Exp 1：以太网数据的观察与传送

目的：了解基本以太网络的信息交换并观察其结果。

摘要：本实验利用使用者自定义的规则传送和回应以太网数据包，并凭借MDDL和GUI 工具传送数据包，让学生观察以太网封包的运动。

时数：1.5 hrs。

一、网络拓扑

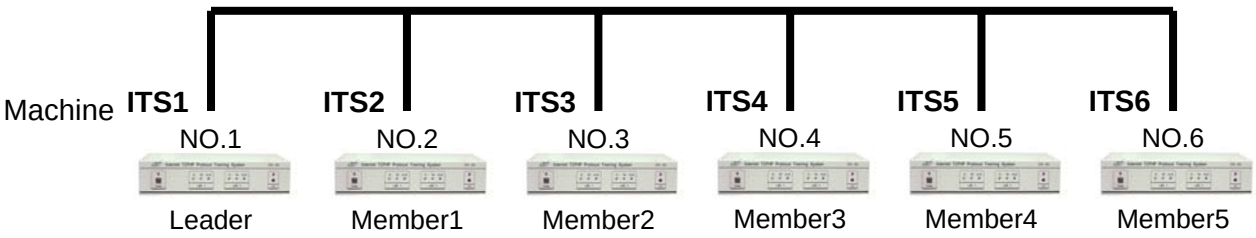

图 1.1

二、技术背景

众所周知以太网络，已是当今最普遍的网络技术，并且被视为网络学术的基础。Ethernet frames 的最大长度为1500 Bytes，表1.1 中记录Ethernet frame 格式其中包含6 Bytes (48Bits) 的source Mac address 及destination Mac address，及16-bit 分组类型 (例如：0x0800为IP 报文、0x0806 为ARP 报文及0x0835 为RARP 报文) 。更详细的以太网数据包的结构请参考附录A。

Ethernet Frame			
Header			**Data**
Destination MAC address	Source MAC address	Type	46(minimum)-1500(maximum)
0	6	12	14 1514

表 1.1

三、实验步骤

1、网络拓扑连线

在 Hubox 上将网络连线如图 1.2 所示（将所有 ITS1 的 LAN1 接通）。本实验可让 6 台 ITS 连接为一个完整的实验网络。其中 1 台 ITS 定义为 Leader，其余 ITS 分别定义为 Members（编号为 1 到 5）。此外，在本实验中学生们可以自行交换角色。

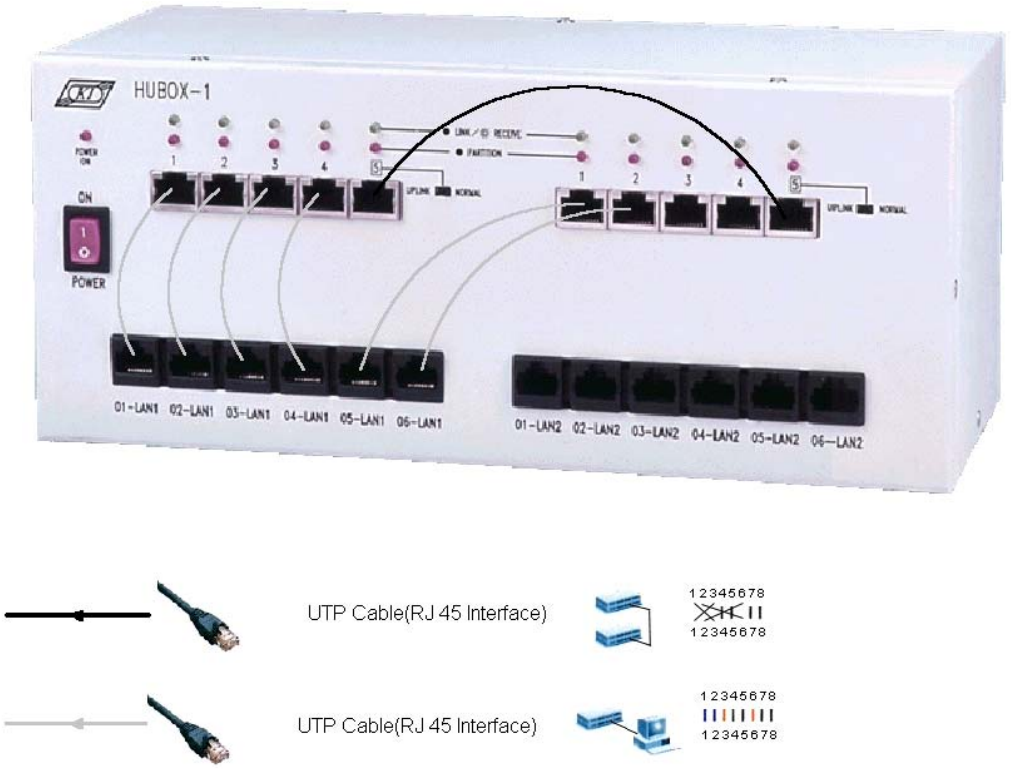

图 1.2

2、发送以太网帧

A. Leader 发送一个全广播帧给所有的 Members

Leader 与 Members

1）执行 **XCLIENT.BAT，** 打开 KCodes Network Explorer.

2）勾选 **Listening On**.

3）从 **Listening menu 中** 选择监听等级（**Listening Level**）. 将 **Interface Frames** 打勾。

4）再从 **Listening menu 中** 选择 **New Memorized Message Brower** 。如图 1.3 所示，网络信息浏览器（Network Message Browser）将会被打开，可以及时监听整个网络上的信息传输。

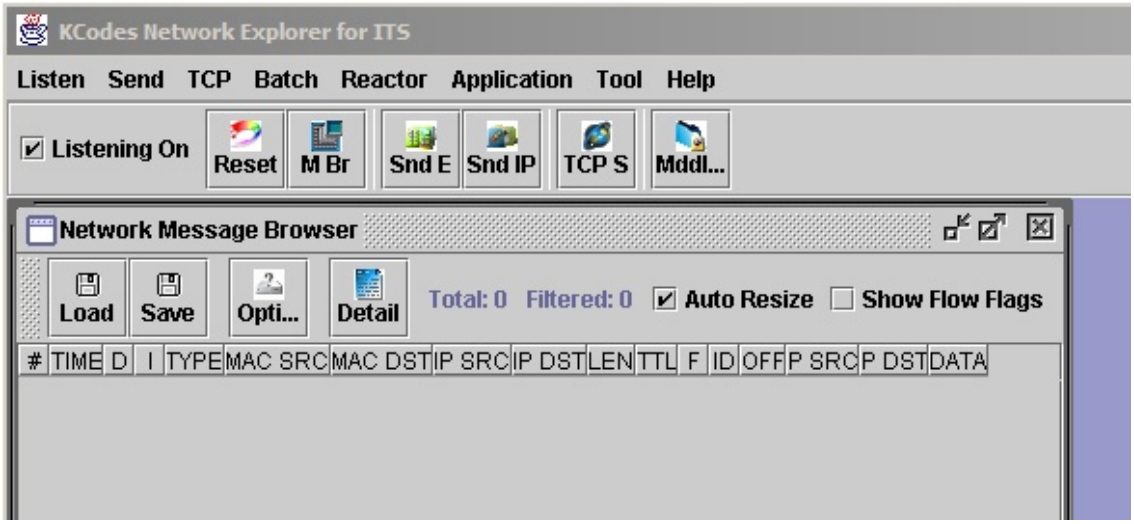

图 1.3

Leader

5）从 **Send menu** 里，选择 **Send Interface Frame** ，我们可以打开一个名称为 **Network Message Sender** 的对话框。

6）在 **Destination MAC Address** 对话框中，输入“**FF:FF:FF:FF:FF:FF**”，**Data** 对话框中输入“**this is a broadcast from leader**”， 如图 1.4 所示，最后按下 **Send** 按钮，Leader 就会发送一个全广播数据帧给所有的组员。

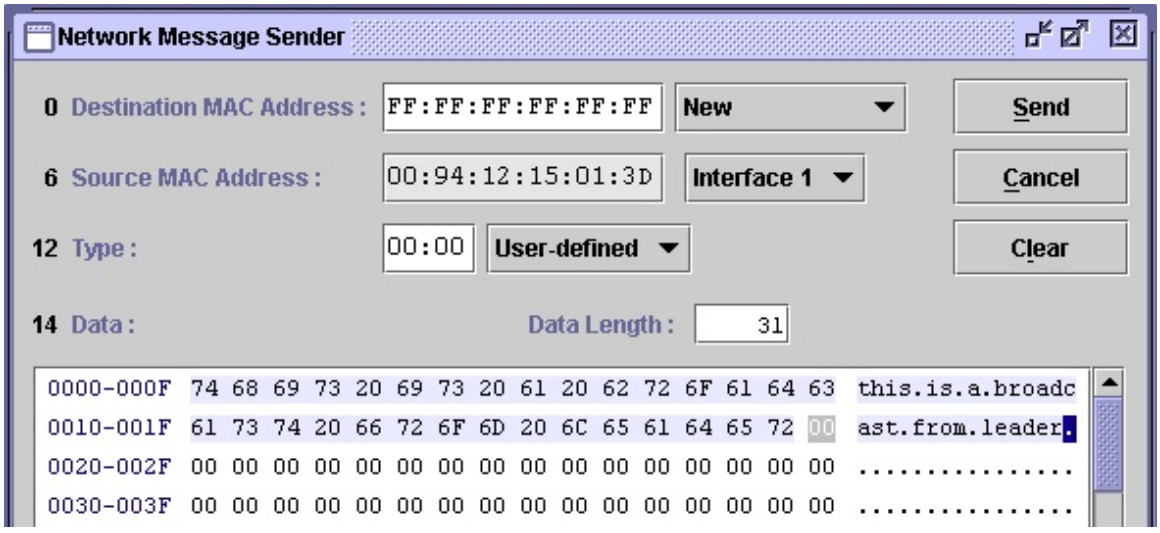

图 1.4

Members

7）当 Leader 发送一个全广播帧后，每一个组员将会收到该帧，并可以通过网络信息浏览器观察到该帧内容，如图 1.5 所示。选择这个帧后，再点击 **Detail** 按钮。

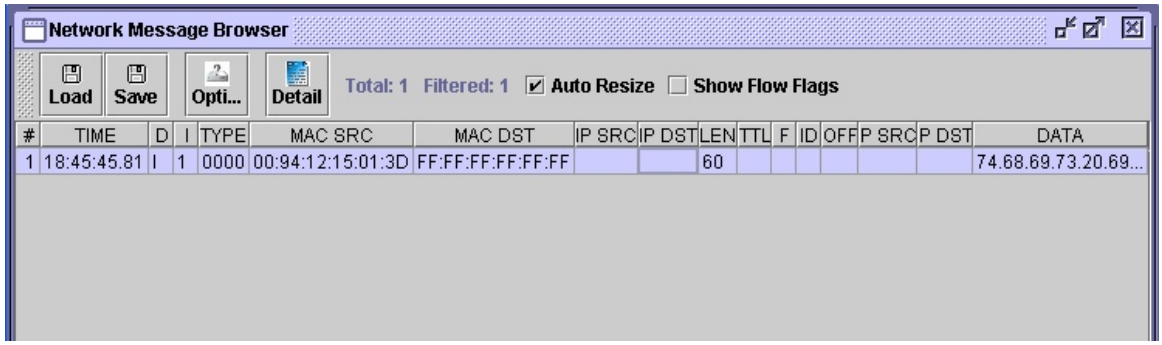

图 1.5

8）见 图 1.6 当按下 Detail 按钮后，将会打开一个名为 Ethernet Frame Viewer 的对话框界面，这里面详细的记录了这个广播帧的所有具体内容。其中，我们可以看到 **Source Ether Address** 为“**00:94:12:15:01:3D**”，这就是本实验中 Leader 机器的网卡 MAC 地址，请将它记录下来，继续完成后面的实验。

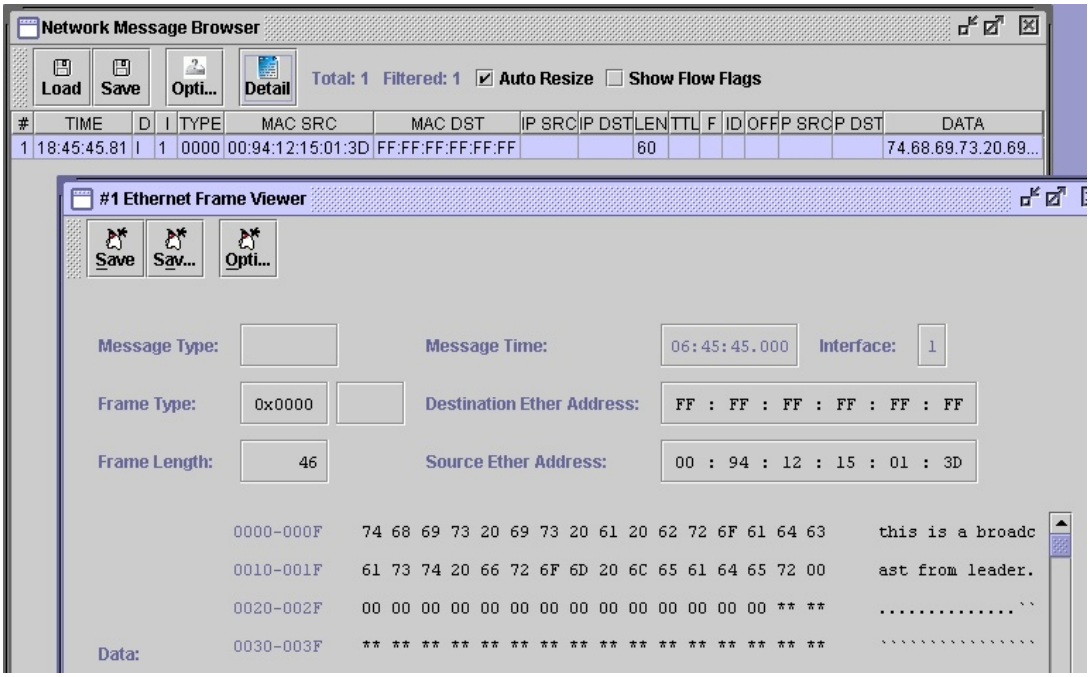

图 1.6

B. Members 发送全广播数据帧给 Leader

Members

9）打开一个新的“**Network Message Sender**”对话框。

10）在 **Destination MAC Address** 对话框中输入 “**FF:FF:FF:FF:FF:FF**” ，**Data** 对话框中输入“**this is a broadcast from member<*your number*>**”，如 图 1.7 所示，最后按下 **Send** 按钮.所有 Member 都会发送一个全广播帧给网络上的所

有成员。

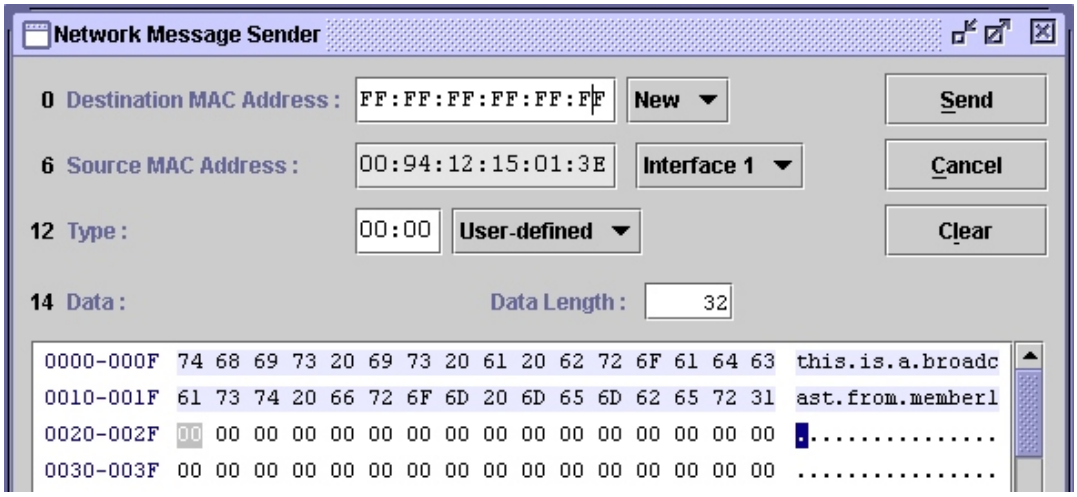

图 1.7

Leader

11）当每个 Member 都送出全广播数据帧后，我们可以从网络信息浏览器（Network Message Browser）中观察到所有的广播帧，如图 1.8 所示。请将所有网卡的 MAC 地址记录下来，并完成表 1.2。

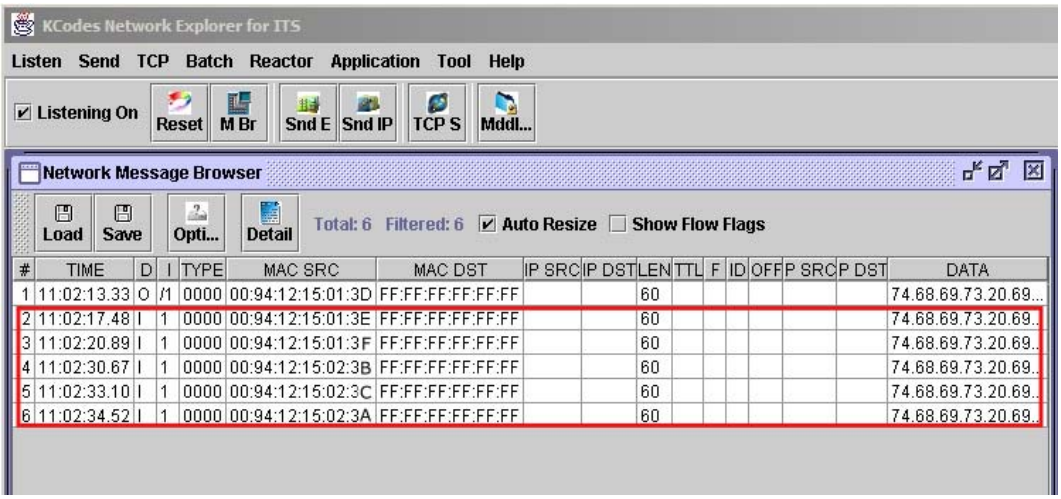

图 1.8

表 1.2

Name	MAC Address
Leader	
Member1	
Member2	
Member3	
Member4	
Member5	

C. Leader 发送一个单播（unicast）帧给所有的 Members

Leader

12）打开一个新的 **Network Message Sender** 对话框。

13)在 **Destination MAC Address** 对话框中输入 Member1 的网卡 MAC 地址，Data 对话框中输入 “**hi member1 this is a unicast from leader**”，如图 1.9 所示。最后单击 **Send** 按钮. Leader 就会发送一个只给 Member 1 的单播数据帧。

14）重复步骤 14 的操作，发送单播帧给其他的 Members.

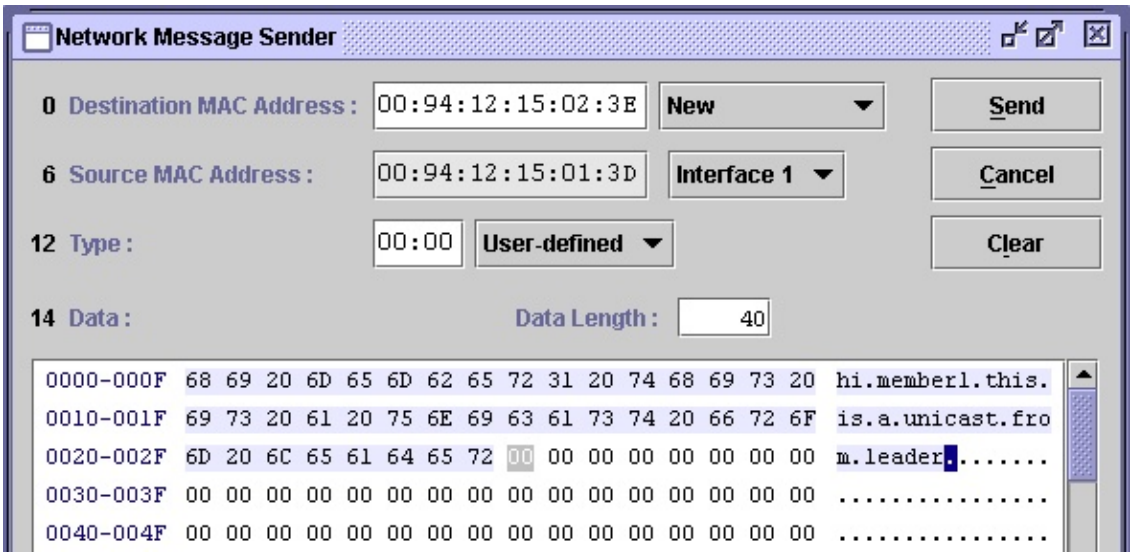

图 1.9

D. Members 发送单播（Unicast）帧给 Leader

Members

15）打开一个新的 **Network Message Sender** 对话框界面。

16)在 **Destination MAC Address** 文本框中输入 Leader 的网卡 MAC 地址，Data

文本框中输入“**hi leader this is a unicast from member<*your number*>**”，如图 1.10 所示。最后单击 **Send** 按钮，每一位 Member 都会发送一个单播帧给 Leader，换言之，Leader 将会收到所有 Members 发送的单播帧。

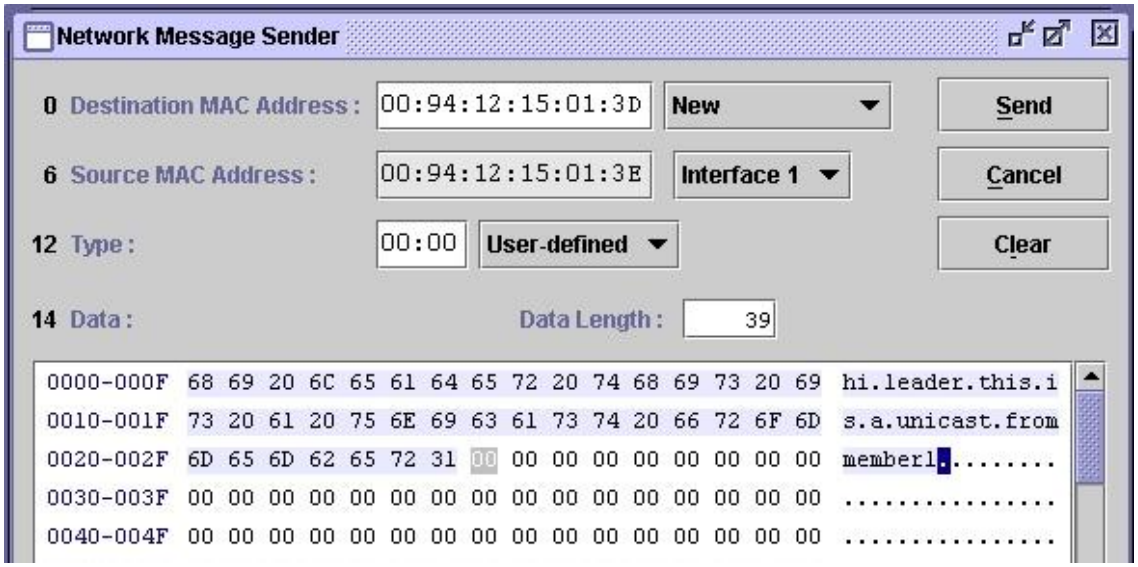

图 1.10

3、利用 MDDL 发送以太网帧

A. 发送并自动恢复广播帧

Leader and Members

17）重新打开一个新的网络信息浏览器“Network Message Browser”，并打勾 **Listening On**。

18）从 **Reactor menu** 中找到并执行 **MDDL Reactor Panel**，会自动打开一个名为 MDDL Editor 的 MDDL 程序编辑界面。

Members

19）在 MDDL 编辑界面里，点击 **Load** 按钮，调用程序 AutoResponser.mddl（路径为 C: \XClient \Data \Mddl \Tutorial \Ex01 \AutoResponser.mddl）如图 1.11 所示，单击 **Upld** 按钮将程序载入 ITS 中。

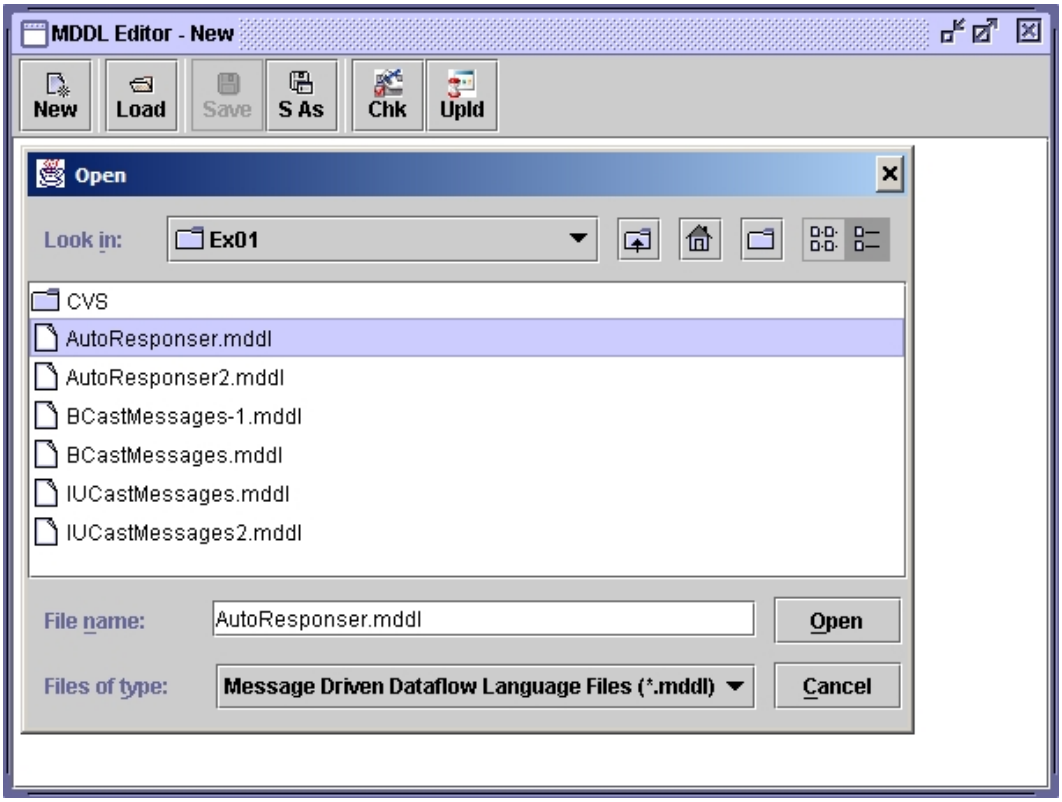

图 1.11

Leader

20）在 MDDL 编辑界面中，单击 **Load** 按钮，调用 BCastMessages.mddl （路径为 C: \XClient \Data \Mddl \Tutorial \Ex01 \BCastMessages.mddl）。

21）单击 **Upld** 按钮载入程序，ITS 将会自动且持续不断的送出一个相同的广播帧，以每 5 秒一次的频率送给网络上的所有成员。

B. 发送并自动恢复单播帧

Leader and Members

22）单击 **Reset** 按钮，复位网络信息浏览器“Network Message Browser”勾选 **Listening On**。

Members

23）在 MDDL 编辑界面中，点击 **Load** 按钮，调用 AutoResponser.mddl 程序（路径为 C: \XClient \Data \Mddl \Tutorial \Ex01 \AutoResponser.mddl），最后单击 **Upld** 按钮将程序载入 ITS 中。

Leader

24）在 MDDL 编辑界面中，点击 **Load 按钮**调用 IUCastMessages.mddl 程序（路

径为 C: \XClient \Data \Mddl \Tutorial \Ex01 \IUCastMessages.mddl）。

25）参考 表 1.1, 把所有 Member 的 MAC 地址添加到 IUCastMessages.mddl 程序中，如 图 1.12 所示。最后单击 **Upld** 按钮，将程序载入到 ITS 中。ITS 将会自动且持续不断的送出单播帧，以每 5 秒一次的频率各自动给网络上的所有成员，而 Member 收到后，将会自动回复 Leader。

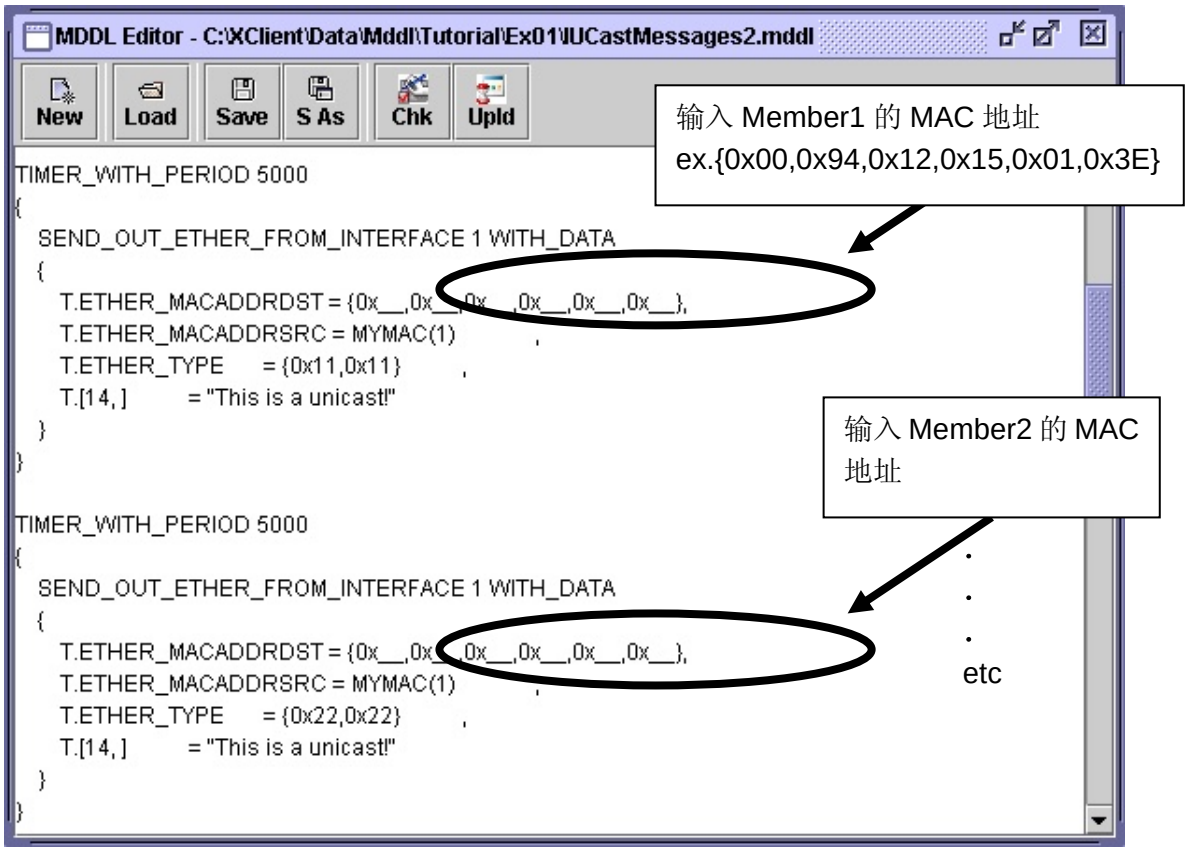

图 1.12

DISCUSSIONS

1、广播帧（broadcast frame）和单播帧(unicast frame)的区别是什么？

2、什么是以太网类型（Ethernet Type）？它的重要性是什么? (请参考附录 A)

3、试着在实验中用 MDDL 平台载入另一个程序“BCastMessages-1.mddl”(路径是 C:\XClient \Data \Mddl \Tutorial \Ex01)，比较“BCastMessages.mddl”和“ BCastMessages-1.mddl”这两个程序的区别 (参考附录 B)。

4、讨论“BCastMessages.mddl ”和“IUCastMessages.mddl”这两个程序代码的内容. 我们如何才能同时发送广播和单播的数据帧？是不是可以将“BCastMessages.mddl”和“IUCastMessages.mddl”两个程序合并?请试着改写程序，并重新做一次实验。

REACTOR PROGRAMS

1、BCastMessages.mddl

```
TIMER_WITH_PERIOD 5000
{
    SEND_OUT_ETHER_FROM_INTERFACE 1 WITH_DATA
    {
        T.ETHER_MACADDRDST = CNST_MACADDR_BROADCAST,
        T.ETHER_MACADDRSRC = MYMAC(1)                ,
        T.ETHER_TYPE       = {0xAA,0xAA}             ,
        T.[14, ]           = "This is a broadcast!"
    }
}
```

2、BCastMessages-1.mddl

```
TIMER_WITH_PERIOD 5000
{
    SEND_OUT_ETHER_FROM_INTERFACE 1 WITH_DATA
    {
        T.[0,5] = {0XFF,0XFF,0XFF,0XFF,0XFF,0XFF},
        T.[6,11] = MYMAC(1)                ,
        T.[12,13]       = {0xAA,0xAA}      ,
        T.[14, ]        = "This is a broadcast!"
    }
}
```

3、IUCastMessages.mddl

```
  TIMER_WITH_PERIOD 5000
  {
      SEND_OUT_ETHER_FROM_INTERFACE 1 WITH_DATA
```

```
        {
            T.ETHER_MACADDRDST = {0x__,0x__,0x__,0x__,0x__,0x__},
            T.ETHER_MACADDRSRC = MYMAC(1)                    ,
            T.ETHER_TYPE       = {0x22,0x22}                 ,
            T.[14, ]           = "This is a unicast!"
        }
    }

    TIMER_WITH_PERIOD 5000
    {
        SEND_OUT_ETHER_FROM_INTERFACE 1 WITH_DATA
        {
            T.ETHER_MACADDRDST = {0x__,0x__,0x__,0x__,0x__,0x__},
            T.ETHER_MACADDRSRC = MYMAC(1)                    ,
            T.ETHER_TYPE       = {0x33,0x33}                 ,
            T.[14, ]           = "This is a unicast!"
        }
    }

    TIMER_WITH_PERIOD 5000
    {
        SEND_OUT_ETHER_FROM_INTERFACE 1 WITH_DATA
        {
            T.ETHER_MACADDRDST = {0x__,0x__,0x__,0x__,0x__,0x__},
            T.ETHER_MACADDRSRC = MYMAC(1)                    ,
            T.ETHER_TYPE       = {0x44,0x44}                 ,
            T.[14, ]           = "This is a unicast!"
        }
    }
```

```
TIMER_WITH_PERIOD 5000
{
    SEND_OUT_ETHER_FROM_INTERFACE 1 WITH_DATA
    {
        T.ETHER_MACADDRDST = {0x__,0x__,0x__,0x__,0x__,0x__},
        T.ETHER_MACADDRSRC = MYMAC(1)                    ,
        T.ETHER_TYPE       = {0x55,0x55}                 ,
        T.[14, ]           = "This is a unicast!"
    }
}

TIMER_WITH_PERIOD 5000
{
    SEND_OUT_ETHER_FROM_INTERFACE 1 WITH_DATA
    {
        T.ETHER_MACADDRDST = {0x__,0x__,0x__,0x__,0x__,0x__},
        T.ETHER_MACADDRSRC = MYMAC(1)                    ,
        T.ETHER_TYPE       = {0x66,0x66}                 ,
        T.[14, ]           = "This is a unicast!"
    }
}
```

4、AutoResponser.mddl

```
ETHER_IN_HANDLER ANY
{
    IF(S.ETHER_MACADDRDST==MYMAC( INTERFACE() ))
    {
        SEND_OUT_ETHER_FROM_INTERFACE INTERFACE() WITH_DATA
        {
            T                    = S                    ,
```

```
                T.ETHER_MACADDRDST    = S.ETHER_MACADDRSRC                ,
                T.ETHER_MACADDRSRC    = MYMAC( INTERFACE() )              ,
                T.ETHER_TYPE          = {0xAA,0xAA}                       ,
                T.[16, ]              = "Received your unicast!"
            }
        }
        IF(S.ETHER_MACADDRDST==CNST_MACADDR_BROADCAST)
        {
            SEND_OUT_ETHER_FROM_INTERFACE INTERFACE() WITH_DATA
            {
                T                     = S                                 ,
                T.ETHER_MACADDRDST    = S.ETHER_MACADDRSRC                ,
                T.ETHER_MACADDRSRC    = MYMAC( INTERFACE() )              ,
                T.ETHER_TYPE          = {0xBB,0xBB}                       ,
                T.[16, ]              = "Received your broadcast!"
            }
        }
    }
```