Raccolta domande Reti di calcolatori
uniVR - Dipartimento di Informatica

Amos Lo Verde
Massimiliano Renso
Gianmaria Forte
Alessio Zattoni

9 gennaio 2022

Domande vecchi appelli

Livello di trasporto

- **Spiegare il concetto di porta a livello di trasporto e le due tipologie esistenti**.
  La porta a livello di trasporto identifica i processi sorgente e destinazione coinvolti nella comunicazione.
  Sia la porta sorgente e destinazione sono definite da 16 *bit* e si dividono in:
  - Porte statiche (da 0 a 1023): sono definite dallo *standard* e assegnate ai processi noti come per esempio HTTP (porta 80), FTP (porta 21), ecc.
  - Porte dinamiche (da 1024 a 65535): sono assegnate dinamicamente e casualmente dal sistema operativo ai processi *client* che si collegano ai *server*.

  Il flusso dati passa attraverso il *socket* definito dalle due coppie IP sorgente/porta sorgente e IP destinazione/porta destinazione, che definiscono in modo univoco la connessione tra *client* e *server*.

- **In riferimento al livello di trasporto, spiegare come viene stimato RTT**.
  L'RTT (*Round Trip Time*) è l'intervallo di tempo che intercorre tra l'invio del segmento e la ricezione dell'ACK (*acknowledgment*).
  La stima dell'RTT è detta SRTT (*Smoothed Round Trip Time*) ed è la media pesata degli RTT istantanei calcolata come:

  $$\text{SRTT}_{\text{attuale}} = \alpha \cdot \text{SRTT}_{\text{precedente}} + (1 - \alpha) \cdot \text{RTT}_{\text{istantaneo}}$$

  con $\alpha = \frac{7}{8} = 0.875$.

- **Descrivere come TCP calcola RTT e RTO**.
  L'RTT (*Round Trip Time*) è l'intervallo di tempo che intercorre tra l'invio del segmento e la ricezione dell'ACK (*acknowledgment*).
  L'RTO (*Retransmission Time Out*) è il tempo di attesa prima di considerare persi i segmenti inviati. Viene calcolato come il doppio della stima di RTT:

  $$\text{RTO} = \beta \cdot \text{SRTT}_{\text{attuale}}$$

  con $\beta = 2$.

- **L'*header* del protocollo UDP contiene solo 4 campi: *Source port*, *Destination port*, *Length* e *Checksum*. Spigare brevemente il loro utilizzo**.
  - *Source/destination port*: contengono le porte sorgente e destinazione del flusso di dati dei processi coinvolti.
  - *Length*: specifica il numero di byte del segmento UDP.
  - *Checksum*: è un codice utilizzato per verificare se sono avvenuti errori nella trasmissione del segmento. Tale verifica è fatta dal destinatario.

- **Descrivere la procedura di TCP per instaurare una connessione**.
  L'instaurazione della connessione TCP è detta *three-way handshake* e avviene in tre fasi:
  1. *Client* TCP invia un segmento al server TCP con flag `SYN = 1` e *payload* vuoto. Inoltre il client sceglie un numero casuale `client_isn` e lo imposta nel campo *sequence number* (SN).
  2. Il *server TCP* alloca i *buffer* e le variabili TCP alla connessione e invia un segmento di connessione approvata al *client* TCP, con flag: `SYN = 1`, `AckN = client_isn + 1`, `ACK = 1` e `SN = server_isn`.
  3. Il *client* alloca i *buffer* e le variabili alla connessione. Successivamente invia un segmento di risposta al *server* con `SN = client_isn + 1`, `AckN = server_isn + 1`, `ACK = 1` e `SYN = 0`.

- **Descrivere la procedura di TCP per chiudere una connessione**.
  Ciascuno dei due processi, ossia quello in lato *client* e quello in lato *server*, possono terminare indipendentemente la connessione TCP quando non hanno più dati da trasmettere.
  I casi di chiusura sono due:
  - Chiusura unilaterale (visto da *client*):
    1. Il *client* invia un segmento al server con *flag* `FIN = 1`.
    2. Il *server* risponde inviando un segmento con *flag* `ACK = 1` e si chiude la connessione lato *client*.
  - Chiusura simultanea (visto da *client*):
    1. Il *client* invia un segmento al *server* con *flag* `FIN = 1`.
    2. Il *server* risponde inviando un segmento con *flag* `ACK = 1` e `FIN = 1`.
    3. Il *client* risponde invia un segmento con *flag* `ACK = 1`. La connessione si chiude sia lato *client* sia *server*.

  Analogamente quando il *server* vuole chiudere la connessione invia gli stessi messaggi, ma in direzione inversa. Alla chiusura della connessione ognuno dealloca le risorse utilizzate.

- **Spiegare cosa succede se MSS è troppo piccolo o troppo grande**.
  L'MSS è la dimensione massima del campo dati dei segmenti che si possono ricevere (MTU - *header*(TCP+IP)).

  - MSS troppo piccolo: avviene uno spreco del mezzo fisico di trasmissione.
  - MSS troppo grande: si possono verificare un aumento delle congestioni e collisioni.

- **Spiegare perché la *congestion window* varia nel tempo**.
  Perché si adatta al variare delle condizioni della rete. Vengono applicati vari algoritmi per l'ottimizzazione del TCP, come *congestion avoidance* e *slow start*, che modificano il valore della *congestion window*.

Livello di rete

- **Definizione di indirizzo privato e indirizzo pubblico, spiegando come vengono utilizzate**.
  - Indirizzi pubblici: sono gli indirizzi utilizzati per accedere alla rete pubblica e definiscono in modo univoco un nodo su Internet.
  - Indirizzi privati: sono utilizzati nelle reti private o LAN per il traffico locale. Inoltre non sono utilizzabili sulla rete pubblica.

- **L'*header* IP contiene un campo di 16 bit denominato "*Identification*": spiegare cosa contiene tale campo e come viene utilizzato**.
  Il campo *identification* ha lo scopo di identificare ogni pacchetto tramite un numero progressivo in maniera univoca.
  Inoltre durante la frammentazione viene utilizzato per riconoscere a quale pacchetto originario appartengono i frammenti.

- **Elencare e spiegare i vari indirizzi IP speciali**.
  - Indirizzo di rete: è formato con tutti i *bit* di suffisso posti a 0.
    Esempio: `xxxxxxxx.xxxxxxxx.xxxx0000.00000000/20`.
  - *Direct broadcast*: è formato con tutti i *bit* di suffisso posti a 1.
    Esempio: `xxxxxxxx.xxxxxxxx.xxxx1111.11111111/20`.
  - *Limited broadcast*: è formato con tutti i bit posti a 1 (`255.255.255.255`). Viene utilizzato in fase di *start-up* quando l'*host* non conosce l'indirizzo di rete.
  - *This host*: durante la fase di *start-up* è l'indirizzo iniziale di default di un *host* prima che DHCP gli assegni un *host* (`0.0.0.0`).
  - Indirizzi di *loopback*: sono gli indirizzi utilizzati dagli sviluppatori per le applicazioni di rete e sono `127.0.0.0` e `127.0.0.1`.

- **Indicare i blocchi di indirizzi IP privati**.
  I blocchi sono:
  - `10.0.0.0/8`
  - `172.16.0.0/12`
  - `192.168.0.0/16`
  - `169.254.0.0/16`

  Questi indirizzi sono riservati per le reti private.

- **Spiegare l'uso dell'*extension header* e il motivo della sua introduzione**.
  Si rende "flessibile" il protocollo grazie al campo *next header*. Questo campo possiede un codice avente un doppio significato:
  - Se non ci sono estensioni, ovvero non si presenta un pezzo addizionale dell'intestazione, allora quel campo identifica il protocollo trasportato dal *payload*.
  - Se c'è un'intestazione aggiuntiva, definita *extension header*, allora quel codice rappresenta il tipo.

  Nell'*extension header* sono presenti due campi indispensabili:
  - *Next header*: è il codice, se esiste, del *next header* aggiuntivo, altrimenti indica il codice del protocollo trasportato.
  - *Header lenght*: permette di conoscere dove finisce l'estensione dell'intestazione.

  Mentre il resto del campo è libero: a seconda del *next header* si hanno formati diversi.

- **L'*header* IP contiene un campo chiamato TTL: spiegare come viene utilizzato tale campo e il perché è stato introdotto**.
  Il campo TTL (*Time To Live*) indica il numero di *hop* entro i quali deve raggiungere la destinazione: ogni volta che ne attraversa uno il campo viene decrementato. Se il valore di TTL raggiunge lo 0, allora automaticamente il *router* lo scarta e restituisce un errore tramite ICMP.
  Questo campo è stato aggiunto per evitare i *routing loop*, ossia una permanenza infinita nella rete del pacchetto.

- **Spiegare il significato e l'utilizzo dei campi *Sequence Number* (SN) e *Acknowledge Number* (AckN)**.
  - Campo *sequence number* (SN): corrisponde al primo *byte* del segmento spedito sommato all'*initial sequence number* (isn).
  - Campo *acknowledgment number* (AckN): corrisponde al primo *byte* del segmento successivo che si aspetta di ricevere.

Il *sequence number* è utilizzato per ordinare i segmenti ricevuti, perché questi ultimi possono seguire strade differenti per raggiungere la destinazione e arrivare in momenti differenti, mentre l'*acknowledgment number* è utilizzato per riscontrare il ricevimento dei pacchetti inviati.

- **Spiegare brevemente la funzionalità di frammentazione dei pacchetti IP, incluso le motivazioni e gli apparati che effettuano frammentazione/deframmentazione**.
  La frammentazione avviene quando si attraversano degli apparati con MTU minori rispetto quello iniziale; questa operazione viene effettuato dai *router*, mentre il riassemblaggio spetta alla destinazione del pacchetto.

- **Come si utilizza il "*fragment offset*" dell'*header* IP**.
  Il campo *fragment offset* corrisponde alla posizione (in *byte*) del frammento rispetto al pacchetto originale. Questo valore è diviso per 8.
  Supponendo di avere un pacchetto da 1000 *byte* di *payload* da dividere in due da 500 *byte* (esclusi gli *header*), il primo frammento avrà il campo *fragment offset* uguale a 0 mentre nel secondo frammento sarà 500 diviso 8. Ciò serve a conoscere la corretta posizione dei frammenti al momento della ricostruzione.

- **Cos'è e perché viene introdotto il NAT**.
  Il NAT (*Network Address Translation*) è una funzione che permette a un *host* con indirizzo IP privato all'interno di una LAN di comunicare con un *host* con indirizzo IP pubblico. Questo avviene tramite la corrispondenza tra indirizzo IP privato e porta osservata nella tabella NAT all'interno del *router*, il quale si occupa di manipolare i campi IP/porta sorgente all'andata e IP/porta destinazione alla risposta.
  È stato introdotto per sopperire alla scarsità degli indirizzi IP pubblici.

- **Descrivere il funzionamento di DHCP**.
  Il DHCP (*Dynamic Host Configuration Protocol*) è un protocollo che permette di assegnare un indirizzo IP a un *host* che si connette a una rete. Il suo funzionamento consiste in:
  - L'*host* invia un messaggio di DHCP *discover* con IP sorgente `0.0.0.0` e destinazione in broadcast `255.255.255.255`, con un ID di transazione impostato casualmente per identificare univocamente l'individuazione.
  - DHCP *server* risponde con un messaggio di DHCP *offer* contenente: IP sorgente del server DHCP, lo stesso ID di transazione precedente, IP proposto al *client*, maschera di sottorete e la durata della concessione.
  - Il client sceglie tra le offerte e risponde con un messaggio di DHCP *request* mandato in broadcast.
  - Il server DHCP risponde con DHCP ACK che conferma i parametri richiesti.

Livello di collegamento

- **Descrivere la modalità di *framing* del *data link* chiamata "*bit stuffing*".**
  La modalità di *bit stuffing* avviene dopo l'inserimento del *byte flag* a inizio e fine della trama, al fine di non confondere in ricezione una sequenza uguale al *byte* di *flag* interna alla trama. Per esempio il *byte flag* è `01111110` e viene posto a inizio e fine della trama; il *bit stuffing* consiste di aggiungere il *bit* 0 alla fine di ogni sequenza che corrisponde a `011111`, indipendentemente dal *bit* che segue, all'interno della trama. In questo modo in ricezione ogni volta che si trova la sequenza `0111110` si rimuove il *bit* 0, così da rimuovere l'alterazione fatta.

- **Cosa si intende per *framing* nel livello 2? Descrivere una delle possibili tecniche con un semplice esempio**.
  Per *framing* si intendono le operazioni che servono a distinguere una trama dall'altra in ricezione.
  Esistono 3 tecniche per delimitare le trame:

  1. Intervalli di tempo: si attende un intervallo di tempo tra la trasmissione di una trama all'altra.
  2. *Character count*: si indica nell'*header* la dimensione della trama in *byte*.
  3. *Byte flag* + *bit stuffing*: si utilizza un *byte* di *flag* per delimitare l'inizio e la fine della trama e si inseriscono *bit stuffing* interne alla trama per evitare ripetizioni del *byte flag*.

  Per esempio si prende il *byte flag* `01111110` e si pone a inizio e fine della trama. Successivamente il *bit stuffing* consiste nell'aggiungere il *bit* 0 alla fine di ogni sequenza che corrisponde a `011111`, indipendentemente dal *bit* che segue, all'interno della trama.
  In questo modo in ricezione ogni volta che si trova la sequenza 0111110 si rimuove il *bit* 0, così da rimuovere l'alterazione fatta, e con il *byte flag* si conosce il corretto *payload* su cui lavorare.

- **Definizione di dominio di collisione e dominio di broadcast**.
  - Dominio di collisione: è la porzione di rete in cui se due stazioni trasmettono contemporaneamente avviene una collisione. Questo dominio è delimitato dallo *switch*.
  - Dominio di *broadcast*: è la porzione di rete raggiunta da un messaggio inviato in *broadcast* a livello 2. I *router* rappresentano sempre un confine del dominio di *broadcast*.

- **Cos'è e come viene utilizzato il NAV**.
  Il NAV (*Network Allocation Vector*) è un meccanismo che consente di risparmiare energia degli apparati connessi in *wireless*: è un vettore che contiene gli intervalli di trasmissione dei canali, nei quali gli altri che non trasmettono disabilitano il circuito d'ascolto del canale.

- **Descrivere attraverso pseudo codice gli algoritmi CSMA puro, no-persistent, p-persistent**.
  - CSMA puro (persistent):
    1. Se ha una trama da trasmettere:
       → Ascolta il canale.
    2. Se il canale è libero:
       → Trasmette la trama.
       Altrimenti
       → Aspetta che il canale si liberi e poi trasmette.
    3. Se c'è collisione:
       → Attende un tempo casuale e torna al punto 1.

  - CSMA no-persistent:
    1. Se ha una trama da trasmettere:
       → Ascolta il canale.
    2. Se il canale è libero:
       → Trasmette la trama.
       Altrimenti
       → Estrae un tempo casuale e torna al punto 1.

  - CSMA p-persistent:
    1. Se ha una trama da trasmettere:
       → Ascolta il canale.
    2. Se il canale è libero:
       → Trasmette la trama.
       Altrimenti
       → Aspetta che il canale si liberi.
       → Con probabilità p trasmette subito.
       → Con probabilità $(1 - p)$ rimango il tentativo di uno *slot* (tempo di *slot* << tempo di trama).

- **Descrivere, anche attraverso l'uso di pseudo codice, l'algoritmo CSMA nella vriante CD, indicando il motivo che ha portato tale introduzione**.
  Algoritmo di CSMA/CD (persistent):
  1. Se ha una trama da trasmettere:
     → Ascolta il canale.
  2. Se il canale è libero:
     → Trasmette la trama.
     Altrimenti
     → Aspetta che il canale si liberi e poi trasmette.
  3. Se c'è collisione:
     → Interrompi la trasmissione.
     → Attendi tempo casuale e torna al punto 1.

Questa variante è stata aggiunta per ottimizzare l'uso del mezzo fisico e interrompere la trasmissione di trame corrotte.

- **Definire il periodo di vulnerabilità e quale è quello di ALOHA, Slotted ALOHA e CSMA persistent**.
  Il periodo di vulnerabilità è definito come l'intervallo di tempo in cui può avvenire una collisione.
  - ALOHA: il suo periodo di vulnerabilità equivale al doppio del tempo di trama.
  - Slotted ALOHA: il suo periodo di vulnerabilità è solo il tempo di trama.
  - CSMA persistent: il suo periodo di vulnerabilità è il doppio del ritardo di propagazione.

- **Protocollo ARP: spiegare il funzionamento e perché è stato introdotto**.
  Il protocollo ARP (*Address Resolution Protocol*) in un *host* permette, dato l'indirizzo IP di destinazione, di ottenere il relativo indirizzo MAC presente nella sua tabella ARP.
  Se una corrispondenza non è presente in tabella, allora il nodo richiedente effettua una richiesta ARP in *broadcast* a livello 2. In questo modo la macchina con l'IP precedentemente indicato risponderà direttamente fornendo il suo indirizzo MAC.
  È stato introdotto perché una macchina per comunicare direttamente con altre macchine deve specificare l'indirizzo MAC di destinazione.

- **Descrivere il problema del terminale nascosto e le sua risoluzioni**.
  In una connessione *wireless* il problema del terminale nascosto si verifica quando l'onda di propagazione di un *host* non riesce a raggiungere gli altri *host*. Per questi ultimi è come se nessuno stesse trasmettendo, di conseguenza potrebbero iniziare la loro trasmissione e causare collisioni.
  Le soluzioni a questo problema sono due:

  1. Soluzione *hardware*: vengono installati più *access point* in modo che ogni *host* sia coperto da almeno due di questi, così che mandino la propria trasmissione a quello più vicino.
  2. Soluzione protocollare: consiste nell'aggiungere i messaggi di CTS e RTS:
     - Messaggio RTS: è un piccolo messaggio che contiene l'identificativo della stazione e la dimensione della trama che vuole trasmettere.
     - Messaggio CTS: è un messaggio inviato in broadcast dove avvisa tutti i nodi che è stato assegnato il diritto a una certa stazione di poter trasmettere. In questo modo si inibiscono le trasmissioni delle altre stazioni per un tempo pari a quello indicato nel messaggio di CTS.

     Tuttavia se due stazioni dovessero mandare contemporaneamente l'RTS entrerebbero in collisione, pertanto non ricevendo nessuna delle due il CTS attendono un tempo casuale, dopo il quale ritentano l'invio di RTS.

Possibili ulteriori domande

Domande concettuali

- **Definire il concetto di protocollo**.
  Un protocollo definisce il formato e l'ordine dei messaggi scambiati tra due o più entità di comunicazione, così come le azioni intraprese in fase di trasmissione e/o ricezione di un messaggio o di un altro evento.

- **Descrivere il core della rete**.
  Il core della rete è la porzione di rete in cui si presenta il percorso di comunicazione e all'interno del core si possono distinguere diverse tipologie di ISP (Internet Service Provider):
  - ISP livello 1: la rete spazia geograficamente su più nazioni (è internazionale).
  - ISP livello 2: viene coperto solo il territorio nazionale.
  - ISP livello 3: la copertura riguarda territori locali.

- **Descrivere le tipologie di commutazione esistenti: circuito e pacchetto**.
  - Commutazione a circuito: consiste nell'installazione di un unico canale, la cui capacità viene interamente dedicata alla comunicazione.
  - Commutazione a pacchetto: l'informazione viene suddivisa in unità indipendenti chiamati pacchetti. A ogni pacchetto viene inclusa un'ulteriore informazione definita intestazione, che permette di arrivare a destinazione in maniera autonoma.

- **Spiegare cosa sono e le differenze dei modelli ISO/OSI e TCP/IP**.
  Entrambi i modelli sono *stack* protocollari, ovvero definiscono i livelli e protocolli per ciascuno di esso. Il modello ISO/OSI è teorico, mentre TCP/IP è quello attualmente utilizzato.
  - Modello ISO/OSI: comprende sette livelli: fisico, collegamento, rete, trasporto, sessione, presentazione e applicazione.

  - Modello TCP/IP: comprende cinque livelli: fisico, collegamento, rete, trasporto e applicazione. A volte potrebbe essere che il livello fisico e collegamento siano unificati in un unico livello.

  Invece i *router* gestiscono sempre e solo i primi tre livelli, cioè: fisico, collegamento e rete.

- **Definire cosa si intende per *subnetting* e blocco CIDR**.
  Il *subnetting* è il processo di suddivisione della rete in sotto reti, mentre il blocco CIDR corrisponde al blocco di indirizzi della rete.

Livello applicativo

- **Descrivere il formato di protocollo HTTP**.
  È un protocollo testuale implementato sia nei *client* sia nei *server*. HTTP si appoggia su TCP e utilizza due tipologie di messaggi:

  - Messaggio di richiesta:
    → Riga di richiesta: metodo, URL, versione.
    → Righe di intestazione: *host* specificato, *connection* (se persistente o no), tipo di *browser* dichiarato dall'*user agent*, *accept-language* specifica la lingua preferita.
    → Corpo dell'entità: si riempie solo con i campi del metodo POST.
  - Messaggio di risposta:
    → Riga di stato: versione, codice di stato e messaggio e di stato.
    → Righe di intestazione: *connection* che comunica al *client* se chiudere o no la connessione TCP, Date che indica la data e l'ora di creazione e invio della risposta, *Server-modified* dichiara l'istante e la data in cui l'oggetto è stato modificato l'ultima volta, *content-length* comunica il numero di *byte* dell'oggetto inviato, *content-type* che indica il tipo dell'oggetto nel corpo.
    → Corpo: contiene l'oggetto richiesto.

- **Descrivere le connessioni persistenti e non persistenti**.

  - Connessioni non persistenti: ogni connessione TCP viene chiusa dopo l'invio dell'oggetto da parte del *text*.
  - Connessioni persistenti: la connessione TCP non viene chiusa dopo l'invio dell'oggetto da parte del *server*.

- **Descrivere il formato e funzionalità dei *cookie*.**
  Sono un meccanismo che consente ai *server* di tener traccia degli utenti. Sono composti da:
  - Riga di intestazione nel messaggio di risposta HTTP.
  - Riga di intestazione nel messaggio di richiesta HTTP.
  - Un *file* mantenuto sul sistema dell'utente e gestito dal *browser*.
  - *Database* sul sito.

- **Descrivere le funzioni del *web caching*.**
  Il *web caching* è un'entità di rete che soddisfa le richieste HTTP al posto del *web server*. In memoria conserva le copie di oggetti recentemente richiesti. Se l'oggetto è vecchio viene controllato dal meccanismo GET condizionale: un messaggio di richiesta HTTP che usa il metodo GET e include la riga di intestazione *If-modified-since*.

- **Descrivere il funzionamento del DNS.**
  Il DNS si occupa di tradurre un indirizzo logico `www.nomesito.tld` nel corrispondente indirizzo IP.
  Prima che *client* e *server* si connettano, viene inviata una richiesta DNS al server DNS che risponde con l'indirizzo IP di destinazione. Esistono tre classi di DNS *server*:
  - *Root-server*: forniscono gli indirizzi IP dei server TLD.
  - *DNS-server*: si occupano dei domini di primo livello e forniscono gli indirizzi IP dei *server* locali.
  - *DNS server* locali: possiedono gli indirizzi di *host/server* pubblici.

- **Descrivere i vari protocolli di posta e il loro funzionamento.**
  - SMTP: trasferisce i messaggi dal *mail server* del mittente a quello del destinatario, facendo uso del servizio TCP; il corpo e intestazione è a 7 *bit*; non vengono *usati mail server* intermedi; è un protocollo *push*: "spinge" i *file* al *mail server* in ricezione.
  - HTTP: trasferisce file da un *web server* a un *web client*; è un protocollo *pull*: i *file* sono caricati e qualcuno li richiede.
  - POP3: protocollo semplice e poco sicuro. Si appoggia su TCP e memorizza in locale i *file*.
  - IMAP: consente una gestione più ampia delle *mail*.

Livello di trasporto

- **Indicare cosa sono e come si calcolano MTU e MSS**.
  - MTU: è la dimensione massima in *byte* di trasmissione, calcolata come somma di *payload* + *header* IP + *header* TCP o UDP.
  - MSS: è la dimensione massima del segmento che limita la quantità di dati prelevabili e posizionabili. Viene calcolato come la differenza tra MTU - (*header* IP + *header* TCP o UDP).

- **Spiegare le differente tra il protocollo UDP e TCP**. Le differenze tra il protocollo UDP e TCP sono:
  - TCP è orientato alla connessione in quanto per stabilire la connessione effettua l'handshake a tre vie, mentre UDP no: invia direttamente i pacchetti.
  - TCP è affidabile in quanto prevede riscontro per i pacchetti, ossia verifica che ognuno arrivi a destinazione altrimenti si occupa di rimandarli.
  - UDP non conserva lo stato della connessione e non tiene traccia dei parametri di numero di sequenza e di *acknowledgment*.
  - UDP utilizza meno risorse poiché pesa 8 *byte*, mentre TCP occupa 20 *byte*.

Livello di rete

- **Descrivere il ruolo e l'uso di ICMP**.
  È un protocollo per scambiarsi informazioni a livello di livello di rete, specialmente per la notifica degli errori.

- **Descrivere il ruolo e l'uso di BGP**.
  È un protocollo di tipo *distance-vector* decentralizzato e asincrono. È l'attuale *standard* dei protocolli di instradamento tra sistemi autonomi in *Internet*.
  BGP mette a disposizione di ciascun *router* un modo per: - Consentire a ogni sottorete di comunicare la propria esistenza al resto di *Internet*. - Determinare il miglior percorso eseguendo BGP localmente.

Livello di collegamento

- **Indicare i metodi d'uso efficienti del mezzo condiviso**.
  Questi metodi servono per minimizzare le collisioni. Si hanno vari protocolli per l'allocazione del canale di trasmissione:

- Statico: FDM (basato sulla frequenza), TDM (basato sul tempo). Per le trasmissioni discontinue non risultano metodi efficienti.
- Dinamico:
  1. A turno, cioè *token ring*. La stazione che trasmette possiede il *token* per un certo tempo, se non ha nulla da trasmettere allora passa il *token* alla stazione successiva.
  2. A contesa, cioè gli algoritmi: ALOHA, slotted ALOHA e CSMA. Questi risultano la soluzione migliore.

- **Descrivere in pseudo codice i passaggi di ALOHA**.

  1. Se ci sono dati da trasmettere:
     → Vengono trasmessi.
  2. Mentre trasmette, ascolta il canale:
     2.1. Se c'è collisione: potenza rilevata > potenza trasmessa.
     → Si estrae un tempo casuale.
     → Si torna al punto 1.

- **Descrivere Slotted ALOHA**.
  Il tempo è suddiviso in intervalli (*slot*) di durata T. Le stazioni seguono il protocollo ALOHA, ma possono trasmettere solo all'inizio di un *slot*.

- **Descrivere cosa sono e il formato degli indirizzi MAC**.
  Gli indirizzi MAC sono indirizzi a livello di collegamento lunghi 6 *byte* (2^{48} combinazioni). Presentano una struttura piatta senza gerarchie e non cambiano mai, ovvero ogni scheda di rete possiede sempre lo stesso indirizzo ovunque si colleghi.

- **Descrivere cosa siano le reti WLAN**.
  Si dispone di un *access point* il quale risulta l'interfaccia tra il mondo *wireless* e quello cablato. La parte gestita dall'*access point* è chiamata BSS (*Basic Service Set*).

- **Descrivere in pseudo codice i passaggi di CSMA/CA**.

  - Per prima cosa ascolta il canale:
    1. Se il canale è libero, allora continua ad ascoltare il canale per un intervallo pari a DIFS. Se dopo questo tempo il canale è ancora libero, allora viene trasmessa la trama.
    2. Se il canale è occupato fin da subito o durante il DIFS, allora continua ad ascoltare il canale fino a quando si libera.

3. Quando il canale è libero, lo si ascolta per un intervallo pari a DIFS. Se il canale torna occupato si torna al punto 2.
4. Se il canale rimane libero per un DIFS, allora la stazione estrae un numero casuale uniformemente distribuito tra (0, cw - 1) (*contention window*). Questo numero casuale viene indicato con S = # *slot* che si deve attendere prima di poter trasmettere.

– Fintanto che il canale rimane libero, la stazione decrementa S: se arriva a 0 allora la stazione trasmette la trama. Al contrario se il canale torna occupato si salva il valore di S e si torna al punto 2, ma una volta arrivati al punto 4 si utilizza il valore precedentemente salvato di S.

– Se dopotutto si dovesse presentare una collisione, allora si interromperebbe la trasmissione, si estrarrebbe un tempo casuale e tornerebbe al punto 1 (raddoppiando cw).