**Assignment 8**

Contribution:

xsh\_udp\_request.c Venkat sambandhan

fcons\_udp\_print.c Harshit Shrivastava

fprod\_udp\_request.c Harshit Shrivastava

server.c Venkat sambandhan

arp.c Venkat sambandhan

clkhandler.c Harshit Shrivastava

Exercise 17.2

Client program written for xinu(xsh\_udp\_request.c) will call server running at Ubuntu VM, server will return the length of the string message passed by client.

Client code:

while(1)

{

printf("\n>"); fgets(message,512,CONSOLE); msg\_len = strlen(message); message[msg\_len-1]='\0'; if(strcmp(message,"exit") == 0)

break;

retval = udp\_send(slot, message, msg\_len);

if(retval == SYSERR)

{

printf(" Error in udp sending datagram ");

return -1;

}

retval = udp\_recv(slot, buffer, sizeof(buffer),3000); if(retval == SYSERR)

{

printf("Error in receiving datagram"); return -1;

}

if(retval == TIMEOUT)

{

printf("Timeout in receiving datagram"); return -1;

}

printf("Length of %s is %s \n",message,buffer); memset(buffer,'\0',sizeof(buffer)); memset(message,'\0',sizeof(message));

}

Server Code: while(1)

{

&from\_len);

bzero(buff,512);

printf("Waiting for requests...\n");

nbytes = recvfrom(socketfd, buff, sizeof(buff), 0, (struct sockaddr\*)&from\_addr,

//fgets(buff);

if (nbytes < 0 )

{

printf("Error in recvfrom");

}

printf("String received at server: %s\n",buff);

//printf("Total bytes received: %d\n",nbytes);

//reply = "I got your message"; len = strlen(buff);

snprintf(reply,sizeof(reply),"%d",len);

sendto(socketfd, reply, strlen(reply), 0,(struct sockaddr\*)&from\_addr,from\_len); memset(buff,'\0',sizeof(buff));

printf("Length of received string sent!\n");

}

Exercise 17.3:

Modified arpcache structure, to store timestamp in variable named ‘timestamp’. This variable will store the timestamp of the most recent cache hit.

New structure:

struct arpentry { /\* Entry in the ARP cache \*/ int32 arstate; /\* State of the entry \*/

uint32 arpaddr; /\* IP address of the entry \*/

pid32 arpid; /\* Waiting process or -1 \*/

byte arhaddr[ARP\_HALEN]; /\* Ethernet address of the entry\*/ int32 timestamp; /\* Timestamp of most recent cache hit\*/

};

extern struct arpentry arpcache[];

Clearing the cache: int32 arp\_cache\_clear()

{

int32 slot;

intmask mask; /\* Saved interrupt mask \*/ mask = disable();

for(slot=0; slot < ARP\_SIZ; slot++){ if(arpcache[slot].arstate == AR\_RESOLVED){

if((clktime - arpcache[slot].timestamp) > 300){

//memset((char \*)&arpcache[slot], NULLCH, sizeof(struct arpentry));

arpcache[slot].arstate = AR\_FREE; arpcache[slot].arpaddr = 0;

arpcache[slot].arpid = -1;

memset(&arpcache[slot].arhaddr, NULLCH, ARP\_HALEN\*sizeof(byte)); arpcache[slot].timestamp = 0;

}

}

}

restore(mask); return 0;

}

Network Futures:

Producer will request server and set the future with the returned value, the consumer will print the value from future.

Consumer Code: while(1)

{

while(fut->state==FUTURE\_EMPTY || fut->state==FUTURE\_WAITING) printf("");

status = future\_get(fut, &i);

if (status < 1)

{

printf("future\_get failed\n"); return -1;

}

if(i==-1)

break;

kprintf("\nConsumer consumed %d", i);

}

Producer Code: while(1)

{

while(f\_exlusive->state==FUTURE\_VALID) printf("");

printf("\n>");

fgets(message,512,CONSOLE); msg\_len = strlen(message); message[msg\_len-1]='\0';

if(strcmp(message,"exit") == 0) break;

retval = udp\_send(slot, message, msg\_len); if(retval == SYSERR)

{

printf(" Error in udp sending datagram "); return -1;

}

retval = udp\_recv(slot, buffer, sizeof(buffer),3000); if(retval == SYSERR)

{

printf("Error in receiving datagram"); return -1;

}

if(retval == TIMEOUT)

{

printf("Timeout in receiving datagram"); return -1;

}

value = atoi(buffer);

printf("\nProducer produced %d",value); status = future\_set(f\_exlusive, &value); if (status < 1)

{

printf("future\_set failed\n");

return -1;

}