Reti di Calcolatori - Laboratorio

Lezione 7

Gennaro Oliva

Opzioni di socket

- Ogni socket aperto ha un insieme di opzioni associate che ne determinano il comportamento
- Distinguiamo due tipi:
  - opzioni binarie che abilitano o disabilitano determinate caratteristiche
  - opzioni con valore (interi, struct, ...)
- Ogni opzione ha un valore di default che in generale possono essere modificati
- Esistono opzioni che non sono modificabili

Opzioni di socket

- Per conoscere e modificare i valori delle opzioni di un socket aperto su possono utilizzare le funzioni:

```
int getsockopt(int sockfd, int level, int optname,
               void *optval, socklen_t *optlen);

int setsockopt(int sockfd, int level, int optname,
               const void *optval, socklen_t optlen);
```

- Entrambi resitutiscono
  - 0 in caso di successo
  - 1 in caso di errore

Argomenti di {set,get}sockopt

int getsockopt(int sockfd, int level, int optname,
void *optval, socklen_t *optlen);

- sockfd è un descrittore di socket aperto
- level specifica il livello dell'opzione
- optname è il nome dell'opzione
- optval è la variabile su cui verrà memorizzato il valore corrente dell'opzione
- optlen specifica la dimensione di optval in input e quella del valore dell'opzione in output

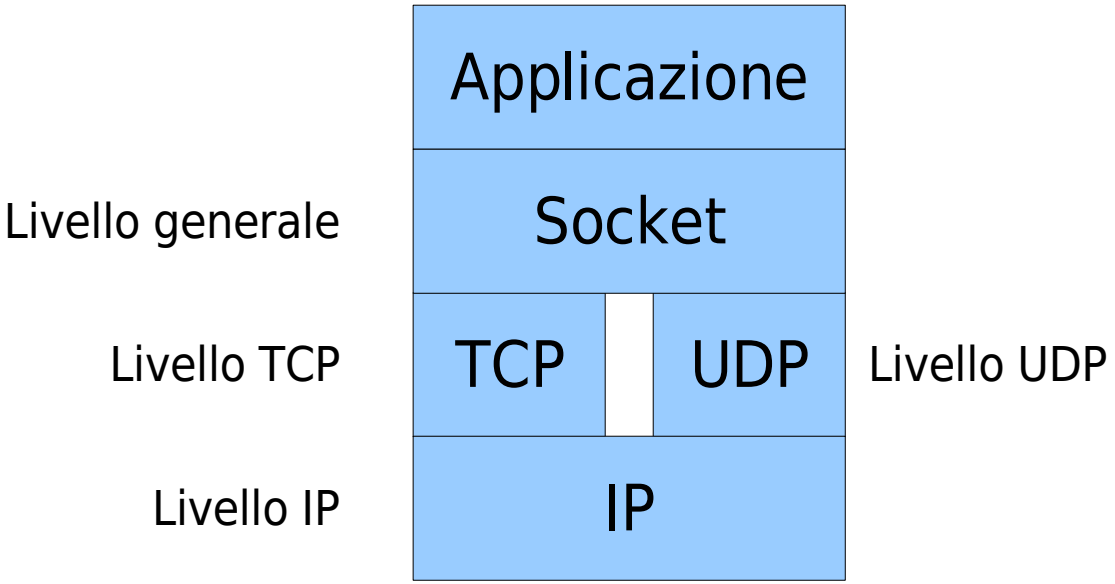
Argomenti di {set,get}sockopt

```
int setsockopt(int sockfd, int level, int optname,
               const void *optval, socklen_t optlen);
```

- sockfd è un descrittore di socket aperto
- level specifica il livello dell'opzione
- optname è il nome dell'opzione
- optval è la variabile che continene il valore dell'opzione da memorizzare
- optlen specifica la dimensione di optval

Livello di un opzione

- Il livello di un'opzione specifica il sottosistema a cui è destinata l'opzione
- Il sottosistema generale delle socket
- Il sottosistema specifico di un protocolol (IPv4, IPv6, TCP, o SCTP, ecc.)



Opzioni binarie

- Accettano come valore una variabile intera:
  - 1 per attivare l'opzione (on)
  - 0 per disattivarla (off)
- Esempio:

```
#include <sys/types.h>
#include <sys/socket>
...
    int     sockfd;
    int on = 1;
 ...
    setsockopt(sockfd, SOL_SOCKET, SO_BROADCAST, &on, sizeof(on));
```

Opzioni supportate

- Non tutte le implementazioni supportano tutte tutte le opzioni
- Per verificare se l'opzione è supportata si possono utilizzare le direttive al preprocessore per la compilazione condizionata

```
#ifdef SO_DEBUG
  setsockopt(sockfd, SOL_SOCKET, SO_DEBUG, on, sizeof(on));
#else
  fprintf(stderr,”SO_DEBUG option not supported\n”);
#endif
```

Opzioni Comuni del livello generico SOL_SOCKET

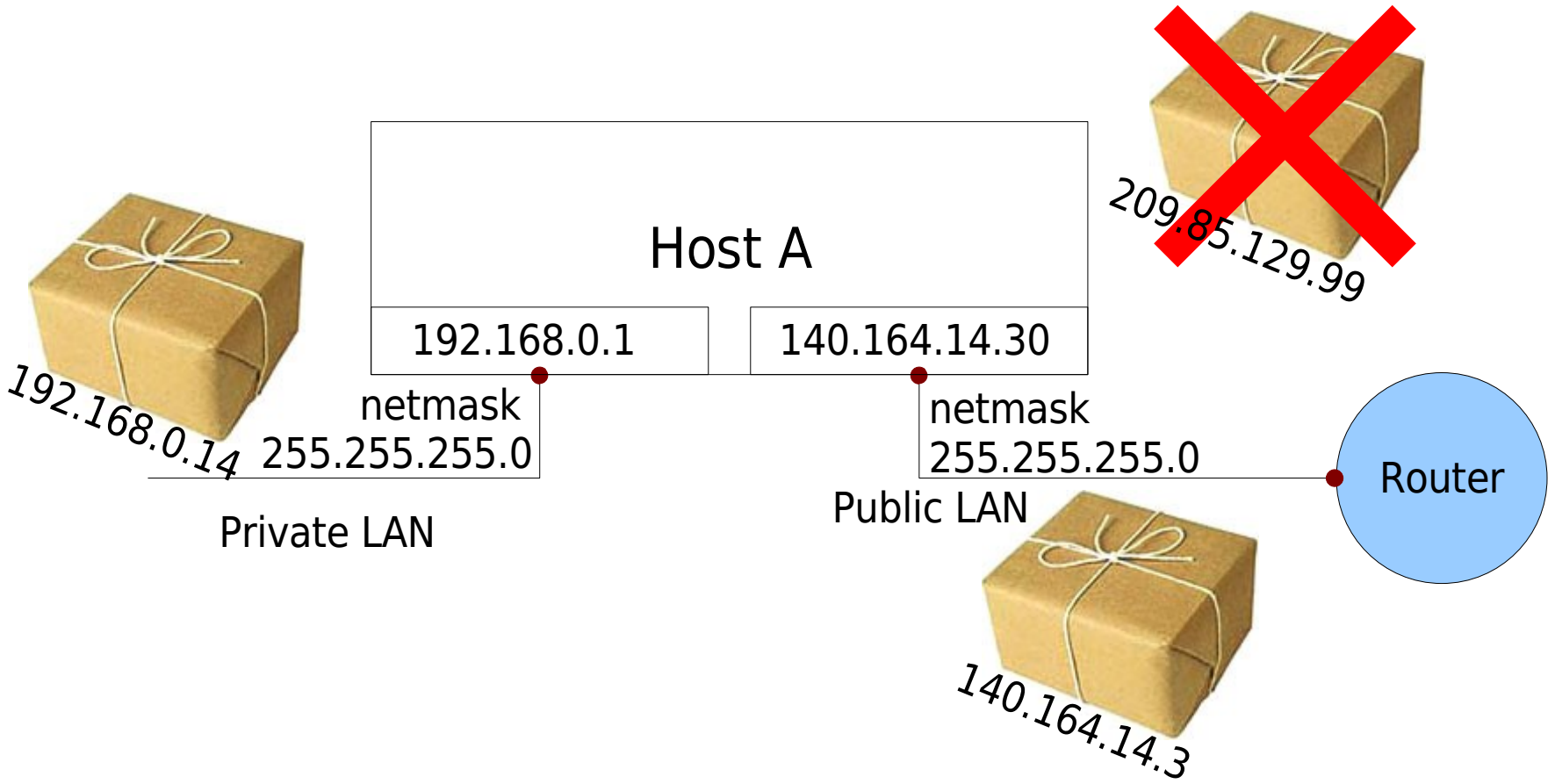
SO_BROADCAST

- SO_BROADCAST è un opzione binaria che abilita il socket ad inviare datagrammi all'indirizzo di broadcast
- Il broadcast deve essere supportato anche a livello fisico (reti ethernet, token ring, ecc.)
- Questa opzione è disabilitata per default in modo da impedire l'utilizzo del broadcast da parte di applicazioni che non lo prevedono
  - Applicazioni che accettano l'indirizzo del server come argomento

SO_DONTROUTE

- SO_DONTROUTE è un opzione binaria che segnala al kernel di ignorare la tabella di routing e cercare un'interfaccia di rete che stia sulla stessa rete del destinatario
- L'indirizzo del mittente condivide con l'indirizzo del destinatario la parte network
- Utilizzando questa opzione è possibile comunicare soltanto sulle reti locali a cui si è connessi

SO_DONTROUTE



SO_ERROR

- Quando si verifica un errore, un codice identificativo intero viene memorizzato nella variabile del kernel so_error
- La notifica dell'errore avviene in 2 modi:
  - in una chiamata a select in lettura o in scrittura, ritorna con il descrittore impostato come pronto
  - con I/O controllato da segnali, viene generato SIGIO
- Il processo può accedere al valore di so_error esclusivamente in lettura utilizzando l'opzione SO_ERROR

SO_KEEPALIVE

- SO_KEEPALIVE è un'opzione binaria che attiva l'invio di un messaggio di controllo nel caso in cui non ci sia stato traffico per 2 ore su un socket connesso (TCP)
- Il messaggio di controllo può avere tre possibili risposte:
  - **ACK di risposta**: la connessione è ancora attiva
  - **RST di risposta**: la connessione era stata chiusa, viene generato l’errore ECONNRESET
  - **Nessuna risposta**: vengono spediti 8 messaggi di controllo ogni 75 secondi

SO_KEEPALIVE

- Se nessun messaggio di verifica ottiene risposta l'errore del socket e' ETIMEDOUT
- Se si riceve una risposta d'errore (ad esempio host unreachable) l'errore relativo al socket viene impostato di conseguenza (nell'esempio EHOSTUNREACH)
- Questo accade generalmente quando ci sono problemi di rete o in caso di crash dell'host remoto

Close

- Per terminare una connessione TCP si utilizza la system call close
- Una volta invocato Il descrittore del socket non e' piu' utilizzabile dal processo per operazioni di lettura o scrittura
- Il sottosistema di rete invia i dati in coda e successivamente chiude la connessione

SO_LINGER

- SO_LINGER permette di controllare le modalita' di chiusura di una connessione
- Il comportamento di default della close è quello di ritornare immediatemente nel processo chiamante mentre il sottosistema di rete si occupa di inviare dati residui nel buffer
- L'opzione richiede una struct di tipo liger come argomento cosi' definita

```
struct linger {    int   l_onoff;        /* 0=off, nonzero=on */
  int   l_linger;       /* linger time, POSIX specifies units as seconds */
};
```

SO_LINGER

```
struct linger {    int   l_onoff;        /* 0=off, nonzero=on */
  int   l_linger;       /* linger time, POSIX specifies units as seconds */
};
```

- l_onoff=0 comportamento di default
- l_onoff≠0, l_linger =0 all'invocazione di close la connessione viene interrotta: i dati in coda non vengono inviati e la connessione viene conclusa con un messaggio di reset (RST)
  - Evita il TIME_WAIT, ma può causare la reincarnazione della connessone

SO_LINGER

- l_onoff≠0, l_linger≠0 la close non ritorna immediatamente ma blocca l'esecuzione del processo in attesa che i dati in coda siano inviati e gli restituisce il controllo quando:
  - tutti i dati sono stati spediti
  - sono trascorsi l_linger secondi
- I dati non vengono inviati entro lo scadere del tempo limite vengono cancellati e la close restituisce EWOULDBLOCK
- In questi casi è opportuno verificare il valore di ritorno

Buffer di ricezione TCP

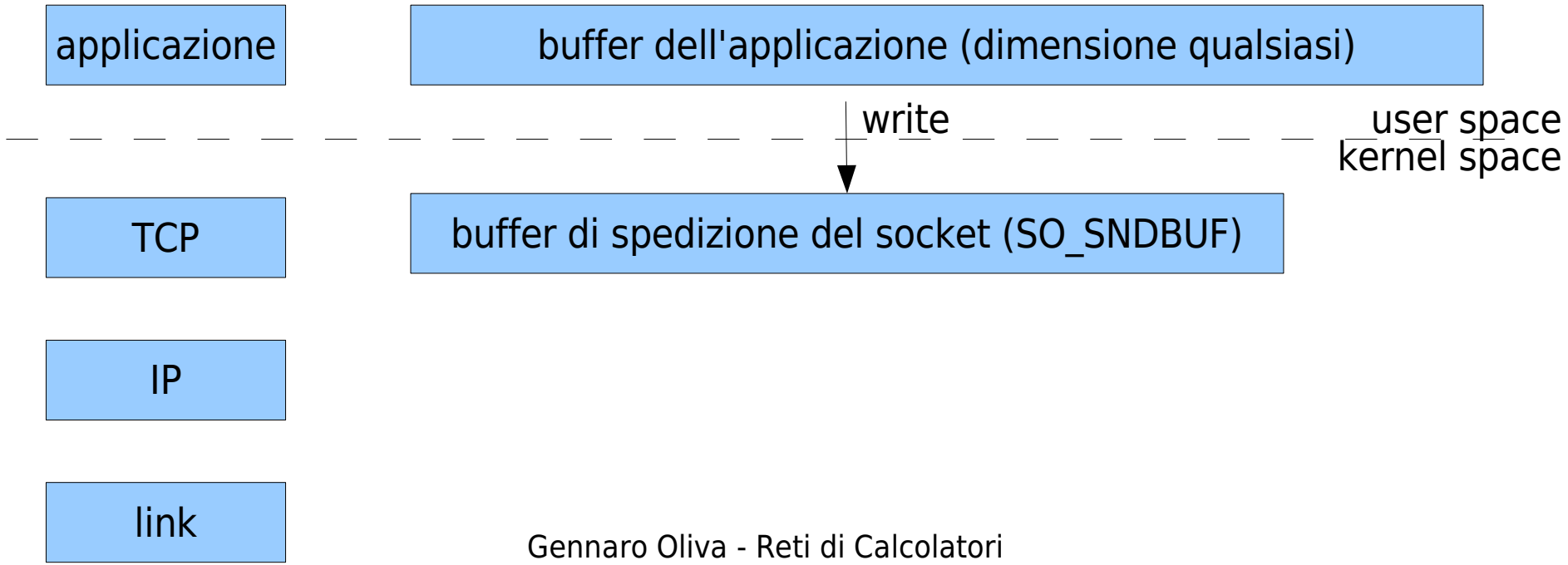
- Ogni socket ha un buffer di ricezione per memorizzare i dati ricevuti fino a quando non vengono letti dall'applicazione
- Con TCP lo spazio disponibile nel buffer di ricezione del socket viene notificato alla controparte (TCP window)
- L'host che invia dati non sfora le dimensioni del buffer e anche se lo facesse i dati in eccesso sarebbero ingorati dall'host che riceve
- Tutto questo avviene a livello TCP (sistema) e non al livello dell'applicazione

Buffer di ricezione UDP

- Con UDP l'host che invia non conosce la disponibiltà di spazio del buffer di ricezione
- I datagrammi che non entrano nello spazio disponibile del buffer di ricezione vengono scartati e quindi persi
- Un host che spedisce pacchetti velocemente può saturare uno che li legge lentamente

Buffer di spedizione TCP

- Quando un'applicazione scrive (write) su un socket TCP il kernel copia i dati dal buffer dell'applicazione al buffer di spedizione del socket

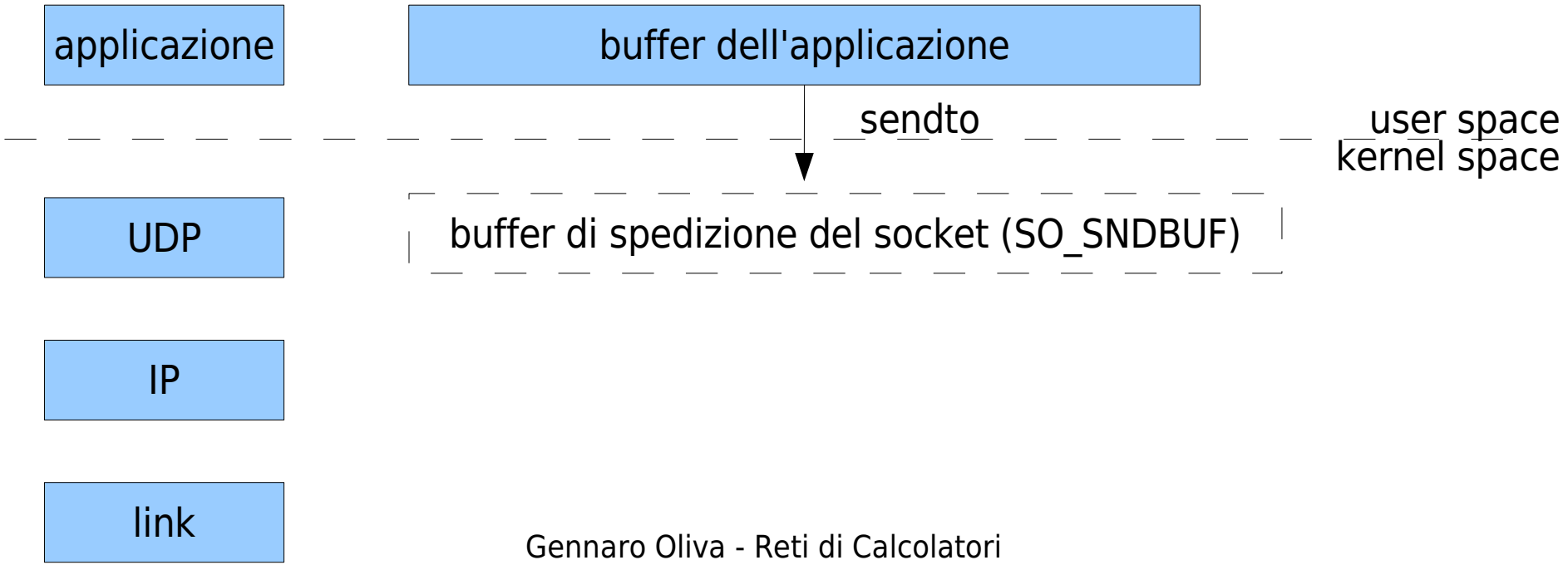


Buffer di spedizione TCP

- Se il buffer del socket non e' sufficientemente capiente (è più piccolo del buffer dell'applicazione o ci sono già altri dati) il processo viene bloccato (sleep) a meno che il socket non sia non-bloccante
- Il kernel restituisce il controllo al programma quando tutti i byte sono stati copiati
- L'esecuzione senza errori di una write significa soltanto che i dati sono stati copiati nel buffer TCP e non che la controparte li abbia ricevuti

Dimensione del buffer di spedizione UDP

- I socket UDP non hanno un buffer di spedizione ma un limite della dimensione massima che un datagramma può avere



Dimensione del buffer di spedizione UDP

- Se il buffer dell'appilicazione eccede la dimensione del buffer del socket UDP viene generato l'errore EMSGSIZE
- Essendo inaffidabile l'UDP non deve conservare la copia dei dati dell'applicazione e quindi non necessita di un buffer di spedizione
- L'esecuzione senza errori di una sendto significa soltanto che i dati sono stati copiati nella coda di spedizione del link layer, in caso contrario viene generato l'errore ENOBUFS

SO_RCVBUF e SO_SNDBUF

- Le opzioni SO_RCVBUF e SO_SNDBUF consentono di modificare il buffer di ricezione e spedizione rispettivamente
- Accettano come opzione un intero che specifica il numero di byte nel buffer

Descrittori pronti in lettura

- Il descrittore di un socket e' “pronto” in lettura quando si verifica una delle 4 condizioni:
  1) Il numero di byte nel buffer di ricezione del socket e' uguale o maggiore di un valore massimo chiamato low-water mark “LWM” per il buffer di ricezione
     - il valore di LWM per un determinato socket puo' essere impostato dal programmatore
     - Il valore di default per UDP e TCP e' 1
     - In questo caso l'accesso in lettura al descrittore non bloccherà l'esecuzione del processo

Dalla lezione sull'I/O multiplex pag. 34

low-water mark

- Ad ogni socket sono associati un low-water mark (LWM) di ricezione e un LWM di spedizione
- Questi valori vengono utilizzati dalla funzione select descritta nella lezione sul multiplex
- Il LWM di ricezione è la quantità di dati che devono essere nel buffer di ricezione del socket affinchè il socket sia considerato “leggibile”
- Per default vale 1 sia per TCP che per UDP

low-water mark

- Il LVM di spedizione è lo spazio disponibile necessario nel buffer di spedizione affichè il socket sia considerato “scrivibile” (solitamente 2048 per TCP)
- UDP non ha un buffer di spedizione pertanto non si può parlare di spazio disponibile
- Se la dimensione del buffer di spedizione (intesa come limite della dimensione massima che un datagramma può avere) è maggiore del LWM di spedizione UDP il socket è sempre “scrivibile”

SO_RCVLOWAT e SO_SNDLOWAT

- Le opzioni SO_RCVLOWAT e SO_SNDLOWAT consentono di modificare questi valori
- Accettano come opzione un intero che specifica il numero di byte del LWM
- In Linux entrambi i valori per default sono impostati a 1 e SO_SNDLOWAT non è modificabile

Inffidabilita'

- La comunicazione tra client e server mediante UDP non e' affidabile:
- Se si perde il pacchetto della richiesta o della risposta l'applicazione client resta bloccata in ricezione
- E' possibile impostare un timeout per la recvfrom
- Non e' possibile sapere quale dei due datagrammi e' andato perso

Dalla lezione sull'UDP pag. 25

SO_RCVTIMEO e SO_SNDTIMEO

- SO_RCVTIMEO e SO_SNDTIMEO consentono di specificare un timeout per ricezione e spedizione
- Il timeout specifica dopo quanto tempo di inattività devono ritornare le funzioni di lettura e scrittura su un socket
- SO_RCVTIMEO agisce sulle funzioni di lettura read, recvfrom, ecc.
- SO_SNDTIMEO sulle funzioni di scrittura: write, sendto, ecc.

SO_RCVTIMEO e SO_SNDTIMEO

- L'argomento è un struct di tipo timeval (la stessa usata per select) che permette di esprimere il timeout in secondi e microsecondi

```
struct timeval {
long tv_sec; /* numero di secondi */
long tv_usec; /* numero di microsecondi */
};
```

- Il timeout viene disabilitato specificando 0 secondi e 0 microsecondi
- Il timeout è disabilitato per default

SO_TYPE

- SO_TYPE restituisce il tipo di socket
- Il suo argomento è un intero il cui valore può essere SOCK_STREAM, SOCK_DGRAM, ecc
- Questa opzione viene tipicamente utilizzata dai processi che ereditano un socket
- Il suo valore può essere letto ma non scritto: non si può cambiare il tipo ad un socket esistente

Livello IPv4 (IPPROTO_IP)

- L'opzione IP_TTL permette di specificare il Time-To-Live del pacchetto
- Il campo TTL è un valore di 8-bit che viene decrementato di un'unità ogni volta che un router instrada il pacchetto
- Il datagramma viene scartato dal primo router che decrementa il suo valore a 0
- Questo limita la vita di un pacchetto a 255 hops (instradamenti)
- Un valore comunemente usato è 64

Esercizi

- Scrivere un programma che mostri il valore di default di tutte le opzioni presentate in questa lezione
- Modificare il client daytime in modo che effettui la richiesta sull'indirizzo di broadcast ed aspetti un numero indeterminato di risposte per un massimo di 2 secondi