Nome e cognome (Name and Surname): Francesco Pirazzini_____
NUMERO MATRICOLA _____

Prova Scritta del corso di Reti di Calcolatori (Computer Networks)
14 Settembre 2020
Docente: Luciano Bononi

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.
[Provide a written answer in the dedicated space only, or in the back of the sheet. Always supply a short motivation and computations in answers who require that.]

1[5]) Che cosa è un algoritmo di error detection e che differenza c'è rispetto a error correction? Fare un esempio di algoritmi visti a lezione.
[Shortly explain what is a error detection algorithm and explain the difference with error correction, with examples.]

2[5]) Cosa significa che un protocollo è connection-less? Ethernet è connection-less? Spiegare
[What does it mean that a protocol is connection-less? Is Ethernet connection-less? Explain.]

3[5]) che cosa è il protocollo AES? Spiegare.
[What is the AES protocol? Explain.]

Il protocollo AES è un protocollo a chiave simmetrica. Ha sostituito DES. Processa i dati in blocchi di 128 bit e utilizza chiavi a 128, 192 o 256 bit.

Nome e cognome (name and surname): Francesco Pirazzini_____(2)

4[15]) Alice spedisce a Bob un messaggio **M** con **garanzia di mittente e privacy**. Inoltre Alice pretende da Bob la prova che il messaggio ricevuto da Bob non possa essere in seguito ripudiato (ovvero Bob non può dire di non averlo ricevuto o di averlo ricevuto diverso). Come può essere realizzato lo schema di cifratura che garantisca i requisiti? Spiegare. [Provide a scheme on how Alice could send to Bob a **message M** with **authenticated sender** and **privacy,** and Bob answers with a **non repudiable confirmation of reception**. Explain your solution.]

Alice manda a Bob il suo messaggio, una Hash del messaggio con la sua firma digitale, il tutto cifrato con la sua chiave privata. Bob alla ricezione del messaggio decifra con la chiave pubblica di Alice il messaggio, esegue la Hash del contenuto e confronta il risultato con l'Hash inviatogli da Alice e grazie alla firma una volta sicuro che il mittente sia Alice manda indietro un acknowledgement con dentro l'Hash del messaggio. In questo modo Alice, confrontando poi l'hash ha la prova che Bob ha ricevuto il messaggio e che il messaggio ricevuto da Bob fosse proprio quello da lei inviato.

ALICE

BOB

Nome e cognome (name and surname): Francesco Pirazzini_____(3)

5[10]) Data questa porzione di codice Python (vista a lezione), spiegare cosa realizza il codice fornito linea per linea. In che modo andrebbe modificato il codice per rendere affidabile la comunicazione? [Explain which function is realized and how by the provided Python code (line per line). How should it be modified to make it based on a reliable communication?]

```
from socket import *

serverName = "130.136.5.36"
serverPort = 12001

clientSocket = socket(AF_INET, SOCK_DGRAM)
message = raw_input('Insert message: ')
clientSocket.sendto(message.encode(), (serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print(modifiedMessage.decode())
clientSocket.close()
```

Spiegazione:

Nella prima linea si include il modulo socket per poter essere in grado di creare socket nel programma

Nelle linee 2 e 3 vengono assegnati i valori alle variabili serverName e serverPort

Nella linea 4 viene creata la socket lato client, memorizzandone il riferimento in una variabile clientSocket. Il parametro AF_INET indica che la rete usa IPv4. Il secondo parametro SOCK_DGRAM indica che è una socket UDP.

Nella linea 5 viene scritto un comando per prendere un input da tastiera che verrà assegnato alla variabile message

Nella linea 6 con message.encode() il messaggio viene convertito da tipo stringa a byte. Poi clientSocket.sendto(…) genera il pacchetto contenente il messaggio convertito e la destinazione (serverName, serverPort) e lo invia alla socket clientSocket.

Nella linea 7 viene definito che una volta arrivato un pacchetto da internet alla socket client, i dati vengano assegnati alla variabile modifiedMessage, invece nella variabile serverAddress viene salvato l'indirizzo sorgente del pacchetto. recvfrom(2048) prende in input la grandezza del buffer.

Nella linea 9 è stato inserito il comando per stampare in output a schermo il messaggio una volta riconvertito da byte a stringa.

Nella linea 10 la socket viene chiusa e il processo termina.

Per rendere affidabile la comunicazione si dovrebbe usare la socket TCP:

```
from socket import *

serverName = "130.136.5.36"
serverPort = 12001

clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
message = raw_input('Insert message: ')
clientSocket.send(message.encode())
modifiedMessage = clientSocket.recv(1024)
print(modifiedMessage.decode())
clientSocket.close()
```

6[10]) Date le seguenti coppie di reti e sottoreti e maschere di rete, in quali casi la fusione delle due reti può realizzare un supernetting corretto? Spiegare [Given the following networks and subnetworks, which cases would realize a correct supernetting. Explain]?

1	10.7.15.99	00001010 00000111 00001111 01100011	netmask /7
	11.7.15.63	00001011 00000111 00001111 00111111	netmask /7
2	129.12.7.10	10000001 00001100 00000111 00001010	netmask /15
	130.11.90.53	10000010 00001011 01011010 00110101	netmask /15
3	200.5.77.8	11001000 00000101 01001101 00001000	netmask /23
	200.5.76.150	11001000 00000101 01001100 10010110	netmask /23
4	199.13.21.9	11000111 00001101 00010101 00001001	netmask /23
	199.13.22.12	11000111 00001101 00010110 00001100	netmask /23

Spiegare:

1) Prendendo solo i primi 7 bit più significativi abbiamo: 0000101 e 0000101 quindi è possibile fare supernetting.
2) Prendendo i primi 15 abbiamo: 10000001 0000110 e 10000010 0000101 quindi non è possibile fare supernetting.
3) Prendendo i primi 23 abbiamo: 11001000 00000101 0100110 e 11001000 00000101 0100110 quindi è possibile fare supernetting.
4) Prendendo di nuovo i primi 23 abbiamo: 11000111 00001101 0001010 e 11000111 00001101 0001011 e quindi essendo diverso il bit meno significativo non è possibile fare supernetting.

7[5]) Cosa indicano le seguenti regole DNS? Spiegare. [What do they mean the following DNS records? Explain]

Record DNS : (Name, Value, Type, TTL)

1) (cs.unibo.it, 130.136.1.110, A, 10) Questo è un record di tipo A, quindi fornisce la corrispondenza tra hostname standard e indirizzo IP.

2) (cs.unibo.it, leporello.cs.unibo.it, CNAME, 10) Questo è un record di tipo CNAME quindi Value rappresenta il nome canonico dell'host per il sinonimo Name

3) (cs.unibo.it, serpina.cs.unibo.it, MX, 10) Questo è un record di tipo MX quindi value è il nome canonico di un mail server avente sinonimo Name

4) (cs.unibo.it, dns1.cs.unibo.it, NS, 10) Questo è un record di tipo NS quindi Name è un dominio e Value è il nome dell'host del server autoritativo DNS che è in grado di ottenere gli indirizzi IP degli host di quel dominio.

8[10]) Chi dovrebbe essere il router (con ultimo indirizzo IP valido) della rete che contiene l'host 211.128.77.15 se la maschera di rete fosse 255.255.255.224? [which IP address should be assigned to the router of network containing the IP 211.128.77.15 when the netmask is 255.255.255.224?]

IPv4 del Router: 211.128.77.30

e se la maschera di rete fosse /25? [and in case the netmask is /25?): 211.128.77.126

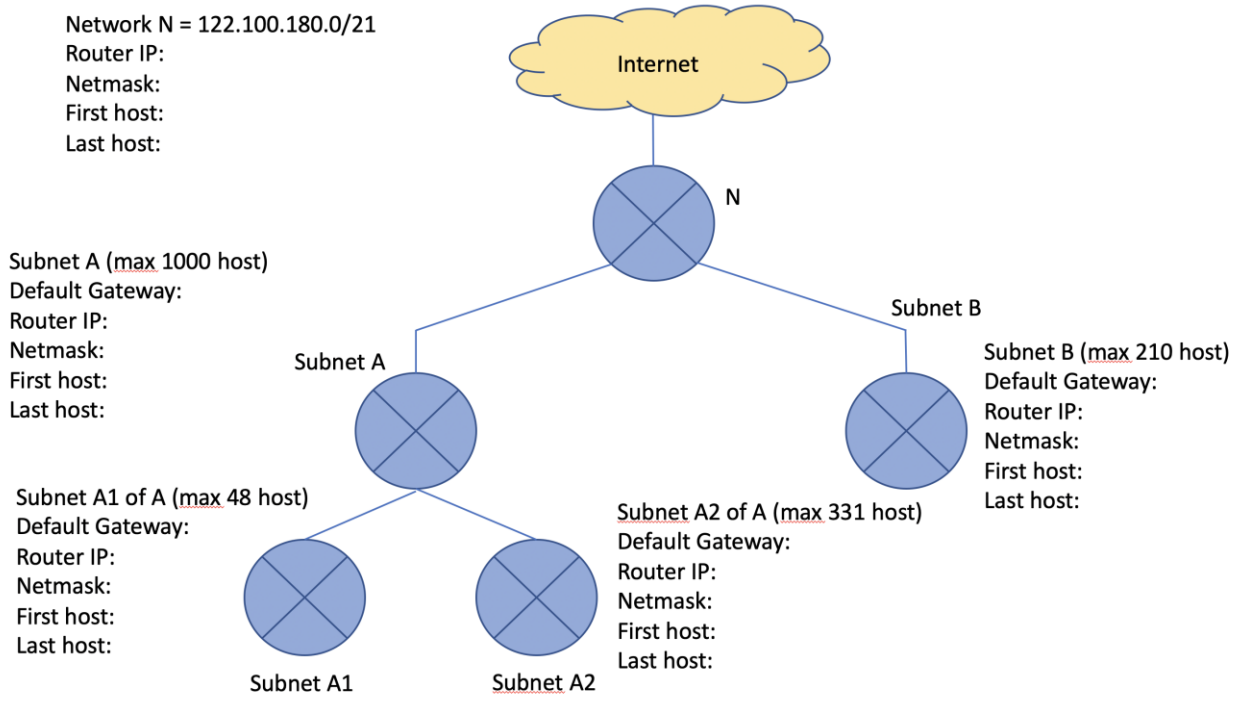
Calcoli [computation]

211 . 128 . 77 . 15 = 11010011 . 10000000 . 01001011 . 000 01111
11111111 . 11111111 . 11111111 . 111 00000

Indirizzo di rete = 11010011 . 10000000 . 01001011 . 000 00000

Netmask = /25 = 11111111 . 11111111 . 11111111 . 1 0000000

9[25] Definire gli indirizzi IPv4 assegnabili nelle reti LOCALI sotto indicate per le esigenze definite:
Usare lo spazio sul foglio per traccia procedimento e calcoli. [Define the IP addressing for the local network below. Use the back sheet for computation.]



Spiegare qui sotto il procedimento [explain how you got the results here]

128 64 32 16 8 4 2 1

Network N = 122.100.180.0/21 ➔ 01111010 . 01100100 . 10110 100 . 00000000
Netmask = 255.255.248.0 11111111 . 11111111 . 11111 000 . 00000000

La rete N può ospitare 2^11 = 2048 host

First Host = 122.100.180.1
Last Host = 122.100.183.254 (Router IP)
Last Host = 122.100.183.255 (Broadcast)

Subnet A = 122.100.180.0/22 In questo modo la rete può ospitare 2^10 = 1024 host che soddisfano il requisito
Default gateway = 122.100.183.254
Netmask = 255.255.252.0

01111010 . 01100100 . 101101 00 . 00000000
11111111 . 11111111 . 111111 00 . 00000000

First host = 122.100.180.1
Last host = 122.100.183.254 (Router IP)
Last host = 122.100.183.255 (Broadcast)

Subet A2 = 122.100.180.0/23 In questo modo la rete può ospitare 2^9 = 512 host che soddisfano il requisito
Default gateway = 122.100.183.254
Netmask = 255.255.254.0

01111010 . 01100100 . 1011010 0 . 00000000
11111111 . 11111111 . 1111111 0 . 00000000

First host = 122.100.180.1
Last host = 122.100.181.254 (Router IP)
Last host = 122.100.181.255 (Broadcast)

Subnet A1 = 122.100.182.0/26 In questo modo la rete può ospitare 2^6 = 64 host che soddisfano il requisito

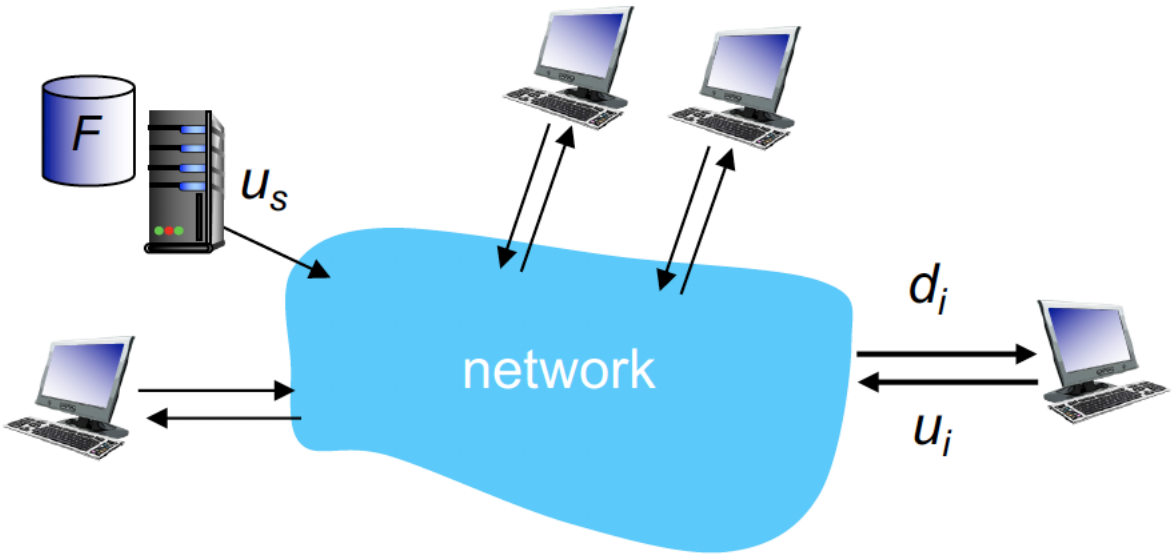
Default gateway = 122.100.183.254
Netmask = 255.255.255.192

01111010 . 01100100 . 10110110 . 00 000000
11111111 . 11111111 . 11111111 . 11 000000

First Host = 122.100.182.1
Last Host = 122.100.182.62 (Router IP)
Last Host = 122.100.182.63 (Broadcast)

Subnet B = 122.100.183.0/24
Default gateway = 122.100.183.254
First host = 122.100.183.1
Last host = 122.100.183.254 (Router IP)
Last host = 122.100.183.255 (Broadcast)

10 [10]) In un sistema di distribuzione di file P2P un server S possiede inizialmente l'unica copia del file da F GByte richiesto da N client. Siano di e ui le capacità omogenee dei canali di comunicazione di rete in download e upload rispettivamente di ogni client Ci, e ds e us le capacità della connessione (download e upload) del server S. Spiegare come si determina il tempo massimo di consegna del file F a tutti i client in modalità sequenziale e Peer to Peer (esprimere, confrontare e spiegare le formule).



Spiegare qui [explain how you got the results here]

In un approccio client-server il server deve fare l'upload del file N volte quindi se il tempo richiesto per fare l'upload una sola volta è la dimensione del file fratto la capacità di upload del server ➔ F/u_s allora per farlo N volte sarà NF/u_s .
Dall'altro lato ci sono però i client che al contrario devono fare ognuno il download del file e prendendo in esame il client che ha la velocità di download più bassa d_{min} allora il tempo di download sarà F/d_{min}. Quindi in un approccio client-server il tempo massimo sarà definito da uno di questi due parametri ➔ Client-server >= max{NF/u_s, F/d_{min}}
La relazione presenta un >= poiché nel migliore dei casi il tempo di attesa sarà pari al maggiore dei due elementi, ma questo non prende in considerazione ritardi imputabili ad altre cause, quindi il tempo potrebbe essere anche maggiore ma non minore.

Nell'approccio P2P invece il server deve fare l'upload di una copia ➔ F/u_s. Ogni client deve scaricare il file ➔ F/d_{min} ma essendo un approccio P2P ogni client poi, man mano che il file viene acquisito, deve ritrasmetterlo a sua volta, quindi alla capacità di upload del server si va a sommare quella dei client ➔ u_s + sommatoria di u_i.
In questo modo nell'approccio P2P si va ad aggiungere un altro parametro ➔
P2P >= max{F/u_s, F/d_{min}, $NF/(u_s$ + sommatoria di $u_i)$}