Exp 13：Congestion Avoidance 壅塞回避

目的：了解TCP 协议中的壅塞(Congestion)算法。

摘要：此实验是在说明，如何利用壅塞算法去解决网络壅塞问题。也可通过MDDL语言，学生可以学习如何去执行这个算法。

时间：3 小时。

一、网络拓扑

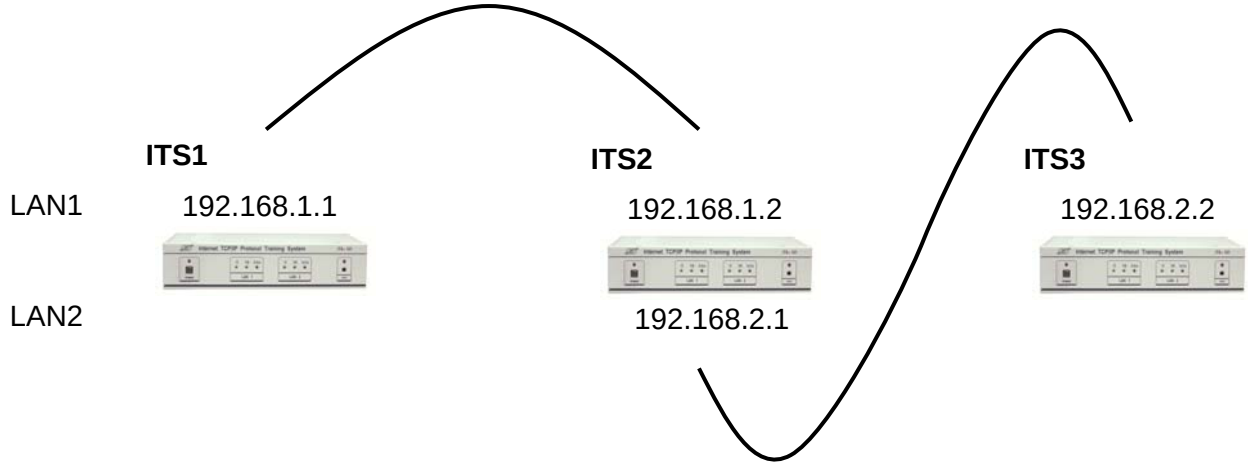


图13.1

二、技术背景

壅塞是由于报文过度负载于一个或多个绕送节点 (例如：路由器)，而引起的严重延迟的状况。当壅塞发生时，除了延迟增加外路由器也会开始占住报文一直到它可以发送。为了避免壅塞的现象，TCP协议标准建议了使用两种技术： 慢起动(Slow-Start) 和倍减(Multiplicative Decrease)，他们是互相有关连性而且容易使用的技术。基本上，对每一个联机(connection)来说，TCP必须记得接收者的窗口大小 (例如：缓冲区还有多少空间，会被记在acknowledgement封包里)，而为了要控制壅塞，或者说防止壅塞所带来的错误，TCP 建立起一机制称做Congestion Window Limit或叫壅塞窗口(Congestion Window, CWND)。当壅塞发生时，它会限制发送端发送的数据必须小于接收端的缓冲区大小。

Allowed_window = min(receiver_advertisement, congestion_window)

了解Congestion Window Limit后，就可以来解释Slow-Start这个机制：每当开启一个新的TCP连结的时候，或是经过一段壅塞期之后想增加数据包的流量，CWND的大小都必须由1个报文开始，收到acknowledgement报文，也就是一个RTT之后，CWND的大小会增加为两倍，并一次一次的递增上去。

而Multiplicative Decrease则是：在联机中，传输顺畅的情形下，壅塞窗口是以Slow-Start的方式指数成长，但只要一发现有数据报遗失后，壅塞窗口将会被减为最小，再重新开始加大。为了避免CWND的尺寸快速的扩张导致的壅塞，在壅塞发生后，TCP会将CWND一半的大小设为Slow-Start Threshold (SSTHRESH)，当壅塞窗口重新成长时，在当未到达SSTHRESH时，以指数成长，但过SSTHRESH 后将进入congestion avoidance状态，以降低传输速率。在壅塞回避期间，CWND 的大小将一次以1个报文大小的方式线性成长，直到所有的报文都被acknowledged。

三、实验步骤

1、网络拓朴

1）在Hubox 上将网络连接如图13.2所示。

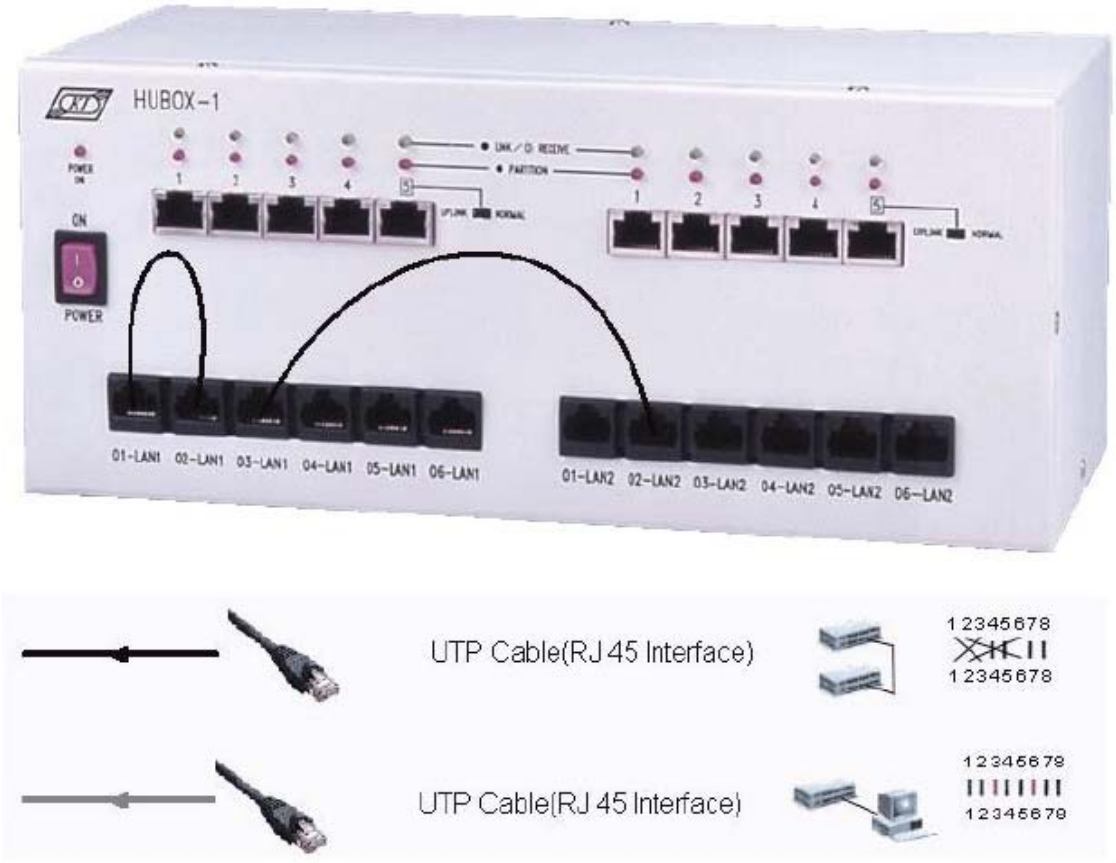


图13.2

2、设置 Host 和 Gateway

1）执行 **XCLIENT.BAT，**打开 ITS 应用软件 KCodes Network Explorer。

2）打开网络封包浏览器 Network Message Browser。

ITS1 (Host) 设置如下：

3）根据拓扑结构. 定义 Interface 1 的 IP 地址为“**192.168.1.1**” 子网掩码设为“255.255.255.0”MTU 设为“1500”。然后点击“**Add new routing entry”按钮**见（图 13.3）。

4）定义 Destination 为“**192.168.2.0**”, MASK 为“**255.255.255.0**”, Gateway 为“**192.168.1.2**” (见图 13.4)最后点击 **Update** 按钮。

5）模式选择“ **Host”**, 之后点击“ **Set & Close** ”按钮。

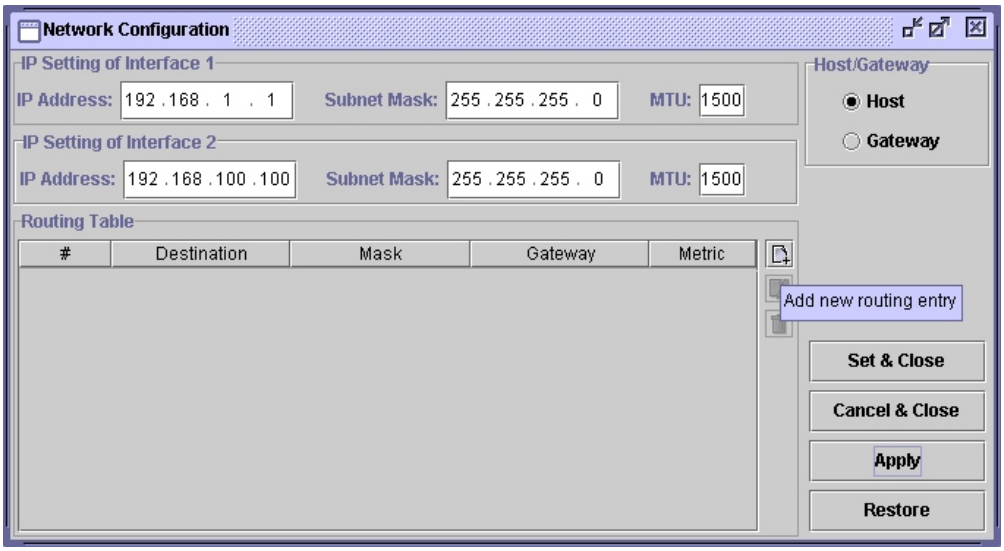


图 13.3

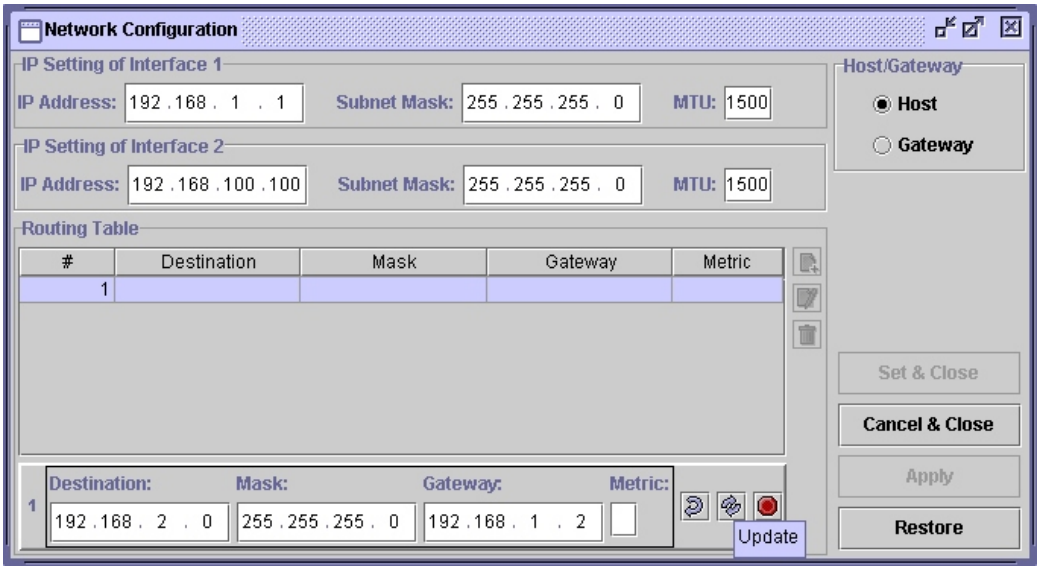


图 13.4

ITS3 (Host) 设置如下：

6）根据拓扑结构定义 Interface 1 的 IP 地址为“**192.168.2.2**”子网掩码设为“255.255.255.0”MTU 设为“1500”。然后点击“**Add new routing entry”按钮。**

7）定义 Destination 为“**192.168.1.0**”, MASK 为“**255.255.255.0**” into Mask, Gateway 为“**192.168.2.1**”. 最后点击 **Update** 按钮。

8）模式选择“ **Host”**, 之后点击“ **Set & Close** ”按钮。

ITS2 (Gateway) 设置如下：

9）根据拓扑结构. 定义 Interface 1 的 IP 地址为“**192.168.1.2**”, 并且定义 Interface 2 的 IP 地址为“**192.168.2.1**”（见图 13.5）

10）模式选择“**Gateway”** 之后点击“ **Set & Close** ”按钮。现在，我们已经设置好了路由表，下面可以开始实验。

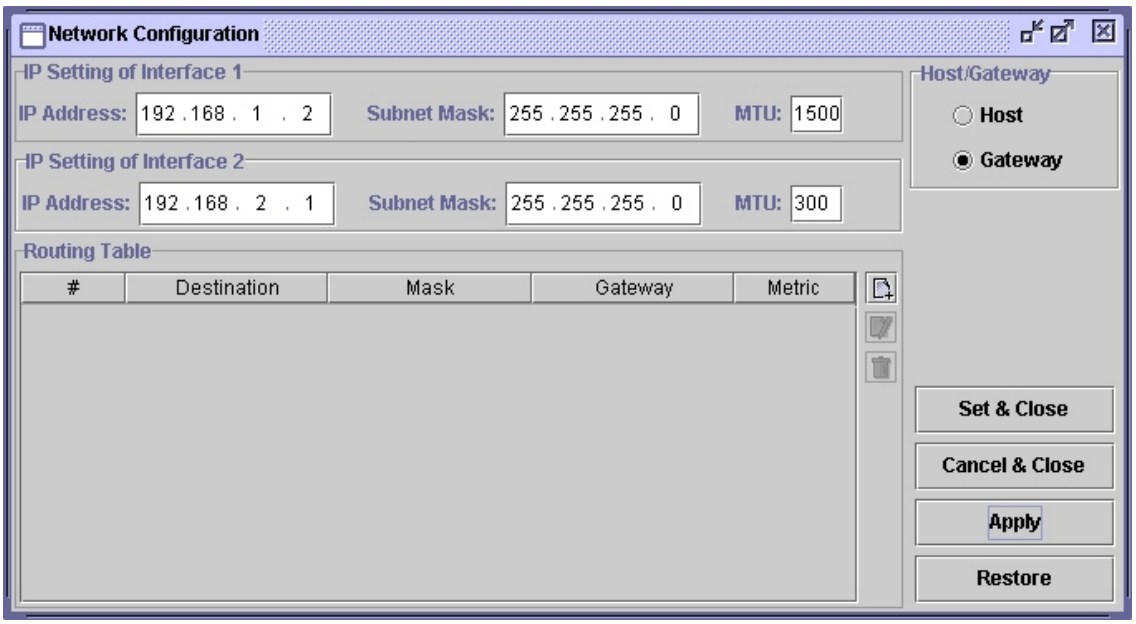


图 13.5

3、Slow-Start and Multiplicative Decrease 实验

ITS2 操作如下：

11）打开网络封包浏览器（Network Message Browser）界面，同时主意是否打开了监听状态。（**Listening On**）

12） 打开 MDDL 平台（MDDL Editor），点击 **Load** 按钮，调用 C: \XClient \Data \Mddl \Tutorial \Ex10 \PktLost4.mddl 程序，最后点击 **Upld** 按钮。（Pktlost4 程序的定义如下：每发送 5 个封包会自动丢弃第四个封包）。

ITS3 操作如下：

13）打开网络封包浏览器（Network Message Browser）界面，同时主意是否打开了监听状态。（**Listening On**）

14）打开 MDDL 平台（MDDL Editor）。

15）点击 **Load** 按钮，调用 C: \XClient \Data \Mddl \Tutorial \Ex13 \CongestionWindowReceiver.mddl，最后点击 **Upld** 按钮。

ITS1 操作如下：

16）打开网络封包浏览器（Network Message Browser）界面，同时主意是否打开了监听状态。（**Listening On**）

17）打开 MDDL 平台（MDDL Editor）。

18）点击 **Load** 按钮，调用 C: \XClient \Data \Mddl \Tutorial \Ex13 \CongestionWindowSender.mddl，最后点击 **Upld** 按钮。

19）打开 IP 封包的发送界面（IP Datagram Sender）。在 Protocol 部分定义“**7**”，输入 Destination IP 为“**192.168.2.2**”，数据段部分输入“**check**”。(见图 13.6)

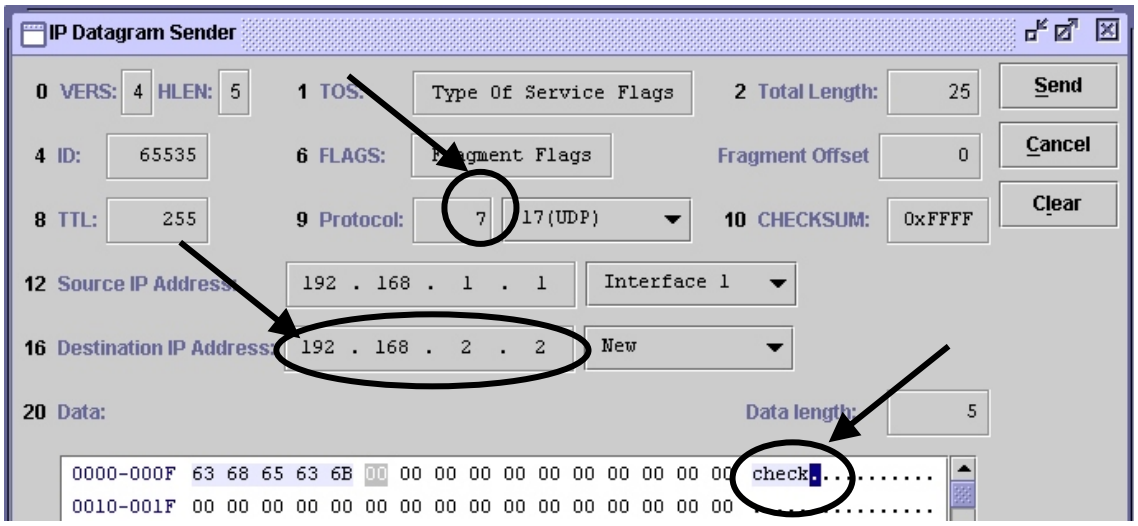


图 13.6

20）最后点击 **Send** 按钮。ITS1 将会发送一个 IP datagram 给 ITS3 ，然后会接收到 ITS3 回应的 ACK（见图 13.7）我们可以看见 congestion window size (CWND) 是‘001’. 如果数据传输正常，CWND 会指数递增到“004”，之后会线性的增长（每次增长 1），见图 13.8。

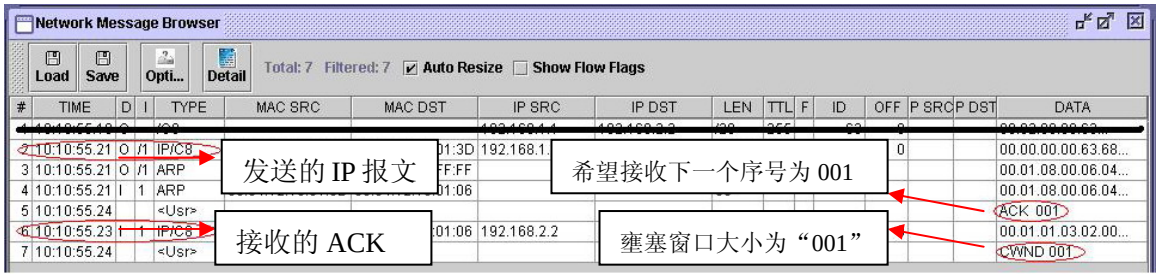


图 13.7

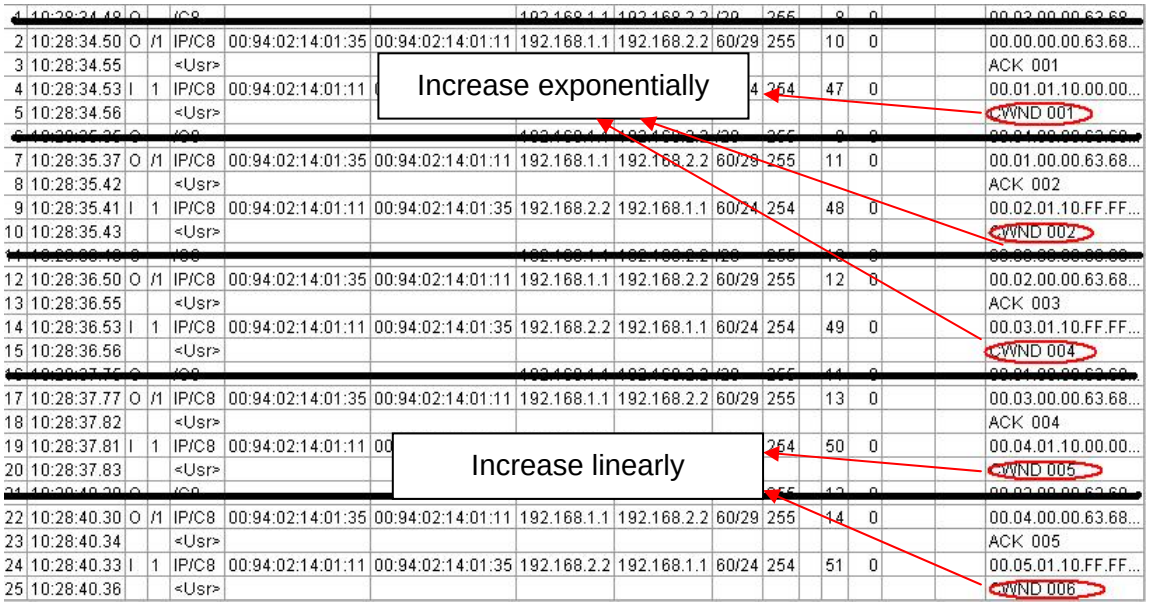
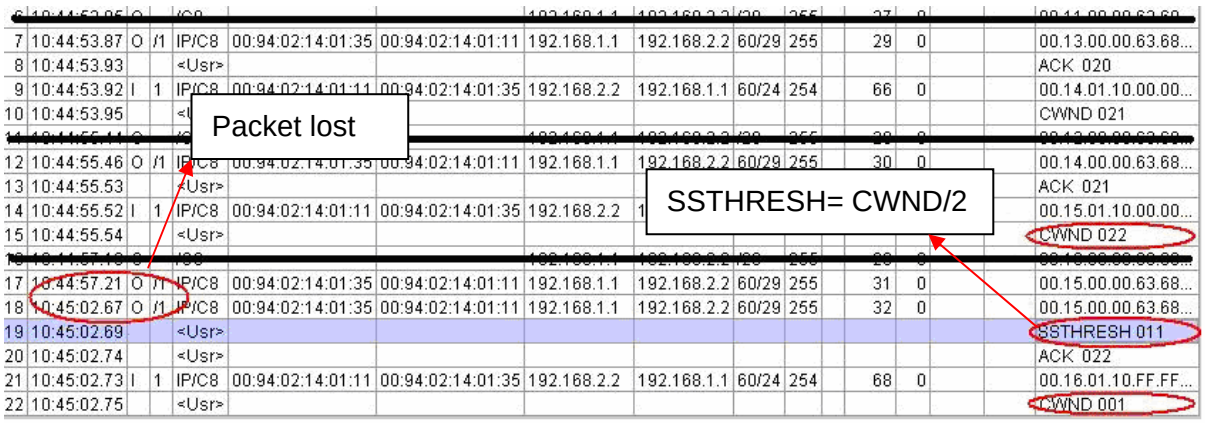


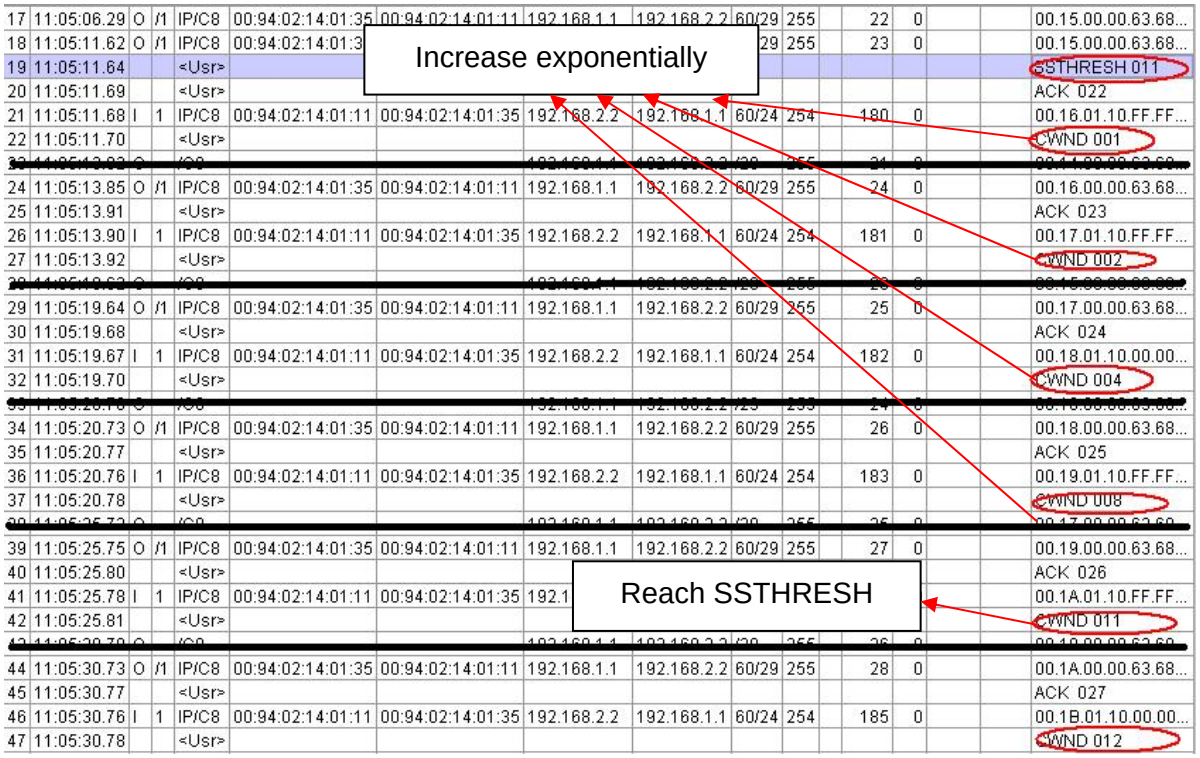
图 13.8

21）由于 ITS2 调用了一个丢包程序（PktLost4），即模拟产生数据传送的壅塞现象．壅塞窗口的大小将会变为 “001” （见图 13.9）。同时慢启动门限(SSTHRESH)为 CWND 的一半大小（即之后，如果网络恢复正常通讯，壅塞窗口的大小会以指数方式递增至 SSTHRESH 值，之后以线性方式递增（每次 CWND 值增加 1）。（见图 13.10）



7	10:44:53.87	O	/1	IP/C8	00:94:02:14:01:35	00:94:02:14:01:11	192.168.1.1	192.168.2.2	60/29	255	29	0	00.13.00.00.63.68...
8	10:44:53.93			<Usr>									ACK 020
9	10:44:53.92	I	1	IP/C8	00:94:02:14:01:11	00:94:02:14:01:35	192.168.2.2	192.168.1.1	60/24	254	66	0	00.14.01.10.00.00...
10	10:44:53.95			<Usr>									CWND 021
12	10:44:55.46	O	/1	IP/C8	00:94:02:14:01:35	00:94:02:14:01:11	192.168.1.1	192.168.2.2	60/29	255	30	0	00.14.00.00.63.68...
13	10:44:55.53			<Usr>									ACK 021
14	10:44:55.52	I	1	IP/C8	00:94:02:14:01:11	00:94:02:14:01:35	192.168.2.2	1					00.15.01.10.00.00...
15	10:44:55.54			<Usr>									CWND 022
17	10:44:57.21	O	/1	IP/C8	00:94:02:14:01:35	00:94:02:14:01:11	192.168.1.1	192.168.2.2	60/29	255	31	0	00.15.00.00.63.68...
18	10:45:02.67	O	/1	IP/C8	00:94:02:14:01:35	00:94:02:14:01:11	192.168.1.1	192.168.2.2	60/29	255	32	0	00.15.00.00.63.68...
19	10:45:02.69			<Usr>									SSTHRESH 011
20	10:45:02.74			<Usr>									ACK 022
21	10:45:02.73	I	1	IP/C8	00:94:02:14:01:11	00:94:02:14:01:35	192.168.2.2	192.168.1.1	60/24	254	68	0	00.16.01.10.FF.FF...
22	10:45:02.75			<Usr>									CWND 001

图 13.9



17	11:05:06.29	O	/1	IP/C8	00:94:02:14:01:35	00:94:02:14:01:11	192.168.1.1	192.168.2.2	60/29	255	22	0	00.15.00.00.63.68...
18	11:05:11.62	O	/1	IP/C8	00:94:02:14:01:3				29	255	23	0	00.15.00.00.63.68...
19	11:05:11.64			<Usr>									SSTHRESH 011
20	11:05:11.69			<Usr>									ACK 022
21	11:05:11.68	I	1	IP/C8	00:94:02:14:01:11	00:94:02:14:01:35	192.168.2.2	192.168.1.1	60/24	254	180	0	00.16.01.10.FF.FF...
22	11:05:11.70			<Usr>									CWND 001
24	11:05:13.85	O	/1	IP/C8	00:94:02:14:01:35	00:94:02:14:01:11	192.168.1.1	192.168.2.2	60/29	255	24	0	00.16.00.00.63.68...
25	11:05:13.91			<Usr>									ACK 023
26	11:05:13.90	I	1	IP/C8	00:94:02:14:01:11	00:94:02:14:01:35	192.168.2.2	192.168.1.1	60/24	254	181	0	00.17.01.10.FF.FF...
27	11:05:13.92			<Usr>									CWND 002
29	11:05:19.64	O	/1	IP/C8	00:94:02:14:01:35	00:94:02:14:01:11	192.168.1.1	192.168.2.2	60/29	255	25	0	00.17.00.00.63.68...
30	11:05:19.68			<Usr>									ACK 024
31	11:05:19.67	I	1	IP/C8	00:94:02:14:01:11	00:94:02:14:01:35	192.168.2.2	192.168.1.1	60/24	254	182	0	00.18.01.10.00.00...
32	11:05:19.70			<Usr>									CWND 004
34	11:05:20.73	O	/1	IP/C8	00:94:02:14:01:35	00:94:02:14:01:11	192.168.1.1	192.168.2.2	60/29	255	26	0	00.18.00.00.63.68...
35	11:05:20.77			<Usr>									ACK 025
36	11:05:20.76	I	1	IP/C8	00:94:02:14:01:11	00:94:02:14:01:35	192.168.2.2	192.168.1.1	60/24	254	183	0	00.19.01.10.FF.FF...
37	11:05:20.78			<Usr>									CWND 008
39	11:05:25.75	O	/1	IP/C8	00:94:02:14:01:35	00:94:02:14:01:11	192.168.1.1	192.168.2.2	60/29	255	27	0	00.19.00.00.63.68...
40	11:05:25.80			<Usr>									ACK 026
41	11:05:25.78	I	1	IP/C8	00:94:02:14:01:11	00:94:02:14:01:35	192.1						00.1A.01.10.FF.FF...
42	11:05:25.81			<Usr>									CWND 011
44	11:05:30.73	O	/1	IP/C8	00:94:02:14:01:35	00:94:02:14:01:11	192.168.1.1	192.168.2.2	60/29	255	28	0	00.1A.00.00.63.68...
45	11:05:30.77			<Usr>									ACK 027
46	11:05:30.76	I	1	IP/C8	00:94:02:14:01:11	00:94:02:14:01:35	192.168.2.2	192.168.1.1	60/24	254	185	0	00.1B.01.10.00.00...
47	11:05:30.78			<Usr>									CWND 012

图 13.10

四、实验讨论

1、连续发出20 个IP报文后制造1个报文丢失，完成后再多送10个可以顺利抵达目的端的IP 数据报，参考每一个IPdatagram 发出后CWND 值的大小，完成图13.11。

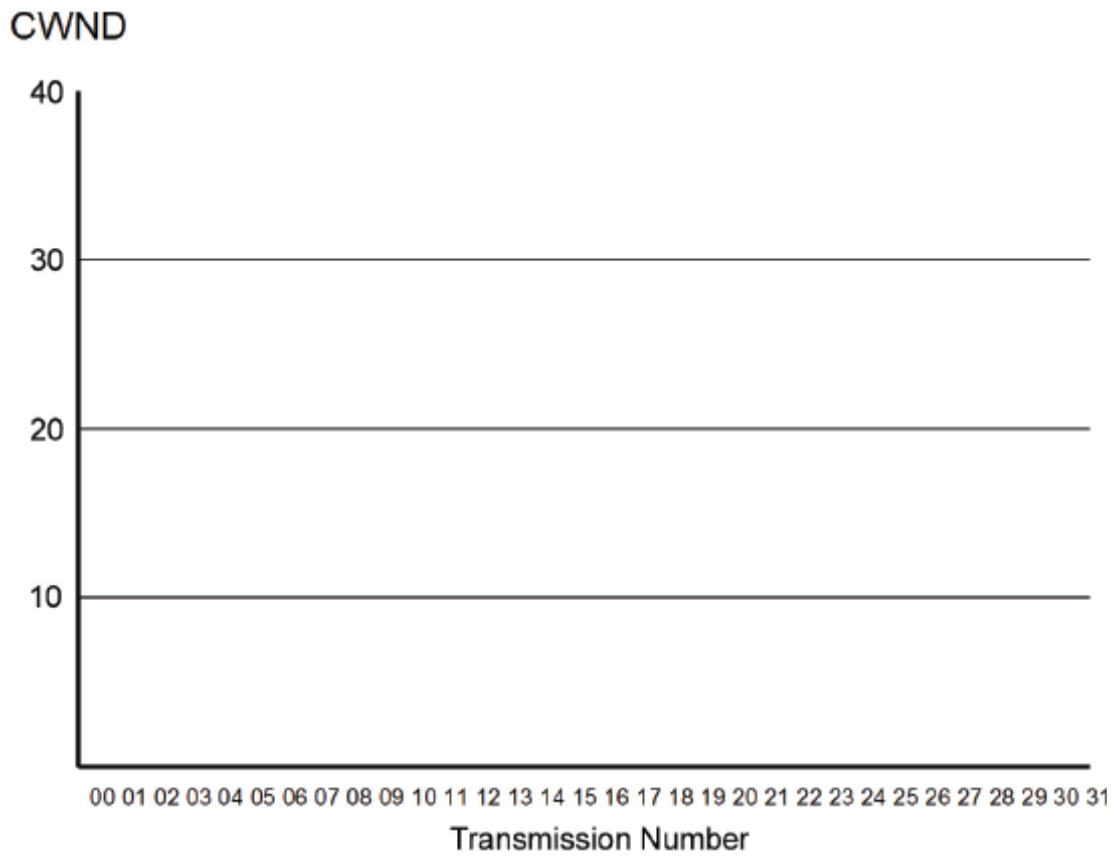


图 13.11

REACTOR PROGRAMS

1、CongestionWindowSender.mddl

```
VAR1.SND_UNA        = 0W;                  // SND_UNA        initialization.
VAR1.SND_NXT        = VAR1.SND_UNA;        // SND_NXT        initialization.
VAR1.SND_WND        = 16W;                 // SND_WND        initialization.
VAR1.SND_CWND       = 1W;                  // SND_CWND       initialization.
VAR1.SND_SSTHRESH   = 4W;                  // SND_SSTHRESH initialization.

VAR2[0, 3]      = {192, 168, 1, 1};     // SRC Address.
VAR2[4, 7]      = {192, 168, 2, 2};     // DST Address.

IP_OUT_HANDLER
{
    IF( S.IP_ADDRDST != VAR2[4, 7] || S.IP_PROT == CNST_IP_PROT_KDP )
        RETURN;

    DISCARD_MESSAGE;
```

```
    IF(VAR1.SND_NXT - (VAR1.SND_UNA + VAR1.SND_WND) < 32768W )
        RETURN;

    IF(VAR1.SND_NXT - (VAR1.SND_UNA + VAR1.SND_CWND) < 32768W )
        RETURN;

    ADD_TO_POOL 20 WITH_DATA
{
        T.[0]                       = 6                     ,
        T.[1]                       = 5                     ,
        T.[2,].KDP_ID               = VAR1.SND_NXT          ,
        T.[2,].KDP_ACK              = 0                     ,
        T.[2,].KDP_WINDOW_SIZE      = 0                     ,
        T.[2,].KDP_DATA             = S.IP_DATA
}

    SEND_OUT_IP WITH_DATA
    {
     T.IP_PROT                  = CNST_IP_PROT_KDP    ,
     T.IP_ADDRDST               = VAR2[4, 7]          ,
     T.IP_DATA.KDP_ID           = VAR1.SND_NXT        ,
        T.IP_DATA.KDP_ACK           = 0                     ,
        T.IP_DATA.KDP_WINDOW_SIZE = 0                       ,
        T.IP_DATA.KDP_DATA          = S.IP_DATA
    }

    VAR1.SND_NXT = VAR1.SND_NXT + 1W;
}

TIMER_WITH_PERIOD 1000
{
    FOR_EVERY_ELEMENT_IN_POOL 20
    {
```

```
PE[0] = PE[0] - 1;
    IF(PE[0] == 0)
    {
        PE[1] = PE[1] - 1;
        IF(PE[1] == 0)
        {
        GENERATE_USER_SYSMSG WITH_DATA
            {
                TARGET = "Communication Aborted!"
            }
            REMOVE_CURRENT_POOL_ELEMENT;
        }
        ELSE IF (PE[1] == 4)
        {
            PE[0] = 6;
            SEND_OUT_IP WITH_DATA
            {
                T.IP_PROT                   = CNST_IP_PROT_KDP      ,
                T.IP_ADDRDST                  = VAR2[4, 7]              ,
                T.IP_DATA                   = PE.[2,]
            }
            VAR1.SND_SSTHRESH =    VAR1.SND_CWND/2 ;
             VAR1.SND_CWND = 1W;

             GENERATE_USER_MSG WITH_DATA
            {
                T.[9] = ((VAR1.SND_SSTHRESH)/100)+0X30,
                T.[10] = (((VAR1.SND_SSTHRESH)%100)/10)+0X30,
                T.[11] = ((VAR1.SND_SSTHRESH)%10)+0X30,
                TARGET = "SSTHRESH "
            }
        }
       ELSE
        {
```

```
                    PE[0] = 6;
                    SEND_OUT_IP WITH_DATA
                    {
                        T.IP_PROT                   = CNST_IP_PROT_KDP      ,
                        T.IP_ADDRDST                  = VAR2[4, 7]              ,
                        T.IP_DATA                   = PE.[2,]
                    }
                }
            }
        }
}

IP_IN_HANDLER
{
 IF(S.IP_ADDRSRC != VAR2[4, 7] || S.IP_PROT != CNST_IP_PROT_KDP ||
     S.IP_DATA.KDP_ACK !=1)
      RETURN;


   GENERATE_USER_MSG WITH_DATA
    {
        T.[5] = ((S.IP_DATA.KDP_ID)/100)+0X30,
        T.[6] = (((S.IP_DATA.KDP_ID)%100)/10)+0X30,
        T.[7] = ((S.IP_DATA.KDP_ID)%10)+0X30,
        TARGET = "ACK    "
    }

 IF(S.IP_DATA.KDP_ID - VAR1.SND_UNA >=    32768W)
      RETURN;

 IF(VAR1.SND_NXT - S.IP_DATA.KDP_ID >= 32768W)
      RETURN;

 DISCARD_MESSAGE;
```

```
FOR_EVERY_ELEMENT_IN_POOL 20
    {
        IF(PE[2,].IP_DATA.KDP_ID - S.IP_DATA.KDP_ID >= 32768W)
            REMOVE_CURRENT_POOL_ELEMENT;
}

VAR1.SND_UNA = S.IP_DATA.KDP_ID;
VAR1.SND_WND = S.IP_DATA.KDP_WINDOW_SIZE;

     GENERATE_USER_MSG WITH_DATA
    {
        T.[5] = ((VAR1.SND_CWND)/100)+0X30,
        T.[6] = (((VAR1.SND_CWND)%100)/10)+0X30,
        T.[7] = ((VAR1.SND_CWND)%10)+0X30,
        TARGET = "CWND "
    }

IF(VAR1.SND_CWND - VAR1.SND_SSTHRESH < 32768W)
      VAR1.SND_CWND = VAR1.SND_CWND + 1W;

                                IF(VAR1.SND_CWND - VAR1.SND_SSTHRESH >= 32768W)
{
                                    IF(VAR1.SND_SSTHRESH - (VAR1.SND_CWND*2) < 32768W)
                                      VAR1.SND_CWND=VAR1.SND_CWND+VAR1.SND_CWND;
                                    ELSE
                                      VAR1.SND_CWND =VAR1.SND_SSTHRESH ;
                                }
}
```

2、CongestionWindowReceiver.mddl

```
VAR1.RCV_NXT      = 0W;                    // RCV_NXT initialization.
VAR1.RCV_WND      = 16W;                     // RCV_WND initialization.

VAR2[0, 3]        = {192, 168, 2, 2};      // SRC Address.
```

```
VAR2[4, 7]          = {192, 168, 1, 1};         // DST Address.

VAR3[4, 5] = 0W;                                  // Some pointer.

IP_IN_HANDLER
{
    IF( S.IP_ADDRSRC != VAR2[4, 7] || S.IP_PROT != CNST_IP_PROT_KDP ||
        S.IP_DATA.KDP_ACK != 0W )
         RETURN;

    DISCARD_MESSAGE;

    IF(S.IP_DATA.KDP_ID - VAR1.RCV_NXT >= 32768W)
         RETURN;

    IF(S.IP_DATA.KDP_ID - (VAR1.RCV_NXT + VAR1.RCV_WND) < 32768W)
         RETURN;

    LOOK_FOR_ONE_ELEMENT_IN_POOL 21 WITH_CONDITION (PE.IP_DATA.KDP_ID ==
   S.IP_DATA.KDP_ID)
         RETURN;

    VAR1.RCV_WND = VAR1.RCV_WND - 1W;

     GENERATE_USER_MSG WITH_DATA
    {
         TARGET = VAR1.RCV_WND
    }

    ADD_TO_POOL 21 WITH_CONDITION (S.IP_DATA.KDP_ID - PE.IP_DATA.KDP_ID < 32768W)
    WITH_DATA
 {
         T = S
 }
```

```
    FOR(VAR3[4, 5] = VAR1.RCV_NXT;;VAR3[4, 5] = VAR3[4, 5] + 1W)
    {

        LOOK_FOR_ONE_ELEMENT_IN_POOL 21 WITH_CONDITION (PE.IP_DATA.KDP_ID ==
        VAR3[4, 5])
        {
            VAR1.RCV_WND = VAR1.RCV_WND + 1W;
            CONTINUE;
        }
        ELSE
            BREAK;
    }

    VAR1.RCV_NXT = VAR3[4, 5];

    FOR_EVERY_ELEMENT_IN_POOL 21 WITH_CONDITION(PE.IP_DATA.KDP_ID –
    VAR1.RCV_NXT >= 32768W)
        REMOVE_CURRENT_POOL_ELEMENT;

 SEND_OUT_IP WITH_DATA
 {
        T.IP_PROT                       = CNST_IP_PROT_KDP,
        T.IP_ADDRDST                    = VAR2[4, 7]        ,
        T.IP_DATA.KDP_ID                = VAR1.RCV_NXT      ,
        T.IP_DATA.KDP_ACK               = 1                 ,
        T.IP_DATA.KDP_WINDOW_SIZE       = VAR1.RCV_WND
    }
}
```