Part 1: Data Link Layer

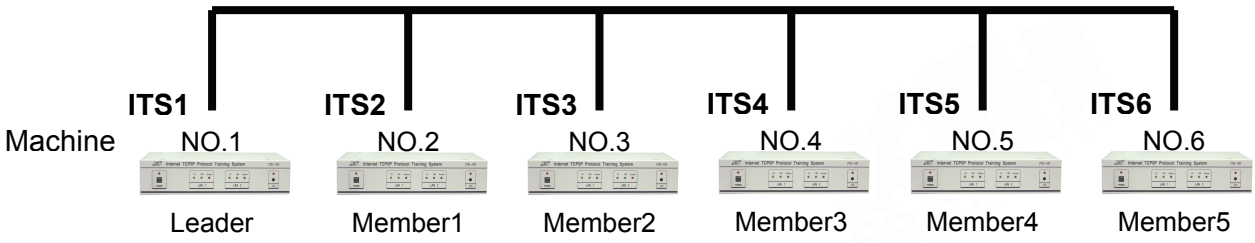
Exp 1. Ethernet Messages Sending & Observing

OBJECTIVE：To understand the basic Ethernet message transfer and observe the result.

BRIEF DESCRIPTION：This experiment examines the Ethernet message transfer using a user-defined protocol. By using MDDL, and GUI tool, students can learn how Ethernet message transfers.

DURATION：1.5 hrs

TOPOLOGY



TECHNICAL BACKGROUND

It is well known that Ethernet has become the most popular LAN technology and can be regarded as a link level connection in network. Ethernet frames are of variable length, with no frame smaller than 60 octets or larger than 1514 octets (header and data). Figure 1.1 shows that the Ethernet frame format contains the physical source address as well as the destination address. The frame type field contains a 16-bit integer that identifies the type of the data being carried in the frame. Appendix A shows some used Ethernet types.

Ethernet Frame			
Header			**Data**
Destination MAC address	Source MAC address	Type	46(minimum)-1500(maximum)

0 6 12 14 1513

Figure 1.1

PROCEDURE

To perform the laboratory experiment, let six students make a group. One student acts Leader and others act Members 1 through 5, respectively. Students play the role of Leader by turns.

Realizing the Network Topology

1. Complete the network connections on HUBOX by referring to Figure 1.2.

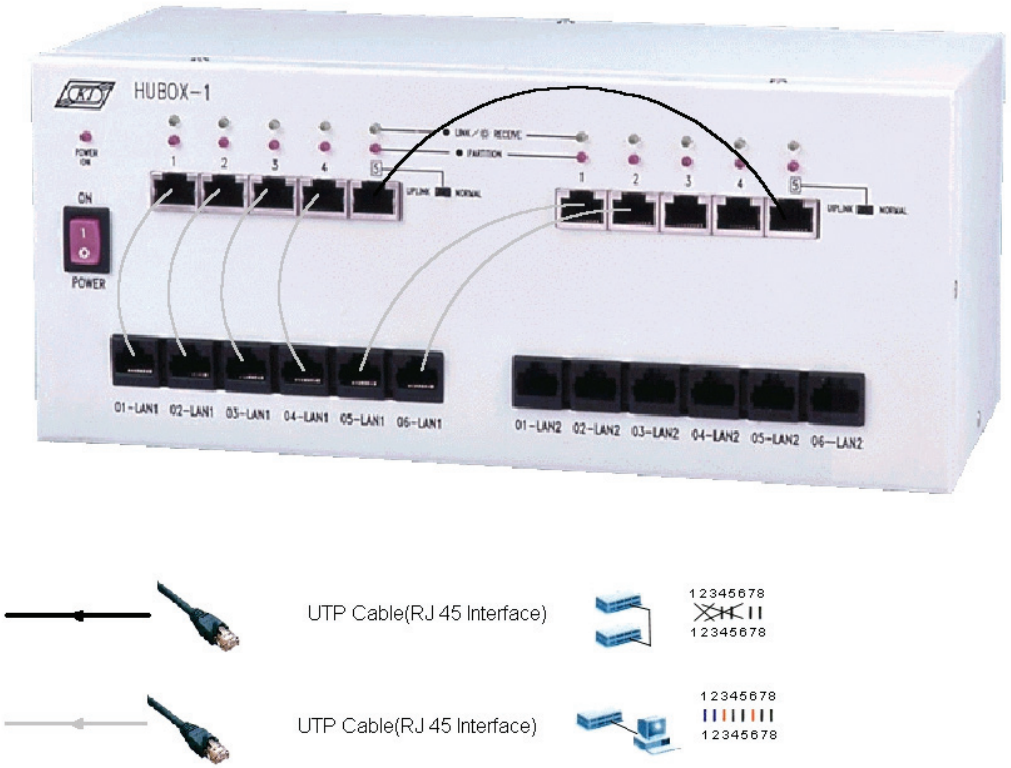


Figure 1.2

Creating Ethernet Messages

A. Leader Sends a Broadcast Frame to Members

Leader and Members

2. Execute **XCLIENT.BAT** to open the KCodes Network Explorer for ITS window.
3. Check the **Listening On**.
4. Choose **Listening Level** from the Listen menu. Check **Interface Frames**.
5. Select **New Memorized Message Brower** from the Listen menu to open the Network Message Browser (see Figure 1.3) that starts to monitor the network transmission.

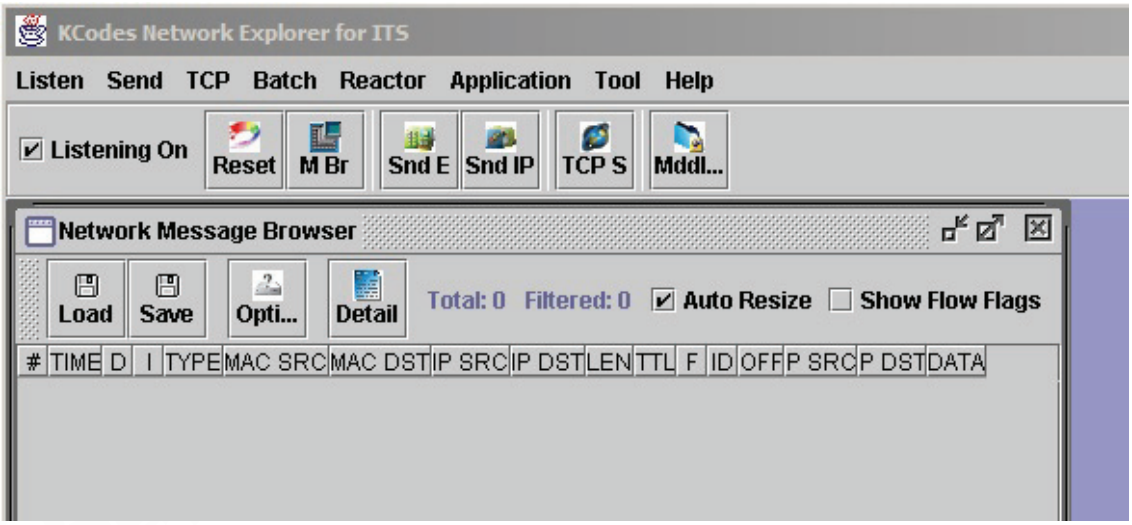


Figure 1.3

Leader

6. Open the Network Message Sender by selecting **Send Interface Frame** from the Send menu.

7. In the Network Message Sender dialog box, type "**FF:FF:FF:FF:FF:FF**" into Destination MAC Address textbox, enter "**this is a broadcast from leader**" into Data (see Figure 1.4), and click the **Send** button.

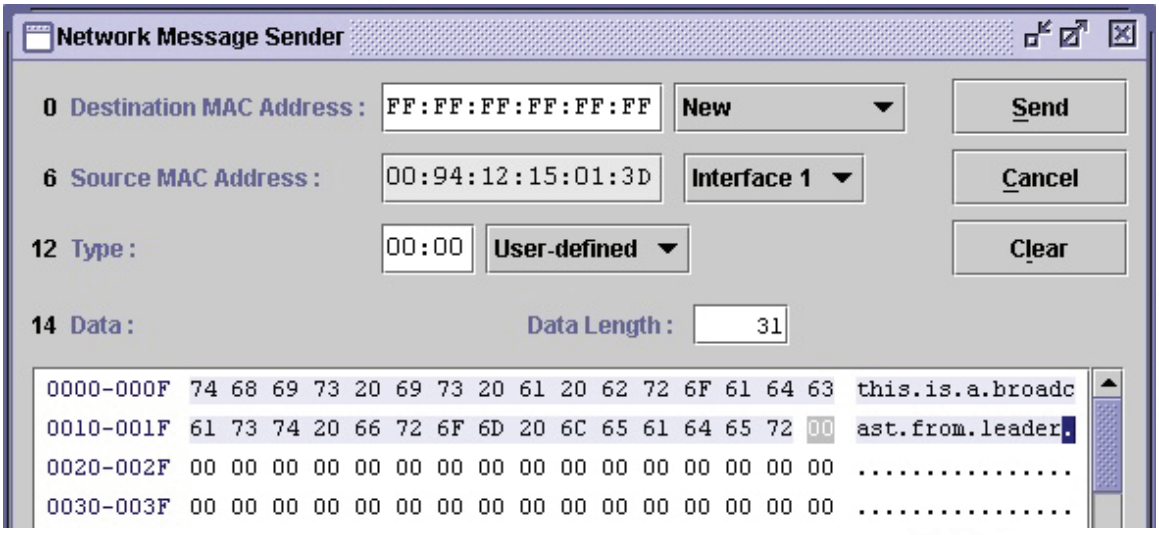


Figure 1.4

Members

8. You should now receive an Ethernet frame as shown in Figure 1.5. Choose this frame by clicking on any field of the frame and press the **Detail** button.

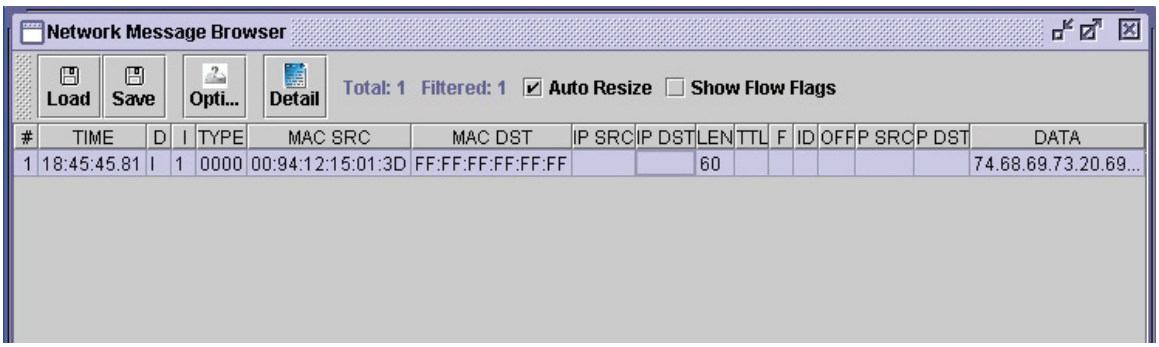


Figure 1.5

9. The Ethernet Frame Viewer appears as shown in Figure 1.6. It shows all details of the frame. Note that the Source Ether Address (00:94:12:15:01:3D) is Leader's address. Write down this address for use in the next steps.

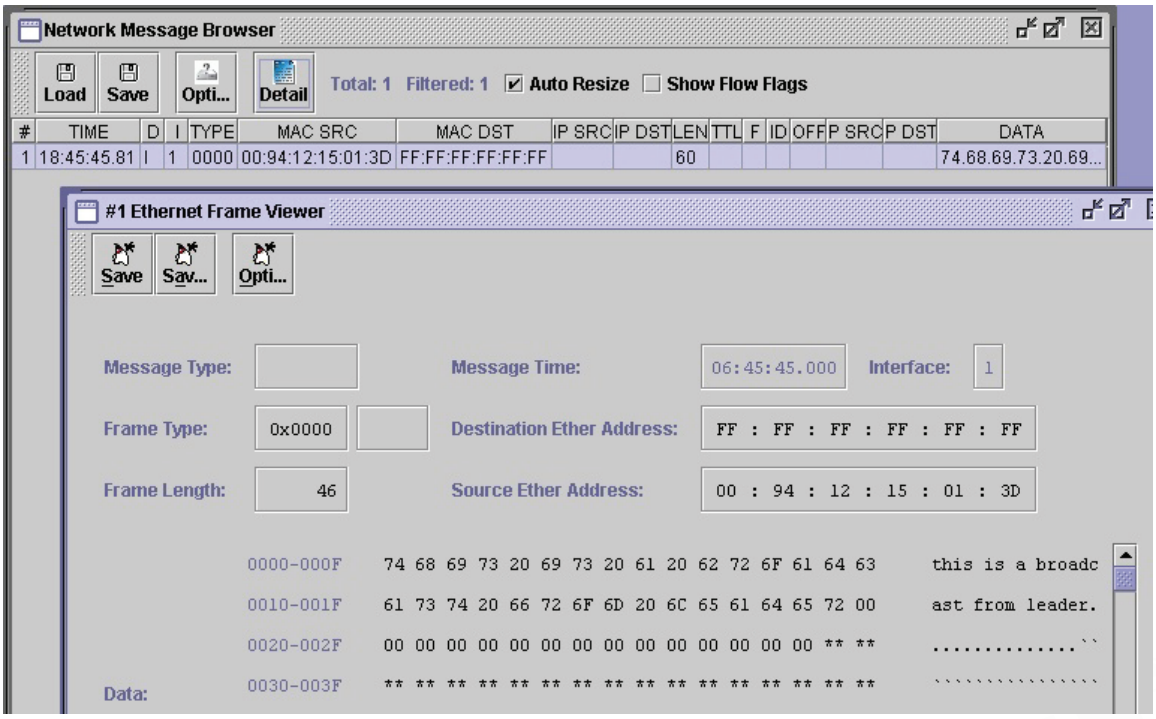


Figure 1.6

B. Members Send Broadcast Frames to Leader

Members

10. Open a new Network Message Sender.
11. Type "**FF:FF:FF:FF:FF:FF**" into Destination MAC Address textbox, enter "**this is a broadcast from member<*your number*>**" into Data (see Figure 1.7), and click the **Send** button.

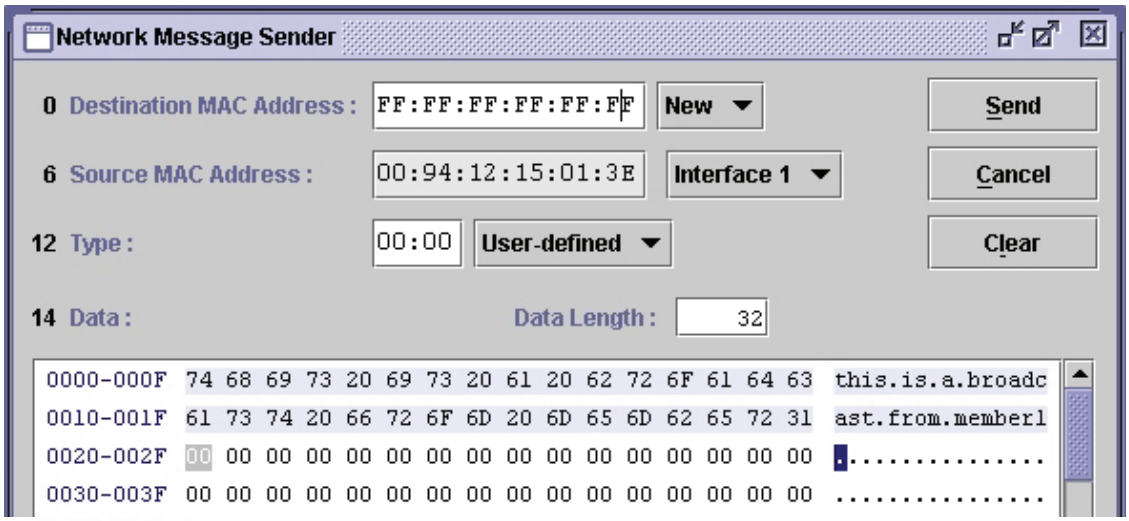


Figure 1.7

Leader

12. After every member has sent their broadcast frames, you should see the other 5 frames in the Network Message Browser as shown in Figure 1.8. Try to look up their MAC Addresses and complete Table 1.1.

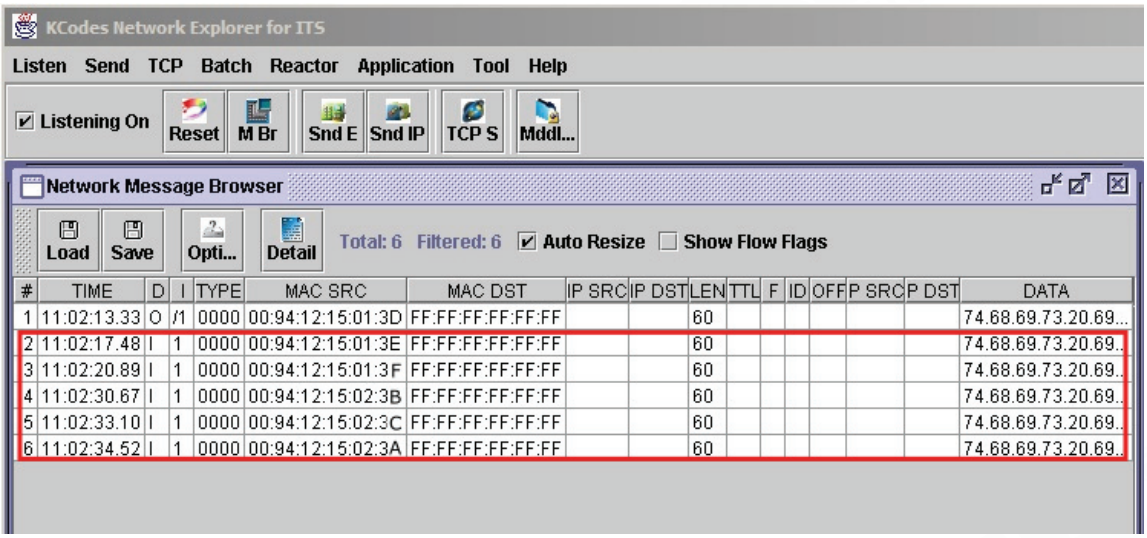


Figure 1.8

Table 1.1

Name	MAC Address
Leader	
Member1	
Member2	
Member3	
Member4	
Member5	

C. Leader Sends a Unicast Frame to Members

Leader

13. Open a new Network Message Sender.
14. Type <***MAC Address of Member1***> into Destination MAC Address textbox, enter "**hi member1 this is a unicast from leader**" into Data (see Figure 1.9), and click the **Send** button. Member 1 should now receive the frame from Leader.
15. Repeat Step 14 for Members 2 through 5.

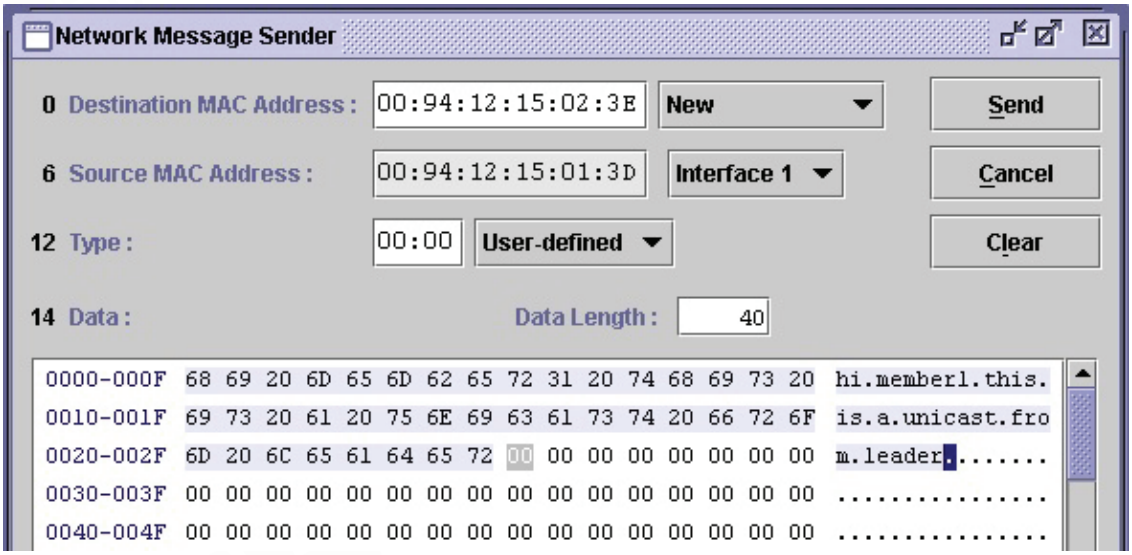


Figure 1.9

D. Members Send Unicast Frames to Leader

Members

16. Open a new Network Message Sender.
17. Type <***MAC Address of Leader***> into Destination MAC Address textbox, enter "**hi leader this is a unicast from member<*your number*>**" into Data (see Figure 1.10), and click the **Send** button. Leader should receive the frames from Members.

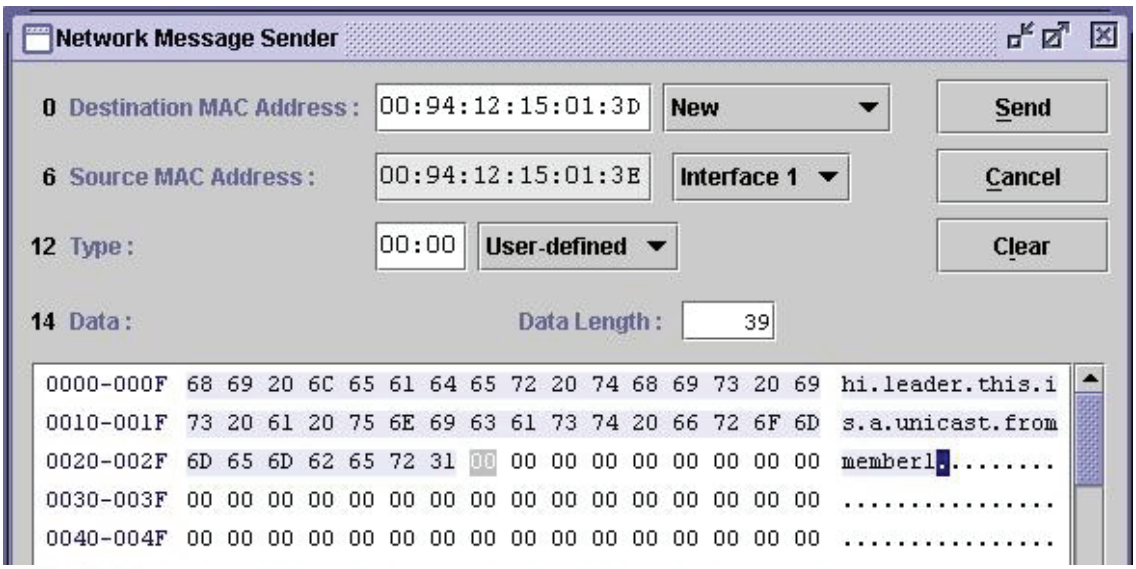


Figure 1.10

Sending Ethernet Messages by MDDL Platform

A. Send and Autorespond Broadcast Messages

Leader and Members

18. Open the Network Message Browser. Check **Listening On**.
19. Open the MDDL Editor by selecting the **MDDL Reactor Panel** from the Reactor menu.

Members

20. In the MDDL Editor window, click the **Load** button and open the file C: \XClient \Data \Mddl \Tutorial \Ex01 \AutoResponser.mddl (see Figure 1.11). Click the **UpId** button.

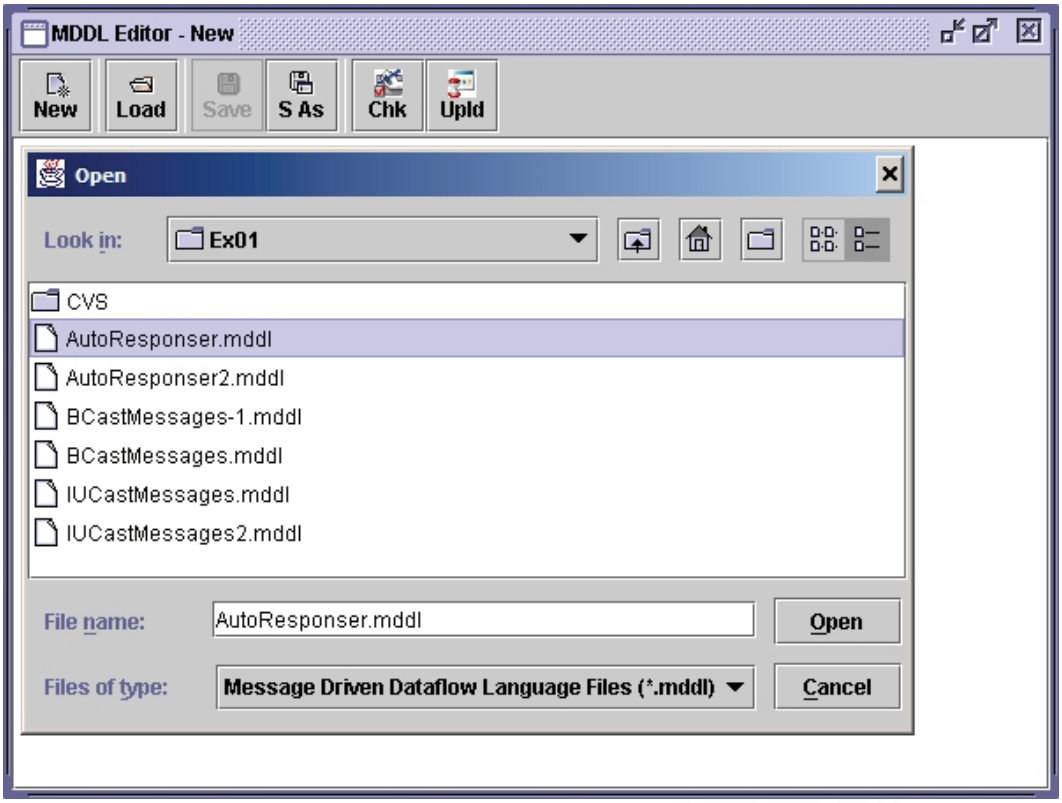


Figure 1.11

Leader

21. In the MDDL Editor window, click the **Load** button and open the file C: \XClient \Data \Mddl \Tutorial \Ex01 \BCastMessages.mddl.
22. Click the **UpId** button. Your ITS will continuously send an identical broadcast Ethernet message to Members at intervals of 5 seconds.

B. Send and Autorespond Unicast Messages

Leader and Members

23. Reset the Network Message Browser. Check **Listening On**.

Members

24. In the MDDL Editor window, click the **Load** button and open the file C: \XClient \Data \Mddl \Tutorial \Ex01 \AutoResponser.mddl. Click the **UpId** button.

Leader

25. In the MDDL Editor window, click the **Load** button and open the file C: \XClient \Data \Mddl \Tutorial \Ex01 \IUCastMessages.mddl.

26. Referring to Table 1.1, add <***all the MAC addresses of Members***> into the program as shown in Figure 1.12. Finally click the **UpId** button. Your ITS will continuously send unicast Ethernet messages to Members at intervals of 5 seconds. Members should receive the frames from Leader than auto-responds it.

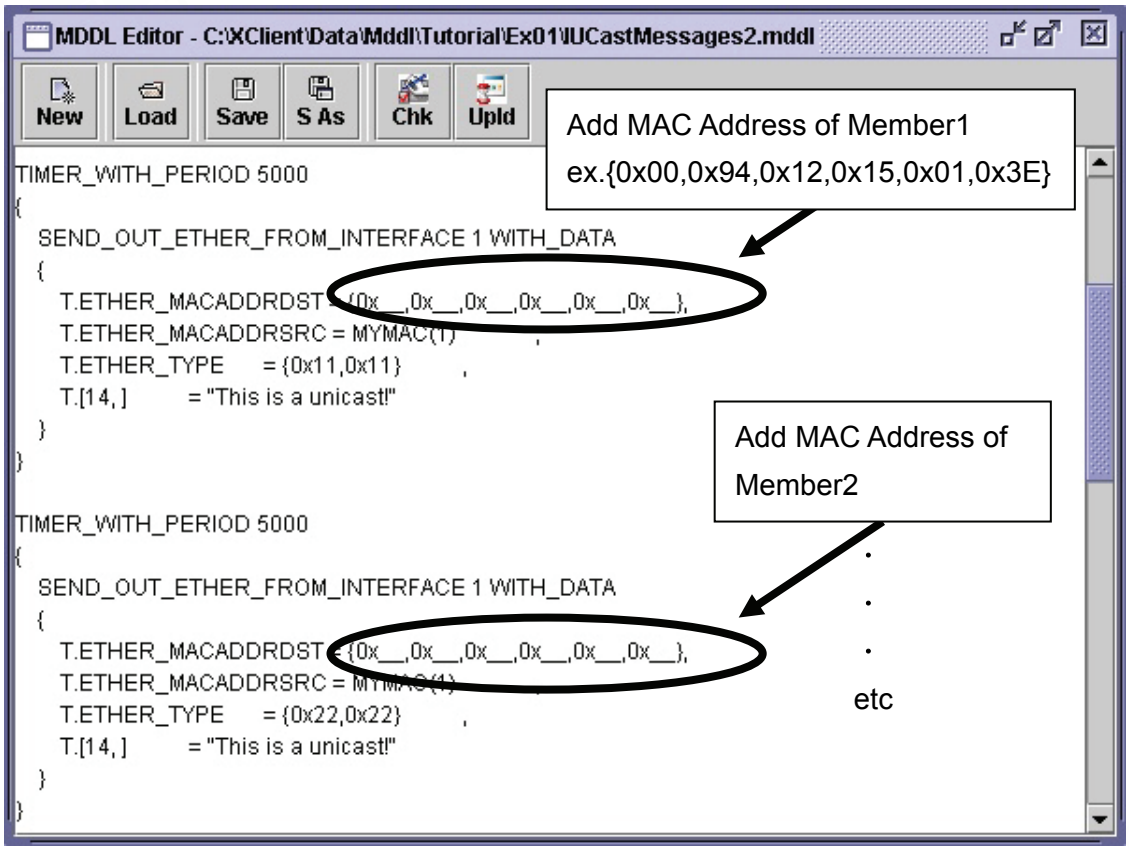


Figure 1.12

DISCUSSIONS

1. What's the difference between broadcast and unicast frame?
2. What's the Ethernet Type? Is it important? (refer to Appendix A)
3. Try to load the BCastMessages-1.mddl file (path: C: \XClient \Data \Mddl \Tutorial \Ex01) in the MDDL Editor, then compare BCastMessages.mddl and BCastMessages-1.mddl (refer to Appendix B).
4. Discuss the programs of BCastMessages.mddl and IUCastMessages.mddl. How we're able to auto-send broadcast and unicast messages together? Can we mix BCastMessages.mddl and IUCastMessages.mddl? Try to mix BCastMessages.mddl and IUCastMessages.mddl, then play again.

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!"
        }
    }
}
```

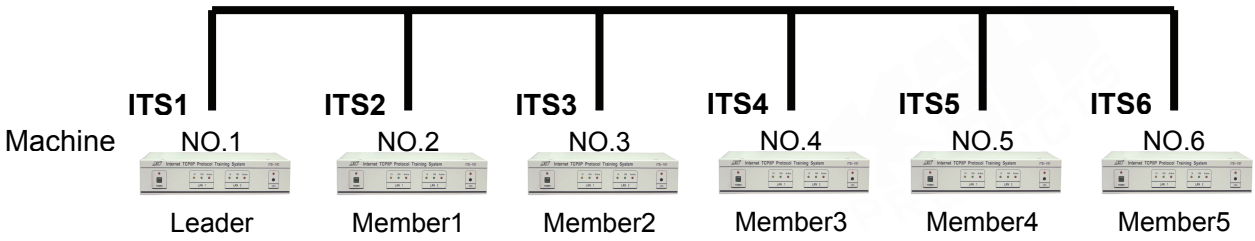
Exp 2. MAC Address Discovery

OBJECTIVE：To learn how to write a simple reactor handler program, and compare the experiment results with the ones by using manual operation.

BRIEF DESCRIPTION：The basic mechanism of network communication, request and reply.

DURATION：3 hrs

TOPOLOGY



TECHNICAL BACKGROUND

A simple non-standard communication protocol (depicted as in Figure 2.1) is defined to query the MAC addresses of ITS interfaces. The protocol consists of the two Ethernet types: one (0xA001) is the *MAC address request* and another (0xA002 or 0xA003) is *MAC address response*. First the students send the MAC address request to query the desired MAC address of the ITS interfaces, which can be identified by the pair of machine number and interface number. When receiving the above request messages, the students choose one of the two formats: Format 1 and Format 2 to reply the messages to tell the senders MAC address.

MAC Address Request

Type	Data	
A0:01	Queried Machine Number	Queried Interface Number

12th 13th 14th 15th byte

(a)

MAC Address Responses

Format 1

A0:02	Queried Machine Number	Queried Interface Number	Queried MAC Address

Format 2

A0:03	Queried Machine Number	Queried Interface number

(b)

Figure 2.1

PROCEDURE

Realize the Network Topology

1. Complete the network connections on HUBOX by referring to Figure 2.2.

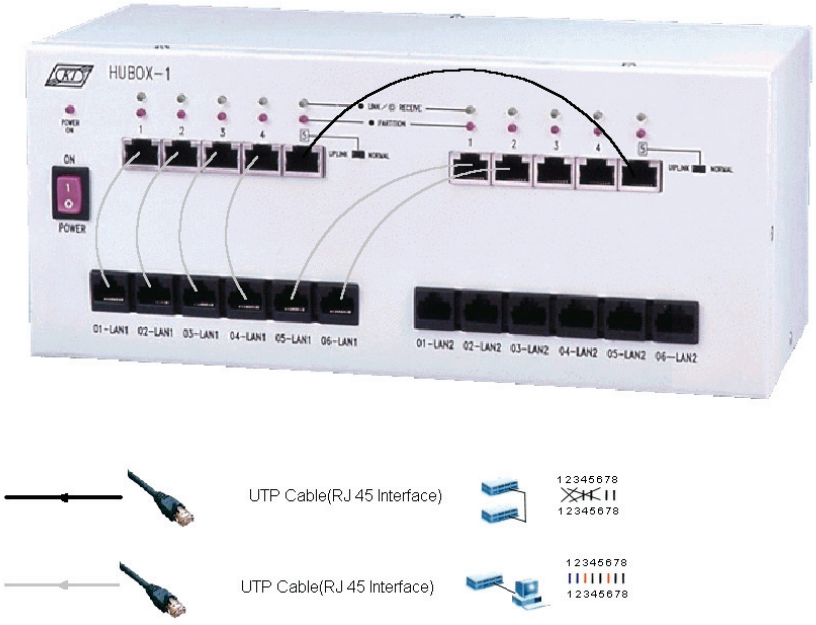


Figure 2.2

Manual Request and Response Frames

Leader and Members

2. Execute **XCLIENT.BAT** to open the KCodes Network Explorer for ITS window.

3. Open the Network Message Browser. Check **Listening On**.

Leader

4. Open the Network Message Sender. Type "**FF:FF:FF:FF:FF:FF**" into Destination MAC Address. Type "**A0:01**" into Type textbox.

5. Refer to Figure 2.1(a). Decide which Member and interface you want to request, then type the value into Data. For example, if you want to request the Interface1 of Member1, just type "**02:01**" into Data (see Figure 2.3) and press the **Send** button.

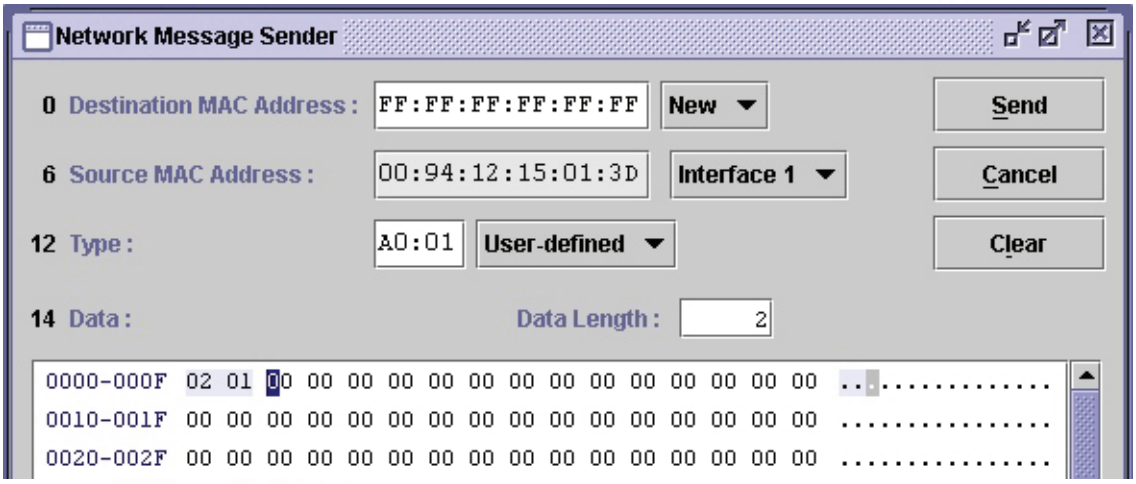


Figure 2.3

Members

6. Look up the detail when the frame is in. If the frame from Leader matches your ITS machine number, open Network Message Sender.

7. Type Leader's MAC address into Destination MAC Address and enter "**A0:02**" or "**A0:03**" into Type as you like. For example, when Member1 receives a request message from Leader, Member1 must type "**00:94:12:15:01:3D**" into Destination MAC Address and "**A0:02**" into Type as shown in Figure 2.4. Then refer to Figure 2.1(b), the protocol of Format 1, type "**02,01,00,94,12,15,01,3E**" into Data. Finally click the **Send**

button.

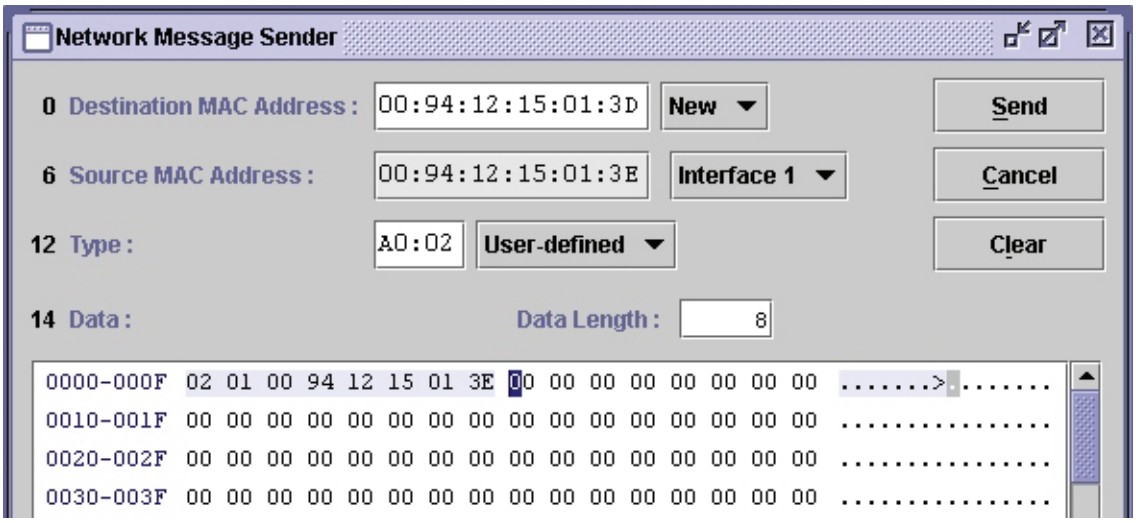


Figure 2.4

Response Frames by MDDL

Leader and Members

8. Open the Network Message Browser window. Check **Listening On**.

Leader

9. Open the MDDL Editor by selecting **MDDL Reactor Panel** from the Reactor menu.
10. Click the **Load** button. Open the file C: \XClient \Data \Mddl \Tutorial \Ex02 \BCastRequest.mddl and click the **UpId** button. Your ITS will continuously send broadcast Ethernet messages to request all Members at intervals of 5 seconds.

Members

11. Open the MDDL Editor by selecting **MDDL Reactor Panel** from the Reactor menu.
12. Click the **Load** button. As you like, open the AutoResponserFormat1.mddl or AutoResponserFormat2.mddl file (path: C: \XClient \Data \Mddl \Tutorial \Ex02).
13. Add <***your machine number***> into IF statement in the AutoResponserFormat1.mddl (see Figure 2.5) or the AutoResponserFormat2.mddl (see Figure 2.6). Then click the **UpId** button.

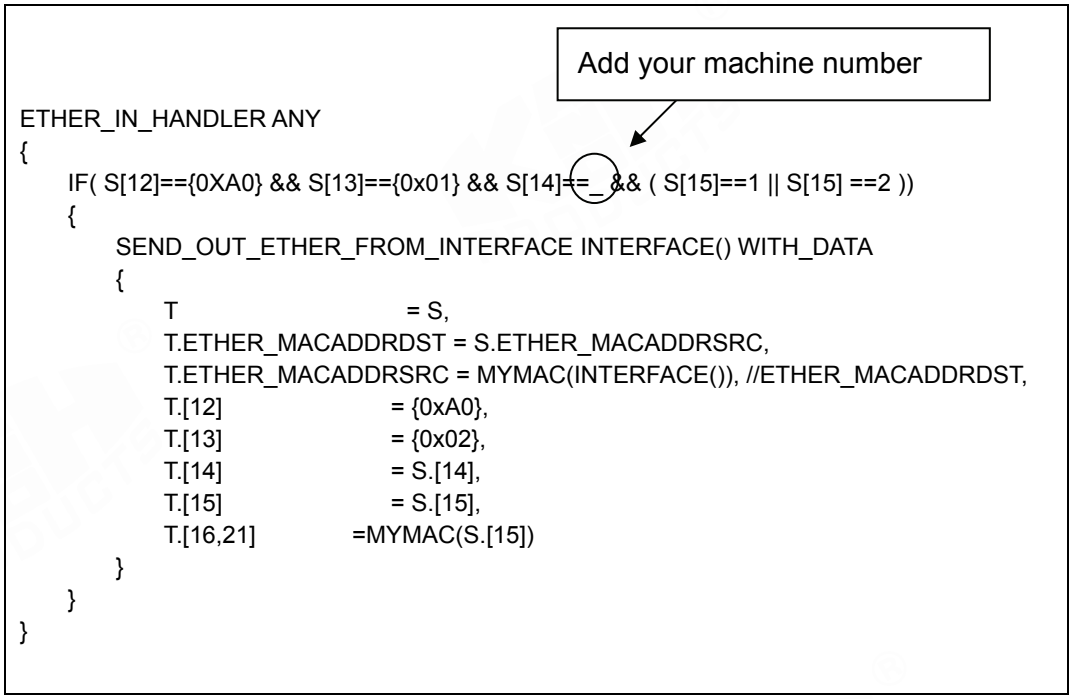


Figure 2.5

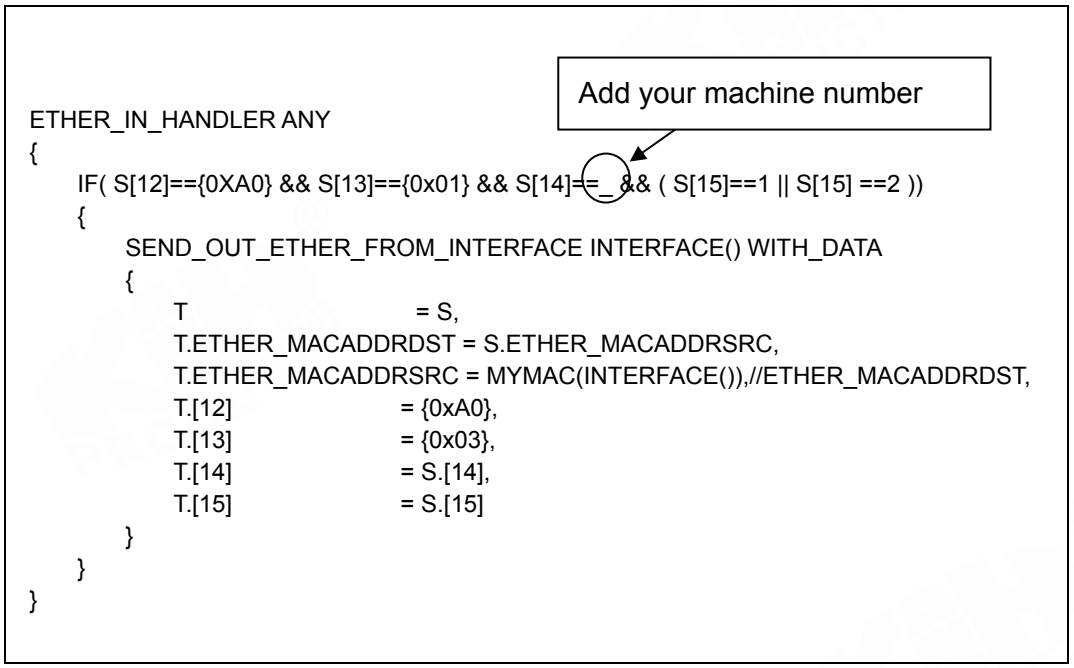


Figure 2.6

14. When Leader requests your MAC address of interface, your ITS will auto-respond with Format 1 or Format 2 depending on which program you chosen.

DISCUSSIONS

1. Discuss the difference between Format 1 and Format 2.

   (Hint: Try to request interface 2.)

2. Discuss with your partners and design a better protocol for Ethernet communication.

REACTOR PROGRAMS

1. **AutoResponserFormat1.mddl**

```
ETHER_IN_HANDLER ANY
{
    IF( S[12]=={0XA0} && S[13]=={0x01} && S[14]==01 && ( S[15]==1 || S[15] ==2 ))
    {
        SEND_OUT_ETHER_FROM_INTERFACE INTERFACE() WITH_DATA
        {
            T                       = S,
            T.ETHER_MACADDRDST   = S.ETHER_MACADDRSRC,
            T.ETHER_MACADDRSRC   = MYMAC(INTERFACE()),//ETHER_MACADDRDST,
            T.[12]                  = {0xA0},
            T.[13]                  = {0x02},
            T.[14]                  = S.[14],
            T.[15]                  = S.[15],
            T.[16,21]           =MYMAC(S.[15])
        }
    }
}
```

2. **AutoResponserFormat2.mddl**

```
ETHER_IN_HANDLER ANY
{
    IF( S[12]=={0XA0} && S[13]=={0x01} && S[14]==01 && ( S[15]==1 || S[15] ==2 ))
    {
        SEND_OUT_ETHER_FROM_INTERFACE INTERFACE() WITH_DATA
        {
            T                       = S,
            T.ETHER_MACADDRDST   = S.ETHER_MACADDRSRC,
            T.ETHER_MACADDRSRC   = MYMAC(INTERFACE()),//ETHER_MACADDRDST,
            T.[12]                  = {0xA0},
            T.[13]                  = {0x03},
            T.[14]                  = S.[14],
            T.[15]                  = S.[15]
        }
    }
}
```

3. **BCastRequest.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       = {0xA0,0x01}       ,
        T.[14, ]           = {0x02,0x01}
    }
}

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       = {0xA0,0x01}       ,
        T.[14, ]           = {0x03,0x01}
    }
}

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       = {0xA0,0x01}       ,
        T.[14, ]           = {0x04,0x01}
    }
}

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       = {0xA0,0x01}       ,
        T.[14, ]           = {0x05,0x01}
    }
}

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       = {0xA0,0x01}       ,
        T.[14, ]           = {0x06,0x01}
    }
}
```