Experiment 5

Simulate sliding window flow control protocols. (Stop and Wait, Go back N, Selective Repeat ARQ protocols)

sliding window flow control protocols

Flow control deals with problem that sender transmits frames faster than receiver can accept, and solution is to limit sender into sending no faster than receiver can handle Consider the simplex case: data is transmitted in one direction (Note although data frames are transmitted in one direction, frames are going in both directions, i.e. link is duplex) Stop and wait: sender sends one data frame, waits for acknowledgement (ACK) from receiver before proceeding to transmit next frame This simple flow control will break down if ACK gets lost or errors occur → sender may wait for ACK
that never arrives

Go-back-n ARQ

The basic idea of go-back-n error control is: If frame i is damaged, receiver requests retransmission
of all frames starting from frame i
Notice that all possible cases of damaged frame and ACK / NAK must be taken into account
In selective-reject ARQ error control, the only frames retransmitted are those receive a NAK or which time out

1. Stop and Wait

Server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```c
#include <time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/socket.h>
#include <unistd.h>
#include <arpa/inet.h>
typedef struct packet{
char data[1024];
}Packet;
typedef struct frame{
int frame_kind; //ACK:0, SEQ:1 FIN:2
int sq_no;
int ack;
Packet packet;
}Frame;
int main(int argc, char** argv){
if (argc != 2){
printf("Usage: %s <port>", argv[0]);
exit(0);
}
int port = atoi(argv[1]);
int sockfd;
struct sockaddr_in serverAddr, newAddr;
char buffer[1024];
socklen_t addr_size;
int frame_id=0;
Frame frame_recv;
Frame frame_send;
sockfd = socket(AF_INET, SOCK_DGRAM, 0);
memset(&serverAddr, '\0', sizeof(serverAddr));
serverAddr.sin_family = AF_INET;
serverAddr.sin_port = htons(port);
serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
bind(sockfd, (struct sockaddr*)&serverAddr, sizeof(serverAddr));
addr_size = sizeof(newAddr);
while(1){
int f_recv_size = recvfrom(sockfd, &frame_recv, sizeof(Frame), 0, (struct
sockaddr*)&newAddr, &addr_size);
if (f_recv_size > 0 && frame_recv.frame_kind == 1 && frame_recv.sq_no ==
frame_id){
```

```
printf("[+]Frame Received: %s\n", frame_recv.packet.data);
frame_send.sq_no = 0;
frame_send.frame_kind = 0;
frame_send.ack = frame_recv.sq_no + 1;
sendto(sockfd, &frame_send, sizeof(frame_send), 0, (struct
sockaddr*)&newAddr, addr_size);
printf("[+]Ack Send\n");
}else{
printf("[+]Frame Not Received\n");
}
frame_id++;
}
close(sockfd);
return 0;
}
```

client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
typedef struct packet{
char data[1024];
}Packet;
typedef struct frame{
int frame_kind; //ACK:0, SEQ:1 FIN:2
int sq_no;
int ack;
Packet packet;

}Frame;
int main(int argc, char **argv[]){
if (argc != 2){
printf("Usage: %s <port>", argv[0]);
exit(0);
}
int port = atoi(argv[1]);
```

```c
int sockfd;
struct sockaddr_in serverAddr;
char buffer[1024];
socklen_t addr_size;
int frame_id = 0;
Frame frame_send;
Frame frame_recv;
int ack_recv = 1;
sockfd = socket(AF_INET, SOCK_DGRAM, 0);
memset(&serverAddr, '\0', sizeof(serverAddr));
serverAddr.sin_family = AF_INET;
serverAddr.sin_port = htons(port);
serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
while(1){
if(ack_recv == 1){
frame_send.sq_no = frame_id;
frame_send.frame_kind = 1;
frame_send.ack = 0;
printf("Enter Data: ");
scanf("%s", buffer);
strcpy(frame_send.packet.data, buffer);
sendto(sockfd, &frame_send, sizeof(Frame), 0, (struct
sockaddr*)&serverAddr, sizeof(serverAddr));
printf("[+]Frame Send\n");
}
int addr_size = sizeof(serverAddr);
int f_recv_size = recvfrom(sockfd, &frame_recv, sizeof(frame_recv), 0 ,(struct
sockaddr*)&serverAddr, &addr_size);
if( f_recv_size > 0 && frame_recv.sq_no == 0 && frame_recv.ack ==
frame_id+1){
printf("[+]Ack Received\n");
ack_recv = 1;
}else{
printf("[-]Ack Not Received\n");
ack_recv = 0;
}
frame_id++;
}
close(sockfd);
return 0;
```

}

OUTPUT

2. Go_Back ARQ

Reciver.c

```c
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
#include<time.h>
#include<stdlib.h>
#include<ctype.h>
#include<arpa/inet.h>
#define W 5
#define P1 50
#define P2 10
char a[10];
char b[10];
void alpha9(int);
int main()
{
struct sockaddr_in ser,cli;
int s,n,sock,i,j,c=1,f;
```

```
unsigned int s1;
s=socket(AF_INET,SOCK_STREAM,0);
ser.sin_family=AF_INET;
ser.sin_port=6500;
ser.sin_addr.s_addr=inet_addr("127.0.0.1");
bind(s,(struct sockaddr *) &ser, sizeof(ser));
listen(s,1);
n=sizeof(cli);
sock=accept(s,(struct sockaddr *)&cli, &n);
printf("\nTCP Connection Established.\n");
s1=(unsigned int) time(NULL);
srand(s1);
strcpy(b,"Time Out ");
recv(sock,a,sizeof(a),0);
f=atoi(a);
while(1)
{
for(i=0;i<W;i++)
{
recv(sock,a,sizeof(a),0);
if(strcmp(a,b)==0)
{
break;
}
}
i=0;
while(i<W)
{
j=rand()%P1;
if(j<P2)
{
send(sock,b,sizeof(b),0);
break;
}
else
{
alpha9(c);
if(c<=f)
{
printf("\nFrame %s Received ",a);
```

```
send(sock,a,sizeof(a),0);
}
else
{
break;
}
c++;
}
if(c>f)
{
break;
}
i++;
}
}
close(sock);
close(s);
return 0;
}
void alpha9(int z)
{
int k,i=0,j,g;
k=z;
while(k>0)
{
i++;
k=k/10;
}
g=i;
i--;
while(z>0)
{
k=z%10;
a[i]=k+48;
i--;
z=z/10;
}
a[g]='\0';
}
```

Client.c

```c
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
#include<time.h>
#include<stdlib.h>
#include<ctype.h>
#include<arpa/inet.h>
#define W 5
#define P1 50
#define P2 10
char a[10];
char b[10];
void alpha9(int);
int main()
{
struct sockaddr_in ser,cli;
int s,n,sock,i,j,c=1,f;
unsigned int s1;
s=socket(AF_INET,SOCK_STREAM,0);
ser.sin_family=AF_INET;
ser.sin_port=6500;
ser.sin_addr.s_addr=inet_addr("127.0.0.1");
bind(s,(struct sockaddr *) &ser, sizeof(ser));
listen(s,1);
n=sizeof(cli);
sock=accept(s,(struct sockaddr *)&cli, &n);
printf("\nTCP Connection Established.\n");
s1=(unsigned int) time(NULL);
srand(s1);
strcpy(b,"Time Out ");
recv(sock,a,sizeof(a),0);
f=atoi(a);
while(1)
{
for(i=0;i<W;i++)
{
```