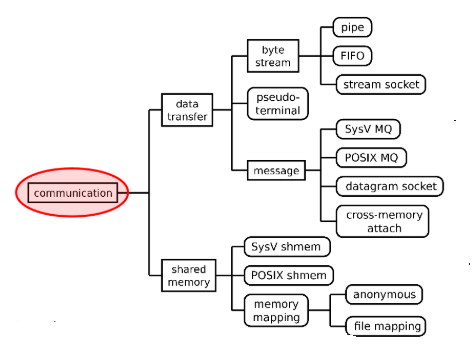
**Linux IPC:**

Below diagram shows different IPC Mechanism.



Pipes are used between related processes, Fifo is used between un related process.

Stream socket uses TCP for communication . Stream sockets are most common because the burden of data transfer is handled application .

Datagram sockets (UDP) uses connectionless service , Datagrams can received out of order.

In Message Queues , readers and writers communicates in units of messages . Messages have priority and are delivered in priority order.

**Socket Programming**



**Sockets**: An interface between application process and transport layer. The application process can send/receive messages to/from another application process (local or remote via a socket).

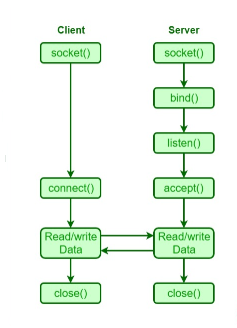
Sockets can be

1. Unix domain sockets AF\_UNIX , communication on a single host , address = file system name
2. IPV4 domain socket AF\_INET , communication on IPV4 network

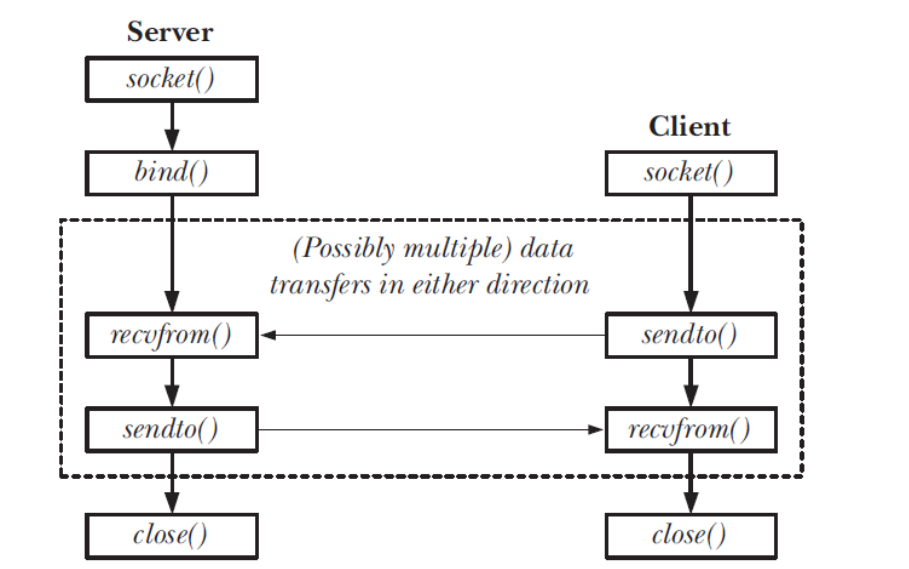
Address = IPV4 address (32 bit) + Port No

1. IPV6 domain socket AF\_INET6 , Communication on IPV6 network , address = IPV6 address(64 bit) + Port No

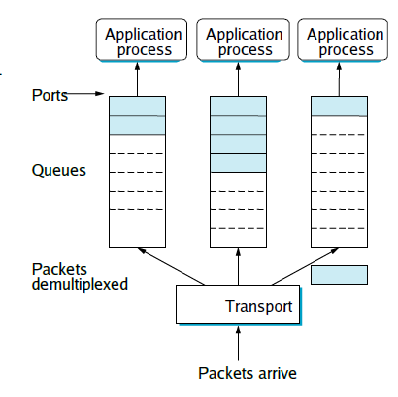
Below diagram shows TCP Connection Oriented Protocol



Below diagram shows UDP Connection , User Datagram Protocol



UDP Sockets are useful for multicast and broadcast



Based on Port No , Data is transferred to application process

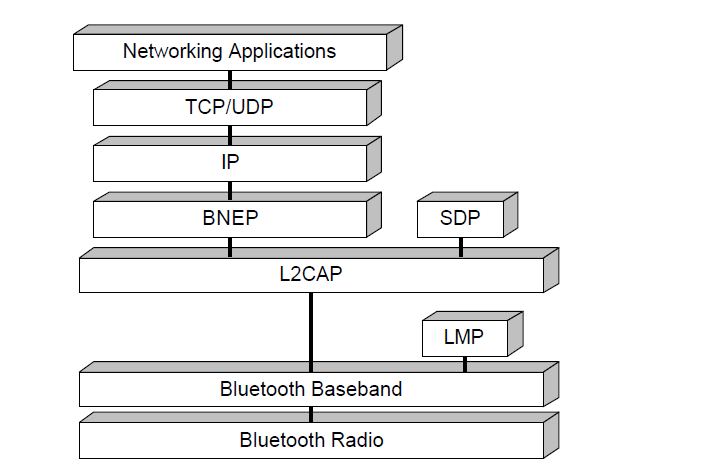
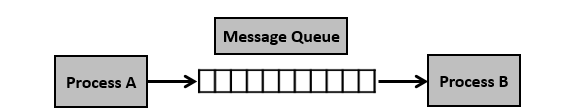


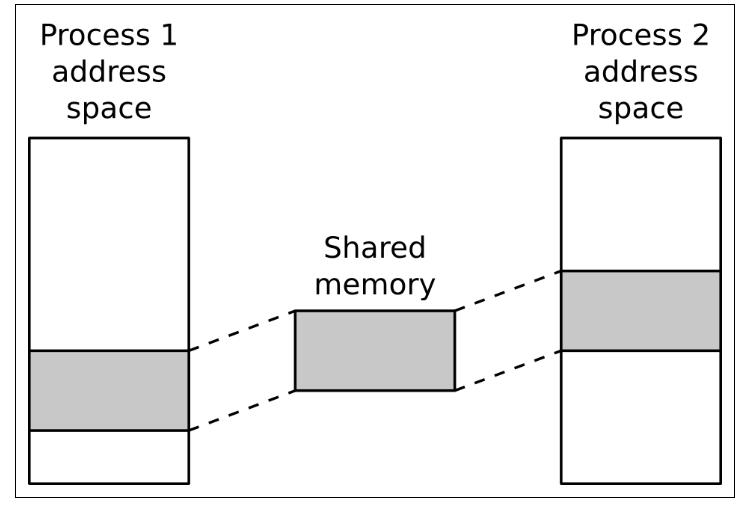
Fig above shows bnep protocol i.e Bluetooth network encapsulation protocol, BNEP is used in PAN Connection , The data from upper layers is received to TCP/UDP then to IP layer and it goes to BNEP protocol .

**Message queue :**

1. **Message queue** allow processes to exchange data in the form of messages
2. Message oriented communication, Receiver reads messages one at a time. Messages have priority, they are delivered in priority order ,
3. Message queue has a form **/ name-of-message-queue.**
4. The information can be found in **/dev/mqueue/<name-of-message-queue> .**
5. **ipcs** command also gives message queue information if they are system v message queues , ipcs command will not work for posix message queues .
6. A process can request asynchronous notification of the arrival of a message on a previously empty queue using **mq\_notify**
7. Additional info can be found from man pages , **man 7 mq\_overview** .

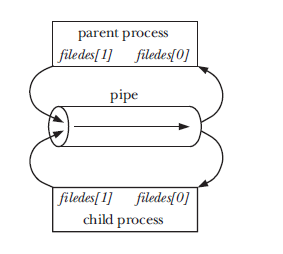


**Shared Memory :**

****

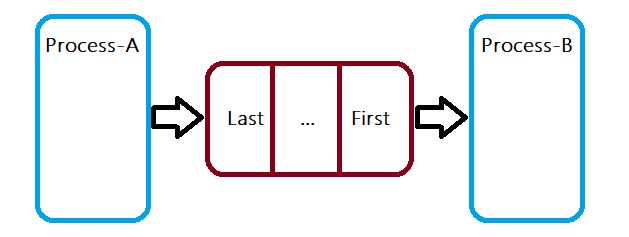
1. The POSIX shared memory API allows processes to communicate information by sharing a region of memory
2. A shared memory object will exist until the system is shut down, or until all processes have unmapped the object and it has been deleted with **shm\_unlink**
3. Can be created with form **/<name-of –shared-memory>**
4. Shared memory normally mounted under /dev/shm
5. No transfer of data between user space to kernel space involved , Applications can directly map top Physical memory
6. Need synchronization between process .

**Pipes :** Setting up a pipe to transfer data from a parent to a child .



1. Pipes are byte stream buffer , Data can travel only in one direction through a pipe. One end of the pipe is used for writing, and the other end is used for reading
2. In Linux Pipes are denoted by symbol **“ | ” ,** a simple vertical bar , Contents to left of | is writer and right of vertical bar | is reader .
3. From Program , user can create pipe using call
4. Pipes communicate only between related process
5. Pipes are anonymous , They don’t have name in file system .Through fork pipes share between process .

**FIFO :**

****

1. Pipes with name in file system is called FIFO (First In First Out)
2. System call mkfifo , helps in creating FIFO , ls –l gives file type indicated by ‘p’
3. I/O is same like pipes
4. Fifo open call is synchronized between process
5. FIFO special file has no contents on the file system; the file system entry merely serves as a reference point so that processes can access the pipe using a name in the files system
6. From command line , mkfifo <fifo-name> can be used to create fifo and echo “data” > <fifo-name> is written to fifo and cat <fifo-name>

Can be used to read fifo .

1. Fifo one end is blocked if there is no other end available i.e if there is no writer , reader gets blocked and if there is no reader , writer gets blocked .