Prova Scritta del corso di Reti di Calcolatori (Computer Networks)

21 Luglio 2023

Docente: Luciano Bononi

Correzione: AZ

Rispondere alle domande scrivendo solo nello spazio consentito, oppure nel retro del foglio. Fornire sempre una breve motivazione o il procedimento di calcolo della risposta, ove previsto.

1[5]) Cosa significa che il livello trasporto realizza un servizio End-to-End? E cosa si intende con multiplexing e demultiplexing dei dati tra mittente e destinatario?

Il livello trasporto realizza un servizio End-to-End nel senso che il livello trasporto viene implementato soltanto sul mittente originale e sul destinatario finale della comunicazione di rete all'interno dell'architettura di Internet. Tutti i nodi intermediari non aprono mai la busta di livello trasporto, il livello trasporto viene gestito solo dal mittente iniziale e dal destinatario finale su Internet. Cosa si intende per mux e demux dei dati tra mittente e destinatario? Come sappiamo i socket possono essere aperti tra mittente e destinatario in numero anche >1, quindi potremmo avere più processi attivi sul mittente che comunicano con uno o più processi attivi sul destinatario generando un'apertura di un socket tra la porta A del mittente e la porta B del destinatario, oltre alla porta A' del mittente e la porta B' del destinatario, ecc. ecc. Avere un numero >1 di processi lato mittente che comunicano con un numero >1 di processi lato destinatario significa che però tutti questi dati comunicati tra questi processi devono essere scambiati usando l'unica architettura di rete/connessione di rete presente sia su mittente che su destinatario, supponiamo (per estremizzare il caso) che abbiamo solo una connessione ethernet sia su mittente che su destinatario, è chiaro che da quella connessione di livello fisico, di livello mac ethernet possono uscire i dati appartenenti ai flussi di tutte queste coppie mittente-destinatario; quindi il multiplexing è quella azione fatta a livello trasporto che prende i dati provenienti da tutti i socket diversi, da porte applicazione diverse su uno stesso indirizzo IP mittente-destinatario e li convoglia all'interno della stessa unica architettura di rete che c'è dal livello rete in giù, come sappiamo dal livello rete in giù abbiamo solo un indirizzo IP e solo un indirizzo mac rispettivamente per il livello rete e l'architettura fisica della scheda di rete. Questi dati partono avendo quegli indirizzi mac e rete ma vengono discriminati dal numero di socket, dal numero di porte in questo caso e quindi il multiplexing è l'operazione di includere tutti questi dati, questi flussi all'interno dell'unico stack di rete che esiste sottostante. Questa operazione però ha un contraltare perché lato ricevente, sul destinatario, emergono dal basso fino a livello rete tutti i pacchetti di questi, questi segmenti (essendo a livello trasporto) che riportano tutti lo stesso indirizzo IP, quello del destinatario ma avranno numeri di porta differenti, quindi a questo punto il Demultiplexing è l'operazione per cui i dati che emergono dal livello rete vengono ri-aperti a ventaglio per andare a inserirsi nei buffer dei socket identificati attraverso il numero di porta, quindi il multiplexing e demultiplexing altri non è quell'operazione di condivisione dello stack sottostante di livello trasporto da parte di flussi di dati originati e generati da numeri di porta differenti.

2[5]) Dove sono implementati, perché sono implementati, da chi sono implementati, e quali aspetti rendono diversi il controllo di flusso e il controllo di congestione?

Siamo sempre a livello trasporto perché il controllo di flusso e il controllo di congestione sono implementati a livello 4, quindi sul *dove* sono implementati: entrambi a livello 4. Da chi sono implementati: sono implementati da TCP, solo da TCP e non da UDP. Perché sono implementati, qual è la loro funzione e quali aspetti diversi li rendono differenti. La funzione di controllo di flusso e controllo di congestione è sostanzialmente la stessa nel senso che entrambi mirano ad aumentare fino al massimo ritmo sostenibile l'invio dei pacchetti dati dal mittente al destinatario, quindi lo scopo di entrambi è accellerare la trasmissione, solo che il limite che devono controllare e gestire rispettivamente il controllo di flusso e il controllo di congestione è diverso, perché il controllo di flusso ha come tetto limite, alla velocità di invio, la massima capacità di ricezione del destinatario, in qualche misura deve essere limitato dal buffer del destinatario; invece il controllo di congestione ha come limite la capacità di inoltro del router più lento presente all'interno del cammino di rete mittente-destinatario su Internet. Quindi è come dire che entrambi debbono accelerare spingendo sul pedale dell'acceleratore ma uno comincia a frenare quando si rende conto che sta saturando il buffer del destinatario, l'altro quando si rende conto che c'è un router che entra in crisi a livello di congestione e quindi comincia a perdere i pacchetti. Il tutto si basa sulla finestra di congestione, ovvero un numero intero che esprime il numero massimo di segmenti di dati che possono essere inviati a livello trasporto in attesa di ricevere l'ACK, come dire un contatore dei dati di cui non sappiamo ancora con certezza l'esito, quelli in sospeso. Il controllo di flusso si realizza mettendo un tetto massimo alla dimensione della finestra di congestione pari alla dimensione del buffer residuo sul ricevente. La finestra di congestione viene gestita da un meccanismo di dimensionamento semi-automatico che prevede due fasi: slow start e congestion avoidance per la quale si parte da una finestra di congestione di livello 1, quindi minima, dimensione 1; se avviene la trasmissione con successo in un RTT di un pacchetto, quindi andata e ritorno si riceve l'ACK senza problemi si aumenta raddoppiando questa dimensione a 2,4,8,16,… fino al raggiungimento di una soglia oltre la quale si va un po' più cauti perché ci si aspetta che da lì in poi possa avvenire la congestione. Controllo di flusso è un tetto massimo alla dimensione, il controllo di congestione è un tetto dinamico della finestra di congestione.

3[5]) I protocolli ICMP e SNMP sono analoghi? spiegare

La risposta può essere sia sì che no, perché certamente hanno analogie, entrambi consentono di gestire e comunicare informazioni relative allo stato di funzionamento della rete. Per ottenere il massimo bisognava sottolineare alcune differenze sostanziali tra questi due protocolli ICMP è un protocollo un po' più banale perché può essere implementato direttamente da uno o due client che comunicano attraverso lo scambio di messaggi definiti da questo protocollo, che consentono di comunicare informazioni su stato di funzionamento della rete, stato di funzionamento dell'host e stato di funzionamento del router. Quindi comunicazione si/no ed eventualmente qual è il problema di comunicazione; host sconosciuto? host non raggiungibile? rete che non sappiamo dov'è… monitoraggio preconfigurato. SNMP è un protocollo di livello più alto che può realizzare un insieme di funzioni più articolate e complesse, è più potente di ICMP che consente di automatizzare e fare un sacco di funzioni più importanti, per esempio il monitoraggio dei servizi, usando ICMP per capire se un servizio è attivo o no è abbastanza limitante perché ti consente solo di arrivare a capire se quell'host è connesso ad internet e ha problemi di connessione, non riusciamo a sapere lo stato di funzionamento del servizio o cose del genere, invece con SNMP, grazie all'implementazione di servizi di monitoraggio ad agenti e grazie alla collaborazione di questi agenti è possibile riuscire a monitorare e a ricevere degli allert qualora un certo servizio venga attaccato, vada in crisi, non sappia più rispondere, …

4[15] aiutate Alice e Bob a definire **un protocollo valido** per il seguente obbiettivo: Alice vuole spedire un numero N di messaggi segreti $m_1…m_N$ (brevi) a Bob, ma né Alice né Bob sanno a priori il numero N (nel senso che prima o poi Alice decide quale sia l'ultimo messaggio m_N della sequenza e lo invia). Nell'inviare l'ultimo messaggio m_N Alice vuole anche specificare a Bob in modo affidabile che quello sia l'ultimo messaggio. Inoltre Alice e Bob vogliono essere sicuri che i messaggi siano tutti ricevuti e possano essere ordinati correttamente da Bob nell'ordine in cui sono stati numerati e spediti ad Alice. Trudy può fare qualunque cosa, ma non cancellare i pacchetti (però li può ritardare quanto vuole purché un tempo finito).
Come si fa a essere sicuri che Bob ottenga in **modo privato** (nessuno a parte Alice conosce e sa il contenuto dei **messaggi $m_1…m_N$** inviati da Alice, sia il **numero di messaggi N**) e sicuro (nessuno abbia **modificato il contenuto di nessuno dei messaggi $m_1…m_N$** inviati da Alice o **modificando il loro ordine e il loro numero** ritardando o aggiungendo o duplicando ma non eliminando alcuni dei messaggi).
Pensare a tutti i modi in cui Trudy potrebbe inserirsi generando problemi al raggiungimento dell'obbiettivo (anche solo potere contare il numero di messaggi N) e cercare di prevenirli, spiegando azioni e motivazioni.

protocollo:

Trudy può effettuare forza bruta
Trudy può ritardare causando timeout ⇒ ritrasmissioni ad Alice

$m_1, m_2, …, m_N$

ALICE: $K_B^+[N, K_A^-(h(N))]$
ALICE: for i = 1 a N {
    $K_B^+[m_i, K_A^-(h(m)), I, K_A^-(h(i))]$
    While(true){
    if (timeout && ricevuto ACK m_i) then → exit()
                                else → $K_B^+[m_i, K_A^-(h(m)), I, K_A^-(h(i))]$
    }
}
BOB: attende e riceve N
BOB: while (#{i] < N) {

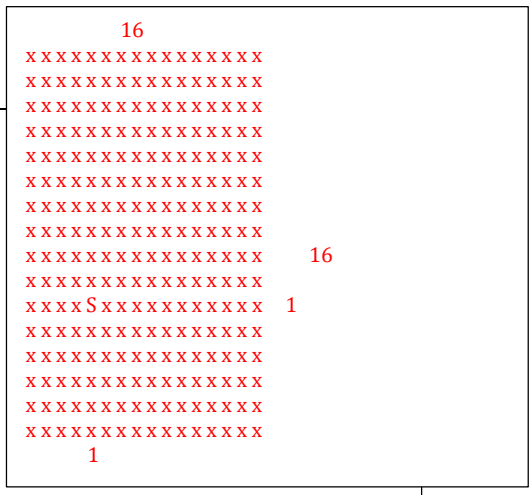
Memorizza m_i (dopo le opportune verifiche)
    Calcola $h(m_i)$ e lo confronta con $K_A^+(K_A^-(h(m_i)))$.
    Se sono uguali → il messaggio è integro.
    Inserisce il messaggio m_i nel buffer di ricezione alla posizione i, dopo avere verificato che il valore i sia uguale a $K_A^+(K_A^-(h(i)))$

}
Per l'affidabilità BOB deve inviare un ACK del messaggio m_i ricevuto ad ALICE
BOB: $K_A^-[ACK(i, K_B^-(h(i)))]$
if i == N then finito()

5[12]) una rete locale di classe C con topologia a stella e uno switch centrale unico ha capacità massima dei collegamenti pari a **10mps** e collega 254 client interni ad un dominio IPv4 unico. In ogni istante al **massimo 100** dei 254 client trasmettono con UDP **X messaggi al secondo**, ognuno della dimensione **costante pari a 25 Bytes** destinati verso il client destinatario identificato con l'indirizzo IPv4 definito prendendo il proprio numero di host (mittente) e sommando il valore 100 e poi facendo il modulo 250 e poi sommando 1 al risultato (ad esempio, il client mittente host 80 manda al destinatario con numero di host (180%250)+1=181, il nodo host 180 manda al nodo (280%250)+1=30+1=31). Ogni destinatario prima riceve per intero gli X messaggi, e poi si attiva e li ri-trasmette al mittente successivo, e la cosa continua all'infinito.
a) Assumendo che a partire con le trasmissioni siano i primi host da 1 a 100, e di raggiungere la saturazione costante della capacità di rete locale, **quanti messaggi X al secondo** potranno essere inviati al massimo dai 100 client attivi in trasmissione in ogni istante nel futuro? Lo switch non fa buffering se non per il pacchetto che è in transito.
b) è possibile aumentare il valore X se inseriamo due sottoreti e rispettivi router nel dominio di classe C, malgrado il limite di Mbps dei collegamenti? Spiegare

Scrivere qui il procedimento e i calcoli
a) 1 ---x--- S ---x--- 101 2 ---x--- S ---x--- 102 100 ---- S ------- 201 Ogni client trasmette max X pacchetti al secondo. X ha capacità massima 10 Mbps Host i può trasmettere a host (100+i) al massimo $25 \cdot 8 = 200$ bit (dim pacchetto) $\cdot X \leq 10$ Mbps $X \leq 10.000.000 / 200 = 50.000$ b) La struttura fisica della rete (stella centrata su S) non cambia, abbiamo solo eletto due nodi con funzioni di router. Quindi rallentiamo il funzionamento, aumentando la latenza.

6[8]) Ragionando solo a livello 1 (fisico), se volessi trasmettere un file da 16 Mbit usando un canale radio a frequenza singola (narrowband) con QPSK e Symbol Rate pari a 500.000 Sym/sec, quanto impiegherebbe la trasmissione? e se dovessi usare una matrice di parità per contrastare 1 bit errato in media ogni 256 bit trasmessi, quanto sarebbe il tempo di trasmissione?
Spiegare



16 Mbit
QPSK = 4 simboli ovvero due bit per simbolo, (codifica)
Symbol rate 500.000 sym/sec * 2 bit/sym = 1.000.000 bit/sec
T_{tras} = 16.000.000 / 1.000.000 = 16 sec.

1 bit errato in media ogni 256 bit. (= 16 x 16)

La matrice di parità mi consente di correggere in media 1 bit ogni 256
Ogni 256 bit trasmessi ne devo aggiungere 16+16 di parità (matrice 256)
Quante matrici devo gestire per i 16 Mbit che voglio trasmettere?

Num matrici = 16.000.000 / 256 = 62.500 matrici * 32 bit = overhead di trasmissione pari a 2.000.000 bit.

Il tempo necessario per trasmettere il tutto:
(16.000.000 + 2.000.000) / 1.000.000 = 18sec

7[5]) Cosa possiamo ricavare come informazioni dalla seguente URL:
https://128.238.251.26:6789/User/Faculty/pippo/html/HelloWorld.html

http = protocollo applicazione (porta 80 well known port number), si effettua una GET
S = versione sicura del protocollo http (porta != da 80) basato su implementazione TLS o SSL
128.238.251.26 = indirizzo IPv4 del server https, rete di classe B 128.238.0.0/16, host 251.26.
Porta socket https = 6789
path nel file system del file richiesto = /User/Faculty/pippo/html/
Nome del file richiesto = HelloWorld.html
Formato del file richiesto = testuale HTML

8[10]) rispondere con evidenza di procedimento e risultati alle seguenti domande:
a) Come sarà possibile fare sub-netting o super-netting che inserisca i seguenti indirizzi IPv4 nella stessa sottorete o super-rete?

host A :55.111.11.32
host B :55.112.24.97

quale sarebbe l'indirizzo e netmask della rete ottenuta?
Indirizzo rete :_____
Netmask rete :_____

b) quale sarà l'indirizzo di router della rete?
_____

Calcoli [procedimento richiesto]

55 rete di classe A 55.0.0.0 / 8
55 . 01101111 (111 decimale)
55 . 01110000 (112 decimale)

55 . 011 00000 .x .y

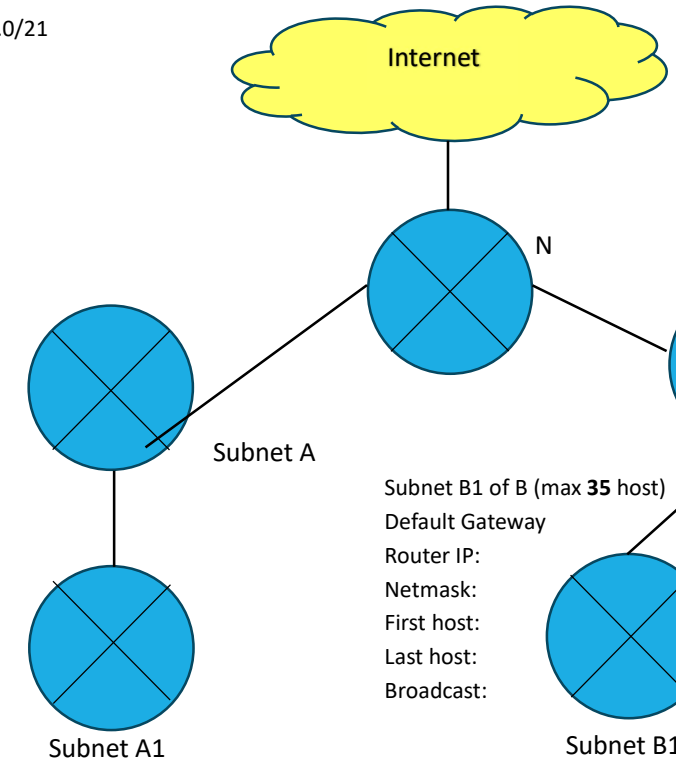
Indirizzo di rete: 55.96.0.0 / 11
Netmask : 255.224.0.0 (/11)

b) indirizzo del router come ultimo post indirizzabile prima del broadcast
55 . 011 11111 .11111111 .11111110 = 55.127.255.254

9[25]) La rete N è connessa a Internet da un Router N collegato a un router A (e alla sua sottorete A) e a un router B con sotterete B. Nella sottorete A a sua volta è collegato un router A1 e alla rispettiva sottorete A1. Nella sottorete B a sua volta è collegato un router B1 e alla rispettiva sottorete B1 e un router B2 e la rispettiva sottorete B2. Lo schema mostra solo i router e i loro collegamenti con interfaccia Ethernet. Definire lo spazio di indirizzi delle reti e sottoreti N, A, A1 e B, B1, e B2, e definire gli indirizzi IPv4 da assegnare agli host e ai router come da schema indicato.
Usare lo spazio sul foglio per fornire traccia del procedimento e calcoli

Network N = 155.136.248.0/21
Router IP:
Netmask:
First Host:
Last Host:
Broadcast:

Subnet B (max **128** host)
Default Gateway
Router IP:
Netmask:
First host:
Last host:
Broadcast:



Subnet A (max **999** host)
Default Gateway
Router IP:
Netmask:
First host:
Last host:
Broadcast:

Subnet A1 (max **44** host)
Default Gateway
Router IP:
Netmask:
First host:
Last host:
Broadcast:

Subnet B1 of B (max **35** host)
Default Gateway
Router IP:
Netmask:
First host:
Last host:
Broadcast:

Subnet B2 of B (max **11** host)
Default Gateway
Router IP:
Netmask:
First host:
Last host:
Broadcast:

N: 155.136.248.0 /21 (subnetting) 5 bit di sottorete.
Spazio di host 256 * 2 * 2 * 2 = 2^11 spazio di indirizzamento per host della subnet N = 2048 -2 host

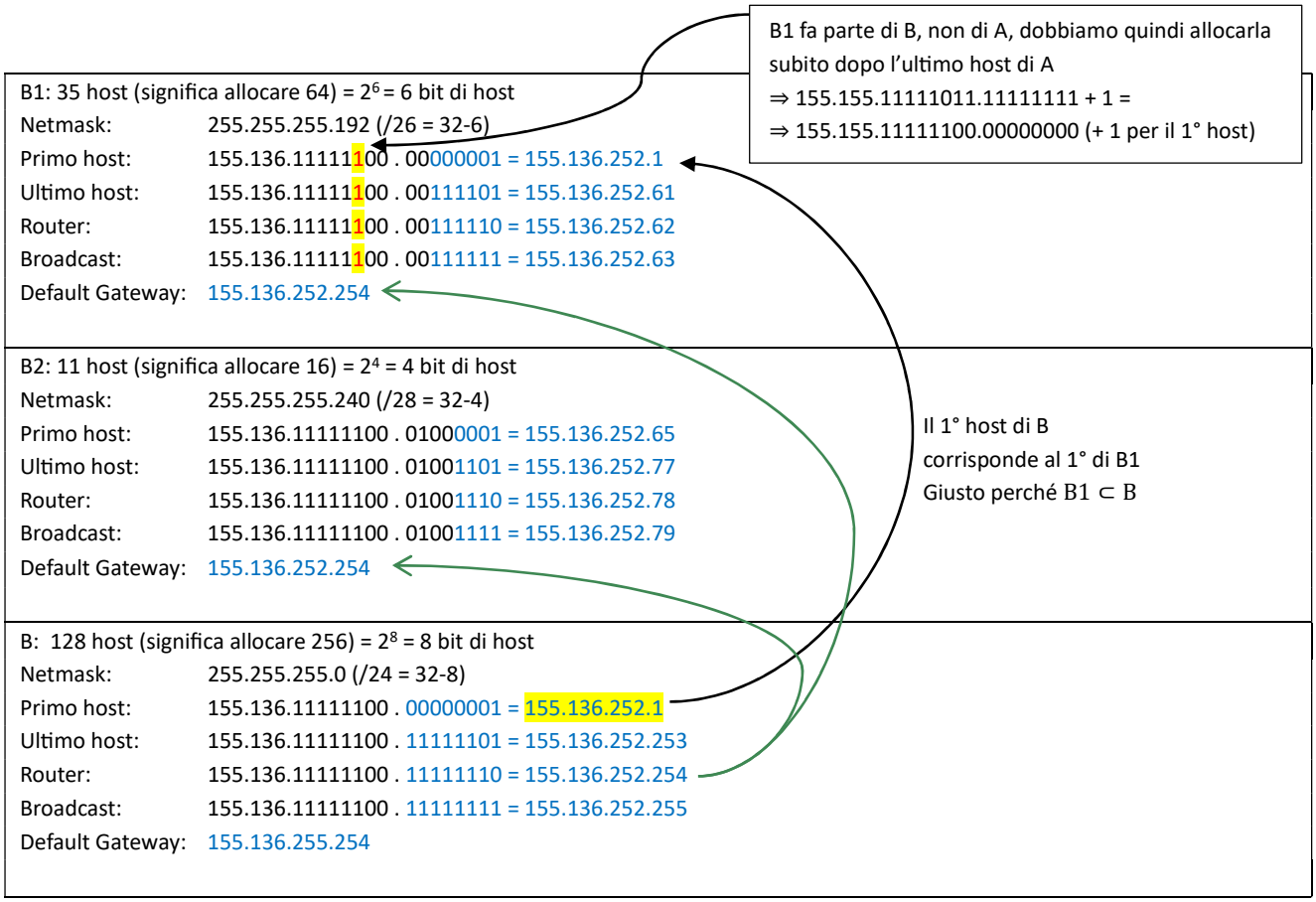
Netmask:	255.255.248.0 (/21 = 32-11)
Primo host:	155.136.11111000 . 00000001 = 155.136.248.1
Ultimo host:	155.136.11111111 . 11111101 = 155.136.255.253
Router:	155.136.11111111 . 11111110 = 155.136.255.254
Broadcast:	155.136.11111111 . 11111111 = 155.136.255.255

A1: 44 host (significa allocare 64 (2^x >)) = 6 bit di host

Netmask:	255.255.255.192 (/26 = 32-6)
Primo host:	155.136.11111000 . 00000001 = 155.136.248.1
Ultimo host:	155.136.11111000 . 00111101 = 155.136.248.61
Router:	155.136.11111000 . 00111110 = 155.136.248.62
Broadcast:	155.136.11111000 . 00111111 = 155.136.248.63
Default Gateway:	155.155.251.254

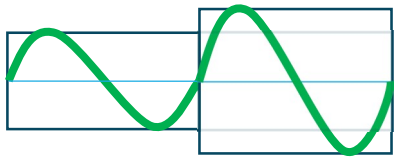
A: 999 host (significa allocare 1024) = 2^{10}=10 bit di host

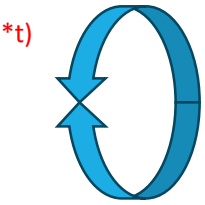
Netmask:	255.255.252.0 (/22 = 32-10)
Primo host:	155.155.11111000 . 00000001 = 155.155.248.1
Ultimo host:	155.155.11111011 . 11111101 = 155.155.251.253
Router:	155.155.11111011 . 11111110 = 155.155.251.254
Broadcast:	155.155.11111011 . 11111111 = 155.155.251.255
Default Gateway:	155.136.255.254

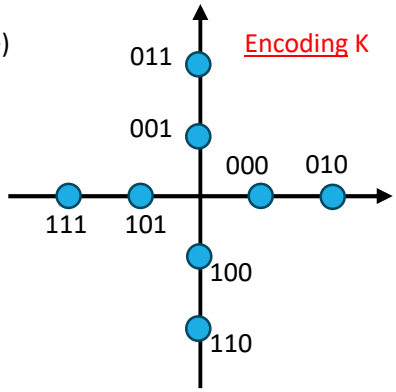


Rimane a disposizione per il futuro:
(primo indirizzo disponibile dopo Broadcast di B) 155.136.253.0 → 155.136.255.255 (Broadcast di N)

10[10]) Un sistema di comunicazione wireless usa la seguente codifica (encoding k): specificare
a) i bit della sequenza binaria di 18 bit che sono trasmessi per la sequenza esadecimale 0EAF…(completare fino a 18 bit aggiungendo zero a destra)
c) esprimere l'etichetta binaria dei simboli trasmessi sotto i relativi simboli della domanda b
b) quali forme d'onda radio saranno generate dall'encoder per trasmettere i 18 bit indicati, disegnandole nei 6 box predisposti sull'asse del tempo, tenendo conto della forma sinusoidale di riferimento A*sin(b*t) fornita (segnale in fase zero nelle due ampiezze utilizzate in esempio).

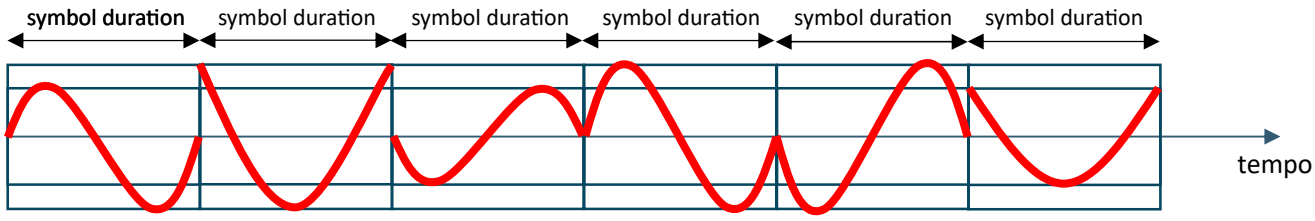






a) Sequenza esadecimale da trasmettere: 0E AF… = sequenza binaria (primi 18 bit): 0000 (0) 1110 (E) 1010 (A) 1111 (F) + 00
000011101010111100 (padding)

b) forme d'onda dei simboli del segnale trasmesso



000 011 101 010 111 100

c) Simboli trasmessi (binario)

000 011 101 010 111 100

000 = sin(t) con fase 0 e ampiezza <
011 = sin(t) con 90° in anticipo e ampiezza >
101 = sin(t) con 180° in anticipo e ampiezza <
010 = sin(t) con fase 0 e ampiezza >
111 = sin(t) con 180° in anticipo e ampiezza >
100 = sin(t) con 90° di ritardo e ampiezza <