

SWE 545

Distributed Systems Programming



Advanced Socket Programming

Outline

- Socket Options
- POSIX name/address conversion
- Out-of-Band Data



Spring'09

SWE 545
Distributed Systems Programming

2

Socket Options

- Various attributes that are used to determine the behavior of sockets
- Setting options tells the OS/Protocol Stack the behavior we want
- Support for generic options (apply to all sockets) and protocol specific options

Option Types

- Many socket options are Boolean flags indicating whether some feature is enabled (1) or disabled (0)
- Other options are associated with more complex types including `int`, `timeval`, `in_addr`, `sockaddr`, etc.



Spring'09

SWE 545
Distributed Systems Programming

3



Spring'09

SWE 545
Distributed Systems Programming

4

Read-Only Socket Options

- Some options are readable only (we can't set the value)



Spring'09

SWE 545
Distributed Systems Programming

5

Setting and Getting Option Values

- `getsockopt ()` gets the current value of a socket option
- `setsockopt ()` is used to set the value of a socket option

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



Spring'09

SWE 545
Distributed Systems Programming

6

getsockopt()

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

`level` specifies whether the option is a general option or a protocol specific option (what level of code should interpret the option)



Spring'09

SWE 545  
Distributed Systems Programming

7

## setsockopt()

---

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



Spring'09

SWE 545
Distributed Systems Programming

8

General Options

- Protocol independent options
- Handled by the generic socket system code
- Some general options are supported only by specific types of sockets (SOCK_DGRAM, SOCK_STREAM)



Spring'09

SWE 545
Distributed Systems Programming

9

Some Generic Options

- SO_BROADCAST
- SO_DONTROUTE
- SO_ERROR
- SO_KEEPALIVE
- SO_LINGER
- SO_RCVBUF,SO_SNDBUF
- SO_REUSEADDR



Spring'09

SWE 545
Distributed Systems Programming

10

SO_BROADCAST

- Boolean option: Enables/Disables sending of broadcast messages
- Underlying DL layer must support broadcasting
- Applies only to SOCK_DGRAM sockets
- Prevents applications from inadvertently sending broadcasts (OS looks for this flag when broadcast address is specified)



Spring'09

SWE 545
Distributed Systems Programming

11

SO_DONTROUTE

- Boolean option: Enables bypassing of normal routing
- Used by routing daemons



Spring'09

SWE 545
Distributed Systems Programming

12

SO_ERROR

- Integer value option
- The value is an error indicator value (similar to errno)
- Readable only
- Reading (by calling `getsockopt()`) clears any pending error



SO_KEEPALIVE

- Boolean option: Enabled means that STREAM sockets should send a probe to peer if no data flow for a “long time”
- Used by TCP. Allows a process to determine whether peer process/host has crashed
- Consider what would happen to an open telnet connection without keepalive



SO_LINGER

- Value is of type:

```
struct linger {
    int l_onoff;    /* 0 = off */
    int l_linger;  /* time in seconds */
};
```
- Used to control whether and how long a call to close will wait for pending ACKS.
- Connection-oriented sockets only



SO_LINGER Usage

- By default, calling `close()` on a TCP socket will return immediately
- The closing process has no way of knowing whether or not the peer received all data
- Setting `SO_LINGER` means the closing process can determine that the peer machine has received the data (but not that the data has been read())



SO_RCVBUF SO_SNDBUF

- Integer values options - change the receive and send buffer sizes
- Can be used with STREAM and DGRAM sockets
- With TCP, this option effects the window size used for flow control - must be established before connection is made



Spring'09

SWE 545
Distributed Systems Programming

17

SO_REUSEADDR

- Boolean option: Enables binding to an address (port) that is already in use
- Used by servers that are transient - allows binding a passive socket to a port currently in use (with active sockets) by other processes



Spring'09

SWE 545
Distributed Systems Programming

18

SO_REUSEADDR

- Can be used to establish separate servers for the same service on different interfaces (or different IP addresses on the same interface)
- Virtual Web Servers can work this way



Spring'09

SWE 545
Distributed Systems Programming

19

IP Options (IPv4)

- IP_HDRINCL: Used on raw IP sockets when we want to build the IP header ourselves
- IP_TOS: Allows us to set the “Type-of-service” field in an IP header
- IP_TTL: Allows us to set the “Time-to-live” field in an IP header



Spring'09

SWE 545
Distributed Systems Programming

20

TCP Socket Options

- `TCP_KEEPALIVE`: Set the idle time used when `SO_KEEPALIVE` is enabled
- `TCP_MAXSEG`: Set the maximum segment size sent by a TCP socket



Spring'09

SWE 545
Distributed Systems Programming

21

Another TCP Socket Option

- `TCP_NODELAY`: Can disable TCP's Nagle algorithm that delays sending small packets if there is unACK'd data pending
- `TCP_NODELAY` also disables delayed ACKS (TCP ACKs are cumulative)



Spring'09

SWE 545
Distributed Systems Programming

22

Socket Options Summary

- This was just an overview
 - There are many details associated with the options described
 - There are many options that haven't been described



Spring'09

SWE 545
Distributed Systems Programming

23

Posix Name/Adress Conversion

- `gethostbyname` and `gethostbyaddr` are protocol dependent; they are not part of sockets library
- POSIX includes protocol independent functions:
 - `getaddrinfo()`
 - `getnameinfo()`



Spring'09

SWE 545
Distributed Systems Programming

24

getaddrinfo, getnameinfo

- These functions provide name/address conversions as part of the sockets library
- In the future it will be important to write code that can run on many protocols (IPV4, IPV6)



Why getaddrinfo()?

- Puts protocol dependence in library (where it belongs)
- Same code can be used for many protocols (IPV4, IPV6)
- Re-entrant function - gethostbyname is not
- Important for threaded applications



getaddrinfo()

```
int getaddrinfo(  
    const char *hostname,  
    const char *service,  
    const struct addrinfo* hints,  
    struct addrinfo **result);
```

`getaddrinfo()` replaces both
`gethostbyname()` and `getservbyname()`



getaddrinfo() Parameters

- `hostname` is a hostname or an address string (dotted decimal string for IP)
- `service` is a service name or a decimal port number string



struct addrinfo

```
struct addrinfo {
    int      ai_flags;
    int      ai_family;
    int      ai_socktype;
    int      ai_protocol;
    size_t   ai_addrlen;
    char     *ai_canonname;
    struct sockaddr *ai_addr;
    struct addrinfo *ai_next;
};
```

Linked list!



Spring'09

SWE 545
Distributed Systems Programming

29

getaddrinfo() Hints

`hints` is an `addrinfo *` (can be `NULL`) that can contain:

- `ai_flags` (`AI_PASSIVE`, `AI_CANONNAME`)
- `ai_family` (`AF_XXX`)
- `ai_socktype` (`SOCK_XXX`)
- `ai_protocol` (`IPPROTO_TCP`, etc.)



Spring'09

SWE 545
Distributed Systems Programming

30

getaddrinfo() Result

- `result` is returned with the address of a pointer to an `addrinfo` structure that is the head of a linked list
- It is possible to get multiple structures:
 - Multiple addresses associated with the `hostname`
 - The `service` is provided for multiple socket types

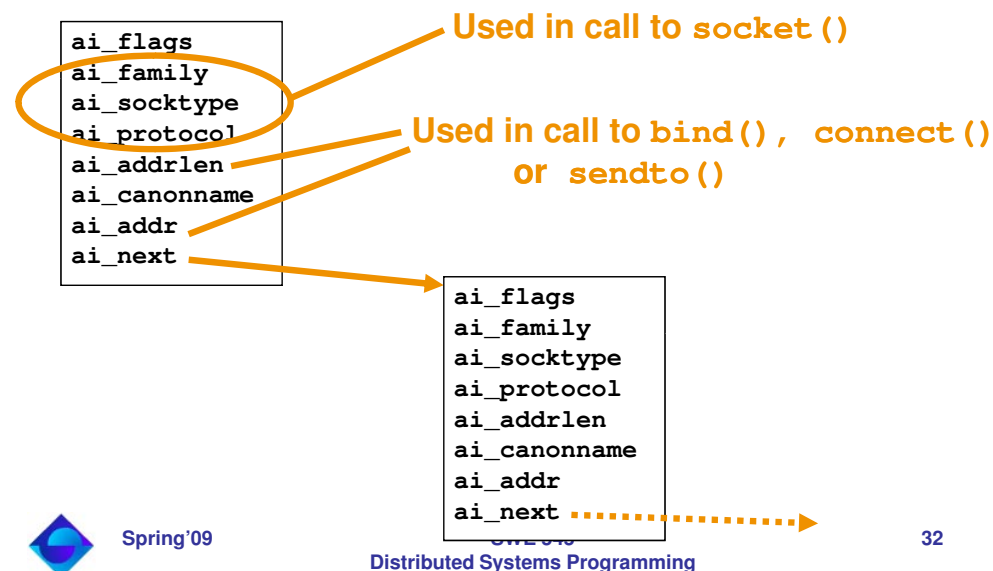


Spring'09

SWE 545
Distributed Systems Programming

31

addrinfo Usage



Spring'09

SWE 545
Distributed Systems Programming

32

getnameinfo()

```
int getnameinfo(  
    const struct sockaddr *sockaddr,  
    socklen_t addrlen  
    char *host,  
    size_t hostlen,  
    char *serv,  
    size_t servlen,  
    int flags);
```

getnameinfo() looks up a hostname and a service name given a `sockaddr`



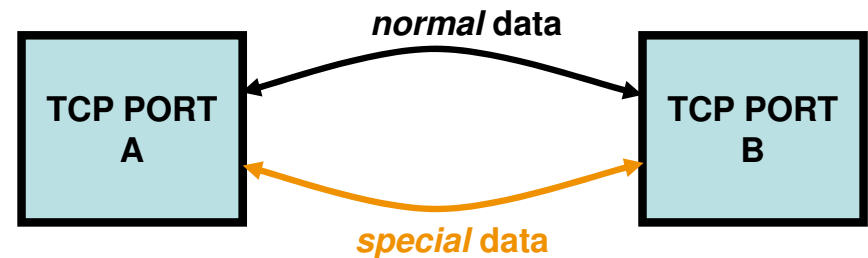
Spring'09

SWE 545
Distributed Systems Programming

33

Out-of-Band Data

- TCP (and other transport layers) provide a mechanism for delivery of "high priority" data ahead of "normal data"
- We can almost think of this as 2 streams:



Spring'09

SWE 545
Distributed Systems Programming

34

TCP OOB Data

- TCP supports something like OOB data using URGENT MODE (a bit is set in a TCP segment header)
- A TCP segment header field contains an indication of the location of the urgent data in the stream (the byte number)



Spring'09

SWE 545
Distributed Systems Programming

35

Sending OOB Data

```
send(sd, buff, 1, MSG_OOB);
```

Use `send()` to put a single byte of urgent data in a TCP stream

The TCP layer adds some segment header info to let the other end know there is some OOB data



Spring'09

SWE 545
Distributed Systems Programming

36

Receiving OOB Data

- The TCP layer generates a **SIGURG** signal in the receiving process
- `select ()` will tell you an exception condition is present



Reading URG data (a.k.a. *re-urg-e-dataing*)

- Depending on how things are set up:
 - The data can be read using `recv ()` with a **MSG_OOB** flag set.
 - The data can be read *inline* and the receiving process can monitor the *out-of-band-mark* for the connection (using `socketmark ()`)



So What?

- OOB data might be used:
 - As a heartbeat between the client and server to detect early failure
 - A way to communicate an exceptional condition to a peer even when flow control has stopped the sender



Signal Driven I/O

- We can tell the kernel to send us a **SIGIO** signal whenever something happens to a socket descriptor
- The signal handler must determine what conditions caused the signal and take appropriate action



Signal Driven UDP

- **SIGIO** occurs whenever:
 - An incoming datagram arrives
 - An asynchronous error occurs
 - Could be ICMP error (unreachable, invalid address, etc.)
- Could allow process to handle other tasks and still watch for incoming UDP messages



Signal Driven TCP (very rare)

- **SIGIO** occurs whenever:
 - an incoming connection has completed.
 - Disconnect request initiated.
 - Disconnect request completed.
 - Half a connection shutdown.
 - Data has arrived.
 - Data has been sent (indicating there is buffer space)
 - asynchronous error

