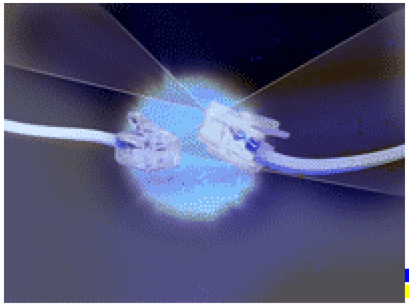
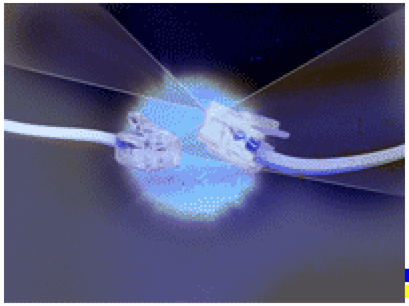


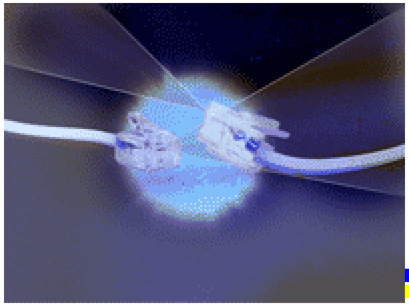
TCP vs. UDP

	TCP	**UDP**
Service	Stream	Datagram
Format	Byte-oriented	Message-oriented
Reliability	Complete	Minimal
Model	Connection-oriented	Connectionless
I/O Mechanism	send/recv	sendto/recvfrom



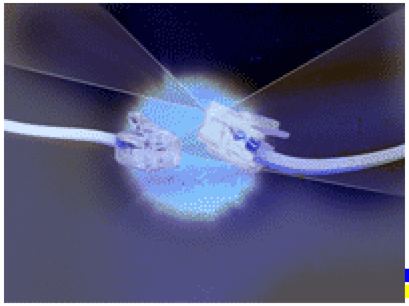
TCP vs. UDP - Format

- TCP is a stream of bytes, like a file. There is no preservation of boundaries in the data you send – you need to add them yourself.
- UDP delivers the message you send – if you put 50 bytes in one message, it delivers the 50-byte message to the receiver



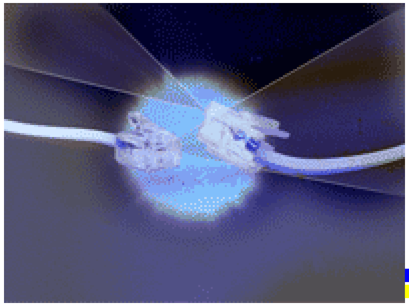
TCP vs. UDP - Reliability

- TCP is completely reliable. It guarantees that every byte is delivered correctly to the receiver, and that all bytes are in the same order.
- UDP checks packets for corruption, but doesn’t provide any other reliability. You have to check for packets in order, and make sure packets are received, if you need that reliability.



TCP vs. UDP - Model

- In TCP, when you call *connect()* on the client and *accept()* on the server, the nodes exchange messages and establish a connection that is maintained as long as the socket is open.
- In UDP, a receiver is not aware of anything about senders other than the source address it sees in packets it gets

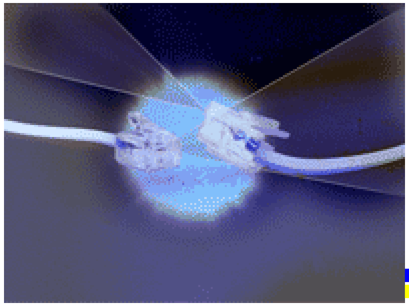


TCP vs. UDP – I/O mechanism

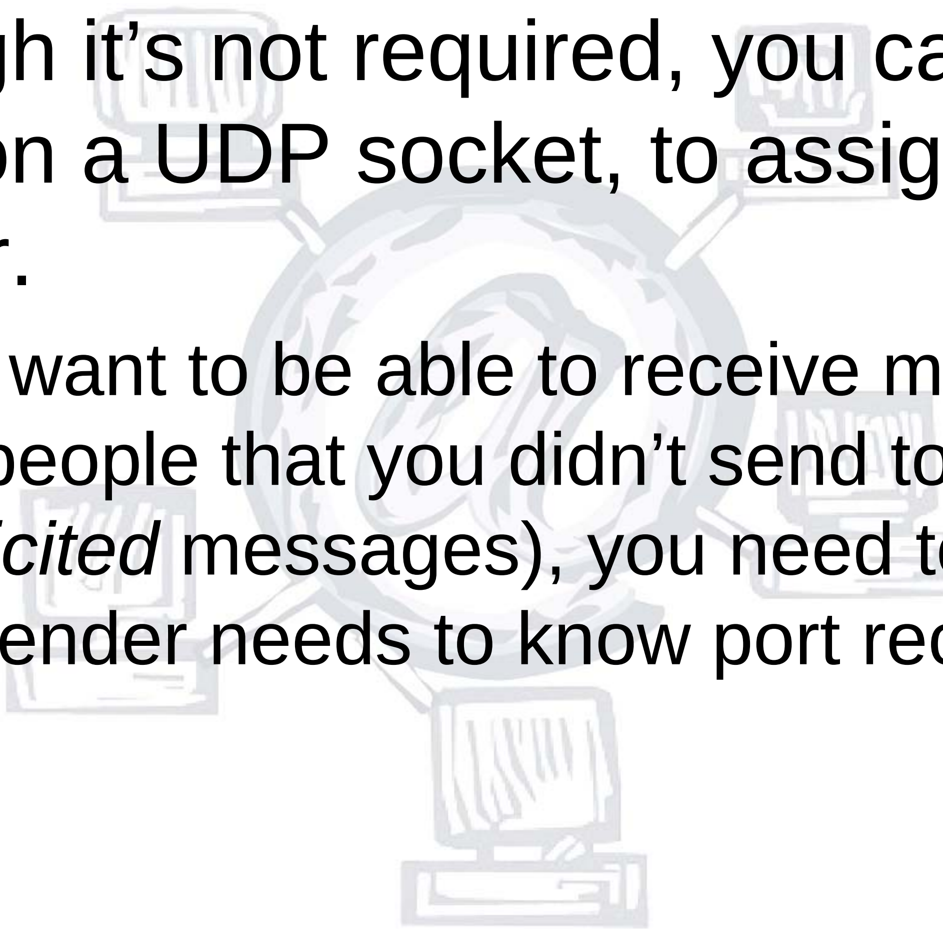
- In TCP, you call *send()* and *recv()* to transmit or read buffers of bytes
- In UDP, you use the functions

```
– int sendto( int sock, char * msg, int len,
              int flags, struct sockaddr * addr,
              int addrlen )
– int recvfrom( int sock, char * msg, int len,
              int flags, struct sockaddr * addr,
              int * addrlen )
```

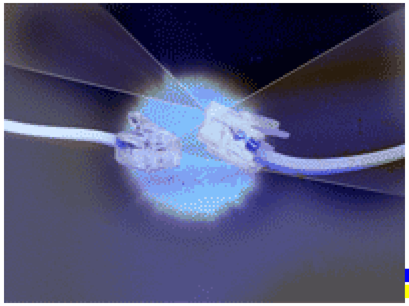
  – Address included in each call – can be different for each invocation



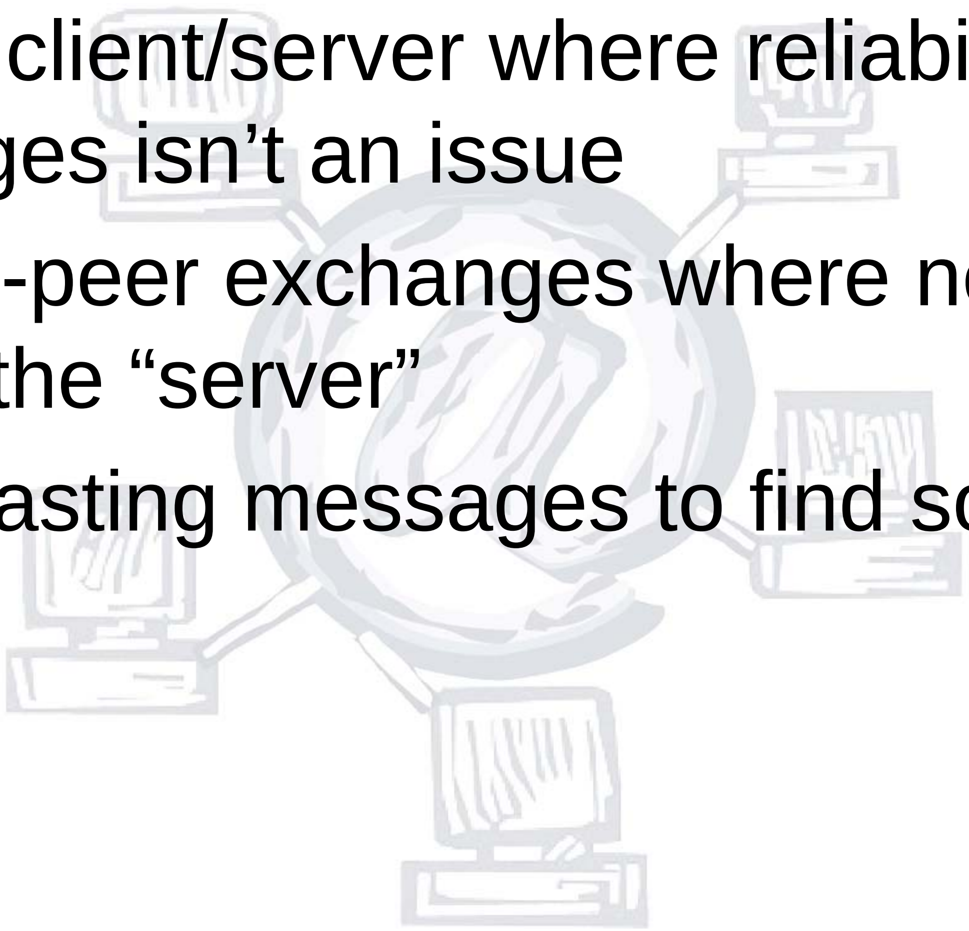
TCP vs. UDP – I/O (cont.)



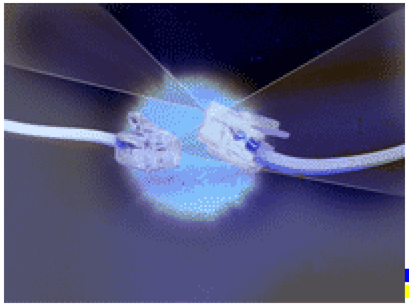
- Although it’s not required, you can call *bind()* on a UDP socket, to assign a port number.
  - If you want to be able to receive messages from people that you didn’t send to first (i.e. *unsolicited* messages), you need to do this. And sender needs to know port receiver is using



Uses for UDP



- Simple client/server where reliability of messages isn’t an issue
- Peer-to-peer exchanges where neither side is the “server”
- Broadcasting messages to find some service



Pros and Cons

- Advantages
  - Simple, low overhead
  - Message-oriented, so you don’t have to add anything to the messages to delimit
  - Don’t need to manage all those sockets to talk to a lot of clients
- Disadvantages
  - Unreliable – may need to implement some parts of TCP on your own