Program Interface To Protocol Software

- An application program uses procedure calls to interact with protocol software
- Set of procedures defines the *Application Program Interface* (API)
- Most communication protocols do not define the precise API that programs use to access them
- API depends on the underlying operating system

Two Prominent APIs

- Socket Interface
  - From BSD UNIX
  - A *de facto* standard
  - Follows UNIX I/O conventions
  - Often provided by other systems
- Transport Layer Interface (TLI)
  - From AT&T System V UNIX
  - Implemented as library functions

Socket I/O

- Program calls *socket* to create a socket
  - Call returns an integer descriptor
- Program calls *recv* or *send* to transfer data
  - Descriptor is an argument in each call
  - read and write work too, but not recommended
- Program calls *close* to terminate the communication
  - Descriptor is an argument to the call

The Socket Interface

- Sockets are flexible; they can be used:
  - For connection-oriented communication (TCP)
  - For connectionless communication (UDP)
  - When the remote endpoint is prespecified
  - When the remote endpoint is specified with each message sent
  - With TCP/IP protocols (e.g., TCP, UDP)
  - With other protocols

The Socket Functions

Name	Meaning
socket	Create socket & descriptor
connect	Connect to a remote peer
close	Terminate all communication
bind	Bind a local address to socket
listen	Place a socket in passive mode
accept	Accept incoming connection (server)
recv	Receive the next incoming message
recvmsg	Receive the next incoming message
recvfrom	Receive the next incoming message & record source address
send	Send outgoing message
sendmsg	Send outgoing message
sendto	Send outgoing message to a specified address
shutdown	Terminate communication in one or both directions
getpeername	Obtain remote machine's address
getsockopt	Obtain current options for a socket
setsockopt	Change options for a socket