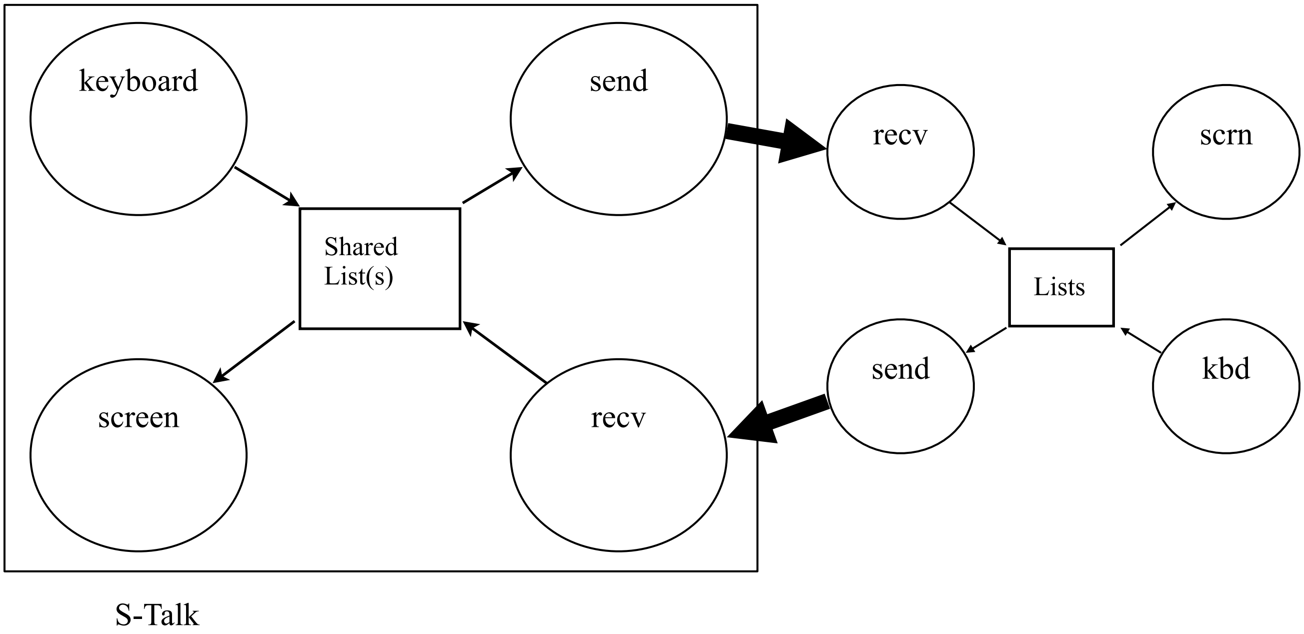
CMPT 300: Sockets

This tutorial focuses on an overview of s-talk and sockets.



The four threads (keyboard/screen/send/recv) communicate with each other via shared lists. One S-Talk “send” thread communicates with another’s “recv” thread via UDP between two sockets. The URL in the assignment has a great discussion of UDP. It is strongly recommended that you read it. It explains why we use AF_INET and not the PF_INET you see on the man page for socket.

A socket is a subclass of a file. A connection between two sockets is like a bi-directional pipe between two processes on (possibly) different machines. To create a socket use the following:

```
s = socket(AF_INET, SOCK_DGRAM, 0);
```

We need a way of “binding” the socket to an internet port that can be named outside the program so that a link can be made with the other s-talk program over the AF_INET protocol. Bind the socket to an IP:port number using:

```
bind(s, (struct sockaddr *)psa_in, sizeof(struct sockaddr_in));
```

where `s` is the socket id returned by “socket”, `psa_in` is a pointer to a `sockaddr_in` struct (see /usr/include/sys/socket.h and /usr/include/netinet/in.h). Since we’re using an internet “address family” socket we use a `sockaddr_in` instead of a `sockaddr`. It needs to be initialized as follows:

```
      struct sockaddr_in addr;

      addr.sin_family = AF_INET;
      sin_port = htons(port);
      addr.sin_addr.s_addr = INADDR_ANY;
      memset(&addr.sin_zero, '\0', 8);
```

“Htons” takes care of the difference between the integer storage format on the host and “network byte order”. “Port” can be anything above 1024 up to 65535. (0 to 1024 are reserved.) INADDR_ANY simply means “use the IP of the local machine”. Our bind call is now:

```
      bind(s, (struct sockaddr *)&addr, sizeof(struct sockaddr_in));
```

To send data or receive data:

```
      sendto(s, &msg, len, 0, (struct sockaddr *)&remote, sizeof(struct
sockaddr_in));
      recvfrom(s, &msg, len, 0, (struct sockaddr *)&remote,
sizeof(struct sockaddr_in));
```

where `len` is the size of `msg`, and remote is another `sockaddr_in` value with the IP address and port of the receiving/sending socket. See the man pages for `getaddrinfo` and `inet_aton`.

Close the socket when you’re done:

```
      close(s);
```