Unit 9: Windows 2000 Networking

9.2. Win32 Socket Programming

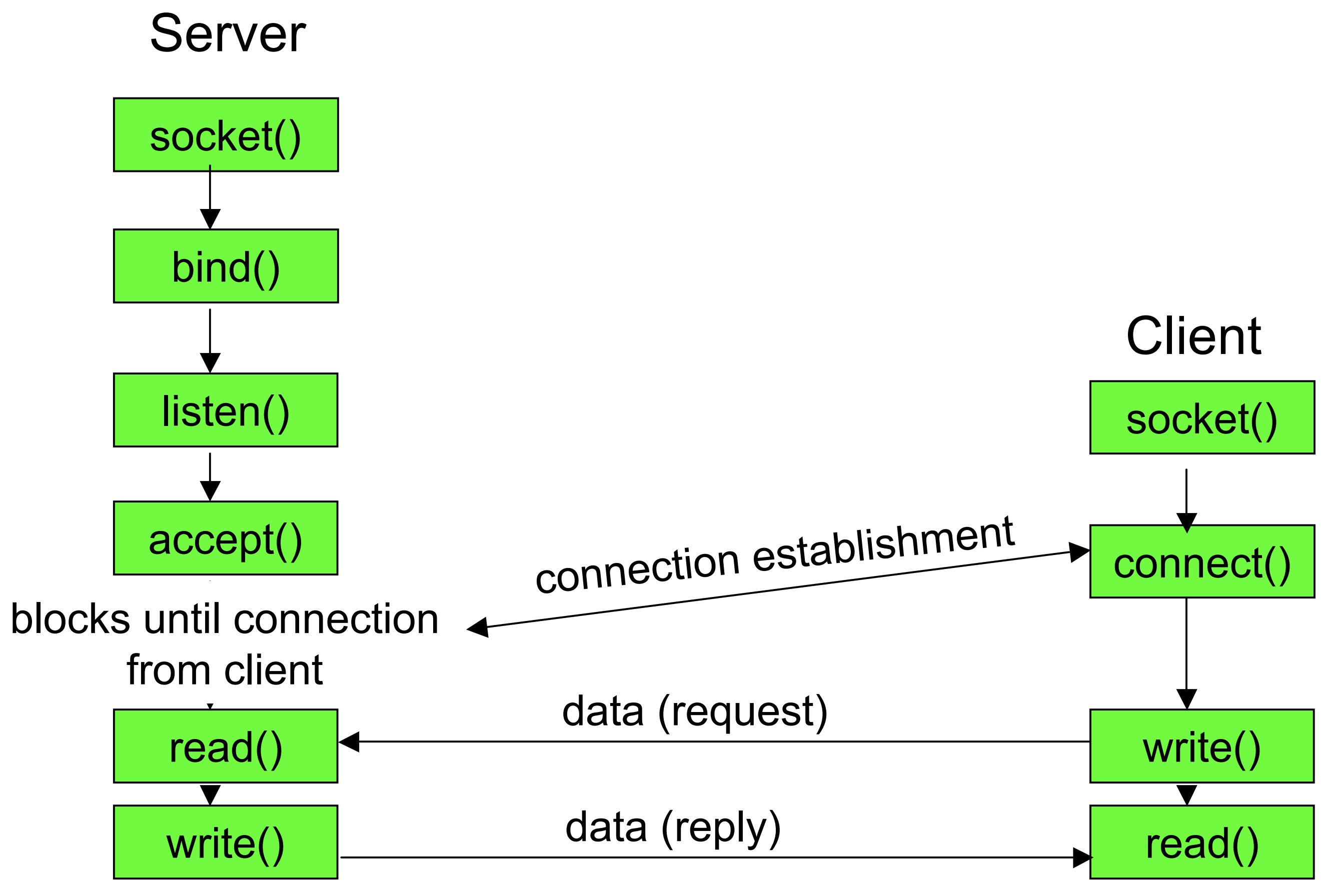
Win32 Socket Programming

Berkeley Socket programs will port to Window Sockets

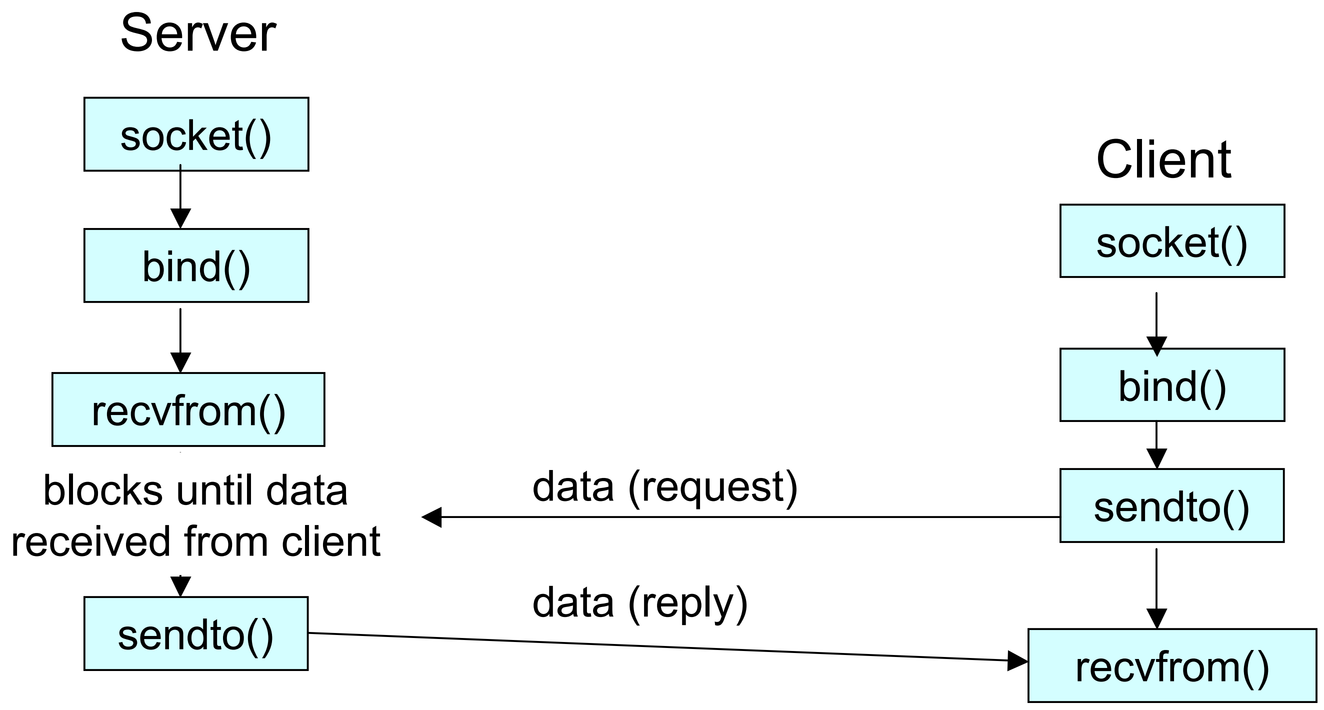
Exceptions:

- Call *WSAStartup()* to initialize Windows Socket DLL
- Use *ioctlsocket()* (non-portable) to configure the socket
- *_read()* and *_write()* can be used on sockets, but only after converting the socket descriptor to a file handle via *_open_osfhandle()*
- Use *closesocket()* (non-portable) rather than close to close a socket
- Call *WSACleanup()* to shut down the DLL

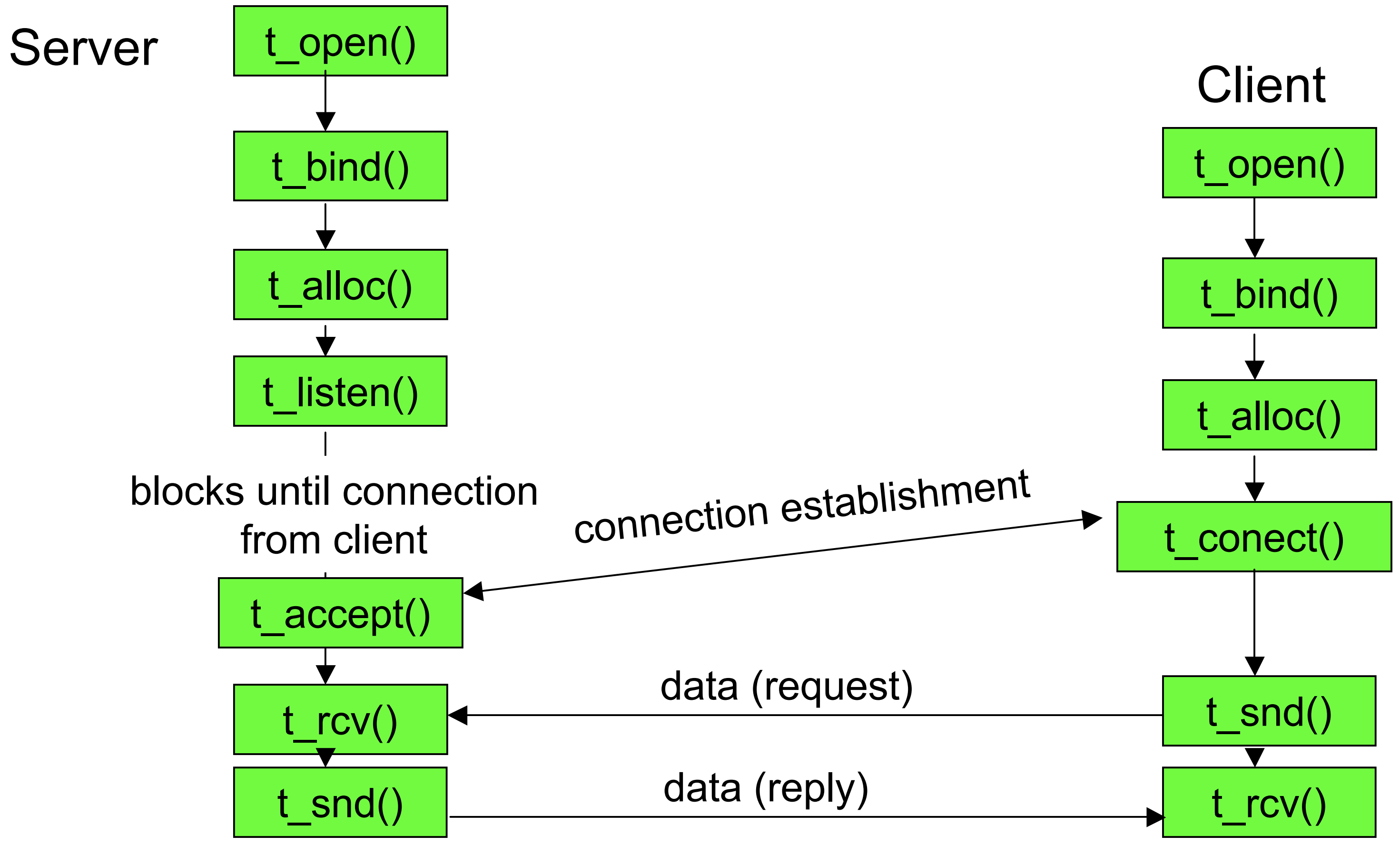
Berkeley 4.3 UNIX Sockets – connection-oriented



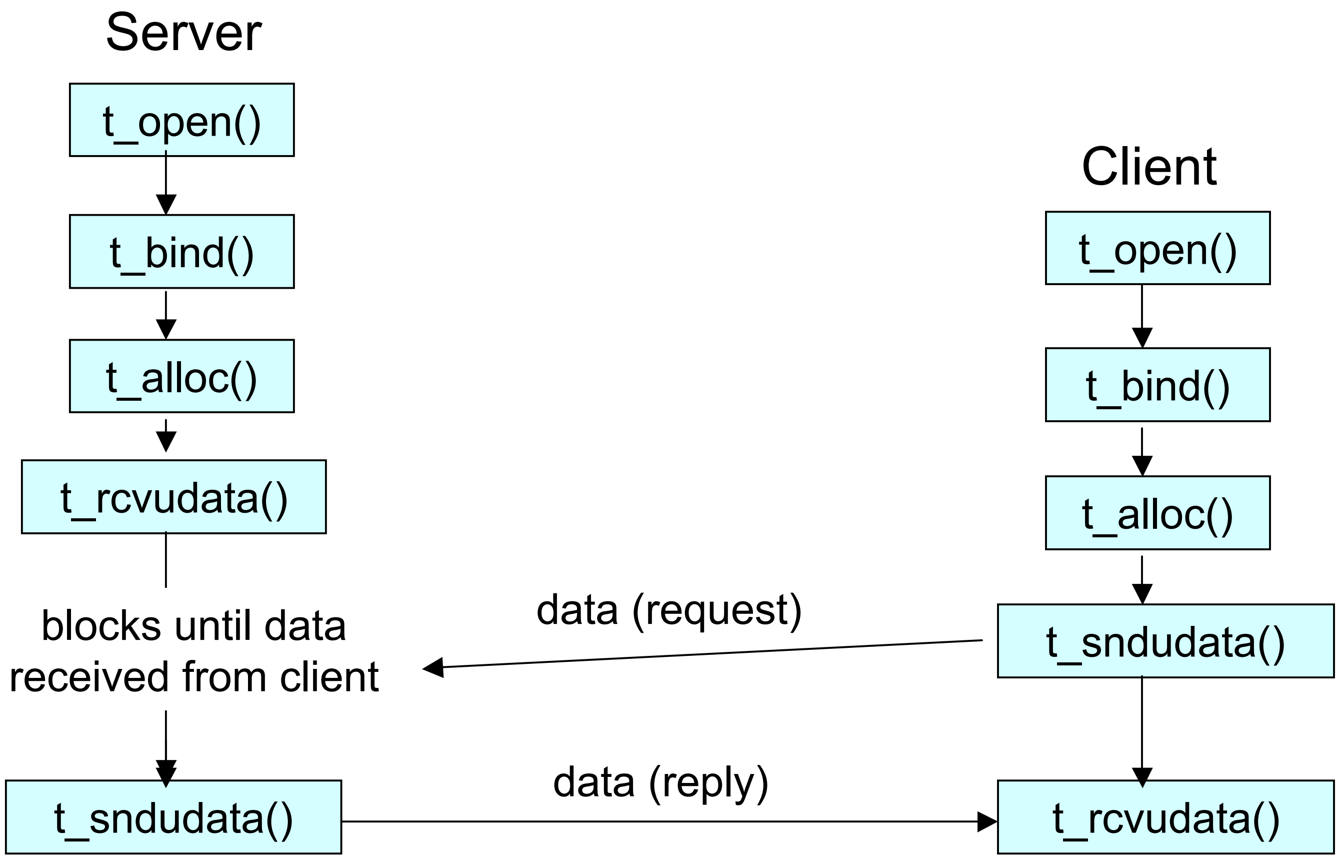
Berkeley 4.3 UNIX Sockets - connectionless



SYS V.3 Transport Layer Interface – connection-oriented



SYS V.3 Transport Layer Interface - connectionless



Accept a connection on a socket

```
#include <winsock.h>
SOCKET PASCAL FAR accept (
            SOCKET s, struct sockaddr FAR * addr,
            int FAR * addrlen );
```

- **s:** A descriptor identifying a socket which is listening for connections after a listen().
- **addr:** An optional pointer to a buffer which receives the address of the connecting entity, as known to the communications layer. The exact format of the addr argument is determined by the address family established when the socket was created.
- **addrlen:** An optional pointer to an integer which contains the length of the address addr.

Associate a local address with a socket

```
#include <winsock.h>
int PASCAL FAR bind (
        SOCKET s, const struct sockaddr FAR * name,
        int namelen );
```

- **s:** A descriptor identifying an unbound socket.
- **name:** The address to assign to the socket.
- **namelen**: length of the name

  -
    ```
    struct sockaddr {
    u_short sa_family;
    char sa_data[14];
    };
    ```

Internet address family

- In the Internet address family, a name consists of several components.
- For SOCK_DGRAM and SOCK_STREAM, the name consists of three parts:
  - a host address, the protocol number (set implicitly to UDP or TCP, respectively), and a port number which identifies the application.
  - If an application does not care what address is assigned to it, it may specify an Internet address equal to INADDR_ANY, a port equal to 0, or both.
  - If the Internet address is equal to INADDR_ANY, any appropriate network interface will be used; this simplifies application programming in the presence of multi-homed hosts.
  - If the port is specified as 0, the Windows Sockets implementation will assign a unique port to the application with a value between 1024 and 5000.
- The application may use getsockname() after bind() to learn the address that has been assigned to it
  - getsockname() will not necessarily fill in the Internet address until the socket is connected; several Internet addresses may be valid if the host is multi-homed.

Example: bind to an reserved port

```
SOCKADDR_IN sin;
SOCKET s;
u_short alport = IPPORT_RESERVED; /* 1024 */
sin.sin_family = AF_INET;
sin.sin_addr.s_addr = 0;
for (;;) {
        sin.sin_port = htons(alport);
        if (bind(s, (LPSOCKADDR)&sin, sizeof (sin)) == 0) {
                /* it worked */
        }
        if ( GetLastError() != WSAEADDRINUSE) {
        /* fail */
        }
        alport--;
        if (alport == IPPORT_RESERVED/2 ) {
        /* fail--all unassigned reserved ports are in use. */
        }
}
```

Close a socket

```
#include <winsock.h>
int PASCAL FAR closesocket ( SOCKET s );
```

- This function closes a socket.
  - releases the socket descriptor s, so that further references to s will fail with the error WSAENOTSOCK.
  - If this is the last reference to the underlying socket, the associated naming information and queued data are discarded.
- Semantics influenced by socket options:

Option	Interval	Type of close	Wait for close?
SO_DONTLINGER	Don‘t care	Graceful	No
SO_LINGER	Zero	Hard	No
SO_LINGER	Non-zero	Graceful	Yes

Establish a connection to a peer

```
#include <winsock.h>
int PASCAL FAR connect ( SOCKET s,
            const struct sockaddr FAR * name,
            int namelen );
```

- **s**: A descriptor identifying an unconnected socket.
- **name**: The name of the peer to which the socket is to be connected.
- **namelen**: The length of the name.
- create a connection to the specified foreign association. The parameter s specifies an unconnected datagram or stream socket

Establish a socket to listen for incoming connection

```
#include <winsock.h>
int PASCAL FAR listen ( SOCKET s, int backlog );
```

- **s**: A descriptor identifying a bound, unconnected socket.
- **backlog**: The maximum length to which the queue of pending connections may grow.
- typically used by servers that could have more than one connection request at a time:
  - if a connection request arrives with the queue full, the client will receive an error with an indication of WSAECONNREFUSED

Receive data from a socket

```
#include <winsock.h>
int PASCAL FAR recv ( SOCKET s,
            char FAR * buf, int len, int flags );
```

- **s**: A descriptor identifying a connected socket.
- **buf**: A buffer for the incoming data.
- **len**: The length of buf.
- **flags**: Specifies the way in which the call is made.

Receive a datagram and store the source address.

```
#include <winsock.h>
int PASCAL FAR recvfrom ( SOCKET s,
            char FAR * buf, int len, int flags,
            struct sockaddr FAR * from, int FAR * fromlen );
```

- **s**: A descriptor identifying a bound socket.
- **buf**: A buffer for the incoming data.
- **len**: The length of buf.
- **flags**: Specifies the way in which the call is made.
- **from**: An optional pointer to a buffer which will hold the source address upon return.
- **fromlen**: An optional pointer to the size of the from buffer.

Determine the status of one or more sockets, waiting if necessary.

```
#include <winsock.h>
int PASCAL FAR select ( int nfds, fd_set FAR * readfds,
fd_set FAR * writefds, fd_set FAR * exceptfds,
const struct timeval FAR * timeout );
```

- **nfds**: This argument is ignored and included only for the sake of compatibility.
- **readfds**: An optional pointer to a set of sockets to be checked for readability.
- **writefds**: An optional pointer to a set of sockets to be checked for writability
- **exceptfds**: An optional pointer to a set of sockets to be checked for errors.
- **timeout**: The maximum time for select() to wait, or NULL for blocking operation.

Send data on a connected socket.

```
#include <winsock.h>
int PASCAL FAR send ( SOCKET s,
const char FAR * buf, int len, int flags );
```

- **s**: A descriptor identifying a connected socket.
- **buf**: A buffer containing the data to be transmitted.
- **len**: The length of the data in buf.
- **flags**: Specifies the way in which the call is made.

Send data to a specific destination

```
#include <winsock.h>
int PASCAL FAR sendto ( SOCKET s,
            const char FAR * buf, int len, int flags,
            const struct sockaddr FAR * to, int tolen );
```

- **s**: A descriptor identifying a socket.
- **buf**: A buffer containing the data to be transmitted.
- **len**: The length of the data in buf.
- **flags**: Specifies the way in which the call is made.
- **to**: An optional pointer to the address of the target socket.
- **tolen**: The size of the address in to.

Create a socket

```
#include <winsock.h>
SOCKET PASCAL FAR socket (
          int af, int type, int protocol );
```

- **af**: An address format specification. The only format currently supported is AF_INET, which is the ARPA Internet address format.
- **type**: A type specification for the new socket.
- **protocol**: A particular protocol to be used with the socket, or 0 if the caller does not wish to specify a protocol.