UNIVERSITÀ DEGLI STUDI DI PALERMO
Dipartimento di Ingegneria Chimica, Gestionale, Informatica, Meccanica
Corso di Laurea in Ingegneria Informatica e delle Telecomunicazioni

Corso di Reti di Calcolatori e Internet
A.A. 2013/2014

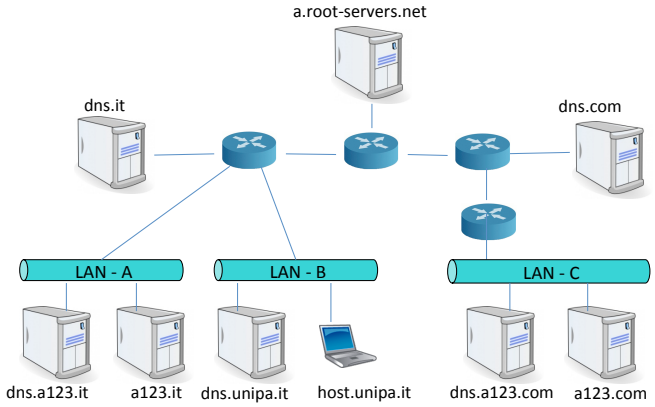
Docente: Ing. Alessandra De Paola
02 Settembre 2014

Quesito 1

Si supponga che il sito web di un'azienda sia ospitato sul server `a123.com` e che consista in una pagina HTML di dimensione 15 kB, contenente 2 immagini di 600 kB ciascuna. Si supponga che l'azienda utilizzi il *web caching* per duplicare il portale sui propri altri server remoti (es. `a123.it`) e che abbia configurato il proprio DNS per risolvere il nome del server nell'indirizzo IP che risulti geograficamente più vicino a chi fa la richiesta.

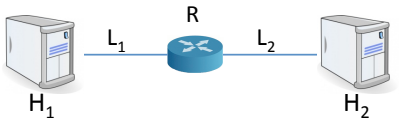
Data la configurazione di rete indicata in figura, si calcoli il risparmio nel tempo totale di trasferimento, incluso il tempo necessario alle richieste DNS, assumendo la modalità ricorsiva, quando la richiesta proviene dall'host `host.unipa.it` rispetto a quanto avverrebbe senza *web caching*.

Si assuma per semplicità che tutti i collegamenti abbiano caratteristiche identiche: ampiezza di banda pari a 100 Mbps, ritardo di propagazione pari a 1 ms, MTU = 1500 byte; l'overhead complessivo di pacchetto sia di 40 byte.



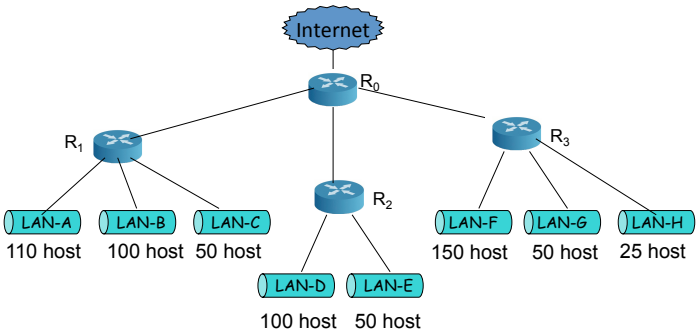
Quesito 2

Dati due host connessi come indicato in figura, si supponga che il primo host inizi a spedire un file di 16 kB verso l'altro mediante il protocollo TCP. Si assuma che i canali abbiano ampiezze di banda rispettivamente pari a 100 Mbps e 50 Mbps, che RTT=0.6 ms, MSS=1024 byte e che gli overhead siano trascurabili. Si determini il throughput della trasmissione, assumendo che non si verifichi congestione. Si evidenzi inoltre l'andamento della finestra di congestione.



Quesito 3

Avendo a disposizione range `10.1.224.0/21`, si proponga uno schema di indirizzamento che minimizzi lo spreco di indirizzi per ciascuna rete e il numero di entry nelle tabelle di inoltro. Si scrivano le tabelle di inoltro per i diversi router della rete.



Quesito 4

Completare il codice fornito per realizzare il server di una semplice applicazione di *random chat* tramite protocollo TCP. L'applicazione prevede che il server attenda la connessione di due client qualsiasi per poi metterli in comunicazione. All'interno di una coppia di client, ogni messaggio digitato dall'uno viene recapitato all'altro e visualizzato sullo standard output, secondo uno schema rigido client1-client2-client1-client2-..... Se uno dei due client digita il comando QUIT la chat viene interrotta.

```c
#include <stdio.h>
#include <stdlib.h>     /* exit() */
#include <strings.h>    /* bzero(), bcopy() */
#include <unistd.h>
#include <netinet/in.h>
#include <sys/types.h>  /* tipi di dati di sistema */
#include <sys/socket.h> /* definizioni utili per le socket() */

#define MAXLINE 4096

void error(char *msg) {
    perror(msg);
    exit(1);
}

int main(int argc, char *argv[]) {

    int sockfd, newsockfd1, newsockfd2;
    int server_port = 6543, clilen;
    char buffer[256];
    struct sockaddr_in servaddr, cliaddr1, cliaddr2;
    pid_t pid;

    if ( (sockd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
        err_sys("errore in socket");
```

```
  bzero((char *) &serv_addr, sizeof(serv_addr));
  servaddr.sin_family = AF_INET;
  servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
  servaddr.sin_port = htons(server_port);

  if (bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0) {
      error("ERROR on binding");}

  listen(sockfd,10);

  while(1){
      clilen = sizeof(cli_addr1);
      newsockfd1 = accept(sockfd,
                          (struct sockaddr *) &cli_addr,
                          (socklen_t *)&clilen1);
      newsockfd2 = accept(sockfd,
                           (struct sockaddr *) &cli_addr,
                           (socklen_t *)&clilen2);
      if (newsockfd1 < 0 || newsockfd2 < 0) {
          error("ERROR on accept");
      }

      pid=fork();
      if (pid == 0) {
          close(sockfd);

           while(1){
               bzero(buffer,256);

               .........................

               .........................

               .........................

               .........................

               .........................

               .........................

               .........................

               .........................

               .........................

               .........................

               .........................

           }

       }

       close(newsockfd1);
       close(newsockfd2);
   }

   return 0;
}
```

Documentazione Programmazione Socket

//Accept an incoming connection on a listening socket
int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);

//Associate a socket with an IP address and port number
int bind(int sockfd, struct sockaddr *my_addr, socklen_t addrlen);

//Connect - initiate a connection on a socket
int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);

int close(int sockfd); *//Close a socket descriptor*

struct hostent *gethostbyname(const char *name); *//Get an IP address for a hostname*

//Convert multi-byte integer types from host byte order to network byte order
uint32_t htonl(uint32_t hostlong);
uint16_t htons(uint16_t hostshort);
uint32_t ntohl(uint32_t netlong);
uint16_t ntohs(uint16_t netshort);

//Convert IP addresses to human-readable form and back
const char *inet_ntop(int af, const void *src, char *dst, socklen_t size);
int inet_pton(int af, const char *src, void *dst);

int listen(int sockfd, int queuelength); *//Tell a socket to listen for incoming connections*

//Receive data on a socket
ssize_t recv(int sockfd, void *buf, size_t len, int flags); *// for TCP socket*
ssize_t recvfrom(int sockfd, void *buf, size_t len, int flags, struct sockaddr *from, socklen_t *fromlen); *// for UDP socket*
ssize_t read(int sockfd, void *buf, size_t count); *// for TCP socket*

//Send data out over a socket
ssize_t send(int sockfd, const void *buf, size_t len, int flags); *// for TCP socket*
ssize_t sendto(int sockfd, const void *buf, size_t len, int flags, const struct sockaddr *to, socklen_t tolen); *// for UDP socket*
ssize_t write(int sockfd, const void *buf, size_t count); *// for TCP socket*

int socket(int domain, int type, int protocol); *//Allocate a socket descriptor*

//Structures for handling internet addresses
struct sockaddr_in {// ...
    short sin_family; *// e.g. AF_INET, AF_INET6*
    unsigned short sin_port; *// e.g. htons(3490)*
    struct in_addr sin_addr; *// see struct in_addr* };

struct in_addr{ unsigned long s_addr; *//e.g. INADDR_ANY* };

//Structure for handling host names
struct hostent{// ...
    char *h_name; *//The real canonical host name.*
    int h_addrtype; *//The result's address type, e.g. AF_INET*
    int length; *//The length of the addresses in bytes, which is 4 for IP (version 4) addresses.*

```
    h_addr; //An IP address for this host. };

//The bzero() function sets the first n bytes of the area starting at s to zero
void bzero(void *s, size_t n);

//The bcopy() function copies n bytes from src to dest.
void bcopy(const void *src, void *dest, size_t n);
```

Nota:
Per le dimensioni relative ai file si considerino le grandezze come potenze di 2 e quindi in particolare:
1 MB = 1.024 kB 1 kB = 1.024 byte
Per le dimensioni relative ai tassi di trasmissione e alle ampiezze di banda si considerino le grandezze come potenze di 10 e quindi in particolare:
1 kbps = 1.000 bps 1 Mbps = 1.000.000 bps

Regolamento di esame La prova scritta, della durata di 2:30 ore, riguarda i contenuti coperti durante l'intero corso.
La consegna del compito equivale all'inizio dell'esame, il cui esito finale dipenderà dalla valutazione della prova scritta e di un esame orale da sostenere successivamente. È consentito agli studenti di non consegnare il compito scritto, nel qual caso la prova non verrà conteggiata nel numero massimo di tre prove d'esame che e' possibile sostenere in uno stesso Anno Accademico.
Durante lo svolgimento della prova valgono le regole riportate di seguito:

- non è assolutamente consentito collaborare;
- non è consentito portare libri, fotocopie, appunti;
- è consentito l'uso di una calcolatrice;
- non è assolutamente consentito tener acceso il telefonino.

Nel caso in cui una delle sopra elencate regole per lo svolgimento degli esami non venga rispettata, si procederà con il ritiro del compito e con il conseguente annullamento della prova.
NB: nella valutazione dell'elaborato si terrà pesantemente conto della chiarezza espositiva.