Soumyadip Pyne 197178 CSEA

2-2

No. of process → 2 + no. of clients.

Process name → ./S ./AS (in other PC)

./c1 ./c2 so on...

IPC → Sockets, UDS, MSG-Queue.

Special BSD calls → recv, send, sendmsg, recvmsg, socket, bind, listen, connect, accept.

I/O Mux → Poll (in clients)

Signals → ~~Yes~~ No

~~Name → sig1-handler fn → AS being notified by S using SIGUSR1~~

FD passing → Yes.

fd-name → wrfd[i] from S to AS.
i ∈ [0, no. of clients]

[illegible]

Soumyadip Payra 197178 CSE A

Steps

1. Original Server S serves clients with thread-service.
2. When required maintanence it ~~signals~~ notifies Alternate server AS, to recieve the sfds it is sending through UDS.
   ↳ through sfd of AS.
3. And S puts message in msg queue notifying the AS service.
4. Clients poll between msg-queue and socket for checking status of S and for service respectively.
5. AS receives the sfd and keep serving.
6. When S is ready it puts msg in msgqueue.
7. Client reads the msg and knows the ready status.
8. client disconnect from AS and reconnects to S.

[illegible]

Soumyadip Pangra 197178 CSEA

for code reuse $\Rightarrow$ service related codes are same with AS and S.

```
void
 echo_service(int nsfd){
        char buf[1024];
        recv(nsfd, buf, 1024, 0);
        printf("%s\n", buf);
}
```

Server S →

```
int main()
{   //msg que
        key_t key = ftok("./queue", 0);
        int msgqid = msgget(key, IPC_CREAT|0666);

    // creation of socket using socket(), bind(),
        followed by listen(), accept().

        int nsfd[10];
        for(int i=0; i<10; i++)
                nsfd[i] = accept(sfd, (struct sockaddr*)&cli_addr,
                                        (socklen_t*)&cli_len);

        pthread_t   id[10];
        for(int i=0; i<10; i++)
                pthread_create(&id[i], NULL, service,
                                        (void*) &nsfd[i]);
```

[illegible]

Soumyadip Pyrag 197178 CSE A

```
//when maintence required:
        msgsnd (msqid, &message, sizeof(message.content), 0);
    // message contain information from S to C.

  ~~kill ( pid_AS , SIGUSR1);~~
  for (int i=0; i<10; i++)
          send_fd (nsfd, nfd[i]);   (//nsfd to
                                     socket connection
                                     between AS
                                     and S)
//when ready put similar msg in
   msgqueue using
              msgsnd (msqid, &msg, sizeof(msg.content)
                                             ,0);
 // reconnect
          -> using accept and then service
               thread like before.

}.
```

AS -> same

except it receives,

using recv_fd () in the main function.

Client

```
int main ()
  {  // create or join msqid -> ftok(), msgget()
   //connects to sfd of S.
```

Pyrag.

Soumyadip Pagra 1917178 CSE A

```
// polls on sfd and msgqid.
// if received from msgqid to join AS,
      it gets notified.
// when received from msgqid that S B
  ready,
       close(sfd);
  // again create and connect socket to
    Server S.
}.
```

```
recv_fd (int socket)
{    int sent_fd, available_b_space;
     struct msghdr socket_msg;
      struct iovec io_vector[1];
      struct cmsghdr *ctrl_msg = NULL);
      char msg_buffer[1];
      char an_e_b[1];
        :
        : as implemented in class
}.
```

same for send_fd (int socket, int sfd_to_send);

Spagra.