Queste sono **alcune** domande estratte da esami dal 2017 al 2021.
Di seguito riportiamo le possibili soluzioni.

unità di misura

Elencare le 4 componenti del ritardo sofferto da un pacchetto nel suo percorso da sorgente a destinazione, evidenziando per ognuno di esse i parametri che ne determinano la durata.

Ci sono 4 tipi di ritardo che un pacchetto può soffrire nel suo percorso da S a D:

- **propagazione** = distanza apparati / velocità di propagazione (dipende dalla natura del mezzo: rame, fibra ottica, wireless)
- **processing**(elaborazione) = dipende da quanto è **veloce** il router.
- **trasmissione** = quantità di dati da trasmettere / bit rate
- **accodamento** = dipende dalla **quantità di lavoro accumulata sugli apparati di rete.** Ha un effetto valanga in alcuni casi, infatti il ritardo in coda cresce esponenzialmente con l'approssimarsi dell'**intensità di traffico** a 1.

la rete

Descrivere in cosa consiste la commutazione di circuito

La **commutazione di circuito** consiste in una connessione tra due host in una rete p2p (point-to-point). Ogni comunicazione effettuata tramite la commutazione di circuito coinvolge 3 fasi:

1. **apertura della connessione**, detto anche handshaking. La sorgente e la destinazione vengono messe in relazione e si scambiano informazioni di controllo.
2. **scambio dati.**
3. **chiusura connessione**.

Una volta aperta la connessione **non c'è ritardo** di processing e di accodamento, solo un minimo ritardo di propagazione. I dati arrivano in ordine.
Questo tipo di commutazione non è molto efficiente, infatti dato che il path è riservato e vengono usati solo qualche kbyte (es.: per la codifica della voce), il resto della **capacità del cavo è sprecata**.
Inoltre ha una **scarsa resistenza ai guasti**.

Quali sono vantaggi e svantaggi delle reti a commutazione di pacchetto rispetto alle reti a commutazione di circuito?
Confrontare pro e contro delle tecniche a commutazione di circuito e commutazione di pacchetto.
Illustrare i motivi per cui nelle moderne reti di trasmissione dati si adotta la tecnica di commutazione di pacchetto invece della tecnica di commutazione di circuito.

Nella **commutazione di pacchetto** i dati vengono rotti in pacchetti di dimensione massima fissata che dipende dalle caratteristiche della rete. Non esiste un cammino unico tra sorgente e destinazione, ogni pacchetto viene mandato sulla rotta migliore usando una tecnica store-and-forward. Ha il vantaggio che si può fare pipelining, cioè avere parallelismo. Questo riduce di molto la latenza.
Gli switch qui sono in grado di memorizzare i pacchetti, oltre che di inoltrarli. Il router possiede una coda dove possono essere memorizzati i pacchetti prima di essere inoltrati e si occupa di fare instradamento (store-and-forward).
La commutazione di pacchetto è più vantaggiosa rispetto alla commutazione di circuito perché usa la tecnica **store-and forward**, cioè ogni pacchetto viene mandato sulla rotta migliore. Inoltre può fare **pipelining**, cioè avere parallelismo. Rispetto alla commutazione di circuito **spreca meno la capacità del cavo ed è più resistente ai danni.**

Nella comunicazione di circuito, invece, il vantaggio è a parte all'inizio per stabilire la connessione, **non ci saranno ritardi di processing e di accodamento** e le informazioni saranno passate attraverso un unico blocco di dati su un **path riservato.** Il lato negativo qui è la s**carsa resistenza ai guasti**, oltre allo **spreco della capacità del link**.

Definire le proprietà dei seguenti servizi:

- connectionless
- connection-oriented
- affidabile
- inaffidabile

e dire quali tipi di servizi sono erogati dai seguenti protocolli: TCP, UDP, IP, DHCP.

le architetture di rete standard

Quali sono le funzionalità in carico al livello di collegamento (Data Link) dello stack TCP/IP?

Si occupa del trasferimento del pacchetto attraverso il collegamento, dopo che i router hanno scelto il migliore link. L'architettura TCP/IP non fornisce protocolli specifici qui (qualsiasi protocollo adatto al compito va bene). I pacchetti qui si chiamano *frame*.

livello applicazione: protocolli HTTP

Spiegare qual'è la funzione dei numeri di porta usati dal livello Trasporto, quali sono gli intervalli di valori definiti da ICANN per tali numeri, e qual è l'utilizzo previsto per ogni intervallo

I **numeri di porta** sono interi corti senza segno su 16 bit, utilizzati per identificare una particolare connessione di trasporto tra quelle al momento attive su un calcolatore.

In base al range cambia il significato :

- **0-255 well-known** per servizi standard. (web, mail, etc..) (standard ITF e IEEE)
- **256-1023 per servizi in test** (oggi c'è qualche servizio standard, perchè non bastano 256 per i well-known)
- **1024-49151 Registered**. Non standard, forniti da aziende. Ad esempio i servizi di Cloud, sono proprietari e non ci sono documenti che dicono come siano fatti all'interno.
- **>49151 effimere/dinamiche**. Servono a sviluppatori, in cui non si usa la rete. Oppure vengono usate regolarmente da processi client per comunicare in rete; nel momento in cui si chiude il processo la porta torna libera.

Illustrare in che cosa consiste il concetto di incapsulamento, quale scopo esso ha, e quale ricaduta ha sulle prestazioni di rete.

Spiegare in cosa consiste la funzionalità di multiplexing e demultiplexing, illustrando in che situazioni è utilizzata.

Ci sono 2 applicazioni in esecuzione sullo stesso host, entrambe devono generare dei dati e quindi chiedono servizio al livello di trasporto sfruttando i service access point.

Il livello di trasporto aggiunge ai dati di queste applicazioni il suo **indirizzo di trasporto**, cioè i numeri di porta.

In seguito inoltra i dati al livello rete che aggiunge ai dati l**'indirizzo di rete** (i dati delle applicazioni risiedono allo stesso indirizzo di rete poiché sono partiti tutti da un unico host). Tutto questo processo è definito incapsulamento o **multiplexing**.

Il **demultiplexing** è l'inverso del multiplexing. Arrivano delle informazioni dal livello rete e queste vengono smistate, grazie prima all'indirizzo di rete e poi al numero di porta, su diverse flussi per far arrivare i dati alle applicazioni destinatarie di livello applicazione.
Ha lo **scopo** di utilizzare al meglio i servizi dei vari livelli dello stack.
Ha una ricaduta sulla rete del tipo...ritardo di processing
è usata in situazioni di … quando ci sono più applicazioni che devono inviare dati a un altro host

Incapsulamento nell'host sorgente

1. A **livello applicazione** i messaggi non contengono solitamente alcuna intestazione (header) o trailer (informazioni di controllo in coda).
2. Il **livello di trasporto** considera il messaggio ricevuto come payload (carico di dati) e gli **aggiunge un'intestazione** con informazioni per la gestione del pacchetto (a quel livello), ovvero lo **incapsula**, rendendolo un segmento (o datagramma utente, a seconda del protocollo).
3. Il **livello rete** aggiunge anche lui un'intestazione che contiene indirizzi degli host (sorgente e destinatario) e altre informazioni per il controllo dell'errore. Il risultato è un datagramma.
4. Il **livello di collegamento** aggiunge la propria intestazione, che contiene l'indirizzo del livello di collegamento dell'host o del next hop (router). Il risultato è un frame.

Decapsulamento nell'host destinatario
Ciascun livello rimuove la propria intestazione dal pacchetto ricevuto e ne estrae il payload, passandolo al protocollo sovrastante (si effettua anche controllo di errori).

Descrivere il formato del messaggio HTTP Request e illustrarne i campi.
La **richiesta HTTP** ha il formato: <riga richiesta, righe di intestazione, riga vuota, corpo entità>.

1. **<riga di richiesta>** è composta da:
    a. metodo (comando che il client manda al server)
    b. URL
    c. versione http che si sta usando
2. **<righe di intestazione>** = (nome campo seguito dal valore) permettono di specificare delle informazioni aggiuntive come cookies e un elenco di cosa il browser client è capace di fare. Nelle righe di intestazione ci può essere il valore *"if modified since"*, in cui si fa una richiesta condizionale al server in cui la data di modifica nello header della copia del server deve essere successiva a quella mandata nella richiesta.
3. **<riga vuota>**
4. **<corpo entità>** = contiene le informazioni da inviare al server, viene usato dal metodo POST.

I metodi più utilizzati sono :

- **GET** <filename> → read file named (ma anche form «?»)
- **HEAD** <filename> → read header of webpage named
- **PUT** <filename> → write data to file named
- **POST** <filename> → append data to file named (form)
- **DELETE** <filename> → delete file named

Illustrare come sono usate le connessioni persistenti adottate a partire da HTTP 1.1, e spiegare quali vantaggi portano rispetto alle connessioni non persistenti.

Confrontare l'uso di connessioni persistenti e non-persistenti adottate in diverse versioni di HTTP.

La versione HTTP 1.1 prevede di default l'uso di **connessioni persistenti**, a meno che il client non lo richieda specificatamente. In questo caso può succedere che:

- il **client non chiude esplicitamente la connessione** con quel server, ma si interrompe perché, per esempio, **chiudendo il browser** si accede a un altro server tramite link o perché se ne inserisce uno nuovo tramite l'url.
- se, dopo un **certo tempo**, non arriva una chiusura esplicita dal client allora la connessione viene automaticamente chiusa, sia da http-client che da http-server perché per un certo tempo non c'è stata alcuna attività su quella connessione. (time-out).

L'impiego di connessioni persistenti consente di **risparmiare tempo e risorse** (messaggi e informazioni di stato per diverse connessioni) rispetto alle **connessioni non persistenti** (venivano usate nelle versioni precedenti di HTTP 1.1).

Descrivere come vengono gestiti i cookie in HTTP, e per quali scopi possono essere usati. Spiegare il funzionamento dei cookie.

In HTTP, illustrare come vengono gestiti i cookies così da mantenere informazioni di stato nonostante la natura stateless del protocollo.

HTTP è senza memoria (stateless). Si aggiunge memoria tramite il meccanismo dei **cookies**. Se i cookies sono abilitati allora fra le righe di intestazione della risposta del server può esserci un nome campo : "*Set_cookie*" seguito da un valore identificativo numerico *IDc* (identificatore numerico del client per il server).

Il database del server ricorda i dati del cliente associati a IDc (file scaricati, dati inseriti in form…) e, poi, l'applicazione (browser client) vede se il sito è già stato visitato e nel database dei cookies prende IDc e lo passa ad HTTP nei messaggi di richiesta (IDc inserito in richieste a successive interazioni col server).

come si generano i cookies ?

1. il server, quando riceve una richiesta, memorizza le informazioni relative al client in una stringa o in un file (es.: salva il nome di dominio del client, i contenuti di un cookie precedente, i dati inseriti in un form, etc...).
2. il server include il cookie nella risposta che invia al client;
3. il browser che riceve la risposta memorizza il cookie in una directory apposita.

Quando un client invia una richiesta ad un server, controlla nella directory dei cookie se c'è un cookie precedente e nel caso viene incluso nella richiesta; il server che la riceve capisce così che si tratta di un vecchio client.

gli **scopi** possono essere:

- vantaggi: identificazione utenti, mantenimento stato, offerta personalizzata dei contenuti.
- svantaggi: violazioni privacy, data mining.

Quali vantaggi si possono ottenere configurando nel proprio browser un proxy http?

Illustrare quale funzione ha e come opera un proxy HTTP

Un **server http proxy** è un server a servizio di una determinata comunità, scelto all'interno o vicino a un rete di tale comunità. Fa caching dei contenuti per tutta la comunità, cioè da qualunque end-system parta una richiesta per procurare un determinato contenuto web all'interno di una comunità, il proxy riceve quel contenuto e ne mantiene una copia per un certo periodo per tutti gli utenti.

I **vantaggi** che si possono ottenere configurando nel proprio browser un proxy http sono la riduzioni di:

- **latenza** → non bisognerà aspettare molto tempo (rispetto al server originario) se il contenuto lo troviamo in locale o su un server vicino.
- **traffico in rete** → non c'è bisogno di attraversare tutto Internet.

Come funziona un proxy server ?

- Tutti i browser di una comunità possono essere configurati per usare un http proxy che si trova nella rete della comunità o ai margini.
- Quando un browser genera una richiesta (parte un messaggio di http request) destinata al server che possiede quel contenuto, questa richiesta viene **incapsulata** una volta di più, in modo che il **destinatario** diventi l'indirizzo di rete del **server http proxy** che ha configurato.
- La richiesta arriva al http proxy che **controlla** nella propria cache se il proxy ha quel contenuto perché precedentemente :
    - lo ha cercato lo stesso utente (e lo ha portato lui stesso nella cache).
    - un'altro utente che si serve del medesimo proxy ha cercato lo stesso contenuto.
- Una volta trovato il contenuto lo fornisce come risposta.
- **Se il contenuto <u>non</u> si trova** il proxy non incapsula niente e fa partire la richiesta originaria http, come se l'avesse generata lui. La **risposta tornerà al proxy** aggiungendo il contenuto nella propria cache. Infine manda il contenuto al richiedente originario.

electronic mail

Schematizzare l'architettura di riferimento per lo scambio di posta elettronica, evidenziando quali entità si comportano come client e quali come server, e quali sono i protocolli usati per la comunicazione tra le varie coppie di entità.

Ci sono diversi componenti dell'architettura di posta elettronica:

- **user agent** = è un software che risiede sull'end-system del client e permette di leggere/inviare mail e allegati. E' anche conosciuto come "mail reader".
- **server di mail** = deve gestire i messaggi di posta in uscita da mandare ad altri server. Quando si manda una mail finisce al server di posta, non direttamente al destinatario utente. Quindi questi mail server hanno ruolo (come i proxy) sia di server sia di client. I mail server, di conseguenza, comunicano in coppia tra di loro, variando il loro ruolo a seconda che stiano inviando o ricevendo. Chi manda la mail ha il ruolo di client mentre fa da server vero è proprio chi riceve perché offre il servizio di ospitare il messaggi.
- **SMTP** = Simple Mail Transfer Protocol, cioè il protocollo di trasferimento di base. Realizza la comunicazione tra i server.

SMTP è un protocollo **push** infatti :

- push che viene utilizzato dallo UA per spingere i messaggi che devono essere spediti verso la MTA mittente.
- push da MTA mittente per spingere i messaggi a MTA destinatario.

Nell'ultimo passaggio si fa **pull** da parte dall'UA destinatario che chiede al proprio server se ci sono mail nuove. Questo perché lo **UA di un destinatario**, diversamente da un MTA che esiste sempre ed è sempre pronto in qualsiasi momento a ricevere un messaggio, sia che il cliente sia uno UA che un altro server, **non esiste sempre** e quindi MTA del destinatario **non può fare push**. Bisogna usare un approccio pull, quando l'utente destinatario vuole guardare la sua posta attiva lo UA e quest'ultimo

chiederà il contenuto della propria mailbox dal proprio MTA di riferimento (in questo caso MTA viene chiamato MAA = Message Access Agent).
Quindi non si usa il SMTP ma 2 protocolli diversi:

- **pop3** → post office protocol. È molto semplice ma limitato. Comunica allo UA che è arrivato un messaggio e permette di fare:
  - download and delete → cancellare
  - download and keep → mantenere una copia
- **IMAP** → Internet Mail Access Protocol. È più complesso e permette più caratteristiche come mantenere dei folder di massaggi sul MAA e di scaricare solo lo header della mail o lo header e il corpo senza gli allegati (comodo per dispositivi con poche risorse). Supporta l'accesso utente da diversi host: non scarica le mail su host utente; POP3 fa lo stesso solo se configurato "download & keep".

Dati due server SMTP X e Y, descrivere in dettaglio i passi seguiti in ognuna delle fasi di comunicazione per la consegna da X a Y di un messaggio di posta.

Discutere le informazioni aggiunte da MIME ad un messaggio di posta elettronica per consentire la gestione degli allegati.
Illustrate le funzionalità svolte da MIME, e i meccanismi usati per fornire servizio.
MIME (Multipurpose Internet Mail Extensions) è un protocollo che serve per estendere il servizio fornito da SMTP per permettere il trasporto di altri tipi di dati. Aggiunge nella mail, sia nello header che nel corpo, delle informazioni di controllo per dialogare con il MIME a destinazione come:

1. **MIME-version** = indica la versione di MIME.
2. **Content-Type** = definisce il tipo di dati nel corpo.
3. **Content-Transfer-Encoding** = definisce il metodo utilizzato per la codifica del messaggio in binario per il trasporto.
4. **Content-id** = individua in modo univoco una parte del messaggio in messaggi composti da più parti.
5. **Content-Description** = indica se il corpo del messaggio contiene un'immagine, un file audio o video.
6. **Content-Length** = numero di caratteri ascii utilizzati.

Se gli allegati sono più di uno, lo header MIME lo indica con *multipart/mixed*. Tutti gli allegati sono separati e descritti dai 5 campi sopra citati.

TELNET

Descrivere il protocollo SSH e le sue varie componenti
SSH (Secure SHell) è un'applicazione nata per sostituire TELNET e **risolvere i suoi problemi** di sicurezza. Permette di aprire una shell su una macchina remota su cui ho un account per lavorare su quella macchina come se fossi in locale.
Il protocollo di livello applicazione è composto da 3 diversi componenti:

- **SSH-TRANS** → ha l'obiettivo di costruire un **canale di comunicazione sicuro** sfruttando la connessione offerta da TCP (il protocollo TCP trasmette tutte le informazioni in chiaro e quindi non è in grado di garantire privacy e confidenzialità). Ciò è possibile grazie ad un

insieme di *tecniche crittografiche* che permettono al client di verificare che il server a cui si sta collegando sia quello legittimo.

- **SSH-AUTH** → è responsabile dell'**autenticazione del client**. Oltre alla normale autenticazione con nome utente e password, sono disponibili altri metodi di verifica, come l'accesso basato su coppie di chiavi crittografiche.
- **SSH-CONN** → offre i **servizi veri e propri di livello applicazione**: accesso al terminale, trasferimento file, esecuzione remota, creazione di tunnel.

domain name server

Illustrare e porre in relazione gli schemi di indirizzamento usati a livello applicazione, trasporto e network, per identificare la destinazione di un messaggio.

Gli schemi di indirizzamento sono:

- nome simbolico → livello 7 applicazione, stringa
- numero di porta →livello 4 trasporto, interi senza segno da 16 bit
- indirizzo di rete IP → livello 3 network, unsigned long su 32 bit
- MAC address o HW address → livello 2 datalink, indirizzi su 48 bit unici per ogni scheda di rete

Illustrare la struttura e il significato dei campi di un resource record DNS.
Descrivere la struttura di un DNS resource record, esemplificando quali valori possono essere contenuti nei vari campi.

Un **record di risorsa** si trova nel database mantenuto da ogni name server (per ogni nome è mantenuto uno (o più) resource record). E' composto da righe di 32 bit contenenti: <nome di dominio, tipo, classe, TTL, valore>.

1. **nome di dominio** (identificativo del record, può avere 255 caratteri al massimo)
2. **tipo** (come interpretare il campo <valore>: determina informazioni mantenute nel record)
3. **classe** (tipo di rete) (di fatto ha valore solo IN, cioè Internet)
4. **TTL** (=time to live, tempo di vita di questo record) (segnala per quanto tempo deve rimanere in cache l'informazione) (generalmente 2 giorni)
5. **valore** (informazione relativa al nome del dominio).

Ad ogni nome di dominio (un nodo) è associato un record di risorsa.

Quale, tra gli schemi iterativo e ricorsivo di risoluzione della query DNS, produce maggior sovraccarico di memoria nei server? motivare la risposta

la risoluzione ricorsiva produce maggior carico di memoria perché ogni server DNS intermedio a cui viene inoltrata la query deve fare caching della reply.

Si descrivano i passaggi eseguiti e i nodi coinvolti nel DNS ai fini della risoluzione del nome simbolico newton.cs.cam.ac.uk per determinare l'indirizzo IP del PC newton del Computer Science Department dell'Università di Cambridge. Si assuma una politica di risoluzione ricorsiva.

i passaggi per la risoluzione del nome simbolico newton.cs.cam.ac.uk usando uno politica ricorsiva sono:

1. Il programma applicativo sull'host sorgente chiede al **resolver DNS** l'indirizzo IP dell'host destinatario.
2. Il resolver DNS, che non conosce tale indirizzo, invia una query al **DNS locale**.

3. Il DNS locale invia la query a un server **DNS root** (il suo indirizzo IP è noto).
4. La query viene inviata a un **server di dominio top level = dns.uk**
5. Il dns.uk conosce l'indirizzo IP del **server di dominio di secondo livello = dns.ac.uk** (.ac = short for academia, is in use in many countries as a second-level domain for academic institutions such as universities, colleges, and research institutes).e gli inoltra la query
6. Il dns.ac. uk conosce l'indirizzo IP del **server DNS subdomain = dns.cam.ac.uk** e gli inoltra la query
7. Il dns.cam.ac.uk conosce l'inidizzo IP del **server DNS locale del Computer Science Department = dns.cs.cam.ac.uk** e gli inoltra la query
8. **dns.cs.cam.ac.uk** conosce l'indirizzo IP del destinatario **newton.cs.cam.ac.uk**
9. L'indirizzo IP viene mandato al server DNS top level domain, al DNS root, al DNS locale e infine viene restituito all'host sorgente.

livello trasporto

Spiegare perchè nell'algoritmo Stop & Wait è sufficiente 1 bit per rappresentare i numeri di sequenza associati ai messaggi

Perché stop&wait può mandare solo un messaggio alla volta (e solo quando riceve un ACK). Per distinguere messaggi diversi con lo stesso numero di sequenza si considerano gli ACK perchè, grazie al campo msg_expected, ci dicono cosa si aspettano.

Messaggi da 1500 bit sono inviati su un canale satellitare con bit rate di 1 Mbps e lunghezza 50 000 Km. Gli ack sono sempre trasmessi in piggybacking; gli header sono di lunghezza trascurabile. Qual'è il massimo utilizzo del canale ottenibile con Idle RQ (stop-and-wait)?

bit data = 1500 bit

bit rate = 1Mbps = 1 000 000 bps

T_p=distanza/V_p= 50000/($3 \bullet 10^5$)= 0,16 sec

T_x= bit data/bit rate = 1500/1 000 000= $1,5 \bullet 10^{-3}$sec

U= $T_x/(T_x+2T_p)$ = $4,6 \bullet 10^{-3}$

Quale limitazione è necessario imporre alla dimensione della finestra di invio di una sorgente che adotta un approccio Go-Back-N per lo scambio ordinato e affidabile di messaggi? Motivare la risposta.

GBN prevede un upper bound sulla dimensione della finestra di invio. Dire quale valore ha tale upper bound e motivare questa limitazione.

la dimensione massima della finestra di invio deve essere uguale al **massimo numero di sequenza** (mentre la finestra di ricezione è uguale a 1).

Infatti utilizzando un numero massimo di sequenza maggiore ci sarebbero errori per quanto riguarda la ricezione degli ACK.

Evidenziare vantaggi e svantaggi della politica Go-Back-N per il recupero dei messaggi smarriti, motivando la risposta anche con un esempio.

TCP adotta, come opzione di default, l'approccio GBN. Discutere vantaggi e svantaggi dell'uso di tale approccio rispetto alle risorse di nodi e rete.

vantaggi: resistente alla perdita di ACK perchè usa ACK cumulativo. Fa uso della tecnica di pipelining e quindi spreca meno tempo (rispetto a S&W).
svantaggi:

Go-Back-N non richiede memoria al receiver che aspetta soltanto il prossimo segmento che deve consegnare in ordine, la gestione è più semplice, ci sono meno computazioni, non c'è controllo di flusso perchè il ricevente è predisposto a ricevere solo il segmento richiesto e il resto viene "buttato via", infatti non ci sono problemi di overflow di memoria. Questo protocollo è però più dispendioso in termini di banda. Selective-Repeat può negoziare il dimensionamento della finestra lato receiver ed eventualmente ri-negoziabile dinamicamente -durante un conversazione la dimensione potrebbe cambiare e quindi essere più piccola dell'upper bound (MAX_SEQ+1)/2, negoziazione max message size per facilità bufferizzazione.

Si consideri l'algoritmo Go-Back-N. Discutere il funzionamento in caso di:

- smarrimenti di un messaggio
- smarrimento di un acknowledgement

In una connessione affidabile che adotta un approccio Selective-Repeat, la rete perde il messaggio 15 mentre il destinatario ha ricevuto in ordine e consegnato correttamente tutti i messaggi precedenti. Descrivere che cosa succede da questo momento a lato sorgente e destinatario.
Il D(=destinatario) manda un ACK cumulativo con numero di sequenza associato pari a 14 (prima dello scadere del timer, altrimenti rischia di ricevere nuovamente tutti i messaggi arrivati finora). Il M(=mittente) capisce che D non ha ricevuto il messaggio con #seq=15 e lo trasmette nuovamente. D finalmente.

TCP - transport control protocol

Quali servizi fornisce TCP al soprastante livello applicazione? Si dia una definizione dei servizi elencati.
controllo di flusso, errore.
TCP è un protocollo di tipo:

- **two-way**= cioè full-duplex, supporta lo scambio di dati tra due parti coinvolte in una conversazione contemporaneamente.
- **connection oriented** → ordinato e affidabile
- **byte-oriented** (stream)= quello che viene scambiato è una sequenza di byte e TCP numera ognuno di questi byte. (è possibile trasferire anche un singolo byte, es.: un comando inviato in remoto fatto di un solo byte).

Elencare i flag presenti nello header dei segmenti TCP e spiegare l'uso di ognuno di essi.
il **code bit** (o flag di controllo) contiene 6 bit di controllo, ciascuno dei quali ha un significato diverso:

1. **URG**: puntatore urgente valido, ci sono dei dati urgenti all'interno del payload
2. **ACK**: riscontro valido
3. **PSH**: richiesta di push
4. **RST**: significa che chi ha generato il segmento, sta resettando la connessione
5. **SYN**: sincronizzazione numeri di sequenza, si tratta di una apertura di connessione
6. **FIN**: chi ha generato questo segmento vuole terminare la connessione

Per una connessione TCP, illustrare la differenza tra dati inviati con il flag PSH alzato e dati inviati con il flag URG alzato.

Elencare le informazioni nello pseudo-header usato per calcolare il checksum a Livello Trasporto, e giustificare la nomenclatura pseudo.

Il **checksum** viene calcolato su una sequenza di bit che rappresentano delle informazioni collezionate nello **pseudo-header**, pseudo perchè non è presente nel segmento. Queste informazioni sono:

- **indirizzo IP sorgente** (fa parte dello header di livello 3, non è presente nel segmento TCP)
- **indirizzo IP destinazione** (header di livello 3)
- **protocol ID** (header di livello 3) (serve per fare multiplexing)
- **lunghezza del segmento TCP**
- **header del segmento TCP**, parte fissa più eventuali opzioni
- **dati del payload**, più un byte aggiuntivo (che può esserci o non esserci) in funzione che il numero di byte nel campo dati sia pari o dispari

Il checksum serve a TCP per verificare che i dati siano arrivati all'host giusto.

Descrivere i passaggi eseguiti dalla procedura three-way handshake di TCP, e motivarne l'utilità rispetto a situazioni di malfunzionamento.

1. S manda il primo segmento con il SYN bit alzato marcato con il proprio numero di sequenza iniziale X;
2. R riceve questo segmento e capisce che le due parti si vogliono sincronizzare, analizza i parametri proposti, algoritmi...risponde con eventualmente i propri parametri con segmento con SYN alzato e numero di #seq=Y e ACK;
3. S invia numero di sequenza X+1 con solamente l'ACK alzato.

utilità rispetto a situazioni di malfunzionamento..Nel **connection record** è indicato il numero di sequenza iniziale, da cui viene poi calcolato lo **spiazzamento dei byte** che ci sono nei payload seguenti. I numeri di sequenza di S e R sono diversi (*calcolati con bit meno significativi dei rispettivi clock*). Viene ricordato per assicurare sicurezza e per evitare che ci sia confusione tra sessioni differenti tra un primario e un secondario oppure che una terza parte possa cercare di replicare una sessione tra sorgente-ricevente per propria utilità (malevola).

Definire il problema del controllo di flusso, fornire un esempio di situazione in cui può verificarsi, e spiegare come esso viene gestito in TCP

Definizione:

I feedback del controllo di flusso vengono trasmessi dal TCP ricevente al TCP mittente e da questo al processo applicativo mittente. La maggior parte delle implementazioni TCP non fornisce feedback del flusso di controllo dal processo destinatario al TCP destinatario, ma consente al processo destinatario di richiedere a suo piacimento i dati dal TCP destinatario.

In altre parole, il TCP destinatario controlla il TCP mittente, il quale a sua volta controlla il processo mittente. Il feedback del **controllo di flusso** dal TCP mittente al processo applicativo mittente è ottenuto tramite il semplice **rifiuto dei dati** da parte del TCP mittente quando la sua finestra è completa.

TCP, per controllare il flusso, forza il mittente e il destinatario a regolare la dimensione delle proprie finestre. La finestra di ricezione si chiude quando arrivano altri byte dal mittente e si apre quando

vengono richiesti altri byte dal processo. La finestra di invio si chiude quando ciò viene consentito da un nuovo riscontro e si apre quando la dimensione della finestra (rwnd) segnalata dal destinatario lo consente. La finestra di ricezione non può essere ridotta, mentre la finestra di invio sì, quando il destinatario definisce un valore rwnd inferiore al precedente (alcune implementazioni tuttavia non lo permettono).
Il destinatario può temporaneamente sbarrare (chiusura totale) la finestra inviando il valore rwnd 0. Questo avviene se per qualche ragione il destinatario non vuole ricevere altri dati dal mittente per un certo periodo. In questo caso il mittente non riduce realmente la dimensione della finestra, ma smette di inviare dati fino a che non riceve nuove istruzioni. Anche quando la finestra viene sbarrata da una richiesta del destinatario, il mittente può sempre inviare un segmento con un byte di dati: si tratta del probing, tecnica utilizzata per prevenire lo stallo.
Esempio:

Definire i problemi di controllo di flusso e controllo di congestione, e fornire esempi - a diversi livelli - che includono meccanismi per risolverli.
Definire il problema di controllo di congestione. Quali protocolli si occupano di gestire il problema e con quali meccanismi?
Definizione:
La finestra di invio viene controllata dal destinatario ma ci potrebbe essere congestione intermedia (nel router). Si utilizza una finestra di congestione, che controlla il numero di pacchetti da inviare. La dimensione finale della finestra di invio è il minimo tra rwnd e cwnd.
Come fa il mittente a sapere se c'è congestione?

- Timeout: se il mittente non riceve riscontri (congestione grave). I messaggi non passano a causa di buffer overflow.
- Tre riscontri duplicati: indicano il segmento smarrito (congestione leggera). I messaggi passano ma si fermano nei buffer e arrivano fuori ordine

Esempio: Algoritmi ARQ (Slow Start e altri), source quench in ICMP.

Se il RRT di TCP è correntemente 30 ms, e i seguenti ack arrivano dopo 26 ms, 32 ms, e 24 ms, dalle rispettive trasmissioni, qual è la nuova stima di RTT usando alfa=0.9?
Se il RTT di TCP è correntemente 30 msec, D è 2, RTO è 38 mesec e i successivi ack arrivano dopo 26 msec e 32 msec dalle rispettive trasmissioni, quali sono le nuove stime di RTT e RTO usando alfa=0.9?
esercizio delle esercitazioni (27).
RTT=30 D=2 RTO=38 1°ACK=26 2°ACK=32
1°ACK:

- RTT= 0.9•30 + (1-0.9)•26= 29.6
- D= 0.9•2 + (1-0.9)•(30-26)= 2,16
- RTO= 29,6 + 4•2,16= 38,24

2°ACK:

- RTT= 0.9•29.6 + (1-0.9)•32= 29,84
- D= 0.9•2,16 + (1-0.9)•(30-32)= 2,14
- RTO= 29,84 + 4•2,144= 38,41

Descrivere quali meccanismi sono adottati da TCP (in entrambe le versioni Taho e Reno) per reagire ad un evento di timeout di ritrasmissione, e motivare l'utilità di tali meccanismi.

Si consideri la connessione TCP in cui il sender osserva il verificarsi di un timeout di ritrasmissione. Si descriva come viene aggiornata la congestion window successivamente al timeout, motivando tale scelta progettuale.

Quando si verifica un timeout di ritrasmissione, in entrambe le versioni di TCP Taho e Reno, succede che:

1. slow start threshold = (congestion window)/2
2. congestion window = maximum segment size
3. si riparte con l'algoritmo slow start

Si setta una nuova soglia di SST perché dopo la SST si passa in congestion avoidance, quindi si evita di congestionare la rete.

Descrivere il funzionamento dell'algoritmo di slow start di TCP per il dimensionamento della finestra di congestione, nel caso non si verifichi alcuna situazione anomala.
Descrivere il funzionamento dell'algoritmo di slow start nel caso di una connessione TCP appena creata e in assenza di errori. Dire quale parametro è determinato da tale algoritmo e dare una motivazione della sua suddivisione in due fasi.

Se non si verificano errori durante l'esecuzione dell'algoritmo di slow start, la congestion window cresce esponenzialmente ($CW_{new}=CW_{old}$• MMS) finché non supera la SST. Allora si passa alla fase di congestion avoidance dove la congestion window ha una crescita lineare ($CW_{new}=CW_{old}+ (1/CW_{old})$). motivazione? Risposto alla domanda precedente. (congestion avoidance evita la congestione)

Descrivere in quali circostanze TCP applica la tecnica di Fast Retransmit e in che cosa essa consiste.

TCP Reno applica la tecnica Fast Retransmit(/Recovery) quando si verifica la ricezione di 3 ACK consecutivi. Succede che:

1. SST = CW/2
2. CW = SST + 3MSS
3. se continuano ad arrivare riscontri duplicati si torna al punto 1.
4. se arriva un nuovo ACK si passa all'algoritmo di congestion avoidance
5. se scade il timer di ritrasmissione si comporta come TCP Taho (SST = CW/2 e CW = MSS. Si passa all'algoritmo di slow start).

Illustrare le operazioni compiute da TCP per chiudere una connessione.
Spiegare quando ha senso adottare la modalità half-close per la chiusura di una connessione TCP, e descrivere la procedura eseguita specificando il contenuto dei messaggi scambiati.

Ci sono due opzioni di chiusura di connessione di TCP:

- chiusura **simmetrica**= si usa una sorta di h**andshaking a tre vie**:
    a. il client (che vuole forzare la chiusura della connessione) invia un segmento con flag FIN=1 (che indica la chiusura) e #seq successivo al numero di byte inviato,
    b. il server TCP utilizza un segmento FIN+ACK e #seq adeguato.
    c. Il client TCP invia l'ultimo segmento ACK. Eventuali dati ancora in buffer di invio possono essere flushed (=TCP manda gli ultimi dati che ha nel buffer al server).
- chiusura **asimmetrica**= si usa una sorta **handshaking a quattro** vie con opzione **half-close**: uno dei due processi può smettere di inviare dati mentre ne sta ancora ricevendo (chiusura a metà). Ad esempio, ciò è **necessario nell'ordinamento** (client manda dati e server li ordina quando li ha tutti).

UDP - user data protocol

Spiegare quando e perché usare il protocollo UDP è preferibile rispetto al TCP

UDP è un protocollo connectionless, non garantisce ordine e affidabilità, privo di controllo di flusso e congestione. Tuttavia, data la sua semplice implementazione, è adatto ad alcune comunicazioni come multicast, dove si parla con un gruppo di apparati con un indirizzo di rete comune e che posso stare su reti differenti. TCP non è adatto a multicast perchè fa fatica a gestire le distanze tra le varie destinazioni e le diverse reti.

UDP è best effort, leggero e semplice ed è per questo che è anche adatto al multimedia soprattutto real time.

network layer

Descrivere il formato dell'header di IPv4, inclusivo di opzioni.

Easy, basta descrivere le robe...

illustrare l'uso dei campi dello header IPv4 previsti per la gestione della frammentazione dei pacchetti.

I due flag sono DF (Don't Fragment) e MF (More Fragments). Il primo è di default alzato e vale uno, ovvero indica che non si deve frammentare il datagramma, ed è usato in TCP, il secondo indica se ci sono ulteriori frammenti (dopo di lui) quando è impostato a 1.

Come è gestita in IPv4 la frammentazione di un datagramma.

Quali sono i campi dello header IPv4 previsti per gestire la frammentazione e quale informazione contiene ognuno di essi?

Spiegare quali informazioni sono usate - e come - nello header IPv4 per la gestione della frammentazione dei pacchetti

Per quale ragione un amministratore di rete può decidere di ricorrere al subnetting nell'assegnazione degli indirizzi agli apparati della propria rete? Secondo quali criteri l'amministratore decide la struttura di tali indirizzi?

Spiegare in che cosa consiste il subnetting, fornendo anche un esempio, e motivarne l'utilità.

Il subnetting è una suddivisione interna di una rete di classe A o B in sottoreti. Lo schema di assegnazione di indirizzi class-based addressing è stato creato per fornire(/affittare) indirizzi IP in base al numero di host che possiede una LAN.

Questa suddivisione è utile perché le classi dello A e B sono molto grandi mentra la classe C è già troppo piccola per essere affittata da certi utenti.

Esempio: Si hanno 300 host:

- Se si sceglie una classe B si può fare broadcast su tutti mettendo tutti i bit a 1 ma si sprecano molti indirizzi (2^{16}-301).
- Se si sceglie una classe C è troppo piccola, ha spazio solo per 255 host.

L'università di Padova si fa assegnare le reti da 194.76.16._ a 194.76.23._; assumendo di utilizzare uno schema di indirizzamento CIDR, calcolare (i) la netmask, (ii) l'indirizzo base, (iii) quanti apparati possono essere indirizzati.

esercizio delle esercitazioni (14).

Ad una rete è stato allocato un blocco di indirizzi di rete da 8.73.112.0 a 8.73.115.255. Assumendo che sia usato CIDR, derivare l'indirizzo base e la netmask da usare, in dotted notation, e il massimo numero di indirizzi disponibili per gli host

8.73.112.0 è l'indirizzo base

numero di indirizzi = (115-112+1)•256=1024

netmask = 255.255.(256-4).0= 255.255.252.0

Ad una rete è stato allocato un blocco di 1024 indirizzi di rete da 200.30.0.0 a 200.30.3.255 . Assumendo che sia stato usato CIDR, derivare -mostrando tutti i calcoli annessi- la netmask da usare (sia in forma binaria sia in decimale puntata), e indicare il netID (o indirizzo base) di questa rete.

esercizio delle esercitazioni (13).

Illustrare la struttura della tabella di traduzione indirizzi di un router NAT.

l NAT si presenta all'esterno con IP address e un numero di porta. Sul gateway d'accesso, di confine, mantiene una tabella NAT in cui ricorda una quadrupla di corrispondenza [<IPA, port#A >, <IPX , port#AX>] :

- IP address dell'host interno,
- numero di porta del processo su quello host (interno che manda verso l'esterno),
- IP address pubblico del gateway NAT che mostra in Internet,
- numero di porta con cui si mostra all'esterno.

Si consideri un host con indirizzo IP 10.152.41.23 collegato in una rete con indirizzi privati, su cui è in esecuzione un browser al quale è richiesto di contattare il web server all'indirizzo 148.159.87.12. Illustrare come il server NAT modifica gli header HTTP request e response.

Modifica gli header del pacchetto uscente con suo il suo indirizzo IP pubblico e il suo numero di porta *(NAT sa lavorare a livello 3 (IP) e 4 (TCP))* e inoltra il pacchetto. Quando il pacchetto rientra da Internet, grazie alla relazione creata in precedenza, sostituisce l'indirizzo IP pubblico con quello dell'host e il numero di porta di NAT con quello originale dell'host e spedisce il pacchetto all'host di destinazione.

IP routing

Si consideri una rete in cui è adottato un algoritmo di instradamento di tipo Distance Vector. In quanto tempo un nodo di tale rete impara l'esistenza di una destinazione a k hop da sé? Motivare la risposta con un esempio.

In tempo O(k) perché...

Derivare, anche usando un esempio, l'ordine di grandezza del numero di scambi di messaggi necessari ad un nodo x che esegue un algoritmo di instradamento di tipo distance vector (es. RIP) per scoprire l'esistenza di un nodo y a distanza di n hop da sé.

Il costo è n.

Illustrare in cosa consiste il problema del conteggio all'infinito sofferto dall'approccio Distance Vector.

Se cade un nodo, non se ne accorge subito in quanto deve aspettare di venire informato dai vicini. Questa informazione si propaga per tutti i router ed alla fine tutti capiscono che il nodo non esiste più. Si spreca molto tempo.

Illustrare la tecnica di split-horizon per evitare la sindrome di conteggio all'infinito in algoritmi di instradamento che adottano l'approccio Distance-Vector.
in che cosa consiste la tecnica di split horizon per risolvere il problema di count to infinity cui sono soggetti i protocolli di instradamento basati su distance vector?
Il nodo invia al vicino X le informazioni di rotta, nella propria tabella routing table, relative a destinazioni per cui non è usato X come next hop. Questo per evitare di tornare indietro a quelle destinazioni nel caso si perdesse il cammino.

In una rete che adotta un algoritmo di instradamento Link-State, quali operazioni vengono eseguite dai nodi per scoprire l'intera tipologia di rete a cui applicare l'algoritmo di Dijkstra?

1. Ogni nodo scopre info sui propri vicini = si inizia con l'invio del pacchetto Hello su ogni linea p2p. Chi riceve risponde con il proprio IP unico. Se ci sono LAN a cui sono connessi più router vengono viste come un nodo fittizio. Ogni router poi invia un pacchetto di echo per scoprire i costi (rttt).
2. ogni nodo dice a tutti le informazioni sui propri vicini tramite flooding. Ogni nodo manda le seguenti informazioni: src, #seq, age, neigh_id, metrica-vicino.
3. ogni nodo impara l'intera topologia della rete.
4. ogni nodo calcola lo shortest path per ogni altro nodo usando **Dijkstra**.

Internet structure - protocolli compagni di IP

Descrivere il formato dei pacchetti ICMP per la segnalazione di errori, fornendo qualche esempio di utilizzo.
sono incerta perchè non capisco cosa intende per formato dei pacchetti...
La segnalazione degli errori (**error reporting**) è uno dei 6 tipi di messaggi del protocollo ICMP. Vi sono 3 sottocategorie di questa tipologia:

- **destination unreachable**= un pacchetto non può essere consegnato perché la destinazione non è raggiungibile. Questo può avvenire per diversi motivi: non c'è il cammino, il pacchetto è arrivato al destinatario ma non esiste il protocollo necessario per fare demultiplexing, la dimensione del pacchetto è troppo grande ma il bit DF(=don't fragment) non permette la frammentazione, oppure il cammino è chiuso al traffico.
- **time exceeded**= il router si trova un pacchetto non destinato a lui perché è esaurito il TTL.
- **parameter error**= qualche parametro dell'header non è valido (es.: checksum sbagliato).

header =

- Tipo (8 bit) tipo di messaggio
- Codice (8bit) specifica ragione del messaggio
- Checksum (16 bit)
- Dati (informazioni utili per identificare il datagramma originale che ha provocato errore)

descrivere in quale modo e a quale scopo e utilizzato il messaggio ICMP di tipo source quench.

source quench= un router non riesce a stare dietro ai pacchetti ricevuti. Per ridurre congestione, il router invia ai router vicini un pacchetto di source quench in cui avvisa che ha i buffer pieni. Fa backpressure. Quando si è decongestionato rimanda un source quench per notificarlo.

routing Internet

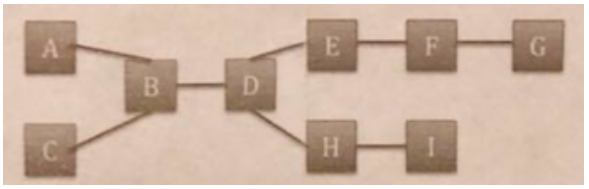
Elencare i tipi di messaggi OSPF specificando per ognuno quale funzione svolge.
Spiegare la funzione di ognuno dei 5 tipi di messaggi utilizzati in OSPF.
Descrivere i tipi di messaggi previsti dal protocollo OSPF.

I tipi di messaggi OSPF sono:

1. **hello** = per scoprire i vicini
2. **link state update** = per ottenere informazioni sul vicinato
3. **link state req** = serve ad un router per aggiornarsi
4. **database description** = contiene l'ultimo #seq posseduto per aggiornare altri router
5. **link state ack** = ack usati per fare flooding

domande miste

Se al tempo t0 i router nella rete in figura vengono tutti contemporaneamente accessi, dire quali informazioni si trovano nella tabella di instradamento del nodo B dopo il 1° e il 2° scambio di vettori distanza.



Spiegare come funziona l'inoltro nel protocollo IPv4

parti di risposta in cui non sappiamo la soluzione
parti di risposta di cui non siamo certi della correttezza