Heuristics

Lab Section 9

Today's Lab

- We will explore:
- No assignment today
  - If you understand everything below and have run auction game code, you can leave
- Discuss Inter-process communication using sockets
- Introduce/install zeromq
- Go over the main functions of zeromq which are used in the Auction game
- Run Auction Project

IPC and Sockets

- For communication between client and server in Auction, sockets are used.
  - Client and server are both processes (programs in execution),
    - Same program could result in multiple processes when executed multiple times
- Inter Process communication (IPC) through message passing using sockets
  - Enable Channel based (stream or datagrams/packets) communication, via Network Control Protocols (TCP/UDP), between
    - Processes on the same physical machine/device (host), or
    - Processes that can run on different hosts (machines)

Socket – Implementation

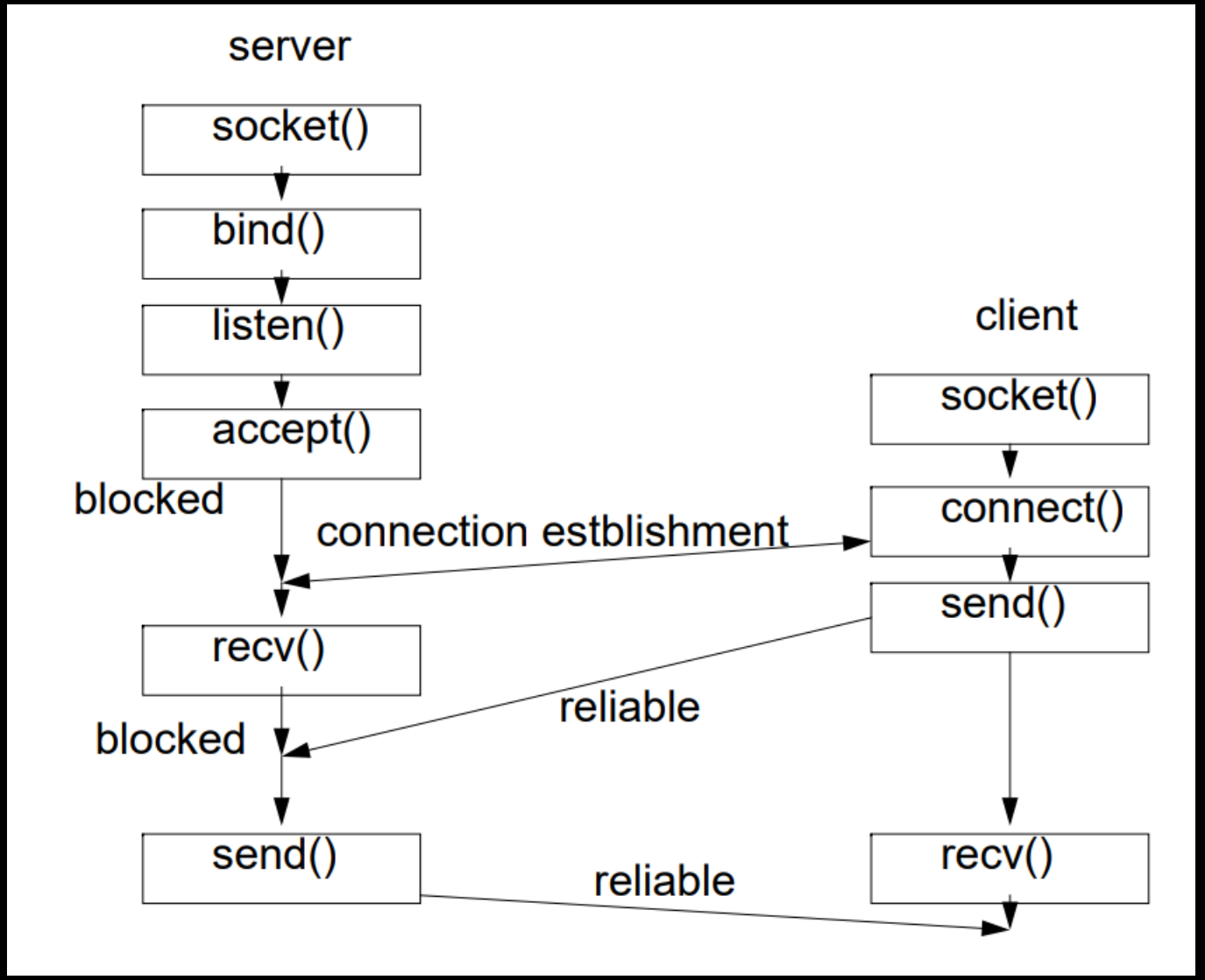
- Sockets are just files. it's a way to talk to other computers using standard Unix file descriptors
- A socket is a pseudo-file that represents a network connection
  - src-ip:port----dest-ip:port. Ip to identify the machine in the network, port to to identify the process that needs to (consume the packet/write to pseudo-file)
    - A port number is a way to identify a specific process to which an internet or other network message is to be forwarded when it arrives at a server. All network-connected devices come equipped with standardized ports that have an assigned number. There are 65,535 port numbers, but not all are used. Restricted port numbers or well-known port numbers are reserved by prominent companies and range from 0 to 1023
- Once a socket/file has been created (identifying the other host and port), writes to that socket are turned into network packets that get sent out, and data received from the network can be read from the socket.
- we are able to read and write using the same socket/file because in sockets, write is not just writing of bytes on a buffer, but also to send this bytes over the network, so

  write in sockets means:
  - write bytes to pseudo file associated to connection
  - send bytes as packets
  - clear content from the file ->
  - now file will be empty and could be used to receive bytes/ read operation on it.

Socket Characteristics

- Server Ip and port, mean address of source-socket remain constant once server process started
- Meaning over the lifetime of server process, server socket will continue to be accessed/communicate with clients-processes using the same address
- client process can not connect before the server binds to a specific port, and any attempt to connect will failed because the server process even if it is running is not identifiable via port, hence oblivious to client processes

Typical Flow of Connection oriented protocol through socket calls



ZMQ Introduction/Installation

- ZeroMQ (also spelled ØMQ, 0MQ or ZMQ) is a high-performance asynchronous messaging library, aimed at use in distributed or concurrent applications. It provides a message queue, but unlike message-oriented middleware, a ZeroMQ system can run without a dedicated message broker.
- ZeroMQ supports common messaging patterns (pub/sub, request/reply, client/server and others) over a variety of transports (TCP, in-process, inter-process, WebSocket and more), making inter-process messaging as simple as inter-thread messaging.
- To install ZeroMQ:
  - pip install pyzmq
    - Replace pip with pip3 for installation in max

ZMQ socket creation

1. To create a Socket, first create a Context:
    - ctx = zmq.Context.instance()
        - Context managers allow you to allocate and release resources precisely when you want to.
2. then call ctx.socket(socket_type):
    - socket = ctx.socket(zmq.REQ)
        - REQ is for request sockets that we will use in clients.
    - socket = ctx.socket(zmq.REP)
        - REP is for replying through the server socket.

ZMQ socket bind/connect

- After socket creation, on the server side, we bind to a particular port/address.
- We need to bind, so that the server process could be identified and reached at a specific address.
  - In our program, for the local server process, we will use the address of localhost: port
  - Where localhost is the loopback ip and port is any free arbitrary port.
- Here is the corresponding code:
  - Port = “50018”
  - Socket.bind(“tcp://*:%s” % port)
- After socket creation, on the client side, we connect to the server listening on a particular port/address.
- We need to connect, so that client can establish communication with the particular server.
  - In our client program, we use the address/port of the server to identify and connect the client with the server.
- Here is the corresponding code:
  - port = "50018"
  - socket.connect ("tcp://localhost:%s" % port)

Send, Recv – call in Server and Client

- Once a connection is established, the application processes invoke the following two functions to send and receive data:
- send(),recv()
- The first operation sends the given message over the specified socket, while the second operation receives a message from the specified socket. By default, both send() and recv(), send and receive the data in bytes. Therefore, we need to encode/decode data when using these functions.
- While sending data, we can encode strings to bytes using the .encode() function, by default it will encode in UTF-8 (character encoding). Similarly, when receiving data, we can decode bytes to string using the .decode() function.
    - txt = "My name is Ståle“ , x = txt.encode()
        - b'My name is St\xc3\xe5le’
    - y = x.decode()
        - My name is Ståle

Running Auction Project

- Copy serverzmq.py and clientzmq.py into a new folder such as Auction
- We will be playing with two bidders.
  - The number of bidders is determined by numbidders line in serverzmq.py.
  - This is currently set for 3 bidders (inspect serverzmq.py to see)
    - I have changed it to 2.
- Open three terminal/command prompt windows and in each change the working directory to Auction (using cd ‘path’)
  - One terminal will be used to run the server process
  - Other two for running client processes.
- In one terminal, run the server by executing the command:
  - python serverzmq.py
- In other terminals, run the client process by executing in each:
  - python clientzmq.py
- For mac, replace python with python3

End