UNIVERSITÀ DEGLI STUDI DI PALERMO — Scuola Politecnica

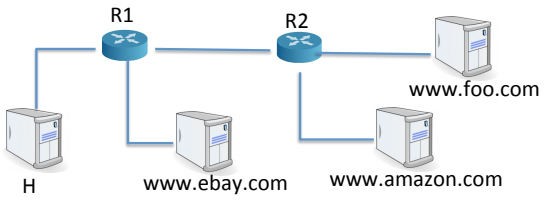
14 Luglio 2016

Reti di Calcolatori e Internet
A.A. 2015/2016

Docente: Ing. Alessandra De Paola
(Durata 2:30h)

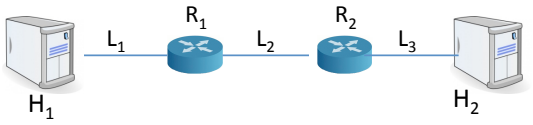
Quesito 1

Data la configurazione schematizzata in figura, si assuma che l'host H richieda una pagina web al server `www.ebay.com`, costituita da un file HTML di 12 KB e dall'immagine `www.foo.it/pics/img1.jpg` di 4 MB, e che, successivamente, lo stesso host richieda una pagina web al server `www.amazon.com` costituita da un file HTML di 10 KB e dall'immagine `www.foo.it/pics/img2.jpg` di 3 MB.
Assumendo che tutte le interazioni avvengano con HTTP 1.1 e che il server `www.foo.com` utilizzi cookies di 100 Byte, si calcoli il tempo totale necessario ad ottenere le due pagine.
Si considerino tutti i collegamenti caratterizzati da ampiezza di banda di 200 Mbps, tempo di propagazione medio pari a 0.1 ms, MTU pari a 1500 Byte e overhead di 40 Byte.



Quesito 2

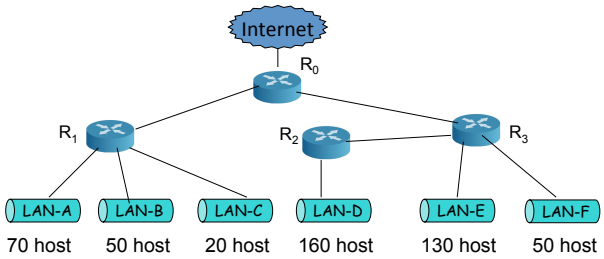
Dati due host collegati come in figura, si supponga che l'host H_1 invii all'host H_2 un file di dimensioni 4KB tramite il protocollo *GoBackN* con finestra statica W=4 MSS. Si assuma che i link abbiamo rispettivamente ampiezze di banda pari a R_1=150 Mbps, R_2=100 Mbps, R_3=150 Mbps, che ci sia RTT pari a 0,5 ms, MSS di 512 byte ed overhead trascurabile, e che i router abbiano risorse illimitate. Si determini il valore del tempo totale necessario a trasmettere il file e del throughput della trasmissione nel caso in cui il valore del timeout sia pari a 1,2 ms, in uno scenario in cui si perda l'ACK relativo al 4^o segmento.



[Continua ...]

Quesito 3

Avendo a disposizione il range di indirizzi `192.168.92.0/22` si proponga uno schema di indirizzamento per la configurazione indicata in figura che minimizzi lo spreco di indirizzi per ciascuna sottorete e il riduca il piu' possibile il numero di righe delle tabelle di inoltro. Indicare le tabelle di inoltro per tutti i router (in notazione binaria).



Quesito 4

Completare il codice fornito per realizzare il server di una semplice random chat, tramite socket TCP. Il server accetta due connessioni TCP qualsiasi, manda al primo client l'autorizzazione ad inviare un messaggio, legge il messaggio ricevuto e lo invia al secondo client. Ripete la procedura autorizzando il secondo client ad inviare un messaggio al primo e chiude entrambe le connessioni.

```c
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

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

int main(int argc, char *argv[]) {
     int sockfd, newsockfd1, newsockfd2;
     int portno, clilen;
     char buffer1[256], buffer2[256];
     struct sockaddr_in serv_addr, cli_addr;
     pid_t pid;
```

[Continua ...]

```
    if (argc < 2) {
        fprintf(stderr,"ERROR, no port provided\n");
        exit(1);
    }

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0) {
       error("ERROR opening socket");
    }

    bzero((char *) &serv_addr, sizeof(serv_addr));
    portno = atoi(argv[1]);
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(portno);


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

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

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

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

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

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

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

while(1){

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

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

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

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

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

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

  pid=fork();
```

[Continua ...]

```
        if (pid == 0) {
    ..........................................................................
   ..........................................................................
   ..........................................................................
   ..........................................................................
   ..........................................................................
   ..........................................................................
   ..........................................................................
   ..........................................................................
   ..........................................................................
   ..........................................................................
   ..........................................................................
   ..........................................................................

          return 0;
    }

     close(sockfd);

  }
  return 0;
}
```

Documentazione Programmazione Socket

```
//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. };

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

//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);

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

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

ssize_t recv(int sockfd, void *buf, size_t len, int flags); //Receive data on a socket
ssize_t send(int sockfd, const void *buf, size_t len, int flags); //Send data out over a socket

//Receive data on a socket
ssize_t recvfrom(int sockfd, void *buf, size_t len, int flags, struct sockaddr *src_addr, socklen_t *addrlen);
//Send data out over a socket
ssize_t sendto(int sockfd, const void *buf, size_t len, int flags, const struct sockaddr *dest_addr, socklen_t addrlen);

ssize_t read (int fd, void *buf, size_t count); //Read data from a stream socket
ssize_t write (int fd, const void *buf, size_t count); //Write data on a stream socket
```

Utility Functions

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

void bcopy(const void *src, void *dest, size_t n); // *Copy n bytes from src to dest.*

int strcmp(const char *s1, const char *s2); // *Compare the two strings s1 and s2.*
int strncmp(const char *s1, const char *s2, size_t n); //*Compare the first n byte of s1 and s2.*

Note:
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 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.
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.