Firmware exercise - Worldsensing

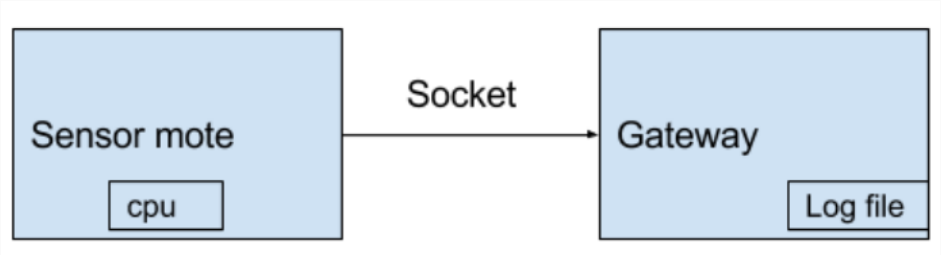
Gian Carlo Gebbia

03/07/19

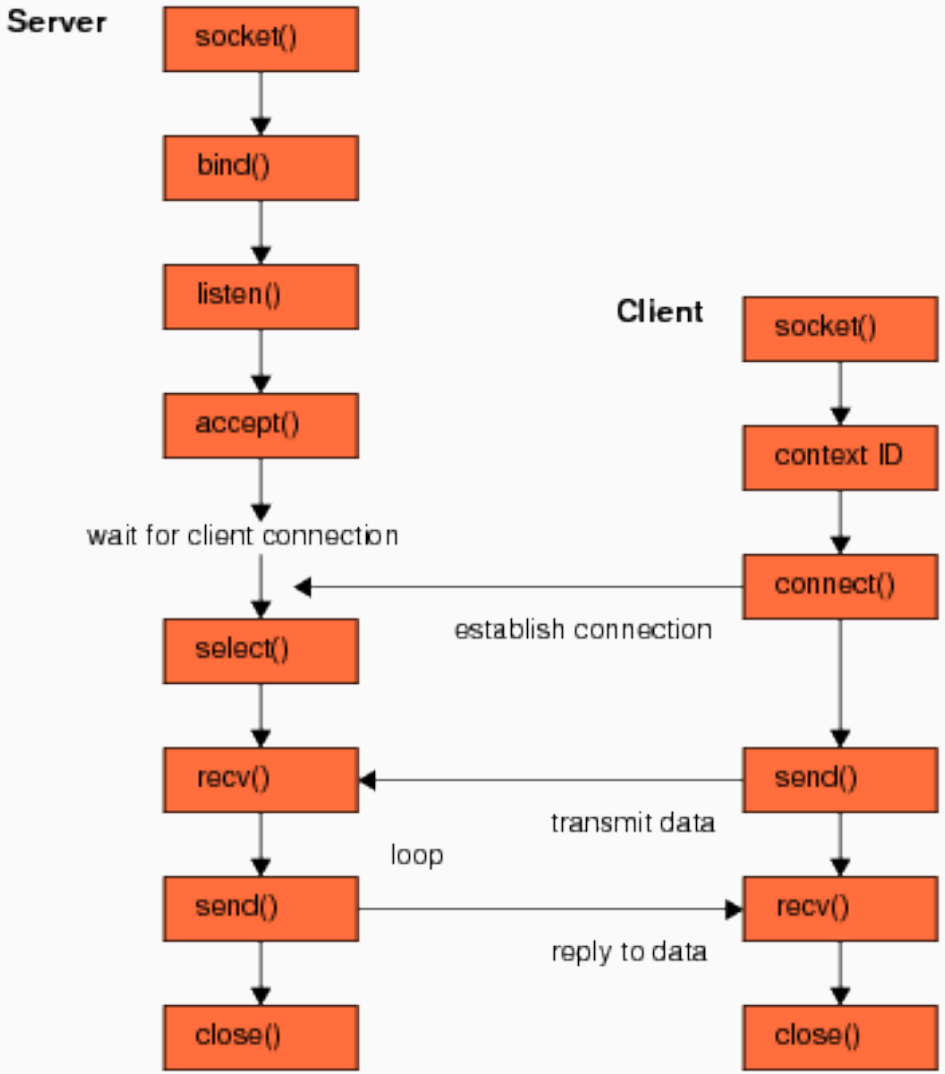
EXERCISE

Build two programs in C:

- Sensor mote: Will read the **temperature of the CPU** and will send it through **socket**.
- Gateway: Receive the messages from motes and will store them in a file.



CONNECTING STREAM SOCKETS - TCP



Client

SOCKET

- AF_INET: IPv4
- SOCK_STREAM: TCP (Transission Control Protocol)
- Protocol: single protocol -> value = 0

```c
int CreateSocket(void){

    int network_socket;
    printf("Creating Socket\n");
    network_socket = socket(AF_INET, SOCK_STREAM, 0);

    return network_socket;
}
```

CONNECTION ESTABLISHMENT

- Family address: AF_INET
- Inet address: Localhost (127.0.0.1)
- Port: 8080

```
int ConnectSocket(int network_socket){

    int conn_status = -1;
    struct sockaddr_in server_address;

    server_address.sin_family = AF_INET;
    server_address.sin_addr.s_addr = inet_addr("127.0.0.1");
    server_address.sin_port = htons(PORT);

    conn_status = connect(network_socket,
                          (struct sockaddr *)&server_address,
                          sizeof(struct sockaddr_in));
    return conn_status;
}
```

DATA SENDING

- timevalue = 10 sec
- level = SOL_SOCKET: Retrieve options at the socket level
- SO_REUSEADDR: Socket may bind, except when there is already listening

```
int SocketSend(int network_socket, char *message, int len_message){

    int send_status = -1;
    struct timeval tv;
    tv.tv_sec = 10;  /* 10 seconds */
    tv.tv_usec = 0;  /* 0 microseconds */

    if(setsockopt(network_socket,
                  SOL_SOCKET,
                  SO_REUSEADDR,
                  (char *)&tv, sizeof(tv)) < 0){
        printf("Time Out\n");
        return -1;
    }
    send_status = send(network_socket, message, len_message, 0);
    return send_status;
```

Server

SOCKET

- AF_INET –> IPv4
- SOCK_STREAM –> TCP (Transission Control Protocol)

```c
int CreateSocket(void){

    int network_socket;
    printf("Creating Socket\n");
    network_socket = socket(AF_INET, SOCK_STREAM, 0);

    return network_socket;
}
```

BINDING

- Family address: AF_INET
- Incoming interface: INADDR_ANY
- Port: 8080

```
int BindSocket(int network_socket){

    int bind_status = -1;
    struct sockaddr_in  address;

    address.sin_family = AF_INET;
    address.sin_addr.s_addr = htonl(INADDR_ANY);
    address.sin_port = htons(PORT);
    bind_status = bind(network_socket,
                       (struct sockaddr *)&address,
                       sizeof(address));

    return bind_status;
}
```

LISTEN, ACCEPT, RECEIVED AND STORAGE

```
//Listen
    listen(network_socket , 3);

//accept connection from an incoming client
    sock = accept(network_socket,
                  (struct sockaddr *)&client,
                  (socklen_t*)&clientLen);

//received
    recv(sock , client_message , 10 , 0)
```

The client message will be stored in the `temperature.txt` file.