato sul web in un'architettura tipica client-server.).

Considerazione: mancanza di stato vuol dire che il server non si ricorda del client, quindi il client invia i cookie che contengono delle informazioni che il server.

1. : non c’è autenticazione, la connessione non è permanente e puoi inviare una richiesta alla volta, header variabile lunghezza

1.1 : La connessione è pemanete e si introduce la tecnica del pipelining (puoi inviare più richieste senza che ti sia arrivata una risposta), header fisso ma con aggiunta di header opzionali.

2.0 : possibilità di inviare le risposte non in ordine in modo tal da ottimizzare la pagina.

**Protocolli livello 2 (data link)**: La famiglia degli standard è la IEEE 802.\* .

Essendo di livello 2, si occupano del collegamento 2+ host in maneira diretta in modo tale che possano comunicare.

Vi è un controllo di integrità del frame (messo in un suo campo ) bastato sul calcolo del CRC32.

Ed il problmea delle collisionmi viene trattato con lagoritmo CSMA/CD

Ethrnet: via cavo, CSMA/CD (CRC32,jamming, round trip time-> attesa casuale aumenta in caso di collisione)

Wifi: via radio ,CSMA/CA (ACK), modalità (Peer-to-peer, Infrastucture), access point (RTS, CTS)

**INODE**: il campo “tipo di file” può essere solo: directory, soft link, file regolare, pipe, socket, dispositivi a blocchi / caratteri.

Funzionamento: contengono i metadati dei file a parte il nome (e puntatore al file in memoria). Sono contenuti nella tabella degli Inode che è contenuta nel **super blocco (che contiene anche la tabella bit-map degli inode se sono usati / e la bit-map dei dati quali blocchi di dati sono utilizzati).** ogni I-node all’inizio ci sono le informazioni sui meta dati, poi ci sono dei blocchi che puntano ai file in memoria in maniera diretta (indicizzata) ai primi blocchi(dell’inode) e in maniera indiretta ad altri blocchi di puntatori che lavorano allo stesso modo (in maniera ricorsiva) fin che non finisce il file.

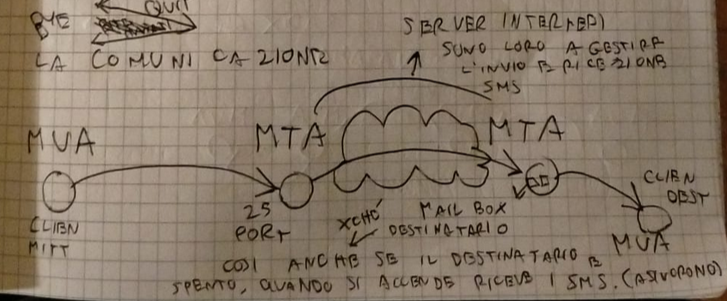
**PROTOCOLLO IP**: protocollo che opera a livello 3 (network), implementa un servizio datagram Best effort (descrive un servizio di rete che garantisce il suo funzionamento se a livelli inferiori come il fisico non ci sono errori che non permettono il suo corretto funzionamento) grazie al protocollo ICMP (è un protocollo di servizio per reti a pacchetto che si occupa di trasmettere informazioni riguardanti malfunzionamenti, informazioni di controllo.)

Ipv4(privati, pubblici, come costruito indirizzo), ipv6 (più opzioni , no frammentazione, ipv6 ridimensiona pacchetto per ipv4).

Instradamento multi-hop: prevede che i router abbiano una tabella della mappatura (Lo scambio di messaggi fra i router per la costruzione della tabella sono di tipo ICMP) dei router precompilata prima di fare un hop per l’instradamento dei datagram (I datagrammi sono di formato prestabilito e adotta una tecnica di trasferimento store and forward, perché i router hanno una cache).

**SMTP**: si appoggia alla porta 25/TCP sulla vecchia, nuova 587/TCP (cifrata).

Questo protocollo per inviare file multimediali utilizza una codifica MIME (utilizza radix 64), l’invio di messaggi è in COD ASCII, se li memorizza li invia (store and forward). Si appoggia al TCP, ma essendo di livello 5 è best effort. Per scaricare la email box-> **POP3 si scarica tutto / IMAP4 scarica gli heder e poi scegli quella vuoi leggere**



**NTP**: è un protocollo applicativo che si occupa della sincronizzazione degli orologi mondiali. È un protocollo gerarchico, ma peer to peer, perché ogni host può essere configurato sia come client che come server. E si appoggia al UDP / 123. (best effort)

**PROTOCOLLI LIVELLO 5 (APPLICATIVO)**: Essendo di livello 5 si basano sui protocolli di livello 4 UDP e TCP.

Questi protocolli comunicano attraverso le porte (socket) stabilite da IANA. Questo livello si occupa di come far scambiare le informazioni fra i vari processi di rete attraverso i socket.

**Schedulign**:

FIFO: il primo processo che arriva viene eseguito fino alla fine

Shortest-Job First: seleziona il processo in attesa con la più piccola sequenza successiva di operazioni(starvation /praticamente non implementabile in quanto non è possibile stabilire con certezza la durata).

Shortest Time-To-Completion First: seleziona il processo in attesa con la più piccola sequenza successiva di operazioni, però se ne arriva uno più corto, mette quello che sta eseguendo in attesa ed inizia ad eseguire quello più corto(starvation/ praticamente non implementabile in quanto non è possibile stabilire con certezza la durata)

Round-Robin: esegue i processi nell'ordine d'arrivo, ma esegue la processo fino alla "quantità di tempo" prestabilita in caso non fosse finito, dopo lo mette in attesa e passa al prossimo fino a quando il ciclo non ricomincia

Multi-level Feedback Queue: sarebbe il round-Robin ma con le priorità, la starvation viene risolta con l’aumento della priorità di tutti.

**SysCall**: sono chiamate molto pesanti perché richiede la CPU in modalità privilegiata, perché sono istruzioni privilegiate (overhead). Le funzioni sono più leggere perché non sono privilegiate.

**TCP**: protocollo di livello 4 affidabile grazie al meccanismo di ritrasmissione in caso di time out.

Per evitare la perdita di informazione, Il controllo di flusso permette siadi evitare perdite di datagrammi indipendentemente dalla velocità, perché nel datagramma c’è scritta la riceve windows quindi il pacchetto non deve superare quella soglia, perché se no sbulacca. (non c’è il controllo crc perché viene garantito dai livelli inferiori, nel UDP si / hand shake / stream/ **Congestion Windows**).

**UDP**: protocollo di livello 4 non affidabile ma veloce rispetto a TCP, utilizza i datagrammi(no connessione) con servizio best effort (crc32) utile per applicazioni real-time

**EXEC**: Facendo la EXEC il codice del processo chiamante viene sostituito con quello che è stato lanciato con la EXEC. Si controlla se il processo può eseguire il file (per via dei permessi e se è possibile farlo) e viene caricato in memoria fisica.

: Il primo numero (in questo caso sempre 1) riguarda i link, il successivo riguarda l’id del user, quello dopo riguarda il gruppo e l’ultimo riguarda la grandezza del file (sempre 0).

**Tabella delle pagine**: Le tabelle delle pagine vengono aperte per i processi per far si che possano trattare la memoria che hanno come contigua. Quindi i numeri delle tabelle sono pari ai numero dei processi.

**File descriptor**: (è un numero intero non negativo che rappresenta un file, una pipe o un socket aperto da un processo e sul quale il processo può effettuare operazioni di input/output) ce n’è uno per File aperto in ogni processo(perché se un file è aperto di conseguenza ha un file descriptor aperto).

**UID processi permessi**: set bit attivo tu esegui il processo con le proprità di quell

Effective: permessi di quello che sta girando (set bit attivo Effective è quello del proprietario)

Real: è UID dell’utente che ha lanciato il processo

Seved: Seved contiene il modo di esecuzione del file, Effective copia il saved ma se il set user id è abilitato il seved e l’ Effective diventano come quello del proprietario, se invece è disabilitato sono tutti e 3 uguali (viene fatto inmodo tale da poter scambiare le priorità fra seved e real)

l kernel identifica ogni utente con un numero intero: UID(0 = ROOT) i gruppi con GID

minimo privilegio: ciascuna entità dovrebbe avere il minimo privilegio, che

gli permetta di eseguire i suoi compiti legittimi

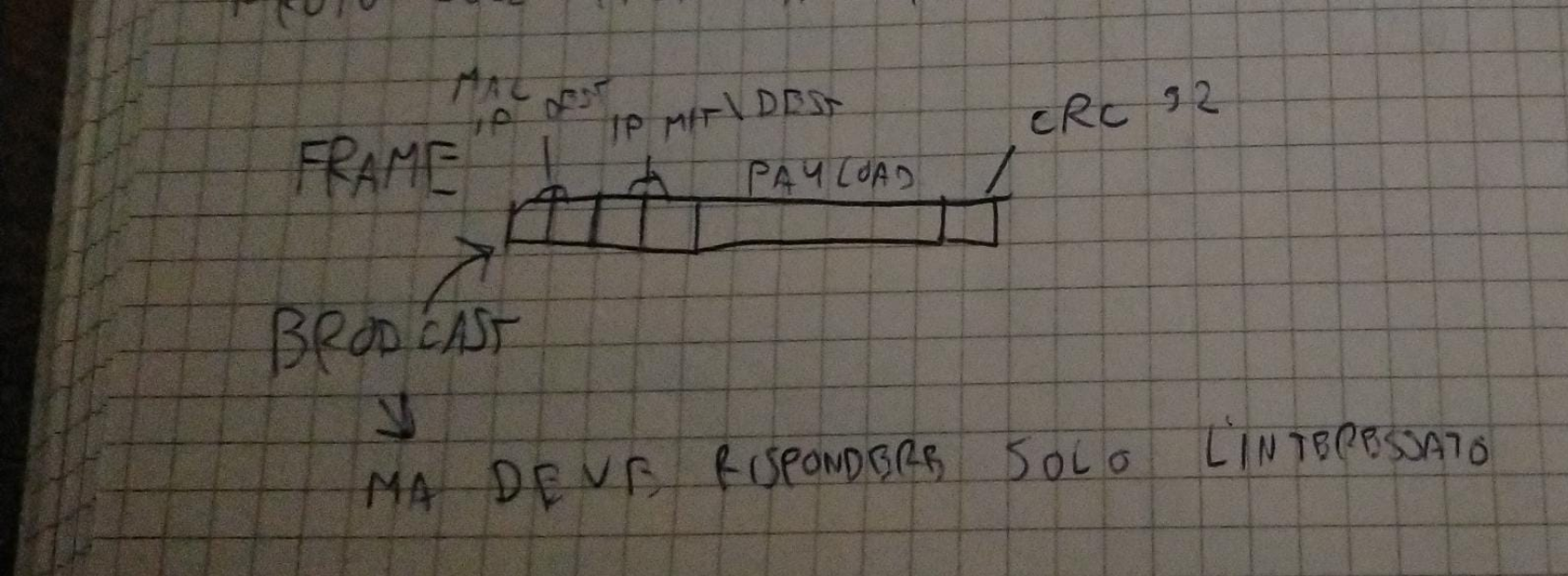
**DHCP**: protocollo applicativo che si appoggia ad UDP/67 SERVER /68 CLIENT

Le 4 fasi sono 2 in brodcast per il messaggio che invia il client e il server invia sempre in unicast.

Questi server DHCP possono essere anche esterni ma hanno un pull di indirizzi privati, di conseguenza operano a livello locale.

**ARP** (requeste- responce): protocollo che opera a livello 2 (data link) che ha il compito di fornire una mappatura fra indirizzo IP e MAC della rete locale ethernet (per far sì che 2 host possano comunicare)

La richiesta è in Brodcast (perché giustamente non sa l’indirizzo MAC, ma conosce l’IP), ma de a questa richiesta deve rispondere solo il destinatario.



**Thread**: permettono di avere più flussi di esecuzione in un processo (normalmente un processo normale ha solo un filo di esecuzione), La differenza principale fra thread e processi è la memoria condivisa (condivide codice e dati con altri thread dello stesso processo), per questo bisogna utilizzare i Mutex per lokare quella zona critica in modo tale che non accedano contemporaneamente. (Thread vengono creati con pthread\_create e terminano con void pthread\_exit(void \*retval); / il mutex si crea con pthread\_mutex\_t / e il processo padre aspetta i Thread con pthread\_join).

**Processo**: per processo si intende un programma in esecuzione, si può originare da una Fork(); e vengono identificati dai pid (il padre pid 0 e il figlio avrà il suo numero, la fork clona il processo padre), i processi non hanno la memoria condivisa.