

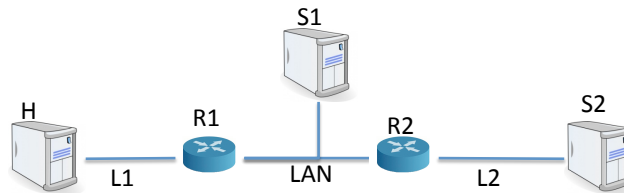


CORSO DI RETI DI CALCOLATORI E INTERNET A.A. 2013/2014

Docente: Ing. Alessandra De Paola
23 Giugno 2014

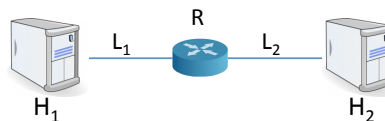
Quesito 1

Data la configurazione schematizzata in figura, si assuma che l'host H scarichi dal web server $S1$ una pagina web di dimensione 10KB, che contiene un video di 3MB ospitato sul server web $S2$. Si assuma inoltre che l'accesso al video richieda l'utilizzo dei cookie e che la dimensione di un cookie sia di 500 Byte, e che sia l'host che i server supportino il protocollo HTTP/1.1. Si assuma che la LAN sia caratterizzata da ampiezza di banda $R_{LAN} = 100$ Mbps, mentre i due link $L1$ e $L2$ siano caratterizzati da ampiezza di banda pari a $R_L = 50$ Mbps; si assuma inoltre che per la LAN e per entrambi i link il tempo di propagazione medio sia pari a 2ms, si abbia una MTU pari a 1500 Byte e un overhead complessivo pari a 48 Byte. Si calcoli il tempo necessario al trasferimento.



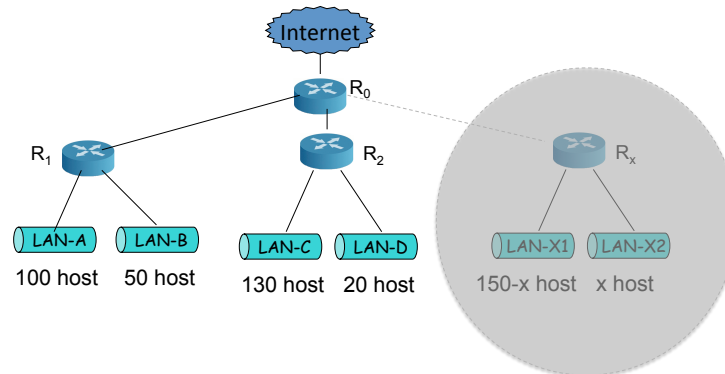
Quesito 2

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 un protocollo di tipo *GoBackN*, con finestra di spedizione statica $W = 5$. Si assuma che il RTT tra i due host sia pari a 1 ms, che le ampiezze di banda per i due link siano rispettivamente $R_1 = 50$ Mbps e $R_2 = 100$ Mbps, e che si abbia un MSS=512 Byte e un overhead di pacchetto pari a 40 Byte. Assumendo che si perda l'ack del 5 pacchetto, indicare quantitativamente, attraverso un opportuno grafico, come varia il *throughput* al variare del valore di timeout.



Quesito 3

Avendo a disposizione il range di indirizzi 153.192.144.0/21 si proponga uno schema di indirizzamento per la configurazione indicata in figura che minimizzi lo spreco di indirizzi per ciascuna sottorete e minimizzi il numero di entry nella tabella di inoltro del router R_0 . Si calcoli inoltre quante ulteriori sottoreti di complessivi 150 host (analoghe a quelle sotto i router R_1 e R_2) possono essere aggiunte sotto il router R_0 , al variare del valore di x .



Quesito 4

Completare il codice fornito per realizzare il server di un'applicazione Echo tramite protocollo UDP. L'applicazione prevede che il client legga un testo dallo standard input e la invii al server; il server legge le linee di testo dalla socket e le rimanda al client; il client legge la risposta ottenuta e la manda sullo standard output.

```
#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;
    int server_port = 6543;
    struct sockaddr_in servaddr, cliaddr;

    if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 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);

    .....

    .....

    my_echo(sockfd, (struct sockaddr *) &cliaddr, sizeof(cliaddr));
}
```



```
void my-echo(int sockfd, struct sockaddr *cliaddr, socklen_t clilen)
{
    int n;
    socklen_t len;
    char msg[MAXLINE];
    while(1) {
        len = clilen;

        .....

        .....

        .....

        .....

        .....

        .....
    }
}
```

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 s, void *buf, size_t len, int flags, struct sockaddr *from, socklen_t *fromlen); // for UDP 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 s, const void *buf, size_t len, int flags, const struct sockaddr *to, socklen_t tolen); // for UDP 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 è 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.