به نام خدا

شبکه ۱
پروژه پایانی
نام : پویا نصیری دهج
شماره دانشجویی :۹۸۲۵۹۴۳

پیاده سازی : Selective Repeat ARQ

پس از اجرای برنامه و انتخاب "Selective Repeat ARQ" دو thread اولیه به شبیه سازی میشود

```
int sr_arq = Scanner.nextInt();
boolean SR = (sr_arq == 1);

System.out.print("Enter the number of data to send you want to store : ");
int size_data_sends = Scanner.nextInt();
int[] datas_send =  new int[size_data_sends];

for(int i=0; i < datas_send.length; i++) {
    System.out.print("Enter data[ " + i + " ] : ");
    datas_send[i] = Scanner.nextInt();
}

System.out.println("Enter a Window Size ( size of data send should be divided by Window Size )  : ");

int pack_lenght = Scanner.nextInt();

Scanner.close();
// so here we make a thread Pool to have two thread to execute togeter ( another way is socket)
SharedMemory shared = new SharedMemory();
Sender sender = new Sender(shared, pack_lenght, datas_send, SR);
Receiver receiver = new Receiver(pack_lenght, sender, size_data_sends, shared);

ExecutorService threads = Executors.newFixedThreadPool(2);
threads.execute(receiver);
threads.execute(sender);
threads.shutdown();
```

در واقع thread های sender و receiver همزمان اجرا می شوند .

```
public void ACK(int index) {
    this.elements_is_sent[index] = 1;
    write("Console : Received ACK relay from recevier for packet with index " + index + "\n");
}
  function do work if packet send on out of time
public void NACK(int index) {
    if (SR) {
        Packet Packet = new Packet(index, packet_transmit[index]);
        write("Console : Sender relaying Packet with index " + index + " again because NACK was relayed.\n");
        shared.send(Packet);
    } else {
        for (int i = pointer; i < pointer + len_window; i++) {
            Packet Packet = new Packet(i, packet_transmit[i]);
            write("Console : relaying Packet to buffer with index : " + Packet.index + " and packet " + Packet.pckt
                    + " (GBN).\n");
            shared.send(Packet);
        }
    }
}
```

همان طور که در بالا مشاهده میکنید تفاوت GBN و SR در زمانی است بسته NACK شود. یعنی بسته در زمان مناسب ارسال شود یا نه .
اگر بسته ما ACK شود داخل این متود می رود می رویم و پیاده سازی این متود برای هر دو حالت GBN و SR یکسان میباشد . اما اگر NACK بیاد این متود میشویم .
در این حالت اگر در مود GBN باشیم باید مجددا کل بسته را بفرستیم .ولی اگر در مود SR باشیم باید همان دیتایی که با مشکل روبرو شده است را بفرستیم .

سپس یک بافر برای قفل کردن مراحل نوشتن تا زمانی که پیامی از گیرنده دریافت شود معرفی می شود که آیا یک بسته تایید شده است یا خیر.
در واقع زمانی که یک ترد مشغول نوشتن در بافر است ، ترد دیگر نمیتواند وارد این بخش شود و همین طور برای خواندن .

که در اینجا در تصویر پایین این پیاده سازی را مشاهده میکنیم .

```java
public void send(Packet Packet){
    lock.writeLock().lock();
    Packets.add(Packet);
    lock.writeLock().unlock();
}
//        in this part just read packet and write locked
public Packet read(int index){
    Packet Packet;
    lock.readLock().lock();
    try {
        Packet = Packets.get(index);
    } catch (IndexOutOfBoundsException e) {
        Packet = null;
    }
    lock.readLock().unlock();
    return Packet;
}
```

در کلاس Receiver مکانیزم پیاده سازی timeout را به شکل زیر اجرا کرده ایم یک عدد رندوم بین ۰ تا ۱۰۱ تولید میکند اگر این عدد رندوم کوچکتر از ۵۰ باشد timeout رخ میدهد و NACK فراخوانی میشود و اگر بزرگتر باشد timeout رخ نمیدهد و ACK فراخوانی میشود .در پایین ای قسمت را مشاهده میکنیم .

```java
if (Packet != null){
    int time_out = new Random().nextInt(101);
    if (time_out < 50){
        write("Console : Received Packet with index " + Packet.index
        + " from buffer , relaying NACK to sender.\n");
        sender.NACK(Packet.index);
    } else {
        packet_transmit[Packet.index] = Packet.pckt;
        is_recevied[Packet.index] = 1;
        write("Console : Received Packet with index " + Packet.index
        + " from buffer , relaying ACK to sender.\n");
        sender.ACK(Packet.index);
    }
    point_to_sharedMe = point_to_sharedMe + 1;
} else {
    write("Console : Receiver has not added index " + point_to_sharedMe + " yet.\n");
}
```

ورودی :

```
Scanner Scanner = new Scanner(System.in);
System.out.println( "\n"+"1 : to run Selective Repeat ARQ  " +
        "\n" + "2 : to run GBN  "+"\nEnter your choice : " );
int sr_arq = Scanner.nextInt();
boolean SR = (sr_arq == 1);

System.out.print("Enter the number of data to send you want to store : ");
int size_data_sends = Scanner.nextInt();
int[] datas_send =  new int[size_data_sends];

for(int i=0; i < datas_send.length; i++) {
    System.out.print("Enter data[ " + i + " ] : ");
    datas_send[i] = Scanner.nextInt();
}
```

در این فرایند ابتدا از کاربر سوال میشود که برنامه در کدام حالت GBN یا SR اجرا شود.
سپس طول آن دیتاهایی را که باید ارسال شود را دریافت میکند.
حال این دیتاها از کاربر دریافت میشود .

```
System.out.println("Enter a Window Size ( size of data send should be divided by Window Size )  : ");

int pack_lenght = Scanner.nextInt();
```

سپس طول بسته هایی که قرار است دیتاها توسط این بسته ها ارسال شوند را دریافت میکند.

خروجی :
مراحل برنامه و بسته هایی که ارسال و دریافت شده اند را به ترتیب در sender.txt و receiver.txt نمایش میدهد .

تفاوت GBN و SR

GBN

در پروتکل Go-Back-N، اگر فریم ارسال شده مشکوک باشد، همه فریم ها از بسته گم شده به آخرین بسته ارسال شده مجددا ارسال می شوند.

n اندازه پنجره فرستنده پروتکل Go-Back-N است.

1 اندازه پنجره گیرنده پروتکل Go-Back-N است.

پروتکل Go-Back-N پیچیدگی کمتری دارد.

در پروتکل Go-Back-N، نه فرستنده و نه در گیرنده نیاز به مرتب سازی ندارند.

در پروتکل Go-Back-N، اگر Receives یک بسته خراب دریافت کند، کل پنجره مجددا ارسال می شود.

SR

در پروتکل تکرار انتخابی، فقط آن فریم هایی که مشکوک هستند، دوباره ارسال می شوند.

اندازه پنجره فرستنده پروتکل Repeat انتخابی نیز N است.

اندازه پنجره گیرنده پروتکل Repeat انتخابی N است.

پروتکل تکرار انتخابی پیچیده تر است.

در پروتکل Repeat انتخابی، سمت گیرنده برای مرتب سازی فریم ها نیاز به مرتب سازی دارد.

در پروتکل Repeat انتخابی، اگر Receives یک بسته خراب دریافت کند، فوراً یک تأیید منفی ارسال می کند و از این رو فقط بسته انتخابی دوباره ارسال می شود.

مثال از ورودی و خروجی :

ورودی :

```
1 : to run Selective Repeat ARQ
2 : to run GBN
Enter your choice :
1
Enter the number of data to send you want to store : 8
Enter data[ 0 ] : 1
Enter data[ 1 ] : 3
Enter data[ 2 ] : 5
Enter data[ 3 ] : 7
Enter data[ 4 ] : 9
Enter data[ 5 ] : 11
Enter data[ 6 ] : 13
Enter data[ 7 ] : 17
Enter a Window Size ( size of data send should be divided by Window Size )  :
2
Process was completed with Window Size : 2 -and the used packet : { 1 3 5 7 9 11 13 17 }
```

خروجی :

.Console : Receiver has not added index 0 yet

] : Receiver
0
0
0
0
0
0
0
0
[
.Console : Receiver has not added index 0 yet

] : Receiver
0
0
0
0
0
0
0
0
[
.Console : Received Packet with index 0 from buffer , relaying NACK to sender

] : Receiver
0
0
0
0
0
0
0
0
[
.Console : Received Packet with index 1 from buffer , relaying ACK to sender

] : Receiver
0
3
0
0
0
0
0
0
[
.Console : Received Packet with index 0 from buffer , relaying ACK to sender

] : Receiver
1
3
0
0
0
0
0
0
[
.Console : Receiver has not added index 3 yet

] : Receiver
1
3
0
0
0
0
0
0
[
.Console : Receiver has not added index 3 yet

] : Receiver
1
3
0
0
0
0
0
0
[
.Console : Receiver has not added index 3 yet

] : Receiver
1
3
0
0
0
0
0
0
[
.Console : Received Packet with index 2 from buffer , relaying ACK to sender

] : Receiver
1
3
5
0
0
0
0
0
[
.Console : Received Packet with index 3 from buffer , relaying ACK to sender

] : Receiver
1
3
5
7
0
0
0
0
[
.Console : Received Packet with index 4 from buffer , relaying ACK to sender

] : Receiver

1
3
5
7
9
0
0
0
[
.Console : Received Packet with index 5 from buffer , relaying ACK to sender

] : Receiver
1
3
5
7
9
11
0
0
[
.Console : Received Packet with index 6 from buffer , relaying ACK to sender

] : Receiver
1
3
5
7
9
11
13
0
[
.Console : Received Packet with index 7 from buffer , relaying ACK to sender

] : Receiver
1
3
5
7
9
11
13
17
[