CS4254

Computer Network Architecture and Programming

Dr. Ayman A. Abdel-Hamid

Computer Science Department

Virginia Tech

Socket Options

Outline

•Socket Options (Chapter 7)

➢ Introduction

➢Checking for Options and default values

➢Some Generic Socket Options

➢TCP Socket Options

Getting and Setting Options 1/2

•Various attributes that are used to determine the behavior of sockets

```
#include <sys/socket.h>
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);
```

Both return 0 if OK, -1 on error

•*sockfd*: an open socket descriptor

•*level*: code in the system that interprets the option (general socket code, or protocol-specific code) (SOL_SOCKET, IPPROTO_IP, IPPROTO_IPv6, IPPROTO_TCP are examples)

•*optname*: see page 193-figure 7.1, and page 194-figure 7.2

Getting and Setting Options 2/2

Some socket options examples (see table on page 193 and 194)

•*Socket Level*

➢SO_SNDBUF, SO_RCVBUF, SO_KEEPALIVE, SO_BROADCAST, SO_REUSEADDR, SO_RESUEPORT

•*IP Level*

➢IP_TTL, IPMULTICAST_IF, IPMUTLICAST_TTL, IP_MULTICAST_LOOP, IP_ADD_MEMBERSHIP, IP_DROP_MEMBERSHIP

•*TCP Level*

➢TCP_KEEPALIVE, TCP_MAXSEG, TCP_NODELAY

Checking for socket Options

- Not all implementations support all socket options
- Source code in **sockopt/checkopts.c**
- Declares 4 different functions to handle the value for a given socket option
- SO_REUSEPORT can be undefined
  - ➢Have to surround with #ifdef
- SO_USELOOPBACK can be undefined
  - ➢Have to surround with #ifdef
  - ➢Need to change the source code for our lab machines

Socket States

- The following socket options are inherited by a connected socket from the listening socket

  SO_DEBUG, SO_DONTROUTE, SO_KEEPALIVE, SO_LINGER, SO_OOBINLINE, SO_RCVBUF, SO_RCVLOWAT, SO_SNDBUF, SO_SNDLOWAT, TCP_MAXSEG, and TCP_NODELAY

- To ensure one of the previous option is set for a connected socket, when 3WHS completes
  - Set the option for the listening socket

Some Generic Socket Options 1/13

- SO_BROADCAST
  - ➢Enable or disable the ability of the process to send broadcast messages (only datagram socket : Ethernet, Token ring..)
- SO_DEBUG
  - ➢Kernel keep track of detailed information about all packets sent or received by TCP (only supported by TCP)
- SO_ERROR
  - ➢When error occurs on a socket, the protocol module in a BSD, kernel sets a variable named **so_error** for that socket (pending error)
  - ➢Process can obtain the value of **so_error** by fetching the SO_ERROR socket option
  - ➢Socket option can be fetched but not set

Some Generic Socket Options 2/13

- SO_KEEPALIVE
  - ➢When set for a TCP socket, and no data has been exchanged in either direction for *two hours*
  - ➢TCP automatically sends a keep-alive probe to the peer
  - ➢Peer must respond
    - ✓Peer responds with expected ACK → OK
    - ✓Peer responds with an RST → peer host has crashed and rebooted. Socket pending error is set to ECONNRESET and socket closed
    - ✓No repsonse from peer
      - ❑BSD TCPs send 8 additional probes, 75 seconds apart
      - ❑Give up if no response within 11 minutes and 15 seconds after first probe
      - ❑Socket pending error set to ETIMEDOUT (or set to ICMP error)
  - ➢See Figure 7.6

Some Generic Socket Options 3/13

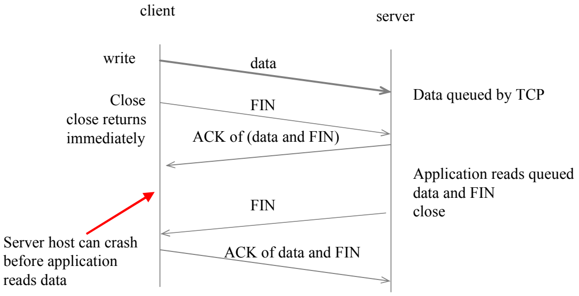
- SO_LINGER
  - ➢specify how the ***close*** function operates for a connection-oriented protocol (default: close returns immediately)

```
struct linger{
    int l_onoff;     /* 0 = off, nonzero = on */
    int l_linger;    /*linger time : seconds*/
};
```

- ➢***l_onoff*** = 0 : turn off , ***l_linger*** is ignored
- ➢***l_onoff*** = nonzero and ***l_linger*** is 0:TCP aborts the connection, discard any remaining data in send buffer.
- ➢ ***l_onoff*** = nonzero and ***l_linger*** is nonzero
  - ✓process waits until *remaining data sent and ACKed*, or until linger time expired
  - ✓If socket has been set non-blocking, it will not wait for the ***close*** to complete, even if linger time is nonzero

Some Generic Socket Options 4/13

- SO_LINGER



client

server

write

data

Data queued by TCP

Close
close returns
immediately

FIN

ACK of (data and FIN)

Application reads queued
data and FIN
close

FIN

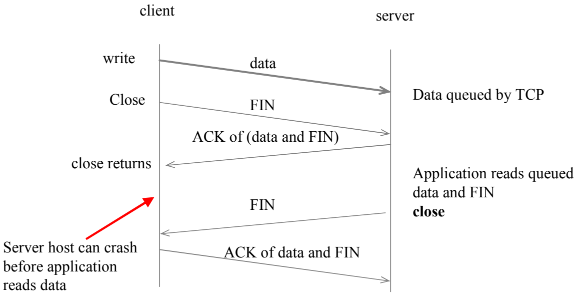
Server host can crash
before application
reads data

ACK of data and FIN

Default operation of close:it returns immediately

Some Generic Socket Options 5/13

- SO_LINGER



client

server

write

data

Data queued by TCP

Close

FIN

ACK of (data and FIN)

close returns

Application reads queued
data and FIN
close

FIN

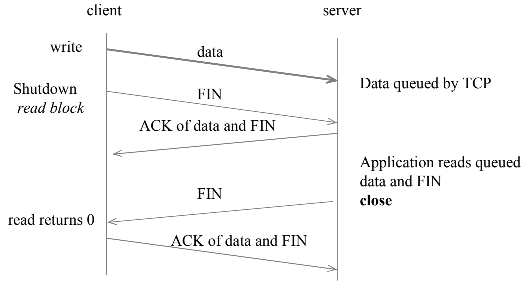
Server host can crash
before application
reads data

ACK of data and FIN

Close with SO_LINGER socket option set and l_linger a positive value

Some Generic Socket Options 6/13

- SO_LINGER (making sure receiver reads the data)



client

server

write

data

Data queued by TCP

Shutdown
read block

FIN

ACK of data and FIN

Application reads queued
data and FIN
close

FIN

read returns 0

ACK of data and FIN

Using shutdown (with 2nd argument SHUT_WR) to know that peer has received our data

Some Generic Socket Options 7/13

•SO_LINGER (making sure receiver reads the data → Application-level ACK)

➢Please see Figure 7.12 for a summary of ***shutdown*** and **SO_LINGER** scenarios

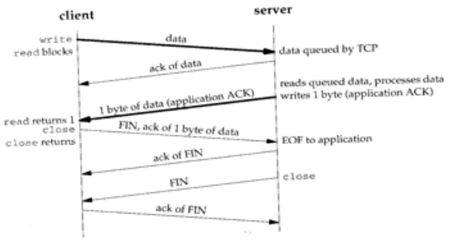


Figure 7.9 Application ACK.

Some Generic Socket Options 8/13

•SO_RCVBUF and SO_SNDBUF

➢Change the default send-buffer, receive-buffer sizes

➢Default TCP send and receive buffer size

✓Older BSD implementations → 4,096 bytes

✓Newer → 8,192-61,440 bytes

➢Default UDP buffer size

✓send 9,000bytes, receive 40,000 bytes

➢SO_RCVBUF option must be set before connection is established (calling **connect** for a client)

➢TCP socket buffer size should be at least 4 times the MSS

Some Generic Socket Options 9/13

•SO_RCVLOWAT and SO_SNDLOWAT

➢Every socket has a receive low-water mark and send low-water mark (used by **select** function)

➢*Receive low-water mark*

✓Amount of data that must be in the socket receive buffer for **select** to return "readable"

✓Default receive low-water mark : 1 for TCP and UDP

➢*Send low-water mark*

✓Amount of available space that must exist in the socket send buffer for **select** to return "writable"

✓Default send low-water mark : 2048 for TCP

✓UDP send buffer never changes (UDP does not keep a copy of datagram sent by application → see Figure 2.16 in section 2.11)

Some Generic Socket Options 10/13

•SO_RCVTIMEO and SO_SNDTIMEO

➢Allows us to place a timeout on socket receives and sends.

➢By default disabled

➢Argument is a pointer to a **timeval** structure (same as **select**)

➢Later, disable a timeout by setting its value to 0 (seconds and microseconds)

➢See Figure 14.5 (source code is in **advio/dgclitimeo2.c**)

Some Generic Socket Options 11/13

•SO_RCVTIMEO and SO_SNDTIMEO

```
struct timeval  tv;   tv.tv_sec = 5;   tv.tv_usec = 0;

Setsockopt (sockfd, SOL_SOCKET, SO_RCVTIMEO, &tv,
sizeof(tv));

n = recvfrom (sockfd, recvline, MAXLINE, 0, NULL, NULL);
        if (n < 0) {
                if (errno == EWOULDBLOCK) {
                        fprintf (stderr, "socket timeout\n");
                        continue;
                } else
                        err_sys ("recvfrom error");
        }
```

Some Generic Socket Options 12/13

•SO_REUSEADDR and SO_REUSEPORT

- ➢Allow a listening server to start and bind its well known port even if previously established connections exist that use this port as their local port
- ➢Possible scenario
  - ✓Listening server started
  - ✓connection accepted
  - ✓a child process is spawned
  - ✓listening server terminates (child is still there)
  - ✓listening server is restarted
- ➢Call to **bind** will fail because listening server is trying to bind a port that is part of an existing connection

Some Generic Socket Options 13/13

•SO_REUSEADDR and SO_REUSEPORT

- ➢Allow multiple instance of the same server to be started on the same port, as long as each instance binds a different local IP address
  - ✓Common for a site hosting multiple HTTP servers while using IP alias technique
  - ✓TCP does not allow *completely duplicate bindings* across multiple *servers* (same IP address and port)
  - ✓What about TCP clients? (see exercise 7.4)
- ➢Allow a single process to bind the same port to multiple sockets, as long as each bind specifies a different local IP address
- ➢Allow completely duplicate bindings : multicasting
- ➢4.4 BSD introduced SO_REUSEPORT socket option

TCP Socket Options 1/2

•SO_MAXSEG

- ➢Set or get the MSS for a TCP connection
- ➢Often is the MSS announced by the other end with its SYN
- ➢MSS can change during the lifetime of the connection if TCP supports path MTU discovery
- ➢Setting the socket option is not available on all systems
- ➢4.4BSD limits the application to decreasing the value

TCP Socket Options 2/2

- SO_NODELAY
  - ➢If set, disables TCP’s Nagle Algorithm (by default enabled)
  - ➢Nagle algorithm aims to reduce the number of small packets on a WAN
    - ✓If a given connection has outstanding data, then no small packets will be sent on the connection (small means smaller than the MSS)
  - ➢Common generators of small packets are Rlogin and Telnet clients (normally send each keystroke as a separate packet)
    - ✓Might be OK on a LAN, but problematic on a WAN because of RTT