实验一

一、实验目的（明确学生应达到的基本能力要求）

1.熟悉数据链路层协议，并能用 PROMELA 语言正确描述

2.掌握用 SPIN 验证协议的方法

二、实验原理

对于给定的一个使用 PROMELA 描述的协议系统，SPIN 可以对其执行任意的模拟，也可以生成一个 C 代码程序，然后对该系统的正确性进行有效检验，并报告系统中出现的死锁，无效的循环，未定义的接受和标记不完全等情况。

三、实验仪器

PC 机

四、实验内容及步骤

实验内容：熟悉数据链路层协议，并将协议条件修改为：报文和应答均会出错，且都丢失，接受方没有无限接受能力。

实验步骤：

1.熟悉数据链路层协议，并进行相应修改

2.用 PROMELA 语言描述协议

3.用 SPIN 对协议进行验证

五、实验结果

1. 熟悉数据链路层协议

对于题目所描述的简单的停等协议，即 RDT 3.0 协议，这个协议的条件是报文和应答均有可能出错，但都不会出现报文丢失，且接收方没有无限接受能力。该协议的主要过程为：发送方发送数据报，等待应答，如果是肯定应答则发送下一帧，如果是否定应答或应答帧错则重发；接收方接收报文，如果是期望的报文则发送肯定应答。否则发送否定应答，给报文加序号。

在简单的停等协议的基础上，根据实验要求，我们多考虑了一种情况，即报文丢失(Miss)。协议具体内容为，在 RDT 3.0 的基础上，在接受方和发送方的两个进程中添加收到 Miss 的情况反馈，当发送方收到 Miss 时，重发上一个数据；当接受方接收到 Miss 时，则直接跳过。

2. 用 PROMELA 语言描述协议

在代码部分，其主题与书中代码一致，但加入了 Miss 的情况，整体分为发送方和接收方两个进程。

在发送方，当发送方收到 Miss 时，选择 goto again，重发数据；

在接收方进程里，大体分为接收到正常的报文、接收有误码的报文、Err 和丢失 Miss。在接收到正常的报文中，如果出现 ACK 丢失，则和 Err 的情况一样，但当接收方接收到 Miss 时，则直接跳过。

修改后的代码具体如下：

```
#define MAXSEQ 5
mtype={Msg,Ack,Nak,Err,Miss};
chan SenderToReceiver=[1]of{mtype,byte,byte};
chan ReceiverToSender=[1]of{mtype,byte,byte};
proctype SENDER(chan InCh,OutCh)
{
byte SendData;
byte SendSeq;
byte ReceivedSeq;
SendData=0;
SendData=0;
do
::skip
again: if
::OutCh!Msg(SendData,SendSeq)
::OutCh!Err(0,0)
::OutCh!Miss(0,0)
fi;
if
::timeout -> goto again
::InCh?Miss(0,0)-> goto again
::InCh?Err(0,0)-> goto again
::InCh?Nak(ReceivedSeq,0)->
end1: goto again
::InCh?Ack(ReceivedSeq,0)->
if
::(ReceivedSeq== SendSeq)->
SendSeq = (SendSeq+1)%MAXSEQ;
SendData = (SendData+1)%MAXSEQ;
::(ReceivedSeq!=SendSeq)->
end2: goto again
fi;
```

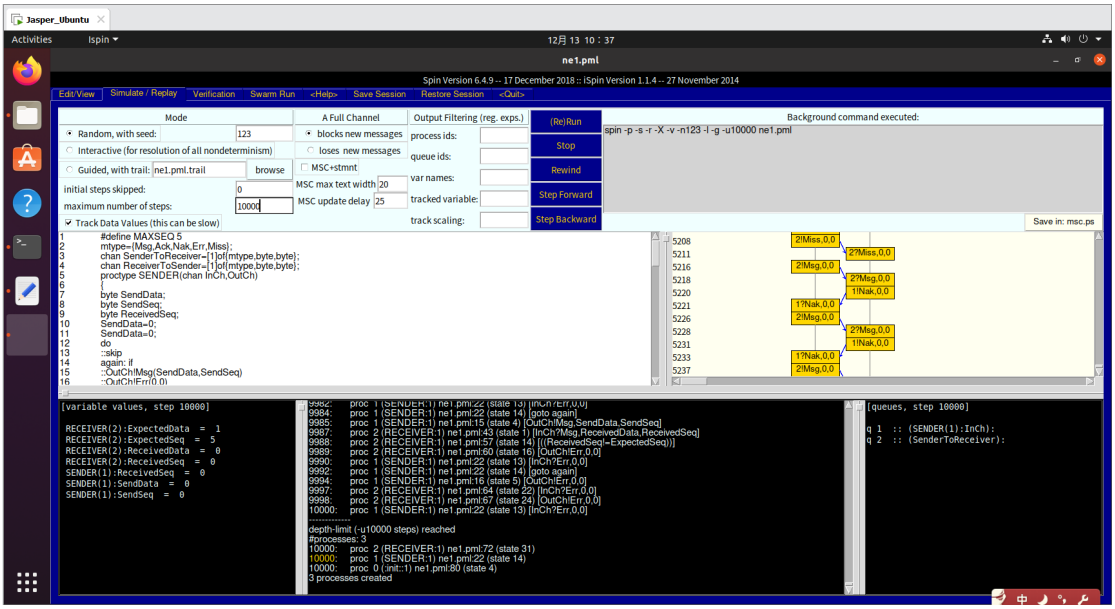
```
fi;
od;
}
proctype RECEIVER(chan InCh,OutCh)
{
byte ReceivedData;
byte ReceivedSeq;
byte ExpectedData=0;
byte ExpectedSeq;
do
::InCh?Msg(ReceivedData,ReceivedSeq)->
if
::(ReceivedSeq==ExpectedSeq)->
progress: ExpectedSeq=1+ExpectedSeq;
ExpectedData=(ExpectedData+1)%MAXSEQ;
if
::OutCh!Miss(0,0)
ExpectedSeq=ExpectedSeq-1;
ExpectedData=(ExpectedData-1)%MAXSEQ;
::OutCh!Ack(ReceivedSeq,0)
::OutCh!Err(0,0)
ExpectedSeq=ExpectedSeq-1;
ExpectedData=(ExpectedData-1)%MAXSEQ;
fi;
::(ReceivedSeq!=ExpectedSeq)
if
::OutCh!Nak(ReceivedSeq,0)
::OutCh!Err(0,0)
::OutCh!Miss(0,0)
fi;
fi;
::InCh?Err(0,0)
if
::OutCh!Nak(ReceivedSeq,0)
::OutCh!Err(0,0)
::OutCh!Miss(0,0)
fi;
::InCh?Miss(0,0)->skip;
od;
}
init
{
atomic
{
```

```
77. run SENDER(ReceiverToSender,SenderToReceiver);
78. run RECEIVER(SenderToReceiver,ReceiverToSender);
79. }
80. }
```

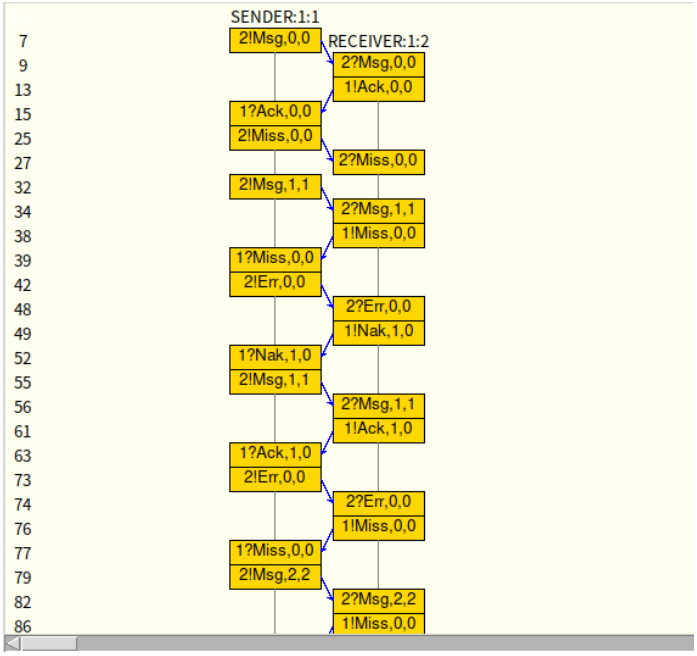
3. 用 Spin 进行协议验证

（1）模拟运行

在 Edit/view 界面，打开事先保存的代码，之后进入 Simulate/Replay 界面，选择 Random，with seed，然后点击 run，观察运行结果。



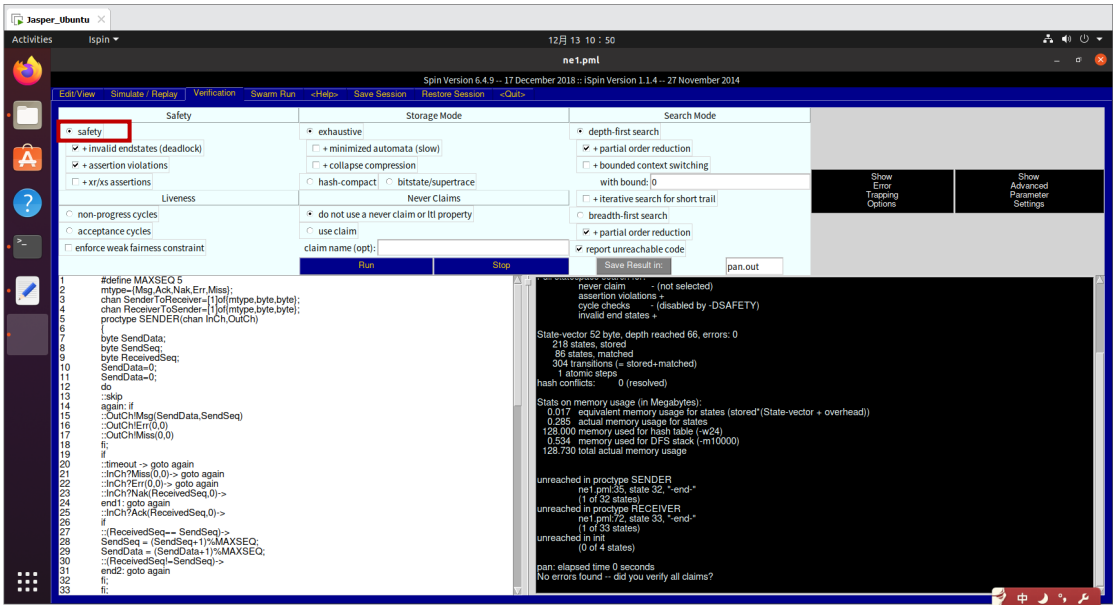
其中，放大后的实用停等协议的模拟运行结果如图：

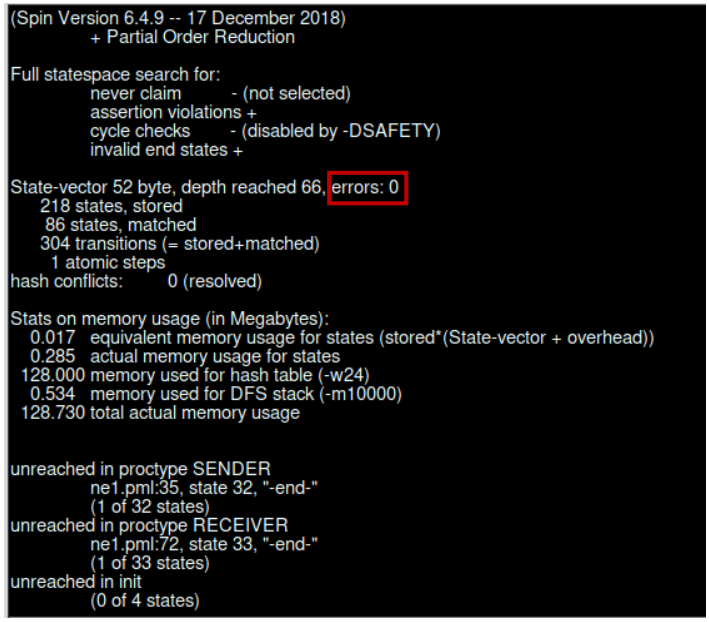


从时序图可以看到第一次进行了一轮正常的收发消息，第二轮 SENDER()丢失了报文，第三轮 RECEIVER()给 SENDER 的 ACK 丢了。从时序图可知，协议修改结果正确。

（2）一般性验证

点击 Verification，选择 Safety，点击 Run。

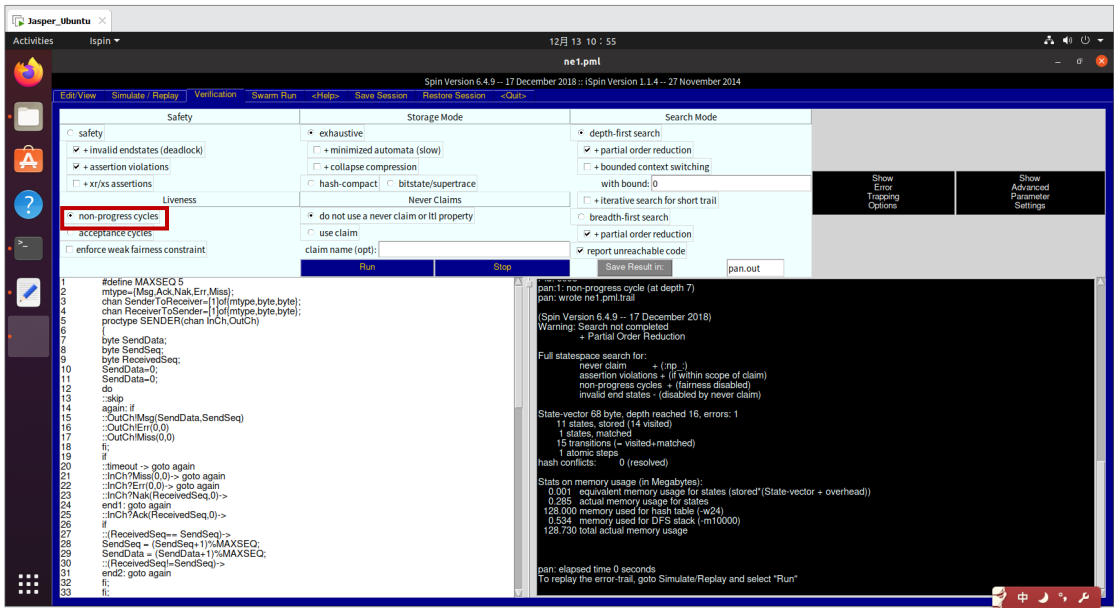


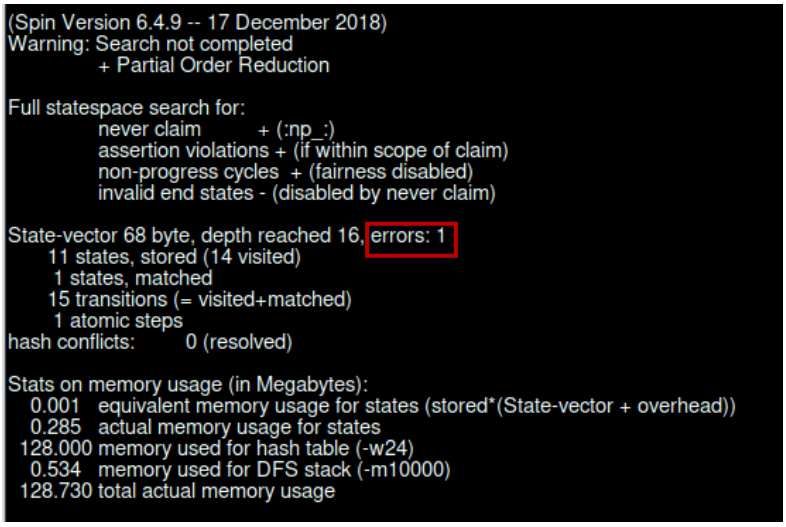


由图中可以看出，errors：0，说明无一般性验证错误，即该协议通过一般性验证

（3）无进展循环验证

选择 Liveness 下的 non-progress cycles，点击 Run





从上图可以得出，errors=1，验证不通过。可以找出验证不通过的原因是由于发送方发送错误报文后不会改变发送序号，导致发送进程中的 process 语句不能执行，接收方受到错误报文后也不会改变接收序号，导致接收方的 process 语句不能执行。但是这是协议要求的情况，是本身协议包含的内容，所以，此处的验证不通过，我们不能把它看做是协议的错误，而是在描述协议正确性要求时，process 标签设置不当。

（4）人为加入错误

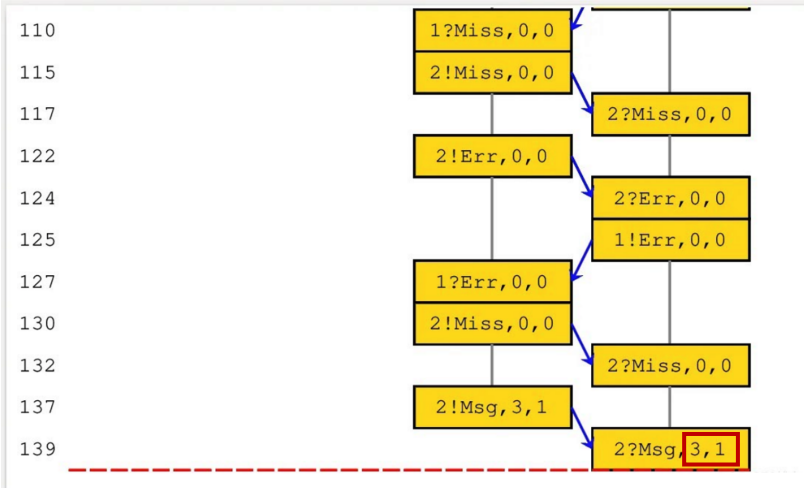
SENDER()部分：goto again 改成 goto progress，通过对协议的修改，再次

模拟运行进行验证。

```
Open   *ne1.pml  ~/Documents   Save

 6 {
 7 byte SendData;
 8 byte SendSeq;
 9 byte ReceivedSeq;
10 SendData=0;
11 SendData=0;
12 do
13 ::skip
14 again: if
15 ::OutCh!Msg(SendData,SendSeq)
16 ::OutCh!Err(0,0)
17 ::OutCh!Miss(0,0)
18 fi;
19 if
20 ::timeout -> goto again
21 //::InCh?Miss(0,0)-> goto again
22 ::InCh?Miss(0,0)-> goto progress
23 ::InCh?Err(0,0)-> goto again
24 ::InCh?Nak(ReceivedSeq,0)->
25 end1: goto again
26 ::InCh?Ack(ReceivedSeq,0)->
27 if
28 ::(ReceivedSeq== SendSeq)->
29 SendSeq = (SendSeq+1)%MAXSEQ;
30 SendData = (SendData+1)%MAXSEQ;

Plain Text   Tab Width: 8   Ln 22, Col 33   INS
```



设定最多跑 10000 步，但是 139 步即停止。发现 Msg 序号出现了 3，显然错误。

实验二

一、实验目的（明确学生应达到的基本能力要求）

1.熟悉 AB 协议，并能用 PROMELA 语言正确描述

2.掌握用 SPIN 验证协议的方法

二、实验原理

对于给定的一个使用 PROMELA 描述的协议系统，SPIN 可以对其执行任意的模拟，也可以生成一个 C 代码程序，然后对该系统的正确性进行有效检验，并报告系统中出现的死锁，无效的循环，未定义的接受和标记不完全等情况。

三、实验仪器

PC 机

四、实验内容及步骤

实验内容：熟悉 AB 协议，并根据状态图用 PROMELA 语言描述该协议

实验步骤：

1.熟悉 AB 协议

2.用 PROMELA 语言描述协议

3.用 SPIN 对协议进行验证

五、实验结果

1. 熟悉 AB 协议

AB 协议（Alternating Bit Protocol）是一种简单的数据传输协议，通常用于计算机网络中。其基本思想是发送方和接收方交替发送和接收数据，以确保数据的可靠传输。

AB 协议有两个状态，分别是 0 和 1。这两个状态表示数据包的两个可能的取值。发送方和接收方交替发送和接收数据包。例如，如果发送方发送了一个状态为 0 的数据包，接收方收到后会确认，并等待发送方发送一个状态为 1 的数据包。发送方在收到确认后才能继续发送下一个状态为 1 的数据包。这样一直交替进行。接收方在成功接收到一个数据包后，会发送一个确认（ACK）给发送方。如果发送方在一定时间内未收到确认，则会认为数据包丢失，需要重新发送。发送方在发送数据包后启动一个定时器，如果在规定时间内未收到确认，就会重新发送相同的数据包。这确保了数据的可靠性。为防止过多的数据堆积在网络中，

可以引入流程控制机制，限制发送方一次发送的数据包数量。

在本次实验中，题目所给的 AB 协议状态转换图中没有 miss 状态，只有收发报文和 err，可以确定的是题文中所给的协议是信道有误码、无丢失下的 AB 协议。

发送方初始状态处于 S5 状态，此时发送端发送报文 a0，继而处于 S4 等待应答。AB 协议只有两种序号的报文，回送报文 b0 和 b1 分别是接收方对报文对 a0 和 a1 的确认。

当接收方处于 S4 并接收到报文， 如果是 a0 或者是 a1 则转向 S1 状态，并发送报文 b1 且转到 S2 状态；如果是 Err 则转向 S5 状态，并发送报文 b0 且转到 S4 状态。

发送方处于等待状态 S4 收到回送报文 b0 或者 b1 后，则转向 S1 状态，并发送报文 a1 且转到 S2 状态。

接收方在 S2 状态下，如果是 a0 则转向 S3，如果是 a1 则转向 S1 状态，并发送报文 b1 且转到 S2 状态；如果是 Err0 则转向 S5 状态，并发送报文 b0 且转到 S4 状态。如果接收方收到的是报文 b0，说明发送方刚刚发给接收方的报文 a1 接收方没有能够收到，接收方会再次重传报文 a1，如果接收方收到回应报文 b1 后，则会重复这样的过程（一直给接收方发送报文 a1），直到出现意外情况打破这种平衡：收到的回应报文错误的消息，回到初始状态 S5。

2. 用 PROMELA 语言描述协议

```
#define MAXSEQ 5
#define timeout 1
#define a 1
#define b 2
chan AtoB = [1] of {byte,byte}
chan BtoA = [1] of {byte,byte}
chan guard

proctype TA()
{
S1: if
::AtoB!a,1->
goto S2;
fi;
```

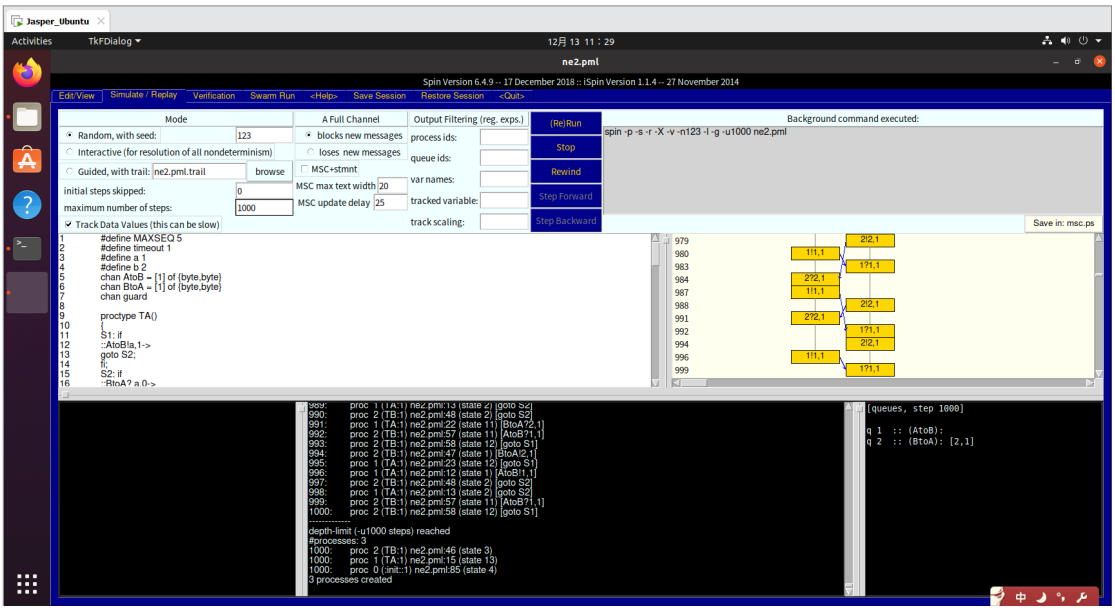
```
S2: if
::BtoA? a,0->
goto S5;
::BtoA? a,1->
goto S5;
::BtoA? b,0->
goto S3;
::BtoA? b,1->
goto S1;
fi;
S3: if
::BtoA?a,1->
goto S2;
fi;
S4: if
::BtoA?b,1->
goto S1;
::BtoA?b,0->
goto S1;
::BtoA?a,0
goto S5;
::BtoA?a,1->
goto S5;
fi;
S5: if
::AtoB!a,0->
goto S4;
fi;
}
proctype TB()
{
S1: if
::BtoA!b,1->
goto S2;
fi;
S2: if
::AtoB?b,0->
goto S5;
::AtoB?b,1->
goto S5;
::AtoB?a,0->
goto S3;
::AtoB?a,1->
goto S1;
```

```
fi;
S3: if
::AtoB?b,1->
goto S2;
fi;
S4: if
::AtoB?a,0->
goto S1;
::AtoB?a,1->
goto S1;
::AtoB?b,0->
goto S5;
::AtoB?b,1->
goto S5;
fi;
S5: if
::BtoA!b,0->
goto S4;
fi;
}
init
{
atomic
{
run TA();run TB();
}
}
```

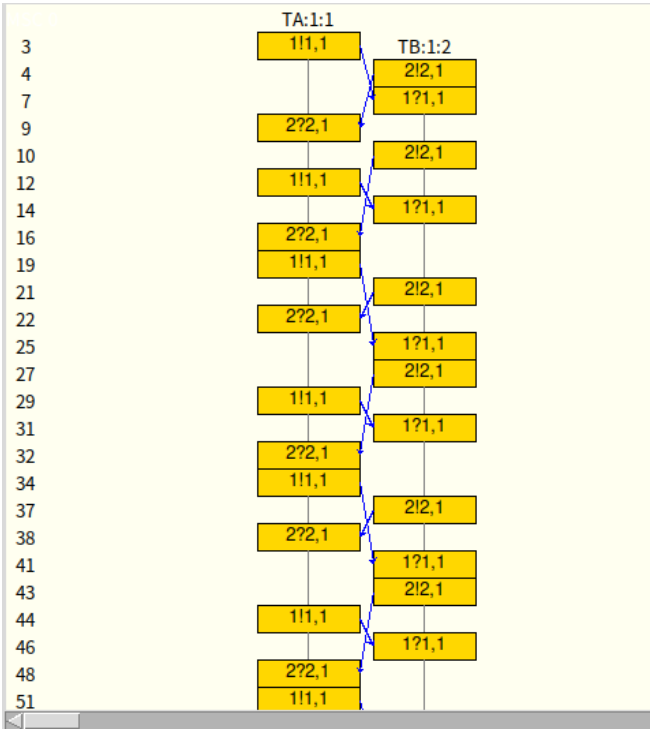
3. 用 spin 进行协议验证

（1）模拟运行

在 Edit/view 界面,打开事先保存的代码,之后进入 Simulate/Replay 界面，选择 Random，with seed，并设置 maximum number of steps 为 1000，缩短检验时间，然后点击 run，观察运行结果。



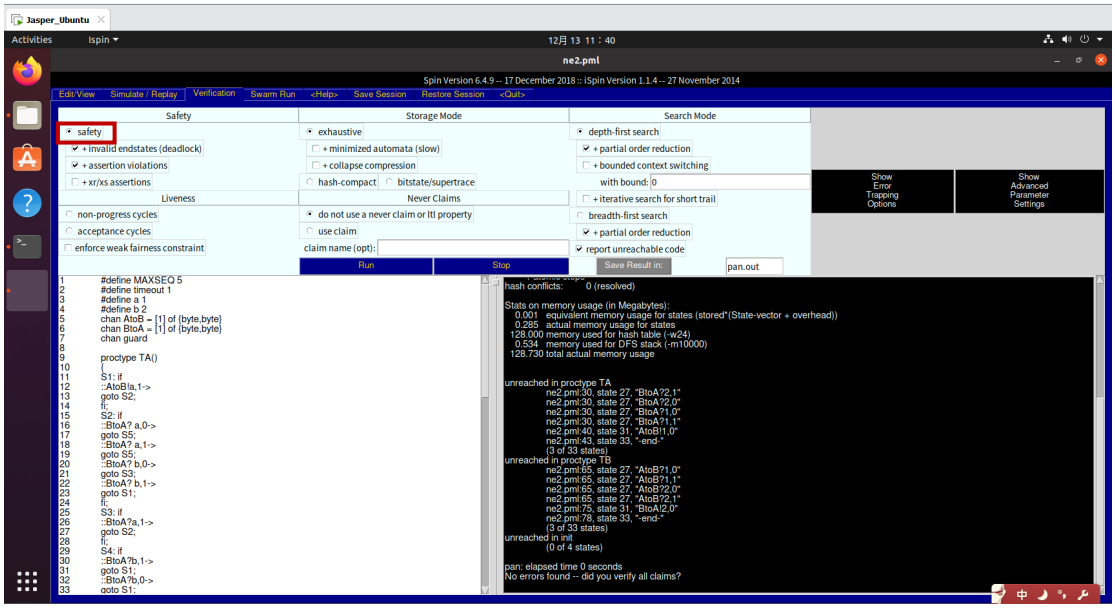
其中，放大后的 AB 协议的模拟运行结果如图：

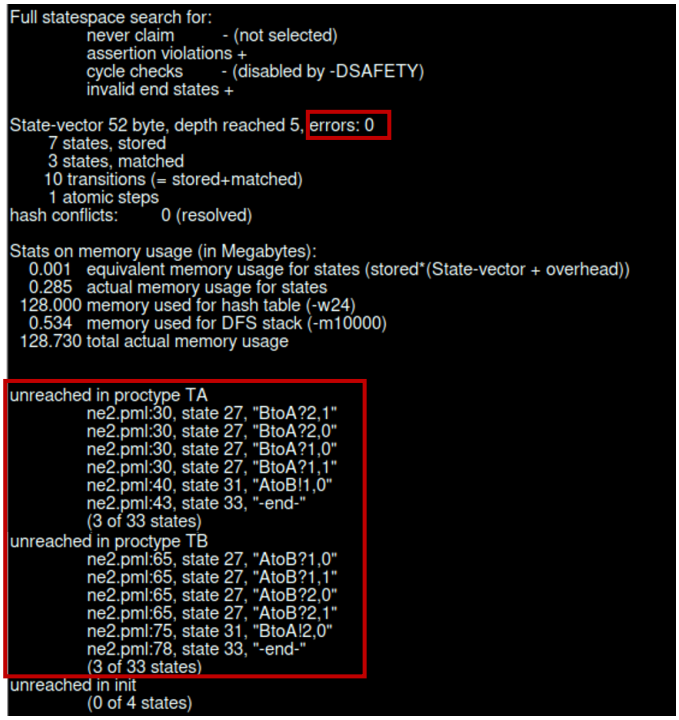


从时序图可知，协议结果正确。

（2）一般性验证

点击 Verification，选择 Safety，点击 Run。

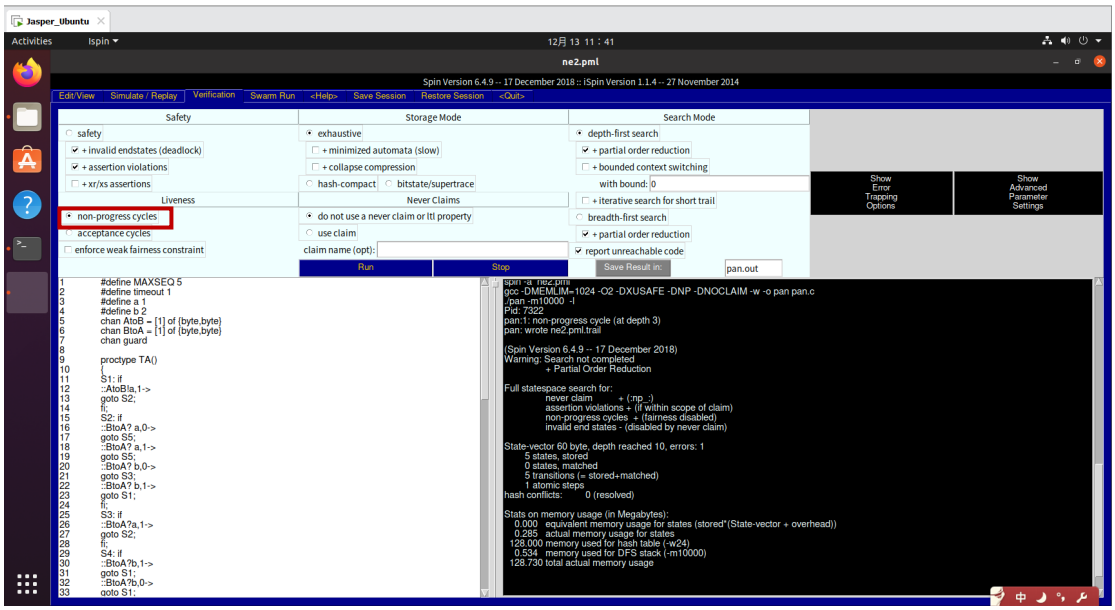




由图中可以看出，errors：0，说明无一般性验证错误，即该协议通过一般性验证。并且可知，发送端 TA 和接收端 TB 都是循环执行。

（3）无进展循环验证

选择 Liveness 下的 non-progress cycles，点击 Run




```
pan:1: non-progress cycle (at depth 3)
pan: wrote ne2.pml.trail

(Spin Version 6.4.9 -- 17 December 2018)
Warning: Search not completed
            + Partial Order Reduction

Full statespace search for:
            never claim         + (:np_:)
            assertion violations + (if within scope of claim)
            non-progress cycles  + (fairness disabled)
            invalid end states - (disabled by never claim)

State-vector 60 byte, depth reached 10, errors: 1
       5 states, stored
       0 states, matched
       5 transitions (= stored+matched)
       1 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
   0.000   equivalent memory usage for states (stored*(State-vector + overhead))
   0.285   actual memory usage for states
 128.000 memory used for hash table (-w24)
   0.534   memory used for DFS stack (-m10000)
 128.730 total actual memory usage
```

由上述结果可知，A 获取的每一个报文至少有一次是正确的，而 B 接收的每一个报文至多有一次是正确的。

实验三

一、实验目的（明确学生应达到的基本能力要求）

1.熟悉 GO-BACK-N 协议，并能用 PROMELA 语言正确描述

2.掌握用 SPIN 验证协议的方法

二、实验原理

对于给定的一个使用 PROMELA 描述的协议系统，SPIN 可以对其执行任意的模拟，也可以生成一个 C 代码程序，然后对该系统的正确性进行有效检验，并报告系统中出现的死锁，无效的循环，未定义的接受和标记不完全等情况。

三、实验仪器

PC 机

四、实验内容及步骤

实验内容：熟悉并用 PROMELA 语言描述 GO-BACK-N 协议

实验步骤：

1.熟悉 GO-BACK-N 协议

2.用 PROMELA 语言描述协议

3.用 SPIN 对协议进行验证

五、实验结果

1. 熟悉 GO-BACK-N 协议

Go-Back-N (GBN) 协议是一种数据链路层的协议，用于在不可靠的通信信道上进行数据的可靠传输。它是一种基于窗口的协议，允许发送方连续发送多个数据包而无需等待确认，从而提高了通信的效率。GBN 协议的核心思想是使用窗口来管理发送方和接收方之间的数据流。

GBN 协议使用滑动窗口的概念，其中有一个发送窗口和一个接收窗口。发送窗口表示允许发送的未确认数据包的范围，而接收窗口表示允许接收的未确认数据包的范围。发送方可以连续发送窗口内的多个数据包，而无需等待确认。这提高了通信效率，允许在等待确认期间继续发送数据。接收方收到数据包后，会发送一个累积确认，表示成功接收了该数据包以及该数据包之前的所有数据包。这样，发送方就知道哪些数据包已经成功接收，可以将发送窗口向前滑动。如果发送方在一定时间内未收到确认，就会认为窗口内的数据包有可能丢失，因此会重新发送窗口内的所有数据包。接收方负责按照顺序将数据包交付给上层应用，即

使接收到的数据包是乱序的，也会按照正确的顺序交付。窗口的大小是有限的，过小的窗口可能导致通信效率低下，而过大的窗口可能引入更多的错误和丢失的数据包。

2. 用 PROMELA 语言描述协议

ACK 的接收:收到序号为 n 的确认分组,则表明 n 之前的报文都正确收到,此时发送方将窗口基序号修改为 n。对序号为 n 的分组的确认用于累计确认。

超时事件:发送方使用一个定时器。如果超时发生,发送方将重发所有已发送过但还未被确认过的分组。如果收到一个有效 ACK,则定时器被重新启动。

GBN 协议的接收方动作很简单,如果一个序号为 n 的分组被正确收到并按序,则接收方为分组 n 发送一个 ACK,并将分组中的数据交付到上层,在所有其他情况下,接收方丢弃该分组并为最近按序接收的分组重发 ACK。

代码具体如下：

```
#define MAXSEQ 5
#define SIZE 4
mtype={Msg,Ack,Nak,Err,Miss};
chan SenderToReceiver=[SIZE] of {mtype,byte,byte};
chan ReceiverToSender=[1]of{mtype,byte,byte};
proctype SENDER(chan InCh,OutCh)
{
byte SendData,SDt;
byte SendSeq,SSt;
byte ReceivedSeq;
SendData=0;
SendSeq=0;
byte count = 0;
do

::skip->
timet: count = count - (SDt-SendData);
SSt = SendSeq;
SDt=SendData;
again: if
::OutCh!Msg(SDt,SSt)
::OutCh!Err(0,0)
::OutCh!Miss(0,0)
fi;
count = count+1;
```

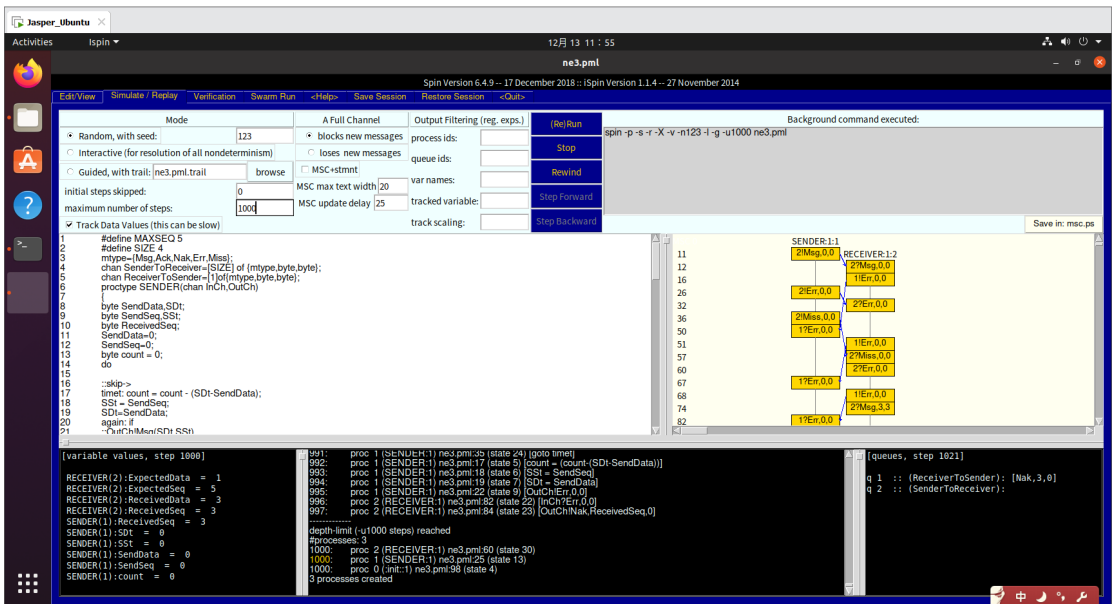
```
26. if
27. :: count<4 ->
28. SDt = SDt+1;
29. SSt = SSt+1;
30. goto again;
31. :: count>=4->
32. count=4;skip;
33. fi;
34. if
35. ::timeout -> goto timet
36. ::InCh?Miss(0,0)->
37. end3: count = count-1; goto again
38. ::InCh?Err(0,0)->
39. end2: count = count -1;goto again
40. ::InCh?Nak(ReceivedSeq,0)->
41. end1: count = count -1;goto again
42. ::InCh?Ack(ReceivedSeq,0)->
43. if
44. ::(ReceivedSeq== SendSeq)->
45. SendSeq = (SendSeq+1)%MAXSEQ;
46. SendData = (SendData+1)%MAXSEQ;
47. count = count-1;
48. ::(ReceivedSeq!=SendSeq)->
49. end4: goto again
50. fi;
51. fi;
52. od;
53. }
54. proctype RECEIVER(chan InCh,OutCh)
55. {
56. byte ReceivedData;
57. byte ReceivedSeq;
58. byte ExpectedData=0;
59. byte ExpectedSeq=0;
60. do
61. ::InCh?Msg(ReceivedData,ReceivedSeq)->
62. if
63. ::(ReceivedSeq==ExpectedSeq)->
64. ExpectedSeq=1+ExpectedSeq;
65. ExpectedData=(ExpectedData+1)%MAXSEQ;
66. if
67. ::OutCh!Miss(0,0)
68. ExpectedSeq=ExpectedSeq-1;
69. ExpectedData=(ExpectedData-1)%MAXSEQ;
```

```
::OutCh!Ack(ReceivedSeq,0)
::OutCh!Err(0,0)
ExpectedSeq=ExpectedSeq-1;
ExpectedData=(ExpectedData-1)%MAXSEQ;
fi;
::(ReceivedSeq!=ExpectedSeq)
if
::OutCh!Nak(ReceivedSeq,0)
::OutCh!Err(0,0)
::OutCh!Miss(0,0)
fi;
fi;
::InCh?Err(0,0)
if
::OutCh!Nak(ReceivedSeq,0)
::OutCh!Err(0,0)
::OutCh!Miss(0,0)
fi;
::InCh?Miss(0,0)->skip;
od;
}
init
{
atomic
{
run SENDER(ReceiverToSender,SenderToReceiver);
run RECEIVER(SenderToReceiver,ReceiverToSender);
}
}
```

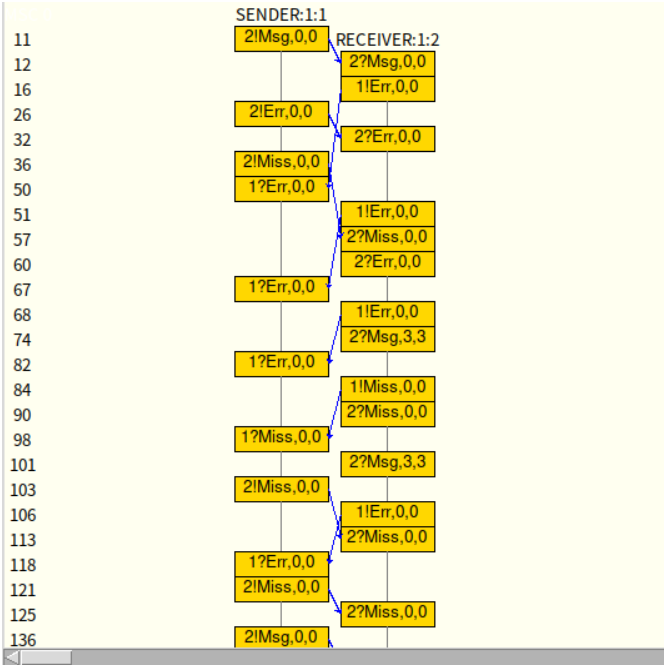
3. 用 Spin 进行协议验证

（1）模拟运行

在 Edit/view 界面，打开事先保存的代码，之后进入 Simulate/Replay 界面，选择 Random，with seed，然后点击 run，观察运行结果。



其中，放大后的 GO-BACK-N 协议的模拟运行结果如图：



从时序图可知，协议结果正确。