Exp 11: Error Control 错误控制

目的：使学生了解错误控制(error control)机制如何解决数据包丢失 及数据包延迟等问题。

摘要：本实验主要介绍错误控制机制如何让TCP成为一个可靠的通讯协议，实验中并借着MDDL程序语言，让学生进一步了解其算法及工作原理。

时间：6 hrs。

一、网络拓扑

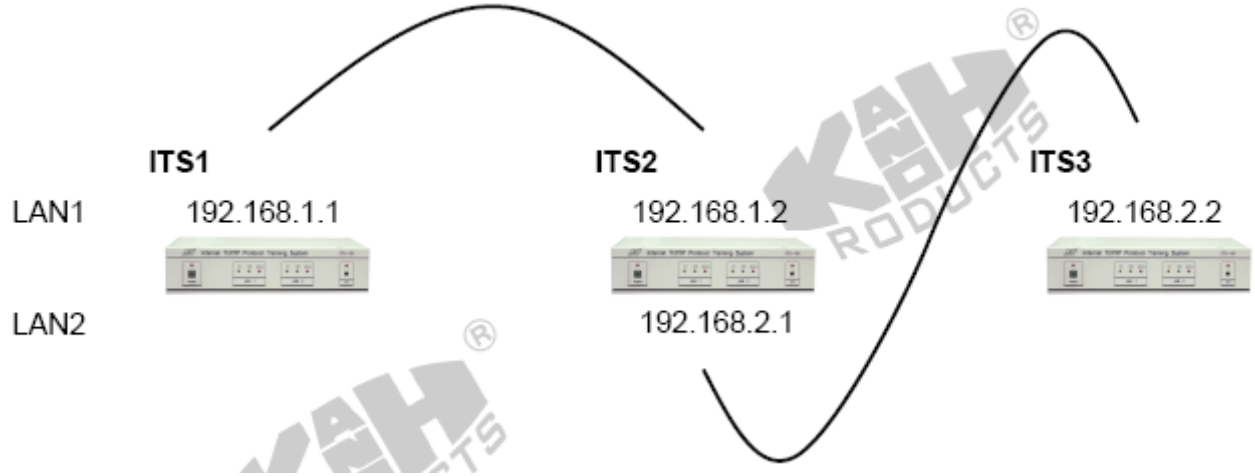

图11.1

二、技术背景

错误控制(error control)机制是用来发现或者更进一步修正网络间的错误封包，在数据链路(data link)层中最常见的错误控制方法，即是自动重发请求(ARQ，Automatic Repeated Request)，即当接收端发现接获一错误报文时，会持续要求发送端重送此报文直到这个错误被更正或此错误报文相关的处理程序已逾时。

ARQ 总共可以分为以下几种不同的型式：

1、Idle RQ

1）Implicit retransmit request

2）Explicit retransmit request

2、Continuous RQ

1）Selective repeat

2）Go-back-N

本实验中我们将只列举Idle RQ 中的隐式重发请求(Implicit retransmit request)及Continuous RQ 中的选择性重发(selective repeat)作说明：

1、Idle RQ - Implicit retransmit request

1）发送端传送一个I-frame(information bearing frame)报文给接收端。

2）发送端等待接收端的确认回应报文(ACK)。

3）当接获此ACK回应报文后，发送端再传送另一新的I-frame给接收端。

4）所有的运作是处在半双工的联机模式下，此外I-frames及ACKs在传送过程中都有可能遗失或重复。

2、 Continuous RQ - Selective repeat

1）发送端不等待接收端的确认信息(ACK)而持续的发送I-frame 给接收端。

2）接收端接获封包时传送ACK 确认信息给发送端。

3）发送端会建立起一个重送表(retransmission list)。

4）接收端会建立起一个接收表(receive list)。

5）因为收发双方都有列表，所以传输中，有发生状况的报文(序号X)会在下个报文(序号X+1)抵达时被查觉。

6）报文(序号Y)的确认响应报文(ACK)将会一同响应retransmission list中序号Y之前的所有报文。

为了模拟IP 传输过程中的ARQ机制，我们在IP封包内自行定义了一种通讯协议叫KDP，其格式如下：

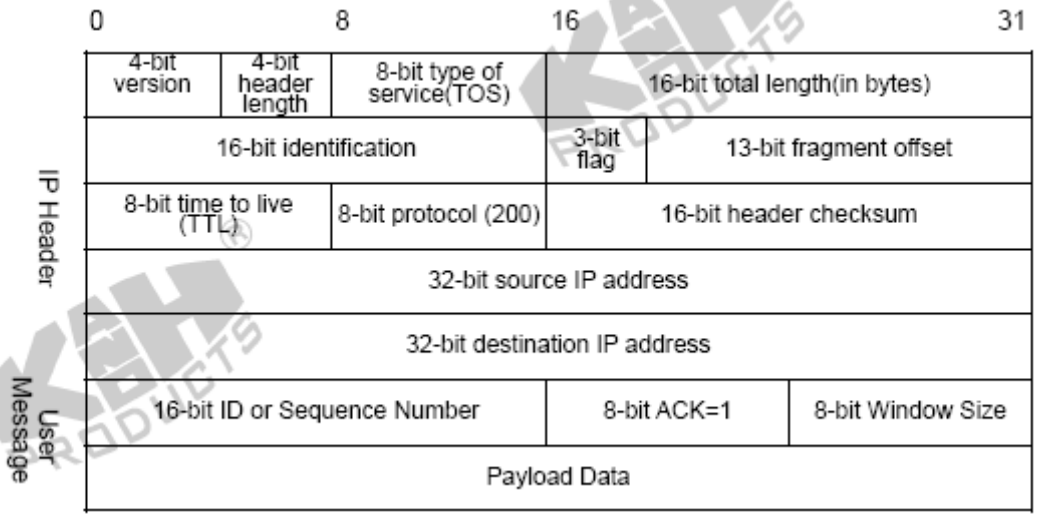

表11.1

8-bit protocol：通讯协议码。我们将此自订的通讯协议KDP 在IP 标头的通讯协议码设为200，用以区别其它协议的报文。

16-bit ID or Sequence Number：认证码。当设为确认ID 时表示此数据包为Idle RQ 类型，设为确认Sequence Number 序号时表示此报文为Continuous RQ 类型。

8-bit ACK：回应确认。设为1时表示其报文为响应确认报文。

8-bit Window Size：未使用。

Payload Data：KDP 标头后的数据域位，用以承载使用者数据。

此外实验中还会将Idle RQ 再细分，探讨隐式重发请求中两种不同的型态：

1、Idle RQ without Packet Identification：见图11.2，这是一种很简单的间歇性重复发送要求的错误控制方式，此型态的报文不管是发送端发出的I-frame 或是接受端发出的ACK 都不会包含任何关于报文ID 的信息。

2、Idle RQ：见图11.3，此图为一个标准的隐式重发请求错误控制机制，报文内包含ID 号码。

Idle RQ without Packet Identification

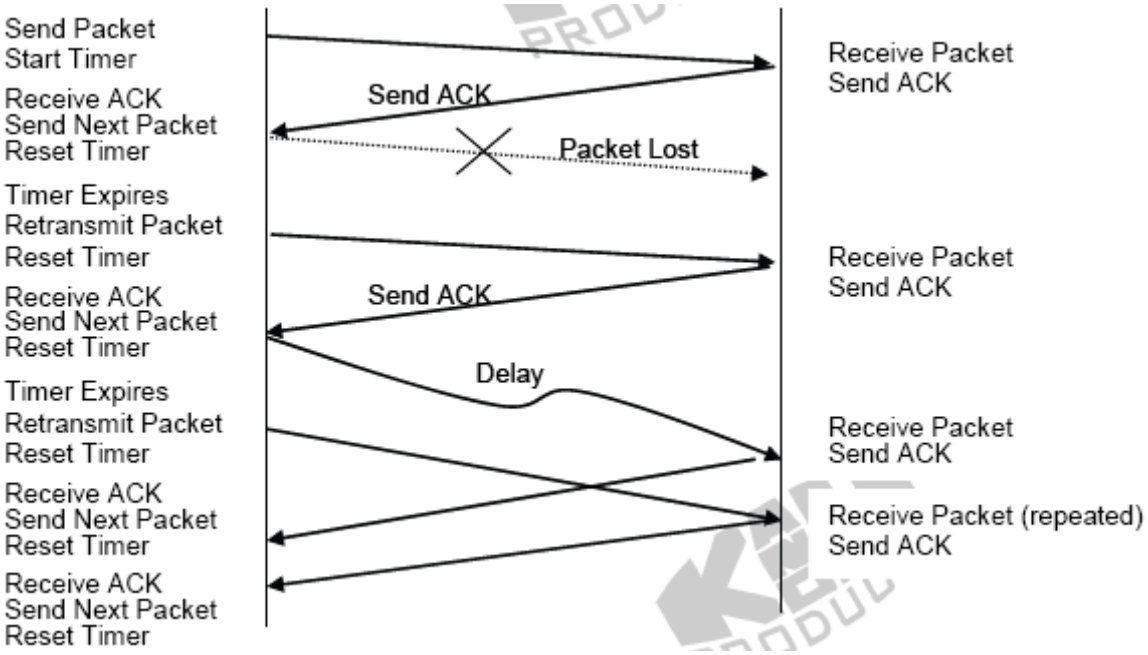

图11.2

Idle RQ

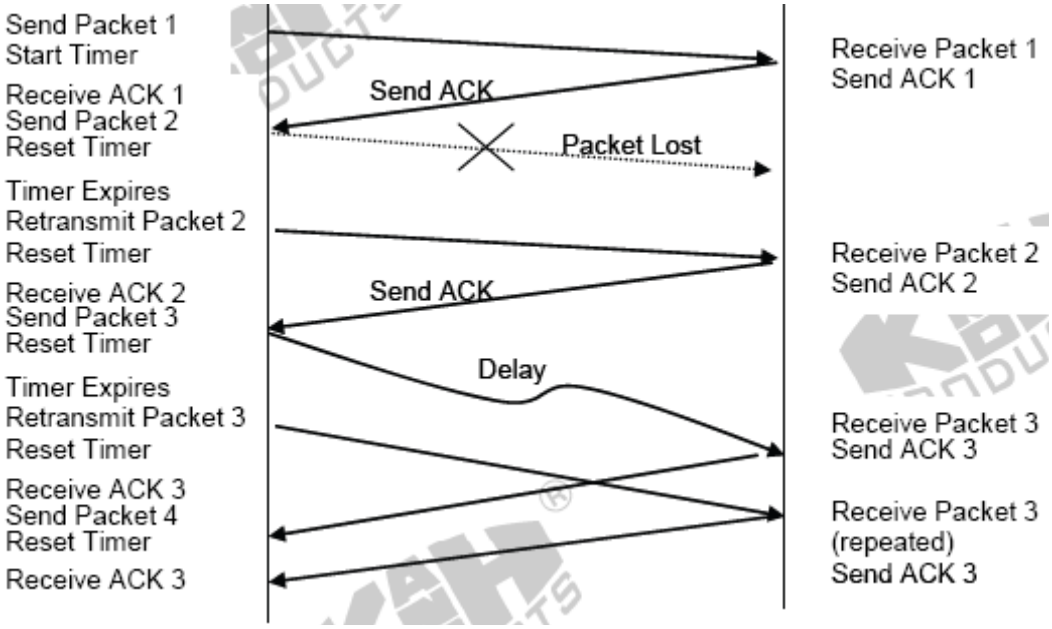

图11.3

Continuous RQ

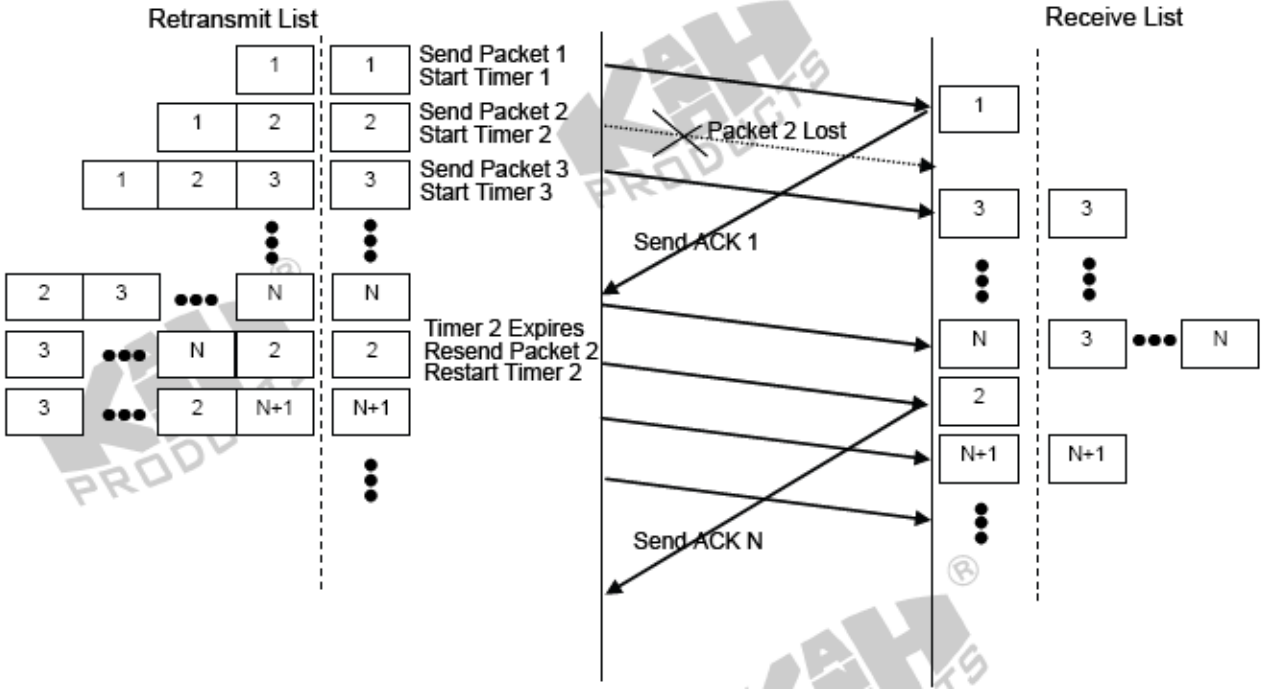

图11.4

三、实验步骤

1、网络拓扑

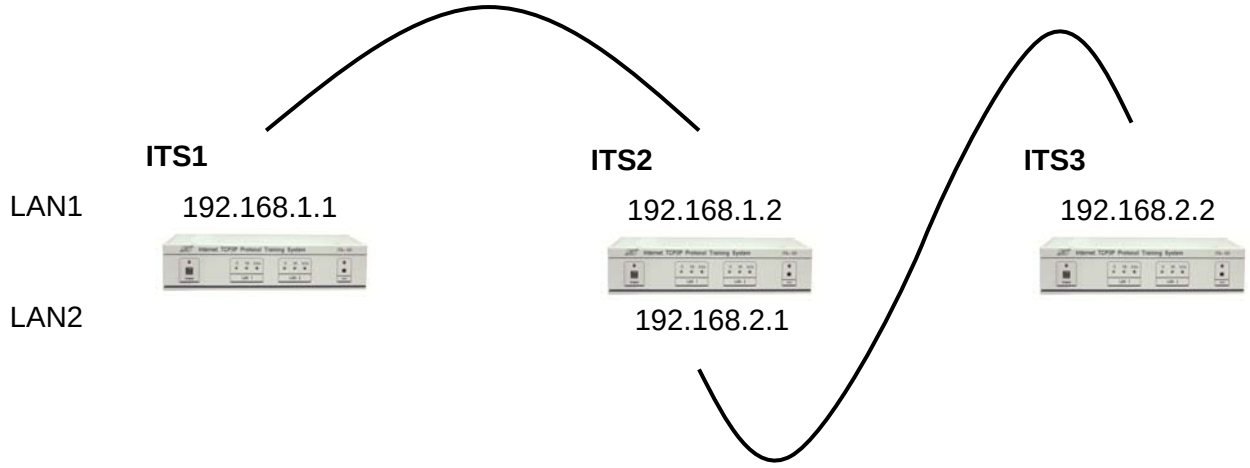

图11.5

2、设置 Host 和 Gateway

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

2）从 Tool 菜单打开网络设置对话框（**Network Configuration**）界面。

ITS1 (Host) 设置如下：

3）根据拓扑结构定义 Interface1 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 11.6) 。

4)定义 Destination 为“**192.168.2.0**”, MASK 为“**255.255.255.0**”, Gatewa 为“**192.168.1.2**” Gateway (见图 11.7), 最后点击 **Update** 按钮。

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

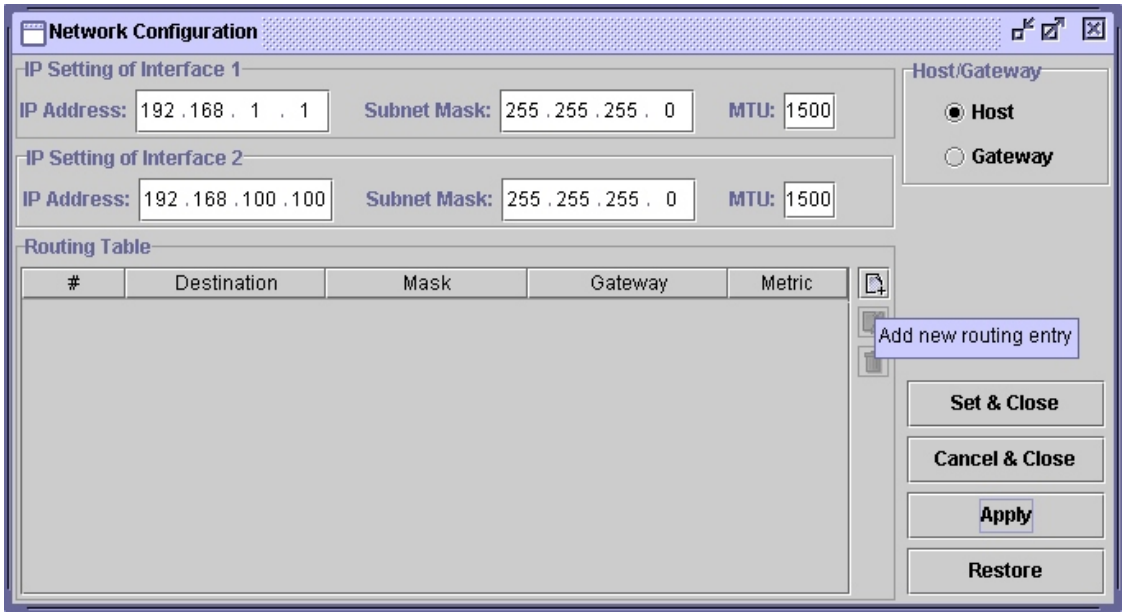

图 11.6

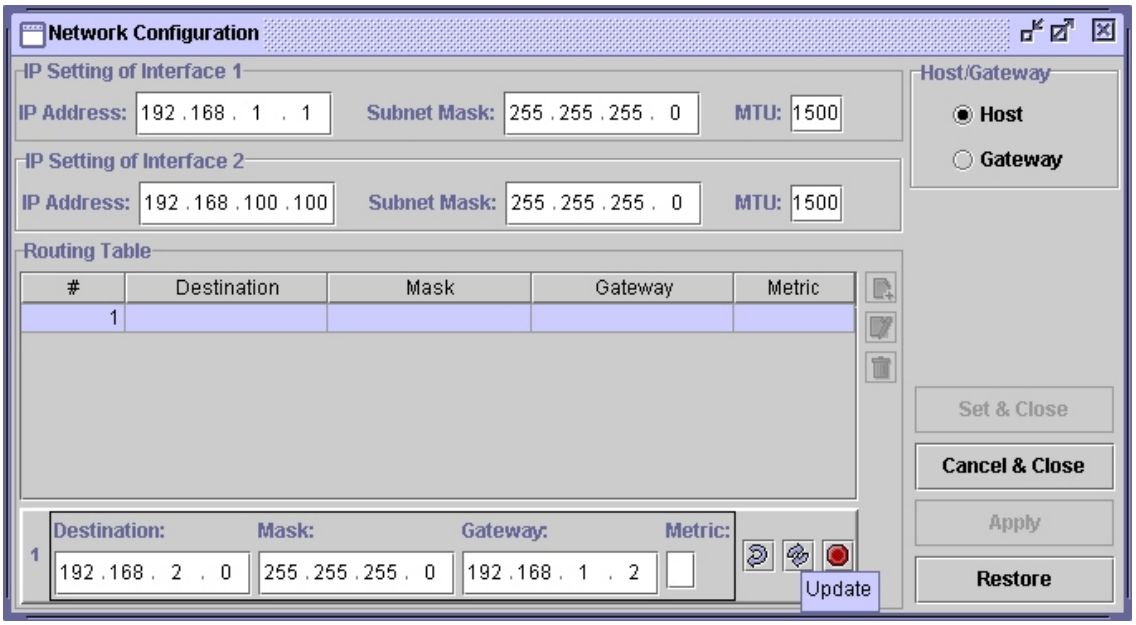

图 11.7

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 11.8.)

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。

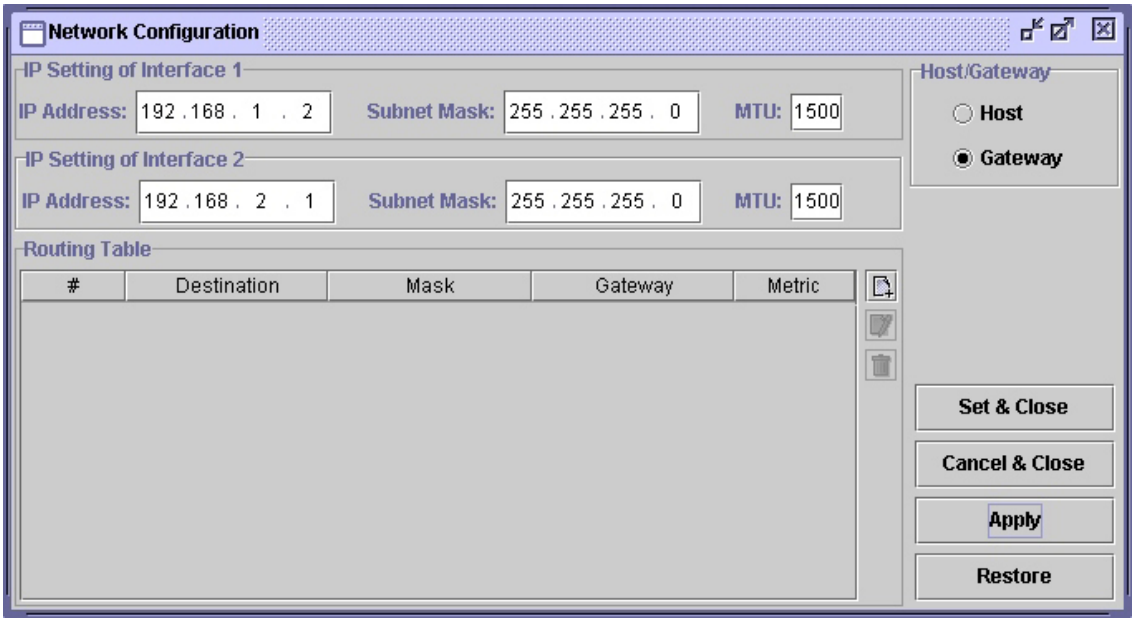

图 11.8

4、Retransmission and RTT (Round Trip Time)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）。

13）点击 **Load** 按钮，调用 C: \XClient \Data \Mddl \Tutorial \Ex11 \PktLostDelay-11-1.mddl, 最后点击 **Upld** 按钮。PktLostDelay-11-1 程序定义如下：当由 ITS1 发送的封包经过 ITS2（router）时，第一个封包会被正常转发；第二个封包会被延迟 4 秒后转发；第三个封包会被延迟 7 秒后转发；第四个封包会被正常转发；第五个封包会被丢弃。以此循环执行。

ITS3 操作如下：

14）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**）

15）打开 MDDL 平台（MDDL Editor）。

16）点击 **Load** 按钮，调用 C: \XClient \Data \Mddl \Tutorial \Ex11 \NoIDIdleRQReceiver.mddl, 最后点击 **Upld** 按钮。

ITS1 操作如下：

17）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**）

18）打开 MDDL 平台（MDDL Editor）。

19）点击 **Load** 按钮，调用 C: \XClient \Data \Mddl \Tutorial \Ex11 \NoIDIdleRQSender.mddl, 最后点击 **Upld** 按钮。

20）从 Send 菜单中，打开“ **Send IP Packet** ”对话框界面。

21）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 11.9。

22）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。尝试发送更多的 IP 报文给 ITS3，我们降会发现封包丢失和封包延迟的区别。

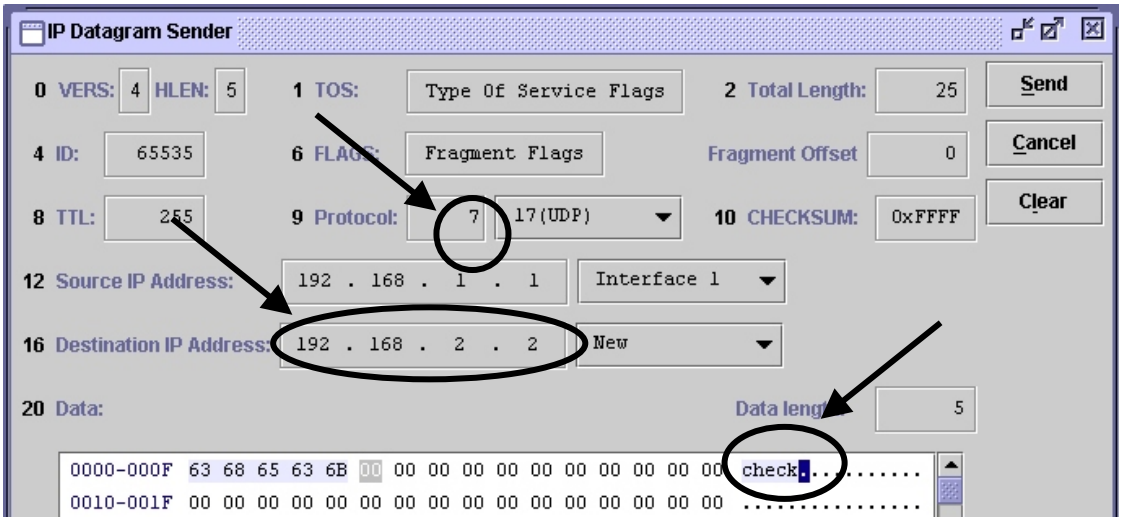

图 11.9

图 11.10 通过观察 ITS1 的网络封包浏览器，我们可以观察到封包的延迟和丢包 。

2	14:00:13.67	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		8	0	00.00.00.00.63.68...
3	14:00:13.70	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		11	0	00.00.01.00.00.00...
4	14:00:13.71			<Usr>										ACKED
6	14:00:17.64	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		9	0	00.00.00.00.63.68...
7	14:00:18.54			<Usr>										05
8	14:00:19.54			<Usr>										04
9	14:00:20.54			<Usr>										03
10	14:00:21.54			<Usr>										02
11	14:00:21.58	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		12	0	00.00.01.00.00.00...
12	14:00:21.59			<Usr>										ACKED
14	14:00:29.29	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		10	0	00.00.00.00.63.68...
15	14:00:29.55			<Usr>										05
16	14:00:30.55			<Usr>										04
17	14:00:31.55			<Usr>										03
18	14:00:32.55			<Usr>										02
19	14:00:33.55			<Usr>										01
20	14:00:34.55			<Usr>										00
21	14:00:34.56			<Usr>										03
22	14:00:34.57			<UsrSys>										Retransmission!
23	14:00:34.58	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		11	0	00.00.00.00.63.68...
24	14:00:34.61	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		13	0	00.00.01.00.00.00...
25	14:00:34.62			<Usr>										ACKED
26	14:00:36.31	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		14	0	00.00.01.00.00.00...
27	14:00:36.32			<Usr>										ACKED
29	14:00:39.57	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		12	0	00.00.00.00.63.68...
30	14:00:39.59			<Usr>										05
31	14:00:40.59			<Usr>										04
32	14:00:41.59			<Usr>										03
33	14:00:42.59			<Usr>										02
34	14:00:43.59			<Usr>										01
35	14:00:44.59			<Usr>										00
36	14:00:44.60			<Usr>										03
37	14:00:44.61			<UsrSys>										Retransmission!
38	14:00:44.62	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		13	0	00.00.00.00.63.68...
39	14:00:44.65	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		15	0	00.00.01.00.00.00...
40	14:00:44.66			<Usr>										ACKED

图 11.10

5、Error Control 实验

A. Idle RQ without Packet Identification 实验

ITS2 操作如下：

23）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**）

24）打开 MDDL 平台（MDDL Editor）。

25）点击 **Load** 按钮，调用 C: \XClient \Data \Mddl \Tutorial \Ex10 \PktLost4.mddl，最后点击 **Upld** 按钮。

其中 PktLost4 的程序定义如下：对于 router，没转发五个封包，将会丢弃第四个封包。

ITS3 操作如下：

26）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**）

27）打开 MDDL 平台（MDDL Editor）。

28）点击 **Load** 按钮，调用 C: \XClient \Data \Mddl \Tutorial \Ex11 \NoIDIdleRQReceiver.mddl, 最后点击 **Upld** 按钮。

ITS1 操作如下：

29）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**）

30）打开 MDDL 平台（MDDL Editor）。

31）点击 **Load** 按钮，调用 C: \XClient \Data \Mddl \Tutorial \Ex11 \NoIDIdleRQSender.mddl, 最后点击 **Upld** 按钮。

32）根据前面的实验操作，通过 IP 封包的发送界面，发送一个 IP 报文给 ITS3。之后，会得到由 ITS3 反馈的 ACK。观察 ITS1 的网络封包浏览器见图 11.11

2	14:27:59.38	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	14	0			00.00.00.00.63.68...
3	14:27:59.42	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	20	0			00.00.01.00.00.00...
4	14:27:59.43			<Usr>											ACKED
6	14:28:02.22	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	15	0			00.00.00.00.63.68...
7	14:28:02.26			<Usr>											ACKED
8	14:28:02.26	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	21	0			00.00.01.00.00.00...
10	14:28:13.22	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	16	0			00.00.00.00.63.68...
11	14:28:13.26			<Usr>											ACKED
12	14:28:13.26	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	22	0			00.00.01.00.00.00...
14	14:28:16.04	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	17	0			00.00.00.00.63.68...
15	14:28:16.08			<Usr>											05
16	14:28:17.08			<Usr>											04
17	14:28:18.08			<Usr>											03
18	14:28:19.08			<Usr>											02
19	14:28:20.08			<Usr>											01
20	14:28:21.08			<Usr>											00
21	14:28:21.09			<Usr>											03
22	14:28:21.10			<UsrSys>											Retransmission!
23	14:28:21.11	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	18	0			00.00.00.00.63.68...
24	14:28:21.14	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	23	0			00.00.01.00.00.00...
25	14:28:21.15			<Usr>											ACKED
27	14:28:23.18	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	19	0			00.00.00.00.63.68...
28	14:28:23.21	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	24	0			00.00.01.00.00.00...
29	14:28:23.22			<Usr>											ACKED

图 11.11

B. Idle RQ 实验

ITS2 操作如下：

33）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**）

34）打开 MDDL 平台（MDDL Editor）。

35）点击 **Load** 按钮，调用 C: \XClient \Data \Mddl \Tutorial \Ex10 \PktLost4.mddl, 最后点击 **Upld** 按钮。

ITS3 操作如下：

36）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**.）

37）打开 MDDL 平台（MDDL Editor）。

38）点击 **Load** 按钮，调用 C: \XClient \Data \Mddl \Tutorial \Ex11 \SIDIdleRQReceiver.mddl, 最后点击 **Upld** 按钮。

ITS1 操作如下：

39）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**）

40）打开 MDDL 平台（MDDL Editor）。

41）点击 **Load** 按钮，调用 C: \XClient \Data \Mddl \Tutorial \Ex11 \SIDIdleRQSender.mddl, 最后点击 **Upld** 按钮。

42）根据前面的实验操作，通过 IP 封包的发送界面，发送一个 IP 报文给 ITS3。当封包发生丢失现象，由 ITS1 立即再发一个新的 IP 报文。我们可以看见新发的 IP 报文将会保存在发送端的 buffer 内，一直到 Retransmission 结束为止，再传送出去。(见图 11.12)

2	14:37:18.04	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		00.00.00.00.63.68...
3	14:37:18.04	O	/1	ARP	00:94:02:14:01:35	FF:FF:FF:FF:FF:FF			60						00.01.08.00.06.04...
4	14:37:18.04	I	1	ARP	00:94:02:14:01:11	00:94:02:14:01:35			60						00.01.08.00.06.04...
5	14:37:18.05			<Usr>											UNA 00
6	14:37:18.08	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		28	0		00.01.01.00.00.00...
7	14:37:18.09			<Usr>											ACK 01
8	14:37:20.43	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		00.01.00.00.63.68...
9	14:37:20.41	O		/94			1.53.192.168	1.1.0.0	/2048	0		1,540	1		00.00.00.00.C0.A8...
10	14:37:20.44			<Usr>											UNA 01
11	14:37:20.47	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		29	0		00.02.01.00.00.00...
12	14:37:20.48			<Usr>											ACK 02
13	14:37:21.90	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		00.02.00.00.63.68...
15	14:37:21.91			<Usr>											UNA 02
16	14:37:21.94	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		30	0		00.03.01.00.00.00...
17	14:37:21.95			<Usr>											ACK 03
18	14:37:22.99	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		00.03.00.00.63.68...
20	14:37:23.00			<Usr>											UNA 03
21	14:37:23.87			<Usr>											05
22	14:37:24.87			<Usr>											04
23	14:37:25.87			<Usr>											03
24	14:37:25.88	O		/C8			192.168.1.1	192.168.2.2	/29	255		27	0		00.03.00.00.63.68...
25	14:37:25.89			<Usr>											UNA 03
26	14:37:26.88			<Usr>											02
27	14:37:27.88			<Usr>											01
28	14:37:28.88			<Usr>											00
29	14:37:28.89			<Usr>											03
30	14:37:28.90			<UsrSys>											Retransmission!
31	14:37:28.91	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		00.03.00.00.63.68...
32	14:37:28.95			<Usr>											ACK 04
33	14:37:28.94	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		31	0		00.04.01.00.00.00...
34	14:37:28.98	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		00.04.00.00.63.68...
35	14:37:29.01	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		32	0		00.05.01.00.FF.FF...
36	14:37:29.02			<Usr>											ACK 05

图 11.12

C. Continuous RQ 实验

ITS2 操作如下：

43）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**）

44）打开 MDDL 平台（MDDL Editor）。

45）点击 **Load** 按钮，调用 C: \XClient \Data \Mddl \Tutorial \Ex10 \PktLost4.mddl，最后点击 **Upld** 按钮。

ITS3 操作如下：

46）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**）

47）打开 MDDL 平台（MDDL Editor）。

48）点击 **Load** 按钮，调用 C: \XClient \Data \Mddl \Tutorial \Ex11 \SIDCRQReceiver.mddl，最后点击 **Upld** 按钮。

ITS1 操作如下：

49）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**）

50）打开 MDDL 平台（MDDL Editor）。

51）点击 **Load** 按钮，调用 C: \XClient \Data \Mddl \Tutorial \Ex11 \SIDCRQSender.mddl，最后点击 **Upld** 按钮。

52）根据前面的实验操作，通过 IP 封包的发送界面，发送一个 IP 报文给 ITS3。当封包发送丢失现象时，立即由 ITS1 再次发送一个新的 IP 报文给 ITS3，我们可以看见新发的 IP 报文将不会等待上个 IP 封包的 Retransmission 结束，就直接发给 ITS3 了。(见图 11.13)

2	15:02:55.05	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		37	0		00.00.00.00.63.68...
3	15:02:55.05			<Usr>											UNA 00
4	15:02:55.10			<Usr>											ACK 01
5	15:02:55.09	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		37	0		00.01.01.00.FF.FF...
6	15:02:55.11			<Usr>											NXT 01
7	15:02:56.72	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		38	0		00.01.00.00.63.68...
9	15:02:56.73			<Usr>											UNA 01
10	15:02:56.77			<Usr>											ACK 02
11	15:02:56.76	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		38	0		00.02.01.00.00.00...
12	15:02:56.78			<Usr>											NXT 02
13	15:02:58.46	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		39	0		00.02.00.00.63.68...
15	15:02:58.47			<Usr>											UNA 02
16	15:02:58.51			<Usr>											ACK 03
17	15:02:58.50	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		39	0		00.03.01.00.00.00...
18	15:02:58.52			<Usr>											NXT 03
19	15:03:00.05	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		40	0		00.03.00.00.63.68...
21	15:03:00.05			<Usr>											UNA 03
22	15:03:02.35	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		41	0		00.04.00.00.63.68...
24	15:03:02.36			<Usr>											UNA 03
25	15:03:05.46	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		42	0		00.03.00.00.63.68...
26	15:03:05.52			<Usr>											ACK 05
27	15:03:05.51	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		40	0		00.05.01.00.00.00...
28	15:03:05.53			<Usr>											NXT 05

图 11.13

四、实验讨论

1、往返时间(RTT, Round Trip Time)是指一个封包从发送端出发，到某个网络上的节点或目的端，并收到响应所需要的平均时间。我们通常将重送时间(retransmission time)设为两倍的RTT。如果我们将重送时间拉长或变短，对网路传输会有什么样的影响？试着改写发送端的程序并讨论其影响。

2、比较一下，Continuous RQ是否比IdleRQ更有效率？在Idle RQ的方式中，最多可以

扣留几个IPdatagram？试着在packet lost 的情况发生时，多送几个IP datagram 看看。

REACTOR PROGRAMS

1、PktLostDelay-11-1.mddl

```
VAR1[0] =0 ;
IP_ROUTING_HANDLER
{
  IF(ISMYIPADDR(S.IP_ADDRDST))
        RETURN;
  IF(S.IP_ADDRDST.[0,2]!= MYIPADDR(2).[0,2]) //compare with netmask 255.255.255.0
    RETURN;

  VAR1[0] = VAR1[0]+1 ;
  GENERATE_USER_MSG WITH_DATA
  {
    TARGET = VAR1[0]
  }
      IF(VAR1[0] == 1)
      {
      IF(S.IP_TTL>1)
       {
            SEND_OUT_IP WITH_DATA
            {
                 T                        = S                          ,
                 T.IP_TTL                 = S.IP_TTL - 1               ,
                 T.IP_HEADERCHKSUM = {0, 0}                            ,
                 T.IP_HEADERCHKSUM = CHECKSUM(T.IP_HEADER)
            }
      }
```

```
 }
 ELSE IF(VAR1[0] == 2)
{
    ADD_TO_POOL 22 WITH_LIFETIME 20000 WITH_DATA
    {
        T.[0]      = 40                    ,
        T.[6, ]    = SOURCE
    }

 }
 ELSE IF(VAR1[0] == 3)
 {
    ADD_TO_POOL 22 WITH_LIFETIME 20000 WITH_DATA
    {
        T.[0]      = 70                    ,
        T.[6, ]    = SOURCE
    }

 }
 ELSE IF(VAR1[0] == 4)
 {
      IF(S.IP_TTL>1)
       {
          SEND_OUT_IP WITH_DATA
          {
              T                    = S                    ,
              T.IP_TTL             = S.IP_TTL - 1         ,
              T.IP_HEADERCHKSUM = {0, 0}                  ,
              T.IP_HEADERCHKSUM = CHECKSUM(T.IP_HEADER)
          }
```

```
            }


    }
  // else if(VAR1[0] == 5)
  // {
  //            // LOST
  // }
    ELSE IF(VAR1[0] == 6)
    {
                IF(S.IP_TTL>1)
            {
                 SEND_OUT_IP WITH_DATA
                 {
                      T                          = S                          ,
                      T.IP_TTL                   = S.IP_TTL - 1               ,
                      T.IP_HEADERCHKSUM = {0, 0}                              ,
                      T.IP_HEADERCHKSUM = CHECKSUM(T.IP_HEADER)
                 }
            }
       VAR1[0] = 0 ;
    }


       DISCARD_MESSAGE;
}


TIMER_WITH_PERIOD 100
{
       FOR_EVERY_ELEMENT_IN_POOL 22
       {
        PE[0] = PE[0] - 1;
```

```
        IF(PE[0] == 0)
        {
            SEND_OUT_IP WITH_DATA
            {
                TARGET                  = PE[6, ],
                T.IP_TTL                = PE.IP_TTL - 1,
                T.IP_LEN                = LENGTH(T),
                T.IP_HEADERCHKSUM       = {0, 0}      ,
                T.IP_HEADERCHKSUM       = CHECKSUM(T.IP_HEADER)
            }
            REMOVE_CURRENT_POOL_ELEMENT;
        }
    }
}
```

2、NoIDIdleRQSender.mddl

```
VAR2[0, 3]          = {192, 168, 1, 1};         // SRC Address.
VAR2[4, 7]          = {192, 168, 2, 2};         // DST Address.
VAR2[8]              = 0;                        // No output message pending.
IP_OUT_HANDLER
{
    IF( S.IP_ADDRDST != VAR2[4, 7] || S.IP_PROT == CNST_IP_PROT_KDP )
        RETURN;
    DISCARD_MESSAGE;
    IF(VAR2[8]==1)
    {
        ADD_TO_POOL 19 WITH_LIFETIME 1800000 WITH_DATA
        {
            T = S
        }
```

```
        }
        ELSE
        {
            ASSIGN_VARIABLE 3 WITH_DATA
            {
                T.[0]           = 6                         ,           // Retry after 6 seconds
                T.[1]           = 4                         ,           // Retry at maximum 3 times
                T.[2, 3]        = LENGTH(S.IP_DATA) ,
                T.[4,]          = 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               = 0W                    ,
                T.IP_DATA.KDP_ACK               = 0                     ,
                T.IP_DATA.KDP_WINDOW_SIZE = 0                       ,
                T.IP_DATA.KDP_DATA              = S.IP_DATA
            }
            VAR2[8] = 1;                                            // Output message pending
        }
    }

    TIMER_WITH_PERIOD 1000
    {
     IF(VAR2[8] != 1)                                               // Output message pending
          RETURN;

          VAR3[0] = VAR3[0] - 1;
        GENERATE_USER_MSG WITH_DATA
```

```
{
      TARGET = VAR3[0]
}
 IF(VAR3[0] == 0)
 {
       VAR3[1] = VAR3[1] - 1;
      GENERATE_USER_MSG WITH_DATA
      {
            TARGET = VAR3[1]
      }
       IF(VAR3[1] == 0)
       {
            GENERATE_USER_MSG WITH_DATA
            {
                  TARGET = "Communication Aborted!"
            }
            VAR2[8] = 0;
       }
       ELSE
       {
            GENERATE_USER_SYSMSG WITH_DATA
            {
                  TARGET = "Retransmission!"
            }
            VAR3[0] = 6;                                    // Retry after 6 seconds
            SEND_OUT_IP WITH_DATA
            {
            T.IP_PROT                       = CNST_IP_PROT_KDP    ,
                T.IP_ADDRDST                    = VAR2[4, 7]          ,
            T.IP_DATA.KDP_ID                = 0W                  ,
```

```
                    T.IP_DATA.KDP_ACK         = 0                    ,
                    T.IP_DATA.KDP_WINDOW_SIZE = 0                    ,
                    T.IP_DATA.KDP_DATA        = VAR3[4, 4 + VAR3[2, 3] - 1]
                }
            }
        }
}

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
    {
        TARGET = "ACKED"
    }
    DISCARD_MESSAGE;
    VAR2[8] = 0;
    LOOK_FOR_ONE_ELEMENT_IN_POOL 19
    {
        ASSIGN_VARIABLE 3 WITH_DATA
        {
            T.[0]      = 6                    ,          // Retry after 6 seconds
            T.[1]      = 4                    ,          // Retry at maximum 3 times
            T.[2, 3]   = LENGTH(PE.IP_DATA),
            T.[4,]     = PE.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             = 0W                  ,
                T.IP_DATA.KDP_ACK        = 0                   ,
                T.IP_DATA.KDP_WINDOW_SIZE = 0                  ,
                T.IP_DATA.KDP_DATA       = PE.IP_DATA
        }
        VAR8[2] = 1;                                  // Output message pending
        REMOVE_CURRENT_POOL_ELEMENT;
    }
}
```

3、NoIDIdleRQReceiver.mddl

```
VAR2[0, 3]      = {192, 168, 2, 2};      // SRC Address.
VAR2[4, 7]      = {192, 168, 1, 1};      // DST Address.
IP_RECEIVED_HANDLER
{
    IF( S.IP_ADDRSRC != VAR2[4, 7] || S.IP_PROT != CNST_IP_PROT_KDP ||
       S.IP_DATA.KDP_ACK != 0W )
        RETURN;
    DISCARD_MESSAGE;
 SEND_OUT_IP WITH_DATA
 {
        T.IP_PROT                    = CNST_IP_PROT_KDP,
        T.IP_ADDRDST                 = VAR2[4, 7]      ,
        T.IP_DATA.KDP_ID             = 0W              ,
        T.IP_DATA.KDP_ACK            = 1               ,
        T.IP_DATA.KDP_WINDOW_SIZE    = 0
    }
     GENERATE_USER_MSG WITH_DATA
    {
```

```
            TARGET = S[24,]
        }
    }
```

4、SIDIdleRQSender.mddl

```
VAR1.SND_UNA      = 0W;                          // SND_UNA initialization.
VAR2[0, 3]        = {192, 168, 1, 1};        // SRC Address.
VAR2[4, 7]        = {192, 168, 2, 2};        // DST Address.
VAR2[8]           = 0;                           // No output message pending.
IP_OUT_HANDLER
{
    IF( S.IP_ADDRDST != VAR2[4, 7] || S.IP_PROT == CNST_IP_PROT_KDP )
        RETURN;
    DISCARD_MESSAGE;
    IF(VAR2[8]==1)
    {
        ADD_TO_POOL 19 WITH_LIFETIME 1800000 WITH_DATA
     {
            T = S
     }
    }
    ELSE
    {
        ASSIGN_VARIABLE 3 WITH_DATA
        {
            T.[0]          = 6                        ,       // Retry after 6 seconds
            T.[1]          = 4                        ,       // Retry at maximum 3 times
            T.[2, 3]       = LENGTH(S.IP_DATA) ,
            T.[4,]         = 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_UNA        ,
                T.IP_DATA.KDP_ACK               = 0                   ,
                T.IP_DATA.KDP_WINDOW_SIZE = 0                         ,
                T.IP_DATA.KDP_DATA              = S.IP_DATA
            }
            VAR2[8] = 1;                        // Output message pending
        }
      GENERATE_USER_MSG WITH_DATA
        {
            T.[4] = ((VAR1.SND_UNA)/10)+0X30,
            T.[5] = ((VAR1.SND_UNA)%10)+0X30,
            TARGET = "UNA "
        }

}
TIMER_WITH_PERIOD 1000
{
  IF(VAR2[8] != 1)                          // Output message pending
         RETURN;
         VAR3[0] = VAR3[0] - 1;
       GENERATE_USER_MSG WITH_DATA
       {
            TARGET = VAR3[0]
       }
         IF(VAR3[0] == 0)
         {
              VAR3[1] = VAR3[1] - 1;
```

```
        GENERATE_USER_MSG WITH_DATA
        {
            TARGET = VAR3[1]
        }
         IF(VAR3[1] == 0)
         {
              GENERATE_USER_MSG WITH_DATA
              {
                   TARGET = "Communication Aborted!"
              }
              VAR2[8] = 0;
         }
         ELSE
         {
              GENERATE_USER_SYSMSG WITH_DATA
              {
                   TARGET = "Retransmission!"
              }
              VAR3[0] = 6;                    // Retry after 6 seconds
              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_UNA        ,
                  T.IP_DATA.KDP_ACK            = 0                   ,
                  T.IP_DATA.KDP_WINDOW_SIZE = 0                      ,
                  T.IP_DATA.KDP_DATA           = VAR3[4, 4 + VAR3[2, 3] - 1]
              }
         }
    }
```

```
}
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.[4] = ((S.IP_DATA.KDP_ID)/10)+0X30,
          T.[5] = ((S.IP_DATA.KDP_ID)%10)+0X30,
          TARGET = "ACK "
     }
 IF(VAR1.SND_UNA + 1W != S.IP_DATA.KDP_ID)
      RETURN;

     DISCARD_MESSAGE;

     VAR1.SND_UNA = S.IP_DATA.KDP_ID;

     VAR2[8] = 0;
     LOOK_FOR_ONE_ELEMENT_IN_POOL 19
     {
          ASSIGN_VARIABLE 3 WITH_DATA
          {
               T.[0]        = 6                    ,          // Retry after 6 seconds
               T.[1]        = 4                    ,          // Retry at maximum 3 times
               T.[2, 3]     = LENGTH(PE.IP_DATA),
               T.[4,]       = PE.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_UNA        ,
                T.IP_DATA.KDP_ACK               = 0                   ,
                T.IP_DATA.KDP_WINDOW_SIZE = 0                         ,
                T.IP_DATA.KDP_DATA              = PE.IP_DATA
            }
            VAR8[2] = 1;                        // Output message pending
            REMOVE_CURRENT_POOL_ELEMENT;
        }
    }
```

5、SIDIdleRQReceiver.mddl

```
    VAR1.RCV_NXT      = 0W;                     // RCV_NXT initialization.
    VAR2[0, 3]        = {192, 168, 2, 2};       // SRC Address.
    VAR2[4, 7]        = {192, 168, 1, 1};       // DST Address.
    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)
            RETURN;
        VAR1.RCV_NXT = VAR1.RCV_NXT + 1W;
     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       = 0
        }
         GENERATE_USER_MSG WITH_DATA
        {
            TARGET = S[24,]
        }
    }
```

6、SIDCRQSender.mddl

```
    VAR1.SND_UNA        = 0W;                       // SND_UNA initialization.
    VAR1.SND_NXT        = VAR1.SND_UNA;             // SND_NXT 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 >= 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;
         GENERATE_USER_MSG WITH_DATA
     {
          T.[4] = ((VAR1.SND_UNA)/10)+0X30,
          T.[5] =((VAR1.SND_UNA)%10)+0X30,
          TARGET = "UNA "
     }
}
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
            {
                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.[4] =((S.IP_DATA.KDP_ID)/10)+0X30,
        T.[5] = ((S.IP_DATA.KDP_ID)%10)+0X30,
        TARGET = "ACK "
    }
    GENERATE_USER_MSG WITH_DATA
```

```
        {
            T.[4] =((VAR1.SND_NXT)/10)+0X30,
            T.[5] = ((VAR1.SND_NXT)%10)+0X30,
            TARGET = "NXT "
        }

    IF(VAR1.SND_UNA - S.IP_DATA.KDP_ID <    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;
}
```

7、SIDCRQReceiver.mddl

```
VAR1.RCV_NXT       = 0W;                          // RCV_NXT 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;
    LOOK_FOR_ONE_ELEMENT_IN_POOL 21 WITH_CONDITION (PE.IP_DATA.KDP_ID
==
    S.IP_DATA.KDP_ID)
        RETURN;
    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])
            CONTINUE;
        ELSE
            BREAK;
    }
    IF(VAR3[4, 5] == VAR1.RCV_NXT)
        RETURN;
    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   = 0
        }
    }
```