Esame Reti 14 Febbraio 2020
Parte B

- Socket server con Select
- HTTP GET con if-modified-since
- DNS ricorsivo
- SMTP e Mailbox

Esercizio 1) (14 punti)
Si vuole implementare un'applicazione server basata sulla funzione select per la gestione contemporanea di due socket server, uno basato su socket TCP e l'altro su socket UDP, che forniscono un servizio di vocabolario. Il servizio vocabolario riceve in ingresso una stringa e ritorna in uscita la sua dedizione sempre sotto forma di stringa. Si richiede di fornire lo pseudocodice del server basato sulla funzione socket select.
N.B. Le funzioni della libreria socket devono essere proposte in modo completo con tutti i parametri specificati. Non verrà accettato uno pseudocodice che utilizza le librerie socket di Java.

```
int main() {
	int sockTCP, sockUDP, socketPronti, connessione;
	fd_set rset; //Set dei socket in lettura

	struct sockaddr_in serverTCP, serverUDP, clientTCP, clientUDP;

	serverTCP.sin_family = AF_INET;
	serverTCP.sin_port = htons(4000);
	serverTCP.sin_addr.s_addr = htonl(INADDR_ANY);

	serverUDP.sin_family = AF_INET;
	serverUDP.sin_port = htons(4001);
	serverUDP.sin_addr.s_addr = htonl(INADDR_ANY);

	//Socket TCP
	sockTCP = socket(AF_INET, SOCK_STREAM, 0);
	bind(sockTCP, (struct sockaddr *)&serverTCP, sizeof(serverTCP));
	listen(sockTCP, 5);

	//Socket UDP
	sockUDP = socket(AF_INET, SOCK_DGRAM, 0);
	bind(sockUDP, (struct sockaddr *)&serverUDP, sizeof(serverUDP));

	FD_ZERO(&rset);   //Init socket set
	int maxvalsock = (sockTCP > sockUDP ? sockTCP : sockUDP) + 1;

	while(1) {
		FD_SET(sockTCP, &rset);
		FD_SET(sockUDP, &rset);

		socketPronti = select(maxvalsock, &rset, NULL, NULL, NULL);
		char parola[20];
		int dim = sizeof(struct sockaddr_in);

		//Gestisco la richiesta in arrivo dal socket TCP
		if(FD_ISSET(sockTCP, &rset)) {
			int connessione = accept(sockTCP,
				(struct sockaddr *) &clientTCP, (socklen_t*) &dim);
			recv(connessione, &parola, 20,  0);
			char def[200];
			vocabolario(&parola, &def);
			send(connessione, &def, 200, 0);
			close(connessione);
		}
```

```
            //Gestisco la richiesta in arrivo dal socket UDP
            else if(FD_ISSET(sockUDP, &rset)) {
                  recvfrom(sockUDP, &parola, 20,  0,
                        (struct sockaddr *) &clientUDP, (socklen_t*) &dim);
                  char def[200];
                  vocabolario(&parola, &def);
                  sendto(sockUDP, &def, 200, 0,
                        (struct sockaddr *) &clientUDP, sizeof(clientUDP));
            }
      }
      exit(0);
}
```

Esercizio 2) (8 punti)
Nell'ambito del protocollo HTTP si mostri una richiesta GET condizionale tramite l'header if-modied-since. Si discutano inoltre le possibili risposte a tale richiesta e per ognuna delle risposte mostrare il contenuto del messaggio HTTP comprensivo di header.

Richiesta GET

```
GET /prova.html HTTP/1.1
User-Agent: Mozilla/5.0 (Firefox/3.5.5)
Accept: text/html
If-modified-since: Sun, 19 Apr 2020 07:28:00 GMT
```

Quando viene specificato questo campo in una richiesta HTTP, il server, se dispone di una versione più recente della risorsa richiesta la manda, altrimenti trasmette un header che dice che la risorsa è ancora quella e che quindi si può prelevare dalla cache del browser (rispondendo con un codice 304)

Se il server ha una risorsa più recente

```
HTTP/1.1 200 OK
Date: Mon, 20 Apr 2020 09:00:00 GMT
Server: Apache
Expires: Mon, 27 Apr 2020 09:00:00 GMT
Content-Type: text/html
Content-Length: {Lunghezza contenuto}

<!DOCTYPE html>
<html>
      <head><title>prova</title>
      <body>prova</body>
</html>
```

Se il server non ha una risposta più recente

```
HTTP/1.1 304 Not Modified
Date: Mon, 20 Apr 2020 09:00:00 GMT
Server: Apache
```

Domanda 1) (4 punti)
Nell'ambito del DNS, si presenti il processo di risoluzione dei nomi ricorsivo e se ne presenti un esempio.
I server DNS ricorsivi servono come intermediari per trovare a quale server autorevole rivolgersi per la risoluzione di un nome.

Quando viene fatta una richiesta ad un server DNS ricorsivo, esso controlla se nella sua cache è presente un record per il nome cercato, e se è presente e non è scaduto il suo TTL restituirà il valore associato a quel record, diversamente si occuperà di interpellare i relativi DNS server autorevoli.
La risoluzione avverrà partendo dai server di risoluzione dei TLD, i quali indicheranno a quale altro server autorevole rivolgersi per la risoluzione della successiva parte di nome. Il seguente server interpellato si comporterà in maniera speculare. Il processo termina quando viene interpellato il server autorevole che sarà in grado di rispondere con l'IP corrispondente al nome cercato. Il server ricorsivo andrà quindi a salvare la corrispondenza nella sua cache locale, con il relativo TTL.

Un classico esempio di server DNS ricorsivo è quello della rete locale, che viene interpellato per tutte le richieste di risoluzione di nomi.

Esempio domanda: www.ripe.net A

- Il resolver chiede al server ricorsivo la risoluzione di www.ripe.net A
- Il server ricorsivo interpellato (che in questo caso non ha il record in cache) interpella il root server chiedendogli la risoluzione
- Il root server risponderà indicando il server che permetterà di risolvere i TLD ".net" @x.gtld-server.net
- Il server ricorsivo allora rimanderà la query all gtld-server
- Il gtld-server risponderà indicando il server in grado di risolvere "ripe.net" @ns.ripe.net
- A questo punto, il server ricorsivo inoltrerà la query al server ripe indicato
- Il server ripe, il quale disporrà dell'IP corrispondente al nome cercato, risponderà fornendo l'IP
- Il server ricorsivo salva quindi nella sua cache l'indirizzo IP e inoltra la risposta al resolver.

Domanda 2) (4 punti)
Dopo aver discusso le caratteristiche principali del protocollo SMTP, si definisca il concetto di mailbox e se ne discuta la struttura.

SMTP è un protocollo basato su TCP il cui scopo è permettere lo scambio di messaggi usando tecnologie che ne permettano la raccolta e l'accodamento.
Smtp è un protocollo protocollo non interattivo, ossia non prevede l'interazione diretta con l'utente, e prevede che i messaggi vengano accodati e inviati tramite un daemon (chiamato Mail Transfer Agent) che accede ad una apposita directory (directory di spooling) dove vengono salvati i messaggi.

SMTP prevede tre attori nel suo funzionamento:
- user agent: ossia il programma di scrittura della posta
- mail server: server che si occuperà dell'invio dei messaggi utilizzando SMTP ad altri mail server
- mail box: directory che contiene i messaggi recapitati tramite SMTP

La comunicazione, segue tre fasi:
- handshaking (HELO, scambio di identità)
- trasferimento dei messaggi
- chiusura della connessione

La comunicazione tra client e server avviene tramite un paradigma di comando/risposta, dove i comandi sono normale testo ASCII e le risposte sono codici di stato (opzionalmente con delle frasi descrittive).

L'intestazione si compone di un minimo di tre campi:
- From:
- To:

- Subject:
A questi possono essere aggiunti altri campi utili, come Cc, Ccn e così via…

La mailbox, o casella di posta, risiede sul mail server e permette di archiviare la posta in arrivo.

Quando il mittente invia una email, smtp provvede a recapitarla al mail server di destinazione, e qui vi sarà uno user agent pronto che si farà carico di smistare la posta in arrivo e archiviarla nella corretta mailbox del destinatario (il mail server destinatario molto probabilmente avrà più mailbox al suo interno).

Se lo user agent del destinatario non risiede sulla stessa macchina della mailbox, esso vi farà accesso tramite protocolli POP (che prevedono lo scaricamento di tutto il contenuto della mailbox all'interno della macchina del destinatario, con consecutivo svuotamento della mailbox) o IMAP (che permette la lettura dei messaggi che risiedono nella mailbox senza scaricarli).