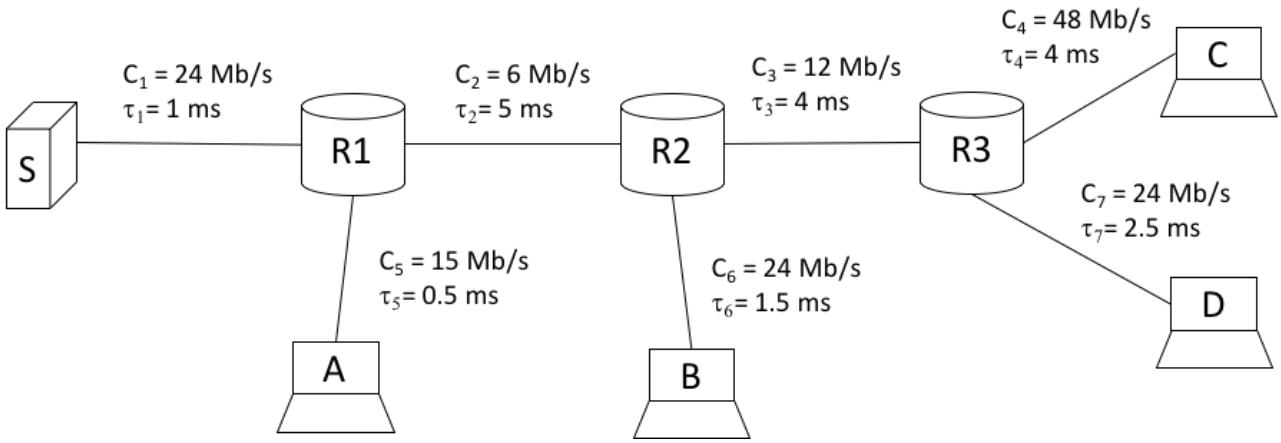
Fondamenti di Internet e Reti
Proff. A. Capone, M. Cesana, I. Filippini

Prova in itinere – 2 Maggio 2018

Cognome	
Nome	
Matricola	

Tempo complessivo a disposizione per lo svolgimento: 2h
Usare lo spazio dopo ogni Esercizio/Quesito per la risposta.

Es1 (6 pt)	Es2 (6 pt)	Es3 (5 pt)	Ques (9 pt)	Lab (6pt)



Compito

1- Esercizio (6 punti)

Si consideri la rete nella figura della prima pagina che utilizza la commutazione di pacchetto di tipo store-and-forward.

a) Al tempo t=0 sono presenti 8 pacchetti in S diretti rispettivamente alle seguenti destinazioni: C,C,D,D,A,B,A,B. I pacchetti abbiano le seguenti dimensioni: L_A=1200 Byte, L_B=1500 Byte, L_C=1500 Byte, L_D=300 Byte. Si calcoli l'istante di ricezione dei soli pacchetti diretti a C, D e B.
b) Al tempo t=0 sono presenti 3 pacchetti in S diretti a C con dimensioni L_C=1500 Byte, e 3 pacchetti in A e diretti a B con dimensione L_B=1500 Byte. Si calcoli l'istante di ricezione dei soli pacchetti diretti a B.

2 - Esercizio (6 punti)

Si consideri la rete nella figura della prima pagina. Una connessione TCP tra l'*host* S e l'*host* C è caratterizzata dai seguenti parametri: lunghezze di *header* e *ack* trascurabili, *link* bidirezionali simmetrici, *MSS* = 1500 Byte, *RCWND* =200 kByte*, SSTHRESH* = 12 kByte.

a) Si calcoli il tempo necessario a trasferire un file di 0.525 MByte.
b) Si ripeta il calcolo assumendo che il 25° pacchetto vada perso e il timeout corrispondente sia T_{out}=200 ms (si assuma i pacchetti fuori sequenza siano memorizzati).

3 - Esercizio (5 punti)

Si consideri la rete nella figura della prima pagina. Si assuma S sia un server http e C un client http. Si assuma, inoltre, tempo di apertura della connessione dovuto al solo tempo di propagazione e RTT pari a 31.75 [ms], indipendente dal rate della connessione. Occorre trasferire un documento base di 15 KByte e 5 immagini di 0.525 MByte.

a) Si calcoli il tempo necessario assumendo 1 flusso interferente tra A e B e 2 flussi tra B e D, e che la connessione sia unica e persistente.
b) Come in a), ma con connessioni non-persistenti e trasferimento in parallelo delle immagini.

Quesiti (9 punti)

Q1

Da un host viene eseguito il comando dig. Rispondere alle domande sotto.

```
$ dig -t ANY polimi.it

; <<>> DiG 9.10.6 <<>> -t ANY polimi.it
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16498
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 11, AUTHORITY: 0, ADDITIONAL: 7

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;polimi.it.              IN    ANY

;; ANSWER SECTION:
polimi.it.         86400 IN    SOA   ns.polimi.it. root.ns.polimi.it. 2017083034 11600 3600 604800 86400
polimi.it.         86400 IN    NS    dns.cineca.it.
polimi.it.         86400 IN    NS    ns.polimi.it.
polimi.it.         86400 IN    NS    ns2.polimi.it.
polimi.it.         86400 IN    A     131.175.187.72
polimi.it.         900   IN    MX    0 polimi-it.mail.protection.outlook.com.

;; ADDITIONAL SECTION:
ns.polimi.it.            86400 IN    A     131.175.12.1
ns2.polimi.it.           86400 IN    A     131.175.12.2
dns.cineca.it.           1543  IN    A     130.186.1.70
polimi-it.mail.protection.outlook.com. 9 IN A     213.199.154.234
polimi-it.mail.protection.outlook.com. 9 IN A     213.199.154.170
dns.cineca.it.           2     IN    AAAA  2001:760:2e0a:8002::20

;; Query time: 4 msec
;; SERVER: 131.175.12.1#53(131.175.12.1)
;; WHEN: Thu Apr 26 10:38:03 CEST 2018
;; MSG SIZE  rcvd: 657
```

Il server da cui arriva la risposta è authoritative? Quali sono i server authoritative?

A cosa è associato l'indirizzo 131.175.187.72?

A quali server (nome ed indirizzo IP) verrebbe spedita una mail indirizzata a rettore@polimi.it?

A che scopo gli indirizzi 213.199.154.234 e 213.199.154.170 sono associati allo stesso nome?

Q2

Un segnale audio è caratterizzato da una banda di 22 kHz. Viene campionato alla minima frequenza di campionamento usando 24 bit per campione. Successivamente viene suddiviso in pacchetti di 1460 Byte di dati e 140 Byte di overhead dei vari protocolli. Calcolare:

Il rate del segnale campionato e quantizzato (in bit a secondo):

Il numero di pacchetti generati ogni secondo:

Il rate dovuto alla trasmissione dei pacchetti generati (in bit al secondo):

Q3

Durante una sessione TCP, l'algoritmo di Jacobson stima valor medio e deviazione standard del RTT come $SRTT^0 = 100$ ms e $SDEV^0 = 40$ ms. I due segmenti successivi registrano un RTT di $RTT^1 = 160$ ms e $RTT^2 = 90$ ms. *Si indichino* nella tabella i valori di SRTT, SDEV, DEV e del Timeout alla ricezione di ciascuno dei due segmenti considerando $(1 - \alpha) = 7/8$ come peso della stima precedente di RTT e $(1 - \beta) = 3/4$ come peso della stima precedente di SDEV. Si usi la tabella per indicare i risultati finali e lo spazio sottostante per mostrare i conti fatti.

	RTT	**SRTT**	**DEV**	**SDEV**	**Timeout**
$SRTT^0 = 100$ $SDEV^0 = 40$	$RTT^1 = 160$	$SRTT^1 =$	$DEV^1 =$	$SDEV^1 =$	$T^1 =$
	$RTT^2 = 90$	$SRTT^2 =$	$DEV^2 =$	$SDEV^2 =$	$T^2 =$

Prova in itinere – 2 Maggio 2018 – Laboratorio (6 punti)

Nome Cognome	
Matricola	

Client

```
from socket import *
clientSocket = socket(AF_INET, SOCK_DGRAM)
clientSocket.settimeout(1)
try:
    while 1:
           message = raw_input('Input a Country name, or Close to finish this client:')
           if message == 'Close':
               break
          clientSocket.sendto(message, ('localhost', 2018))
           modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
          if modifiedMessage ...
```

Server

```
from socket import *
import requests
import json
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('localhost', 2018))
while 1:
    try:
        message, clientAddress = serverSocket.recvfrom(2048)
        modifiedMessage = message.title()
        #title() ritorna una copia della stringa in cui il primo carattere di tutte le
        #        parole è maiuscolo
        r = requests.get('http://api.football-data.org/v1/competitions/467/teams')
        teams_qualificati = parseRisposta(r.json())
        t = '0'
        for team in teams_qualificati:
            if team == modifiedMessage:
                t = '1'
        serverSocket.sendto(t, clientAddress)
    finally:
        serverSocket.close()
```

Q1. Indicare gli errori che ci sono nello script Server **(2 punti)**

Q2. Completare il codice sul client per far sì che stampi a video se il paese inserito da tastiera è qualificato al mondiale di calcio 2018 oppure no. Hint: *parseRisposta(param)* è una funzione che si occupa di parsare *param* e ritorna la lista di team qualificati al mondiale **(4 punti)**

Codice esercizi laboratorio

UDP client

```
from socket import *
serverName = 'localhost'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
message = raw_input('Input lowercase sentence:')
clientSocket.sendto(message, (serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print modifiedMessage
clientSocket.close()
```

UDP server

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print "The server is ready to receive"
while 1:
    message, clientAddress = serverSocket.recvfrom(2048)
    print "Datagram from: ", clientAddress
    modifiedMessage = message.upper()
    serverSocket.sendto(modifiedMessage, clientAddress)
```

UDP error management

```
from socket import *
serverName = 'localhost'
serverPort = 12001
clientSocket = socket(AF_INET, SOCK_DGRAM)
clientSocket.settimeout(5)
message = raw_input('Input lowercase sentence:')
try:
    clientSocket.sendto(message, (serverName, serverPort))
    modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
    # in case of error blocks forever
    print modifiedMessage
except error, v:
    print "Failure"
    print v
finally:
    clientSocket.close()
```

TCP client

```
from socket import *
serverName = 'localhost'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence)
modifiedSentence = clientSocket.recv(1024)
print 'From Server:', modifiedSentence
clientSocket.close()
```

TCP server

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
```

```
print 'The server is ready to receive'
while True:
    connectionSocket,  clientAddress = serverSocket.accept()
    print "Connection form: ", clientAddress
    sentence = connectionSocket.recv(1024)
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence)
    connectionSocket.close()
```

TCP client persistent

```
from socket import *
serverName = 'localhost'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
while True:
    sentence = raw_input('Input lowercase sentence ( . to stop):')
    clientSocket.send(sentence)
    if sentence == '.':
        break
    modifiedSentence = clientSocket.recv(1024)
    print 'From Server:', modifiedSentence
clientSocket.close()
```

TCP server persistent

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
while True:
    print 'The server is ready to receive'
    connectionSocket,  clientAddress = serverSocket.accept()
    print "Connection form: ", clientAddress
    while True:
        sentence = connectionSocket.recv(1024)
        if sentence == '.':
            break
        capitalizedSentence = sentence.upper()
        connectionSocket.send(capitalizedSentence)
    connectionSocket.close()
```

TCP auto client

```
from socket import *
import time
serverName = 'localhost'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
for a in range(100):
    clientSocket.send('A')
time.sleep(1)
clientSocket.send('.')
#clientSocket.recv(1024)
clientSocket.close()
```

TCP auto server

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
```

```
while True:
    print 'The server is ready to receive'
    connectionSocket,  clientAddress = serverSocket.accept()
    print "Connection form: ", clientAddress
    while True:
        sentence = connectionSocket.recv(1024)
        if sentence == '.':
            break
        print len(sentence)
#        connectionSocket.send(capitalizedSentence)
    connectionSocket.close()
```

TCP server thread

```
from socket import *
import thread
def handler(connectionSocket):
    while True:
        sentence = connectionSocket.recv(1024)
        if sentence == '.':
            break
        capitalizedSentence = sentence.upper()
        connectionSocket.send(capitalizedSentence)
    connectionSocket.close()
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
while True:
    print 'The server is ready to receive'
    newSocket,  addr = serverSocket.accept()
    thread.start_new_thread(handler, (newSocket,))
```