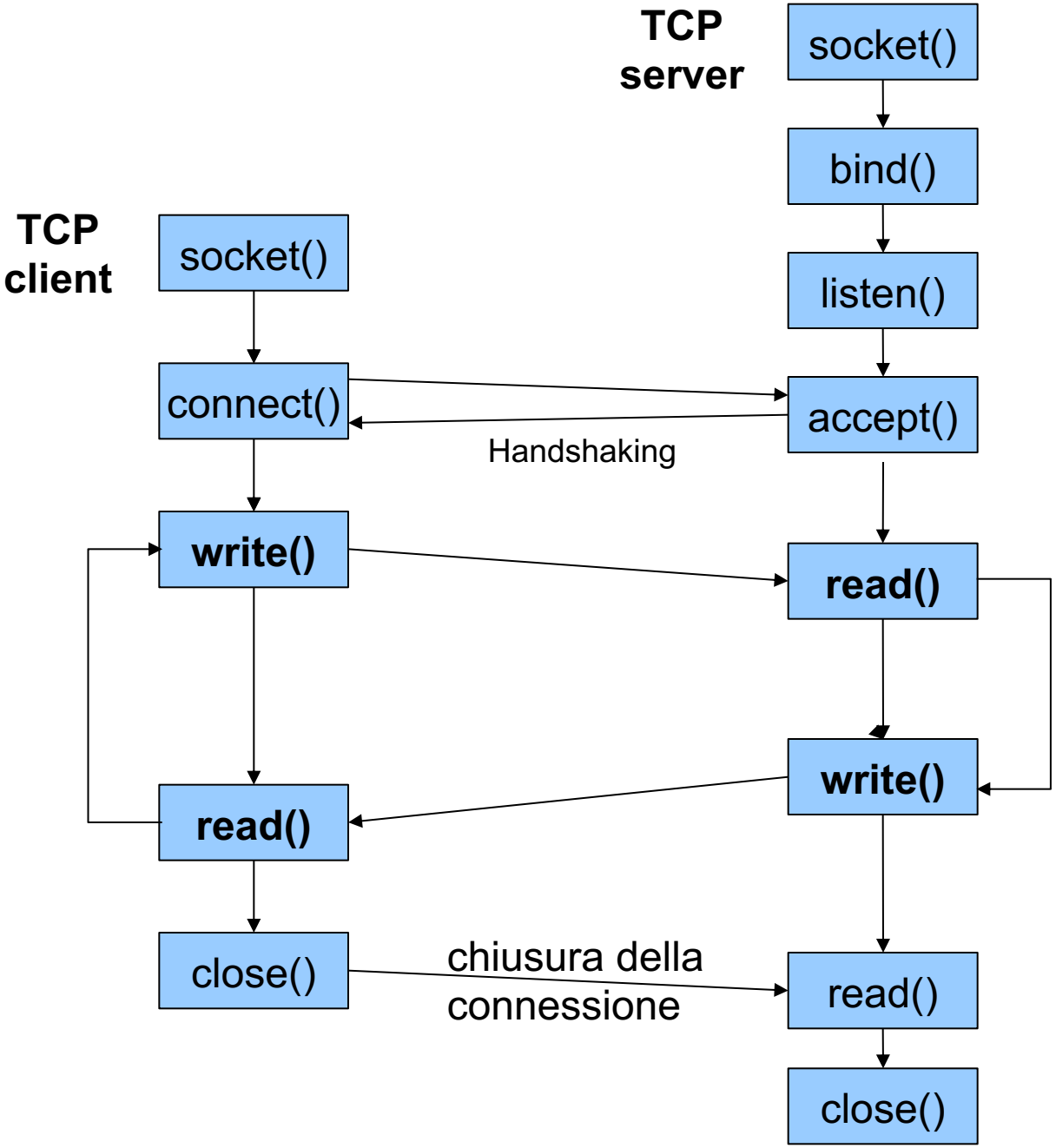
Laboratorio di Reti di Calcolatori

Lezione 4


TCP server
socket()
bind()
listen()
accept()
TCP client
socket()
connect()
Handshaking
write()
read()
write()
read()
close()
chiusura della connessione
read()
close()

Write

- ssize_t **write**(int fd, const void *buf, size_t count);
- si usa per scrivere su un socket
- write restituisce il numero di byte scritti
- Può accadere che si scrivano meno bytes di quelli richiesti
- Sono necessarie chiamate successive
- FullWrite scrive esattamente count byte s iterando opportunamente le scritture

FullWrite

```
#include <unistd.h>
ssize_t FullWrite(int fd, const void *buf, size_t count)
{
    size_t nleft;
    ssize_t nwritten;
    nleft = count;
    …
    return (nleft);
}
```

```
 repeat until no left
        write(fd, buf, nleft)
            if errno == EINTR  /* if interrupted by system call */
                repeat the loop
            else
                exit with error
         set left to write
         set pointer
```

FullWrite

```
#include <unistd.h>
ssize_t FullWrite(int fd, const void *buf, size_t count)
{
    size_t nleft;
    ssize_t nwritten;
    nleft = count;
    while (nleft > 0) {             /* repeat until no left */
        if ( (nwritten = write(fd, buf, nleft)) < 0) {
            if (errno == EINTR) {   /* if interrupted by system call */
                continue;           /* repeat the loop */
            } else {
                exit(nwritten);   /* otherwise exit with error */
            }
        }
        nleft -= nwritten;          /* set left to write */
        buf += nwritten;              /* set pointer */
    }
    return (nleft);
}
```

Read

- ssize_t **read**(int fd, void *buf, size_t count);
- Si usa per leggere da un socket
- La read *blocca* l'esecuzione qualora non ci siano dati da leggere ed il processo resta in attesa di dati
- E' normale ottenere meno bytes di quelli richiesti
- Ottenere 0 bytes significa che il socket e' vuoto ed e' stato chiuso

FullRead

```
#include <unistd.h>
ssize_t FullRead(int fd, void *buf, size_t count)
{
  size_t nleft;
  ssize_t nread;
  nleft = count;
  ….
  return (nleft);
 }
```

```
   repeat until no left
       read(fd, buf, nleft)
           if errno == EINTR    /* if interrupted by system call */
                    repeat the loop
           else
                    exit with error
        if EOF
            break loop here
        set left to read
        set pointer
```

FullRead

```
#include <unistd.h>
ssize_t FullRead(int fd, void *buf, size_t count)
{
    size_t nleft;
    ssize_t nread;
    nleft = count;
    while (nleft > 0) {             /* repeat until no left */
        if ( (nread = read(fd, buf, nleft)) < 0) {
            if (errno == EINTR) {   /* if interrupted by system call */
                continue;           /* repeat the loop */
            } else {
                exit(nread);        /* otherwise exit */
            }
        } else if (nread == 0) {    /* EOF */
            break;                  /* break loop here */
        }
        nleft -= nread;             /* set left to read */
        buf +=nread;                /* set pointer */
    }
    buf=0;
    return (nleft);
}
```

Esercizi

- Completare il client ed il server
  - clientFullRead_incomplete.c
  - serverFullWrite_incomplete.c