Part 5: Firewalls

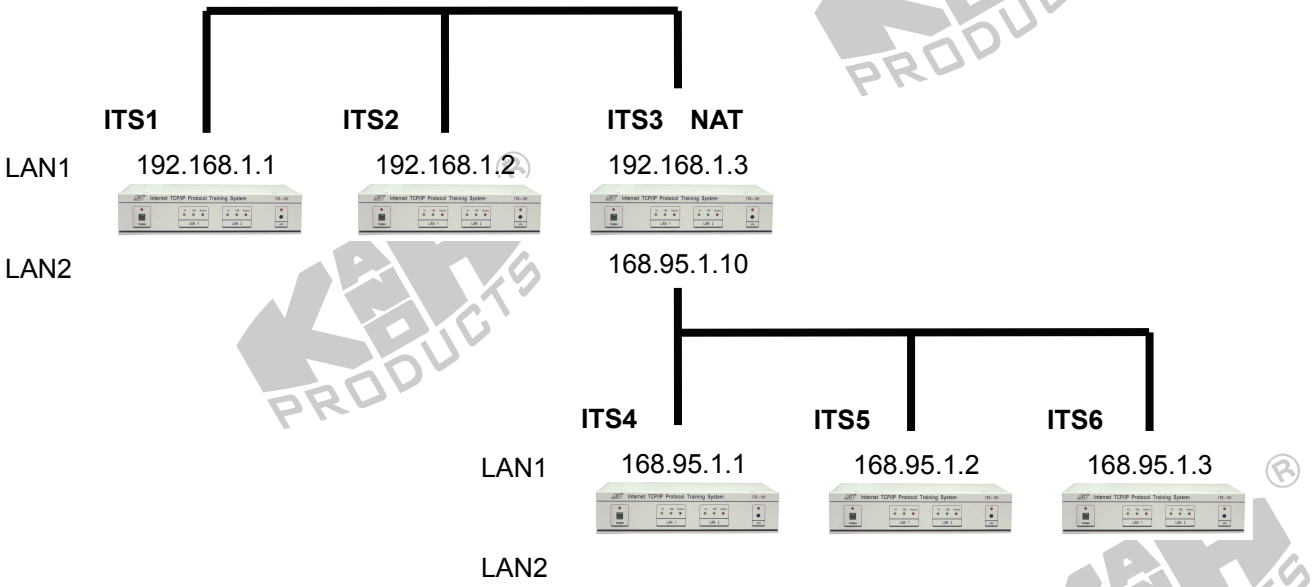
Exp 22. NAT

OBJECTIVE：To understand what NAT is and how to implement it.

BRIEF DESCRIPTION：This experiment examines the Network Address Translation (NAT) that is used to connect multiple computers to the Internet (or any other IP network) using one IP address. By using **MDDL**, students can learn how to implement the NAT mechanism.

DURATION：4.5 hrs

TOPOLOGY



TECHNICAL BACKGROUND

Network Address Translation (NAT) is a method of connecting multiple computers to the Internet (or any other IP network) using one IP address.

To multiplex several TCP connections to a single destination, client computers label all packets with unique "port numbers". Each IP packet starts with a header containing the source and destination addresses and port numbers:

Source address	Source port	Destination address	Destination port

When a packet is received from an internal client, NAT looks for the matching source address and port in the port mapping table. If the entry is not found, a new one is created, and a new mapping port allocated to the client:

a. Incoming packet received on non-NAT port.

b. Look for source address, port in the mapping table.

c. If found, replace source port with previously allocated mapping port.

d. If not found, allocate a new mapping port.

e. Replace source address with NAT address, source port with mapping port.

Packets received on the NAT port undergo a reverse translation process:

a. Incoming packet received on NAT port.

b. Look up destination port number in port mapping table.

c. If found, replace destination address and port with entries from the mapping table.

d. If not found, the packet is not for us and should be rejected.

Each client has an idle time-out associated with it. Whenever new traffic is received for a client, its time-out is reset. When the time-out expires, the client is removed from the table. This ensures that the table is kept to a reasonable size. The length of the time-out varies, but taking into account traffic variations on the Internet should not go below 2-3 minutes. Most NAT implementations can also track TCP clients on a per-connection basis and remove them from the table as soon as the connection is closed. This is not possible for UDP traffic since it is not connection based.

Many higher-level TCP/IP protocols embed client addressing information in the packets. For example, during an "active" FTP transfer the client informs the server of its IP address & port number, and then waits for the server to open a connection to that address. NAT has to monitor these packets and modify them on the fly to replace the client's IP address (which is on the internal network) with the NAT address. Since this changes the length of the packet, the TCP sequence/acknowledge numbers must be modified as well. Most protocols can be supported within the NAT; some protocols, however, may require that the clients themselves are made aware of the NAT and that they participate in the address translation process. (Or the NAT must be protocol-sensitive so that it can monitor or modify the embedded address or port data.)

PROCEDURE

Realizing Network Topology

1. Complete the network connections on HUBOX by referring to Figure 22.1.

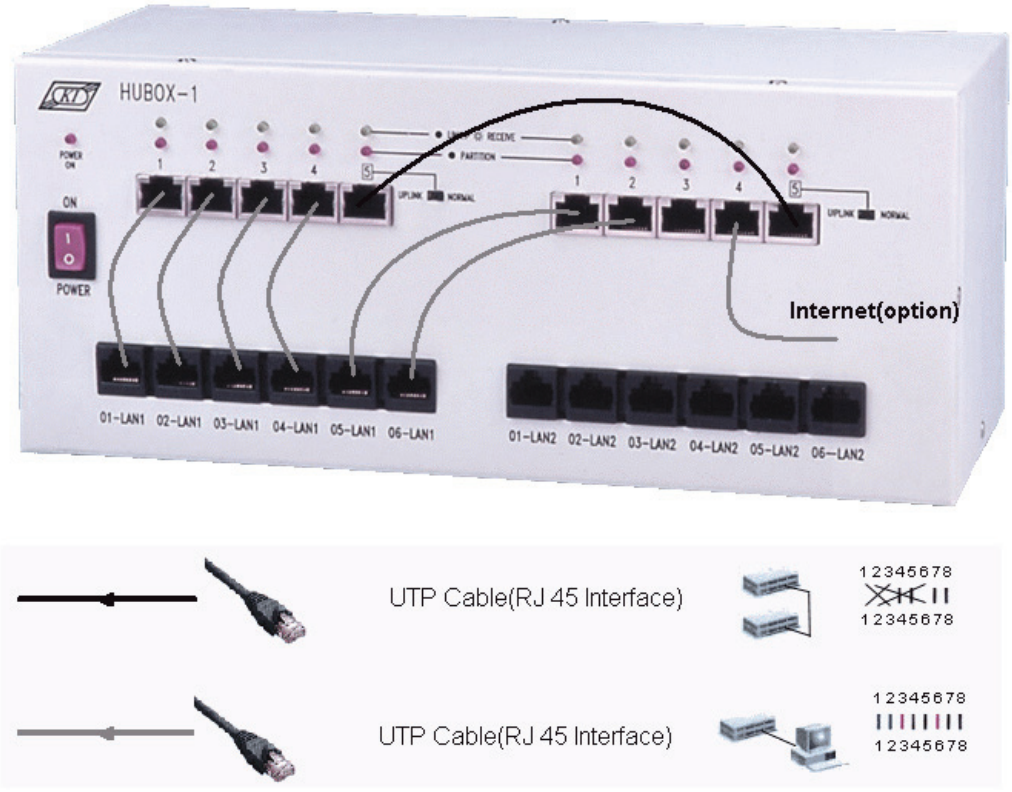


Figure 22.1

Transmission using NAT Table

A. Setup

2. Execute **XCLIENT.BAT** to open the KCodes Network Explorer for ITS window.
3. Select **Network Configuration** from the Tool menu. The Network Configuration dialog box opens as shown in Figure 22.2.

ITS1

4. Type "**192.168.1.1**" into IP Address of Interface 1. In the Routing Table, enter "**192.168.1.3**" into Gateway and "**0.0.0.0**" into Destination and Mask. Set **Host** and click the **Set & Close** button.

ITS2

5. Type "**192.168.1.2**" into IP Address of Interface 1. In the Routing Table, enter "**192.168.1.3**" into Gateway and "**0.0.0.0**" into Destination and Mask. Set **Host** and click the **Set & Close** button.

ITS3 (NAT)

6. Type "**192.168.1.3**" into IP Address of Interface 1 and enter "**168.95.1.10**" into IP Address of Interface 2. Set **Gateway** and click the **Set & Close** button.

ITS4

7. Type "**168.95.1.1**" into IP Address of Interface 1. In the Routing Table, enter "**168.95.1.10**" into Gateway and "**0.0.0.0**" into Destination and Mask. Set **Host** and click the **Set & Close** button.

ITS5

8. Type "**168.95.1.2**" into IP Address of Interface 1. In the Routing Table, enter "**168.95.1.10**" into Gateway and "**0.0.0.0**" into Destination and Mask. Set **Host** and click the **Set & Close** button.

ITS6

9. Type "**168.95.1.3**" into IP Address of Interface 1. In the Routing Table, enter "**168.95.1.10**" into Gateway and "**0.0.0.0**" into Destination and Mask. Set **Host** and click

the **Set & Close** button.

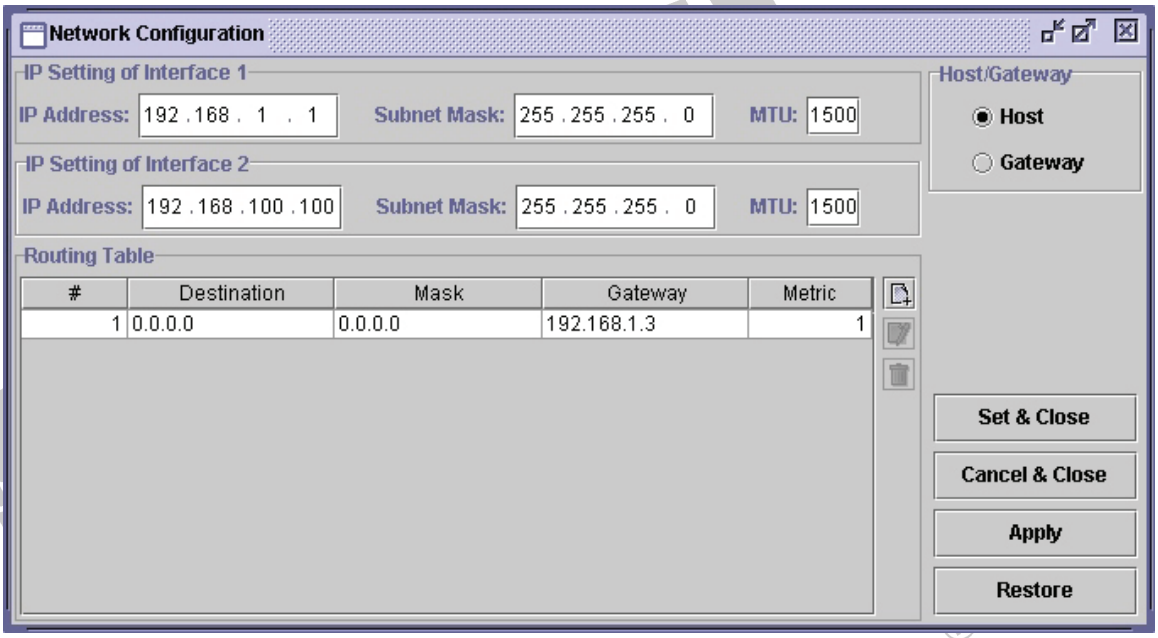


Figure 22.2

B. Transfer

ITS 3

10. Open Network Message Browser. Check **Listening On**.
9. Open MDDL Editor by selecting **MDDL Reactor Panel** from the Reactor menu.
10. Click the **Load** button. Open the file C: \XClient \Data \Mddl \Tutorial \Ex22 \NATSetup.mddl, and click the **UpId** button.

ITS 1, 2, 4, 5 and 6

11. Open the Network Message Browser window. Check **Listening On**.
12. Referring to the Exp4 (Figure 4.1), ITS1 sends ICMP Echo Request to ITS4. Figure 22.3 shows that ITS1 sends ICMP Echo Request to ITS4 and receives ICMP Echo Reply from ITS4.

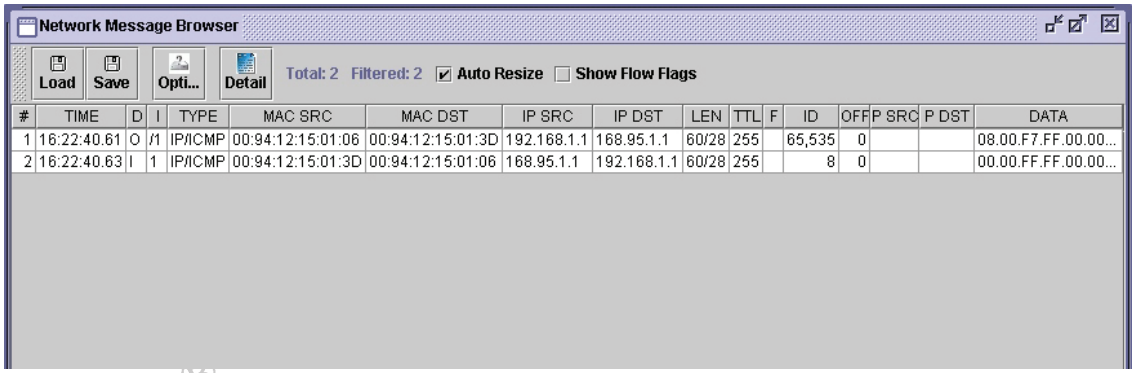


Figure 22.3

Figure 22.4 shows that ITS4 receives ICMP Echo Request from ITS3 then send ICMP Echo Reply to ITS3, so ITS4 doesn't know the IP address of ITS1.

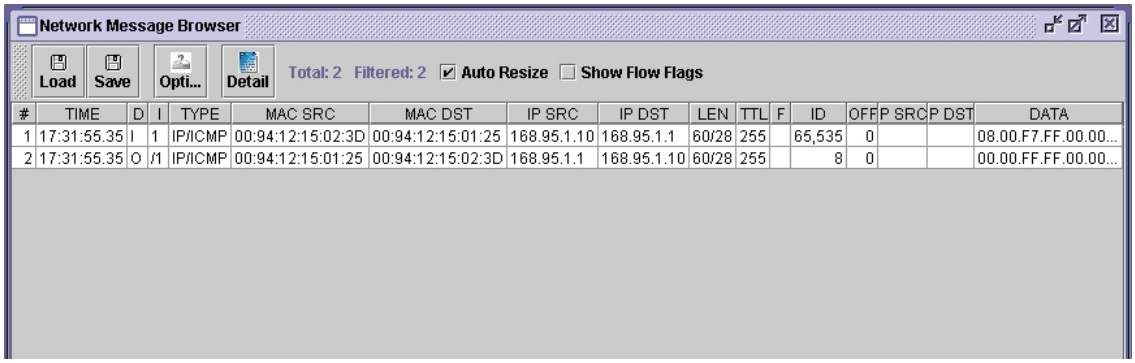


Figure 22.4

Figure 22.5 shows that ITS3 (NAT) changes the IP address of datagrams while delivering IP.

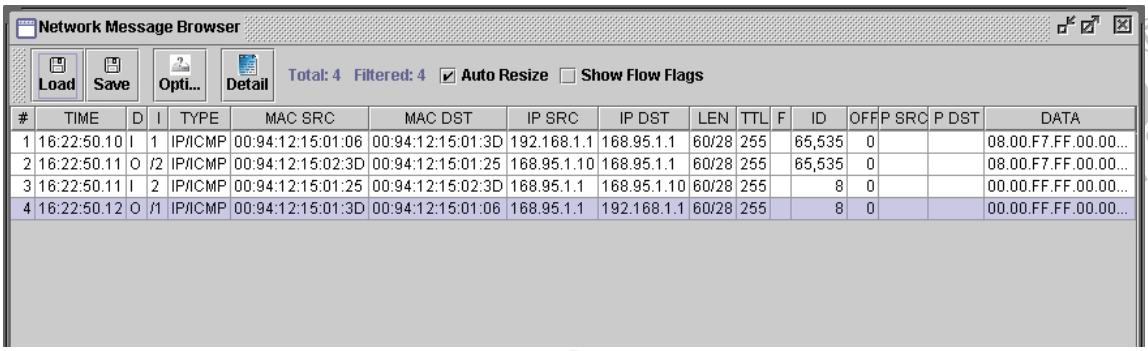


Figure 22.5

DISCUSSION

1. Try to send more datagrams from any ITS to other ITS, what is IP Masquerading, show it?

   Hint: “IP masquerading” is a form of network address translation (NAT) which allows internal computers with no known address outside their network, to communicate to the outside.

REACTOR PROGRAM

1. **NATSetup.mddl**

```
// NAT Setup
VAR1[0,1]              = 2048W       ; //initial port
VAR2[0,3]              = MYIP(2)     ;

IP_RECEIVED_HANDLER
{
    IF(!ISMYIPADDR(S.IP_ADDRDST)) //forward IP
 {
     LOOK_FOR_ONE_ELEMENT_IN_POOL 20 WITH_CONDITION
     (S.IP_PROT==PE[0]&&S.IP_ADDRSRC==PE[1,4]&&S.IP_ADDRDST==PE[5,8])
        {
            IF(S.IP_PROT==CNST_IP_PROT_UDP||S.IP_PROT==CNST_IP_PROT_TCP)
         {
             LOOK_FOR_ONE_ELEMENT_IN_POOL 20 WITH_CONDITION (S[20,21]==PE[9,10]
              &&S[22,23]==PE[11,12])
             {
                 S.IP_ADDRSRC                  = VAR2[0,3]                          ;
                  S.[20,21]                    = PE[13,14]                           ;
                 IF(S.IP_PROT==CNST_IP_PROT_UDP)
                  {
                     S[26,27]                  = 0W            ;     //disable udp checksum
                 }
                  ELSE
                  {
                      VAR3[0,3]                = VAR2[0,3]                          ;
                      VAR3[4,7]                = S.IP_ADDRDST                       ;
                      VAR3[8]                  = 0                                  ;
                      VAR3[9]                  = S.IP_PROT                          ;
                      VAR3[10,11]              = LENGTH(S)-20W                      ;
                      S[36,37]                 = {0, 0}                             ;
                      VAR3[12,]                = S[20,]                             ;
                      IF(VAR3[10,11]%2==1)
                      {
                          VAR3[VAR3[10,11]+12W]    = 0                      ;
                          S[36,37]         = CHECKSUM(VAR3[0,VAR3[10,11]+12W])   ;
                      }
                      ELSE
                      {
                          S[36,37]         = CHECKSUM(VAR3[0,VAR3[10,11]+11W])       ;
                      }
                  }
```

```
                SEND_OUT_IP WITH_DATA
                    {
                        T=S,
                        T.IP_LEN                    = LENGTH(T),
                        T.IP_HEADERCHKSUM           = {0, 0},
                        T.IP_HEADERCHKSUM           = CHECKSUM(T[0,19])
                    }
            }
        }
        ELSE
        {
            S.IP_ADDRSRC = VAR2[0,3];
            SEND_OUT_IP WITH_DATA
                {
                    T=S,
                    T.IP_LEN                    = LENGTH(T),
                    T.IP_HEADERCHKSUM           = {0, 0},
                    T.IP_HEADERCHKSUM           = CHECKSUM(T[0,19])
                }
        }

    }
ELSE                                                                    //add entry
{
    IF(S.IP_PROT==CNST_IP_PROT_UDP||S.IP_PROT==CNST_IP_PROT_TCP)
    {
        VAR1[0,1]                               = VAR1[0,1]+1                   ;
        ADD_TO_POOL 20 WITH_LIFETIME 20000 WITH_DATA
            {
                T[0]                            = S.IP_PROT                     ,
                T[1,4]                          = S.IP_ADDRSRC                  ,
                T[5,8]                          = S.IP_ADDRDST                  ,
                T[9,10]                         = S.[20,21]                     ,
                T[11,12]                        = S.[22,23]                     ,
                T[13,14]                        = VAR1[0,1]
            }
            S.IP_ADDRSRC                        = VAR2[0,3]                     ;
            S[20,21]                            = VAR1[0,1]         ;   //PE[13,14];
            IF(S.IP_PROT==CNST_IP_PROT_UDP)
            {
                S[26,27]                        = 0W                            ;
            }
            ELSE
            {
                VAR3[0,3]                       = VAR2[0,3]                     ;
                VAR3[4,7]                       = S.IP_ADDRDST                  ;
                VAR3[8]                         = 0                             ;
                VAR3[9]                         = S.IP_PROT                     ;
                VAR3[10,11]                     = LENGTH(S)-20W                 ;
                S[36,37]                        = {0, 0}                        ;
                VAR3[12,]                       = S[20,]                        ;
                IF(VAR3[10,11]%2==1)
                {
                    VAR3[VAR3[10,11]+12W]       = 0                             ;
                    S[36,37]                    = CHECKSUM(VAR3[0,VAR3[10,11]+12W]) ;
                }
                ELSE
                {
                    S[36,37]                    = CHECKSUM(VAR3[0,VAR3[10,11]+11W]) ;
                }
```

```
            }

            SEND_OUT_IP WITH_DATA
            {
                T                               = S                     ,
                T.IP_LEN                        = LENGTH(T)             ,
                T.IP_HEADERCHKSUM               = {0, 0}                ,
                T.IP_HEADERCHKSUM               = CHECKSUM(T[0,19])
            }
        }
        ELSE
        {
            ADD_TO_POOL 20 WITH_LIFETIME 20000 WITH_DATA
            {
                T[0]=S.IP_PROT,
                T[1,4]=S.IP_ADDRSRC,
                T[5,8]=S.IP_ADDRDST
            }
            SEND_OUT_IP WITH_DATA
            {
                T                               = S                     ,
                T.IP_ADDRSRC                    = VAR2[0,3]             ,
                T.IP_LEN                        = LENGTH(T)             ,
                T.IP_HEADERCHKSUM               = {0, 0}                ,
                T.IP_HEADERCHKSUM               = CHECKSUM(T[0,19])
            }
        }
    }
}
ELSE                                                        //  IP arrivaling
{
    LOOK_FOR_ONE_ELEMENT_IN_POOL 20 WITH_CONDITION
    (S.IP_PROT==PE[0]&&S.IP_ADDRSRC==PE[5,8])
    {
        IF(S.IP_PROT==CNST_IP_PROT_UDP||S.IP_PROT==CNST_IP_PROT_TCP)
     {
          LOOK_FOR_ONE_ELEMENT_IN_POOL 20 WITH_CONDITION
         (S[20,21]==PE[11,12]&&S[22,23]==PE[13,14])
         {
            S.IP_ADDRDST                = PE[1,4]                       ;
            S[22, 23]                   = PE[9,10]                      ;
           IF(S.IP_PROT==CNST_IP_PROT_UDP)
            {
                S[26,27]        = 0W            ;   //disable udp checksum
            }
            ELSE
            {
                VAR3[0,3]       = S.IP_ADDRSRC          ;   //VAR2[0,3];
                VAR3[4,7]       = PE[1,4]               ;   //S.IP_ADDRDST;
                VAR3[8]         = 0                             ;
                VAR3[9]         = S.IP_PROT                     ;
                VAR3[10,11]     = LENGTH(S)-20W                 ;
                S[36,37]        = {0, 0}                        ;
                VAR3[12,]       = S[20,]                        ;
                IF(VAR3[10,11]%2==1)
                {
                    VAR3[VAR3[10,11]+12W]    = 0                        ;
                    S[36,37] = CHECKSUM(VAR3[0,VAR3[10,11]+12W])        ;
                }
                ELSE
```

```
                    {
                        S[36,37] = CHECKSUM(VAR3[0,VAR3[10,11]+11W])        ;
                    }

                }

                SEND_OUT_IP WITH_DATA
                {
                    T                           = S                         ,
                    T.IP_LEN                    = LENGTH(T)                 ,
                    T.IP_HEADERCHKSUM           = {0, 0}                    ,
                    T.IP_HEADERCHKSUM           = CHECKSUM(T[0,19])
                }
            }
        }
        ELSE
        {
            SEND_OUT_IP WITH_DATA
            {
                    T                           = S                         ,
                    T.IP_ADDRDST                = PE[1,4]                   ,
                    T.IP_LEN                    = LENGTH(T)                 ,
                    T.IP_HEADERCHKSUM           = {0, 0}                    ,
                    T.IP_HEADERCHKSUM           = CHECKSUM(T[0,19]),
            }
        }
       }
    }
    DISCARD_MESSAGE;
}
```

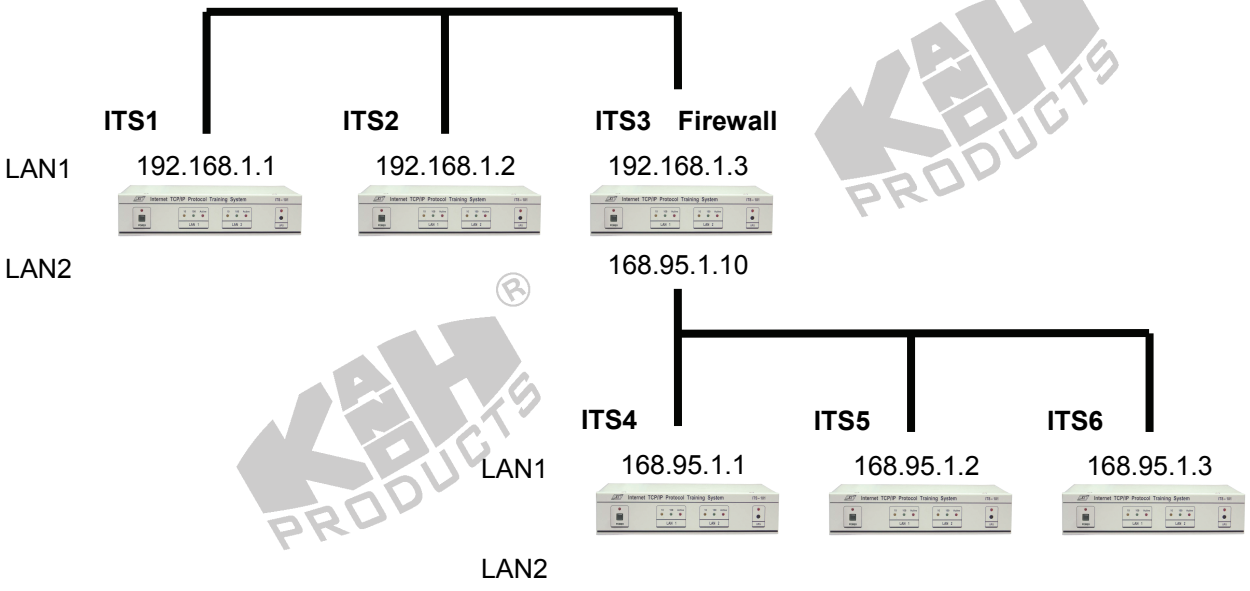
Exp 23. Firewall

OBJECTIVE：To understand what the firewall is and how to implement it.

BRIEF DESCRIPTION：This experiment examines the firewall that is used to protect the resources of a private network from users from other networks. By using **MDDL**, students can learn how to implement the firewall mechanism.

DURATION：4.5 hrs

TOPOLOGY



TECHNICAL BACKGROUND

A firewall is a set of related programs, located at a network gateway that protects the resources of a private network from users from other networks.

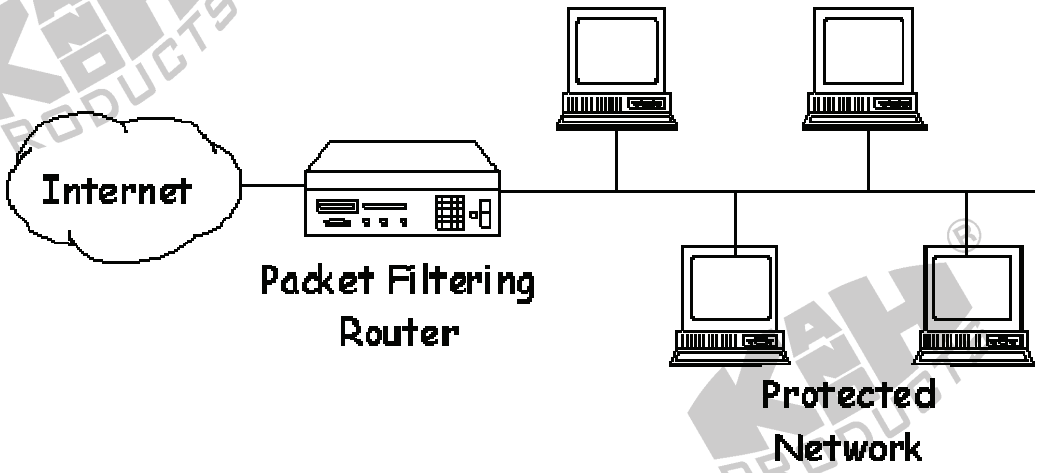
a. A firewall is a program or device that used to control access to a computer or network, much like a security guard protects a business.

b. Firewalls monitor requests entering and leaving restricted systems.

c. A firewall sits between a restricted system and the internet, so all traffic in or out must flow through it.

d. Firewalls monitor the data and requests (called "packets") passing through them, by looking at certain information contained in the packets.

e. If a firewall receives a packet that does not meet the conditions set for passage by the firewall's user, it is denied access and thrown away.



Packet filtering firewall is a common firewall technology which checks versatile protocol header information of packet to decide: reject the packet or accept it. Generally, main firewall rules are to consider IP address (source and destination), IP protocol type, TCP or UDP port numbers (source and destination) and hardware address. For example, to allow SMTP e-mail, the firewall would be configured to allow any IP address to send packet to TCP port 25 on the mail server. To block Telnet access, the firewall could be configured to refuse all traffic to TCP port 23 (generally, ports below 1024 are considered as privileged, and are blocked unless specifically allowed). Rules are searched in order, so only the first match counts.

PROCEDURE

Realizing Network Topology

1. Complete the network connections on HUBOX by referring to Figure 23.1.

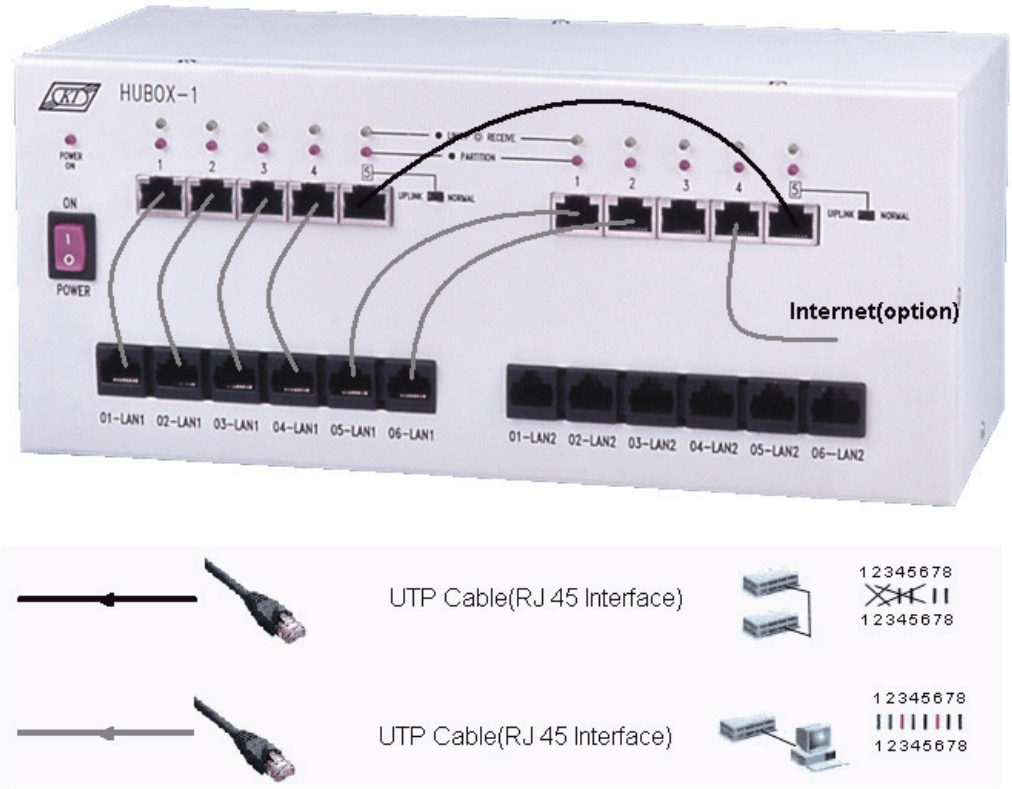


Figure 23.1

Firewall Rules of IP Address Filter

A. Setup

2. Execute **XCLIENT.BAT** to open the KCodes Network Explorer for ITS window.
3. Select **Network Configuration** from the Tool menu to open Network Configuration as shown in Figure 23.2.

ITS1

4. Type "**192.168.1.1**" into IP Address of Interface 1. In the Routing Table, enter "**192.168.1.3**" into Gateway and "**0.0.0.0**" into Destination and Mask. Set **Host**, and click the **Set & Close** button.

ITS2

5. Type “**192.168.1.2**” into IP Address of Interface 1. In the Routing Table, enter “**192.168.1.3**” into Gateway and “**0.0.0.0**” into Destination and Mask. Set **Host**, and click the **Set & Close** button.

ITS3 (Firewall)

6. Type “**192.168.1.3**” into IP Address of Interface 1 and enter “**168.95.1.10**” into IP Address of Interface 2. Set **Gateway**, and click the **Set & Close** button.

ITS4

7. Type “**168.95.1.1**” into IP Address of Interface 1. In the Routing Table, enter “**168.95.1.10**” into Gateway and “**0.0.0.0**” into Destination and Mask. Set **Host**, and click the **Set & Close** button.

ITS5

8. Type ”**168.95.1.2**” into IP Address of Interface 1. In the Routing Table, enter “**168.95.1.10**” into Gateway and “**0.0.0.0**” into Destination and Mask. Set **Host**, and click the **Set & Close** button.

ITS6

9. Type “**168.95.1.3**” into IP Address of Interface 1. In the Routing Table, enter “**168.95.1.10**” into Gateway and “**0.0.0.0**” into Destination and Mask. Set **Host**, and click the **Set & Close** button.

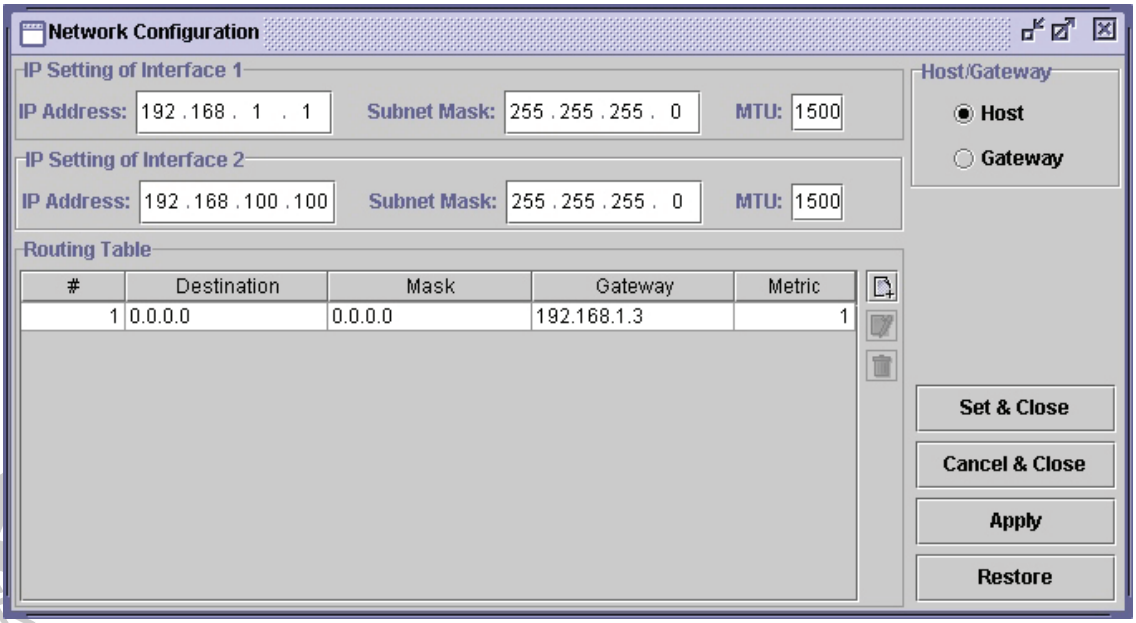


Figure 23.2

B. Reject by Firewall

ITS 3

10. Open Network Message Browser. Check **Listening On**.
11. Open the MDDL Editor by selecting **MDDL Reactor Panel** from the Reactor menu.
12. Click the **Load** button. Open the file C: \XClient \Data \Mddl \Tutorial \Ex23 \Firewall.mddl, and click the **UpId** button.

ITS 1, 2, 4, 5 and 6

13. Open the Network Message Browser window. Check **Listening On**.
14. Referring to the previous experiments, ITS4 opens a TCP active connection by setting source port to 21 and click **Listen**. ITS1 opens a TCP active connection by setting destination port to 21 and setting destination IP address as the IP of ITS4 (see Figure 23.3). Then click the **Connect** button. We should see that ITS3 (Firewall) rejects this IP request as shown in Figure 23.4.

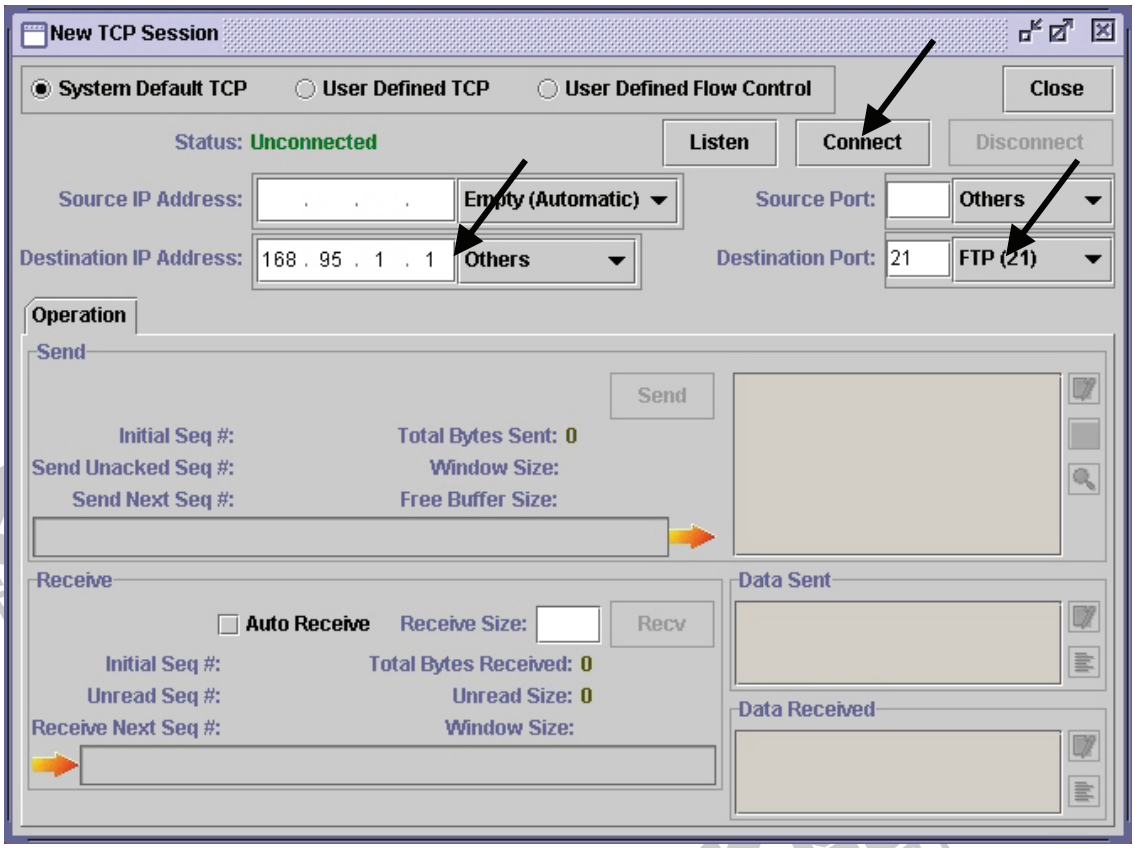


Figure 23.3

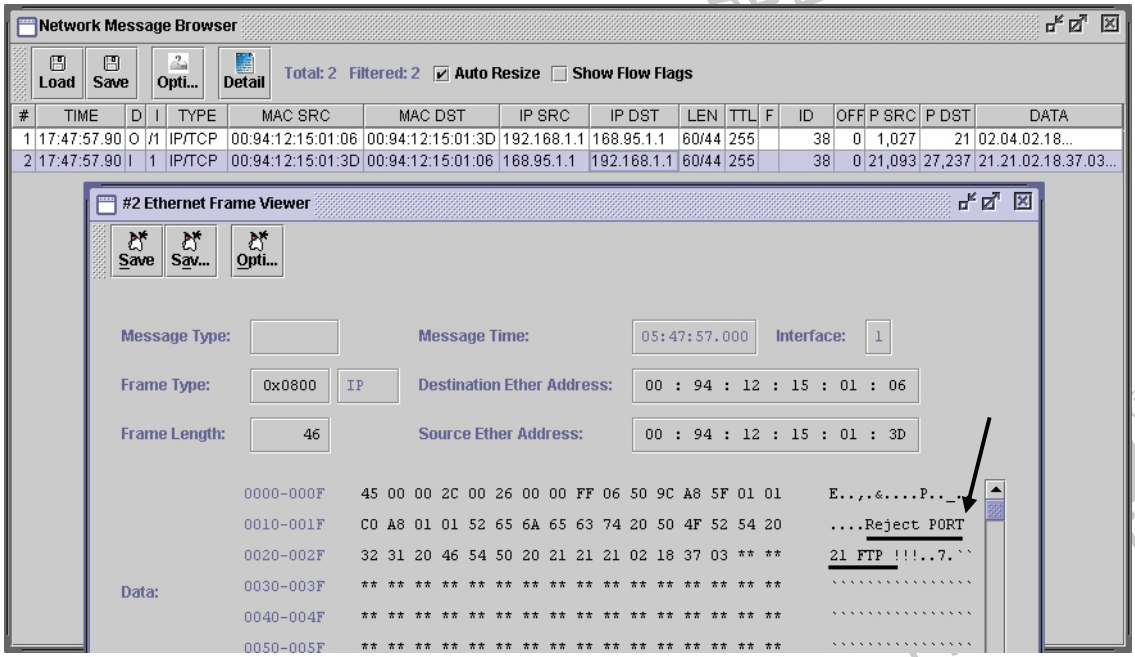


Figure 23.4

15. Referring to the previous experiments, ITS4 sends ICMP Echo Request to ITS1. We will see that ITS3 (Firewall) rejects this IP request as shown in Figure 23.5.

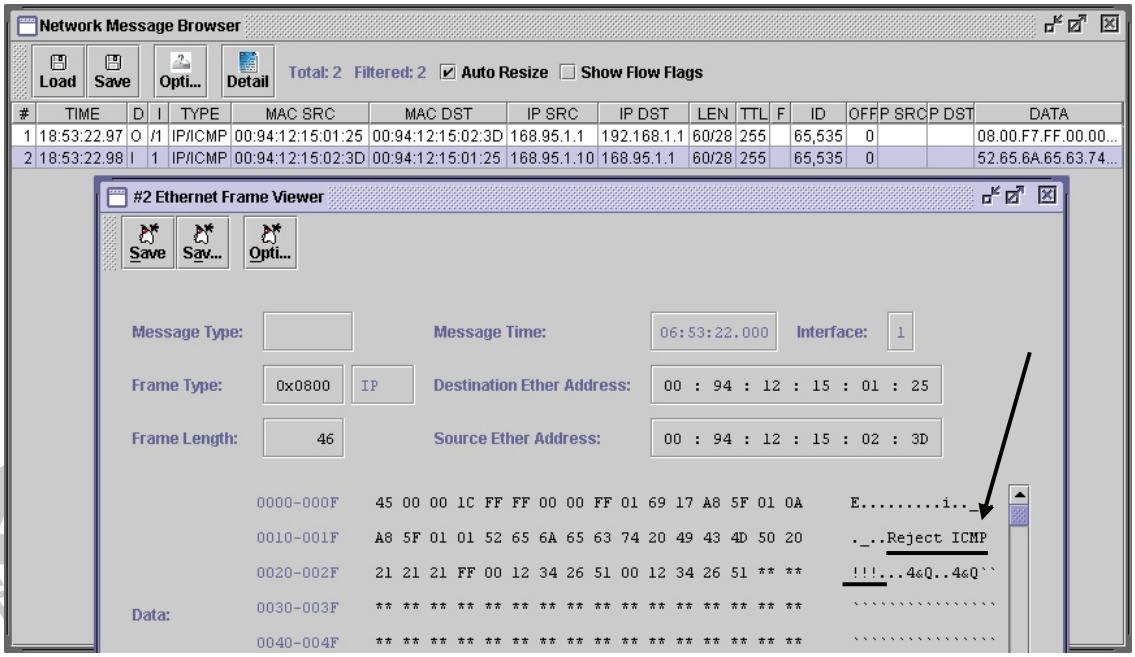


Figure 23.5

DISCUSSION

1. What is IP spoofing? Can you design a firewall to solve it? Explain it.

   Hint: “Spoofing” is the creation of TCP/IP packets using somebody else's IP address. Routers use the "destination IP" address in order to forward packets through the Internet, but ignore the "source IP" address. That address is only used by the destination machine when it responds back to the source.

REACTOR PROGRAM

1. **Firewall.mddl**

```
// Firewall
// Reject Port 80/21 from interface 1
// Reject ICMP from interface 2

ETHER_IN_HANDLER 1
{
    IF(S.ETHER_TYPE==CNST_ETHER_TYPE_IP)
    {
            IF(S.ETHER_DATA.[22,23]== 80W)   // REJECT   PORT 80     HTTP
          {
              DISCARD_MESSAGE;
              SEND_OUT_ETHER_FROM_INTERFACE 1 WITH_DATA
      {
      T = S ,
```

```
        T.ETHER_MACADDRDST         = S.ETHER_MACADDRSRC          ,
        T.ETHER_MACADDRSRC         = MYMAC(1)            ,
        T.ETHER_DATA.IP_ADDRSRC = S.ETHER_DATA.IP_ADDRDST ,
        T.ETHER_DATA.IP_ADDRDST =    S.ETHER_DATA.IP_ADDRSRC ,
        T.ETHER_DATA.IP_DATA       = "Reject PORT 80 HTTP!!!"       ,
        T.ETHER_DATA.IP_HEADERCHKSUM = 0W                                    ,
        T.ETHER_DATA.IP_HEADERCHKSUM = CHECKSUM(T.ETHER_DATA.IP_HEADER)
        }
              }
              ELSE IF(S.ETHER_DATA.[22,23]== 21W)   // REJECT    PORT 21    FTP
              {
                   DISCARD_MESSAGE;
                   SEND_OUT_ETHER_FROM_INTERFACE 1 WITH_DATA
        {
        T = S ,
         T.ETHER_MACADDRDST         = S.ETHER_MACADDRSRC          ,
        T.ETHER_MACADDRSRC         = MYMAC(1)            ,
         T.ETHER_DATA.IP_ADDRSRC = S.ETHER_DATA.IP_ADDRDST ,
         T.ETHER_DATA.IP_ADDRDST =    S.ETHER_DATA.IP_ADDRSRC ,
        T.ETHER_DATA.IP_DATA       = "Reject PORT 21 FTP !!!"       ,
        T.ETHER_DATA.IP_HEADERCHKSUM = 0W                                    ,
        T.ETHER_DATA.IP_HEADERCHKSUM = CHECKSUM(T.ETHER_DATA.IP_HEADER)
        }
              }
    }
}

ETHER_IN_HANDLER 2
{
    IF(S.ETHER_TYPE==CNST_ETHER_TYPE_IP)
    {
                  IF(S.ETHER_DATA.IP_PROT==CNST_IP_PROT_ICMP)
                  {
                      DISCARD_MESSAGE;
                      SEND_OUT_ETHER_FROM_INTERFACE 2 WITH_DATA
       {
       T = S ,
        T.ETHER_MACADDRDST         = S.ETHER_MACADDRSRC          ,
       T.ETHER_MACADDRSRC         = MYMAC(1)            ,
        T.ETHER_DATA.IP_ADDRSRC = MYIP(2) ,
        T.ETHER_DATA.IP_ADDRDST =    S.ETHER_DATA.IP_ADDRSRC ,
        T.ETHER_DATA.IP_DATA       = "Reject ICMP !!!"       ,
        T.ETHER_DATA.IP_HEADERCHKSUM = 0W                          ,
        T.ETHER_DATA.IP_HEADERCHKSUM = CHECKSUM(T.ETHER_DATA.IP_HEADER)
       }
                  }
    }
}
```

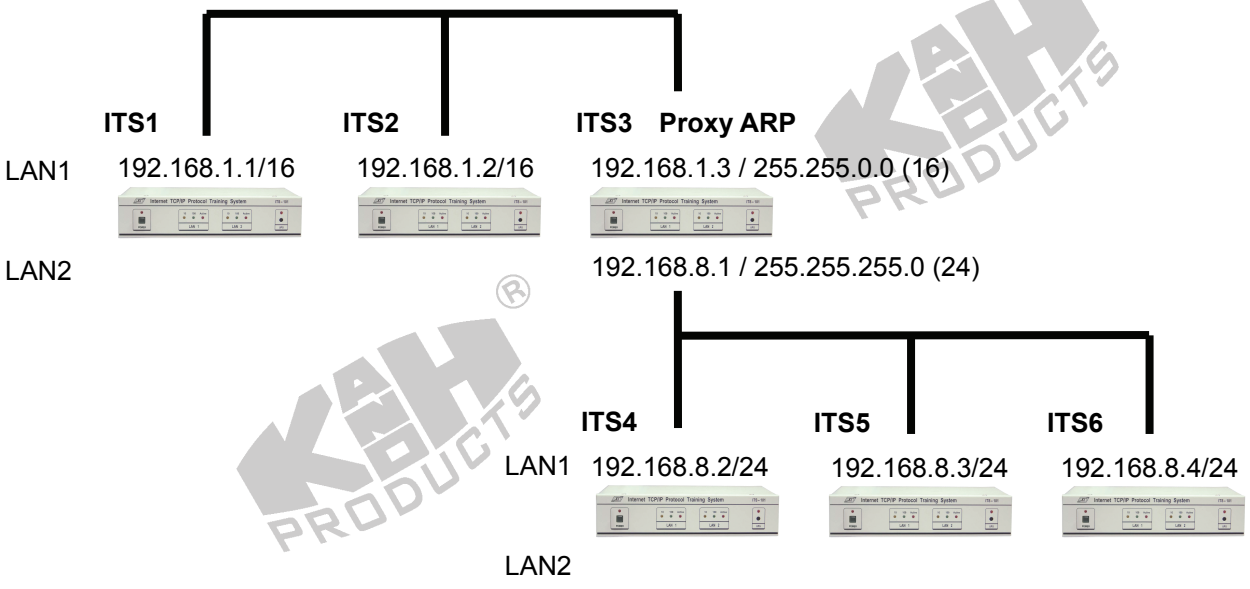
Exp 24. Proxy ARP

OBJECTIVE： To understand what the Proxy ARP is and how to implement it.

BRIEF DESCRIPTION： This experiment examines the Proxy ARP that is used to respond to ARP requests for hosts other than itself. By using **MDDL**, students can learn how to implement the Proxy ARP mechanism.

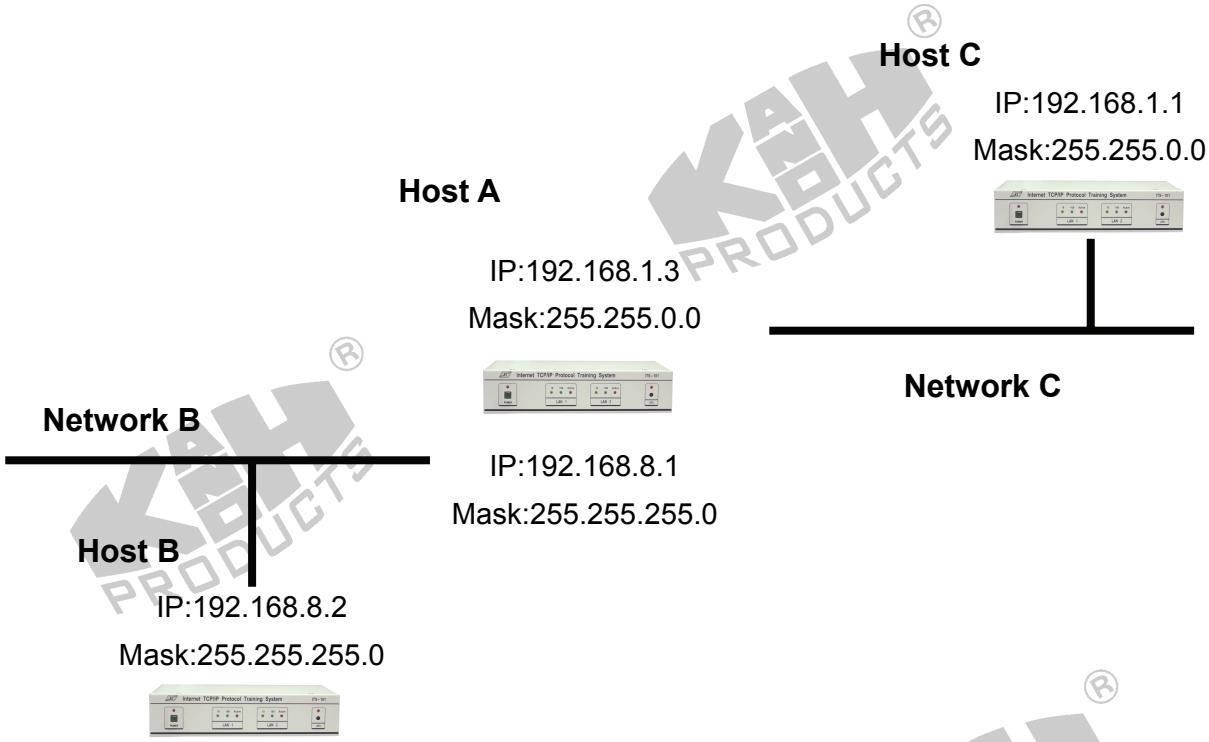
DURATION: 4.5 hrs

TOPOLOGY



TECHNICAL BACKGROUND

Proxy ARP basically means that a particular machine (such as a firewall) will respond to ARP requests for hosts other than itself.



Host C (network 1) is sending an ICMP Echo Request packet to host B (network 2). Since host B's IP is located in network C, the host C sends an ARP Request to network C to query MAC address of host B. However host B and host C are not belonged to the same physical network, host B will not respond this request message. On the other hand, since the host A can do proxy ARP job for host B, the host A will respond the request message by MAC address of host A's interface 2. Host C then updates it's ARP cache with an entry for host B with MAC address of host A's interface 2. Now host C can now send the IP packet to host B, and the host A receives it and then forward the packet to host B via interface 1. Host B gets the packet and then sends an ICMP Echo Response to host C. Host B knows host C is located in a different subnet, and hence, host B sends the packet via default gateway host A to host C.

PROCEDURE

Realizing Network Topology

1. Complete the network connections on HUBOX by referring to Figure 24.1.

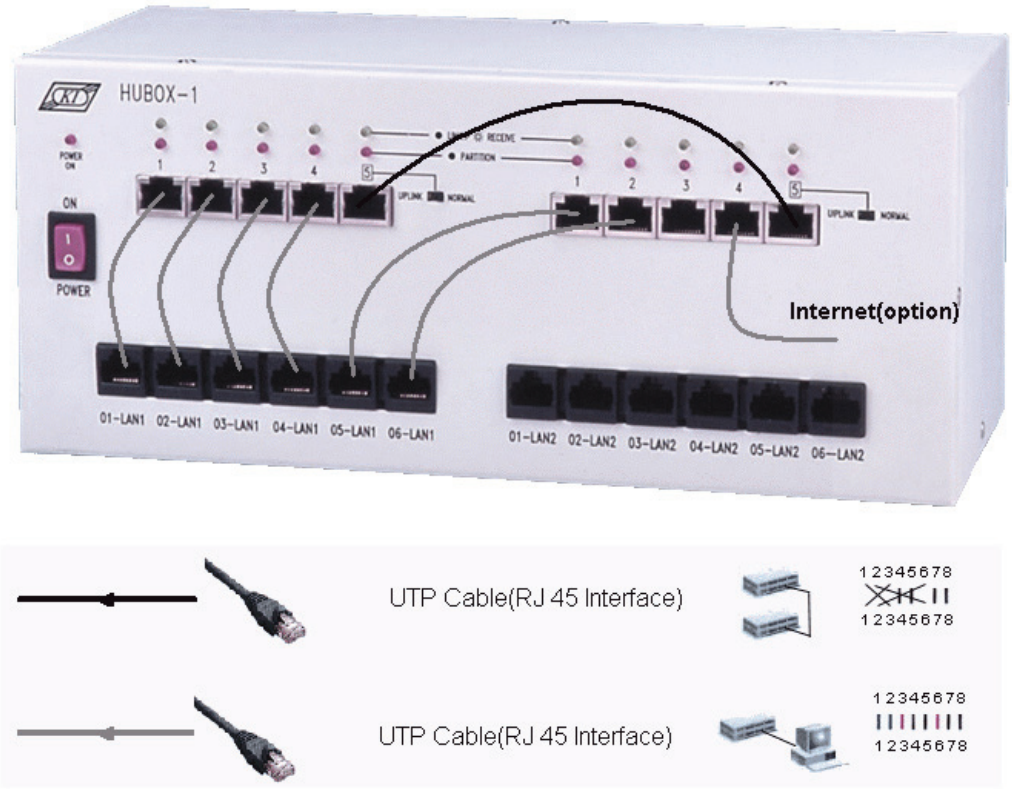


Figure 24.1

Proxy ARP Apply

A. Setup

2. Execute **XCLIENT.BAT** to open the KCodes Network Explorer for ITS window.
3. Select **Network Configuration** from the Tool menu to open the Network Configuration dialog box.

ITS1

4. Type "**192.168.1.1**" into IP Address of Interface 1 and enter "**255.255.0.0**" into Subnet Mask of Interface 1. In the Routing Table, enter "**192.168.1.3**" into Gateway and "**0.0.0.0**" into Destination and Mask. Once completed, choose **Host** and click the **Set & Close** button. (See Figure 24.2.)

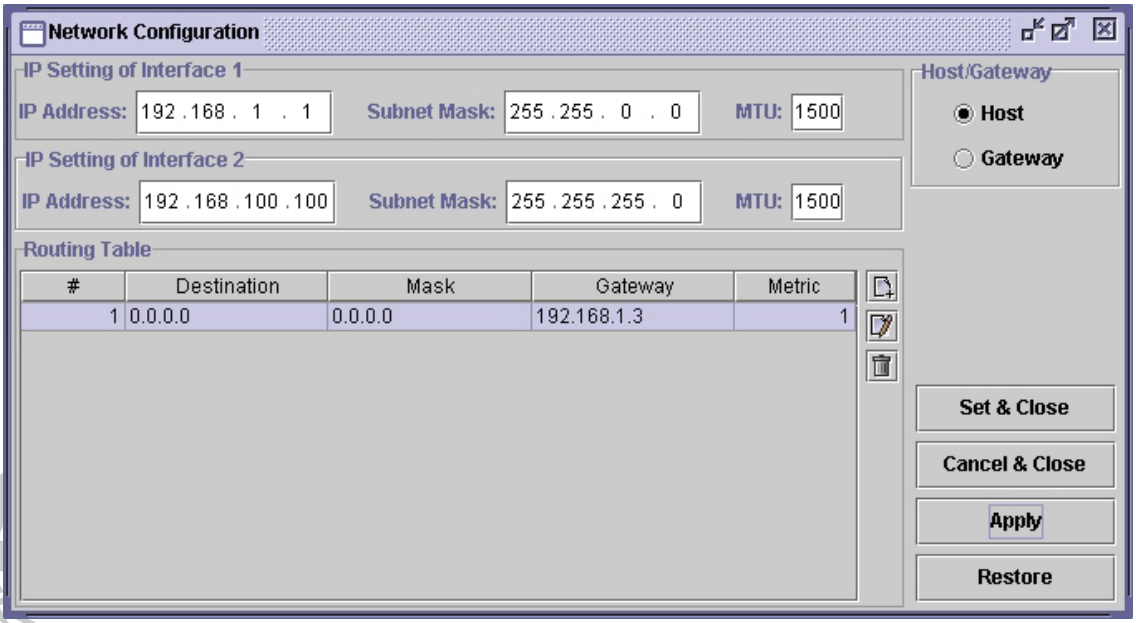


Figure 24.2

ITS2

5. Type "**192.168.1.2**" into IP Address of Interface 1 and enter "**255.255.0.0**" into Subnet Mask of Interface 1. In the Routing Table, enter "**192.168.1.3**" into Gateway and "**0.0.0.0**" into Destination and Mask. Set **Host** and click the **Set & Close** button.

ITS3 (Proxy ARP)

6. Type "**192.168.1.3**" into IP Address of Interface 1 and enter "**255.255.0.0**" into Subnet Mask of Interface 1. Type "**192.168.8.1**" into IP Address of Interface 2 and enter "**255.255.255.0**" into Subnet Mask of Interface 2. Set **Gateway** and click the **Set & Close** button.

ITS4

7. Type "**192.168.8.2**" into IP Address of Interface 1 and enter "**255.255.255.0**" into Subnet Mask of Interface 1. In the Routing Table, enter "**192.168.8.1**" into Gateway and "**0.0.0.0**" into Destination and Mask. Set **Host** and click the **Set & Close** button.

ITS5

8. Type "**192.168.8.3**" into IP Address of Interface 1 and enter "**255.255.255.0**" into Subnet Mask of Interface 1. In the Routing Table, enter "**192.168.8.1**" into Gateway and "**0.0.0.0**" into Destination and Mask. Set **Host** and click the **Set & Close** button.

ITS6

9. Type “**192.168.8.4**” into IP Address of Interface 1 and enter “**255.255.255.0**” into Subnet Mask of Interface 1. In the Routing Table, enter “**192.168.8.1**” into Gateway and “**0.0.0.0**” into Destination and Mask. Set **Host** and click the **Set & Close** button.

   (Because we set the different subnet masks, ITS1 and ITS4 seem to be in the same subnet, but actually they are in the different subnet.)

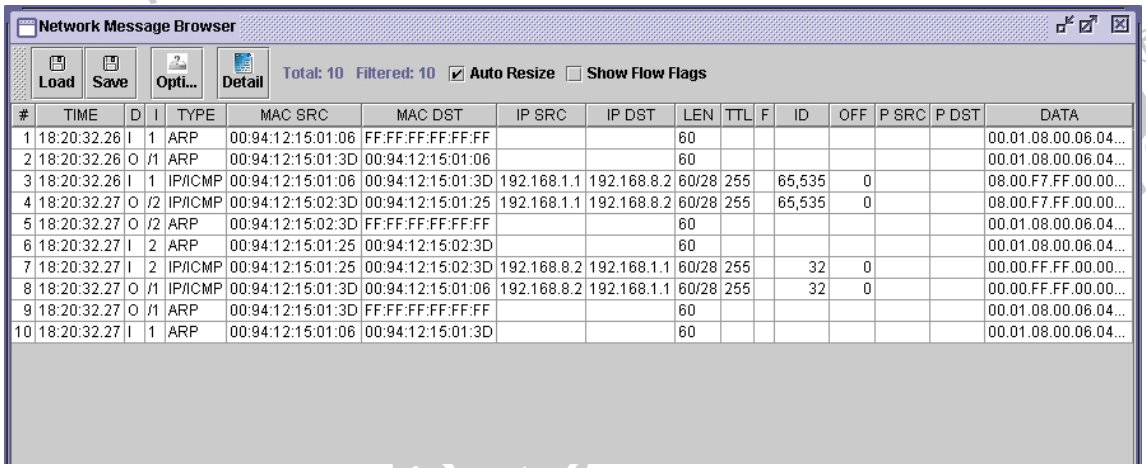
B. Proxy ARP

ITS 3

10. Open the Network Message Browser window. Check **Listening On**.
11. Open the MDDL Editor by selecting **MDDL Reactor Panel** from the Reactor menu.
12. Click the **Load** button. Open the file C: \XClient \Data \Mddl \Tutorial \Ex24 \ProxyArp.mddl, and click the **UpId** button.

ITS 1, 2, 4, 5 and 6

13. Open the Network Message Browser window. Check **Listening On**.
14. Referring to the Exp4 (Figure 4.1), ITS1 sends ICMP Echo Request to ITS4. We will see that ITS3 (Proxy ARP) receives ITS1’s ARP then delivers ICMP to ITS4, as shown in Figure 24.3.



#	TIME	D	I	TYPE	MAC SRC	MAC DST	IP SRC	IP DST	LEN	TTL	F	ID	OFF	P SRC	P DST	DATA
1	18:20:32.26	I	1	ARP	00:94:12:15:01:06	FF:FF:FF:FF:FF:FF			60							00.01.08.00.06.04...
2	18:20:32.26	O	/1	ARP	00:94:12:15:01:3D	00:94:12:15:01:06			60							00.01.08.00.06.04...
3	18:20:32.26	I	1	IP/ICMP	00:94:12:15:01:06	00:94:12:15:01:3D	192.168.1.1	192.168.8.2	60/28	255		65,535	0			08.00.F7.FF.00.00...
4	18:20:32.27	O	/2	IP/ICMP	00:94:12:15:02:3D	00:94:12:15:01:25	192.168.1.1	192.168.8.2	60/28	255		65,535	0			08.00.F7.FF.00.00...
5	18:20:32.27	O	/2	ARP	00:94:12:15:02:3D	FF:FF:FF:FF:FF:FF			60							00.01.08.00.06.04...
6	18:20:32.27	I	2	ARP	00:94:12:15:01:25	00:94:12:15:02:3D			60							00.01.08.00.06.04...
7	18:20:32.27	I	2	IP/ICMP	00:94:12:15:01:25	00:94:12:15:02:3D	192.168.8.2	192.168.1.1	60/28	255		32	0			00.00.FF.FF.00.00...
8	18:20:32.27	O	/1	IP/ICMP	00:94:12:15:01:3D	00:94:12:15:01:06	192.168.8.2	192.168.1.1	60/28	255		32	0			00.00.FF.FF.00.00...
9	18:20:32.27	O	/1	ARP	00:94:12:15:01:3D	FF:FF:FF:FF:FF:FF			60							00.01.08.00.06.04...
10	18:20:32.27	I	1	ARP	00:94:12:15:01:06	00:94:12:15:01:3D			60							00.01.08.00.06.04...

Figure 24.3

DISCUSSION

1. Describe the behavior of Proxy ARP ,refer to Figure 24.3 .

2. What's the difference between router and proxy ARP?

REACTOR PROGRAM

1. **ProxyArp.mddl**

```
// Proxy ARP
VAR1[0,2]     = {192,168 , 1};     // Subnet 1
VAR1[4,6]     = {192, 168, 8};     // Subnet 2
VAR2[0]       = 1                  ;     // 192.168.1.3 interface 1 of ITS3
VAR2[1]       = 2                  ;     // 192.168.8.1 interface 2 of ITS3

ETHER_IN_HANDLER VAR2[0]
{
IF(S.ETHER_TYPE==CNST_ETHER_TYPE_ARP&&S.ETHER_ARP_OP==CNST_ETHER_ARP_OP_A
RPREQ )
    {
 IF(S.ETHER_ARP_IPADDRTRGT.[0,2]==VAR1[0,2]||S.ETHER_ARP_IPADDRTRGT.[0,2]==VAR1[4,6])
        {
            SEND_OUT_ETHER_FROM_INTERFACE VAR2[0] WITH_DATA
            {
                T                                 = S                                ,
                T.ETHER_MACADDRDST        = S.ETHER_MACADDRSRC              ,
                T.ETHER_MACADDRSRC        = MYMAC(VAR2[0])                  ,
                T.ETHER_ARP_OP                = CNST_ETHER_ARP_OP_ARPREPLY      ,
                T.ETHER_ARP_MACADDRSNDR = MYMAC(VAR2[0])                  ,
                T.ETHER_ARP_IPADDRSNDR    = S.ETHER_ARP_IPADDRTRGT          ,
                T.ETHER_ARP_MACADDRTRGT = S.ETHER_ARP_MACADDRSNDR         ,
                T.ETHER_ARP_IPADDRTRGT    = S.ETHER_ARP_IPADDRSNDR
            }
            DISCARD_MESSAGE;
        }
    }
}

IP_RECEIVED_HANDLER
{
    IF(S.IP_ADDRDST.[0,2]==VAR1[4,6])
    {
        SEND_OUT_IP_FROM_INTERFACE VAR2[1] WITH_NEXTHUB S.IP_ADDRDST WITH_DATA
        {
            T = S
        }
        DISCARD_MESSAGE;
    }
   IF(S.IP_ADDRDST.[0,2]==VAR1[0,2])
    {
        SEND_OUT_IP_FROM_INTERFACE VAR2[0] WITH_NEXTHUB S.IP_ADDRDST WITH_DATA
        {
            T = S
        }
        DISCARD_MESSAGE;
    }
}
```

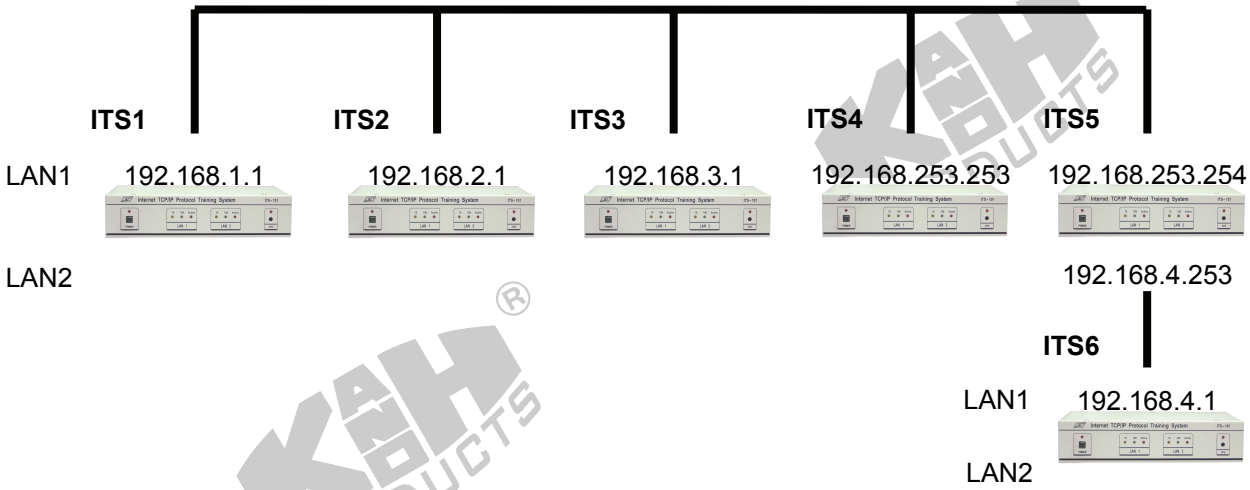
Exp 25. IP Aliasing

OBJECTIVE：To understand what the IP aliasing is and how to implement it.

BRIEF DESCRIPTION：This experiment examines the IP aliasing that is used to adding more than one IP address to a network interface. By using **MDDL**, students can learn how to implement the IP aliasing mechanism.

DURATION: 4.5 hrs

TOPOLOGY



TECHNICAL BACKGROUND

IP aliasing is the process of adding more than one IP address to a network interface. With this, one node on a network can have multiple connections to a network, each serving a different purpose.

PROCEDURE

Realizing Network Topology

1. Complete the network connections on HUBOX by referring to Figure 25.1.

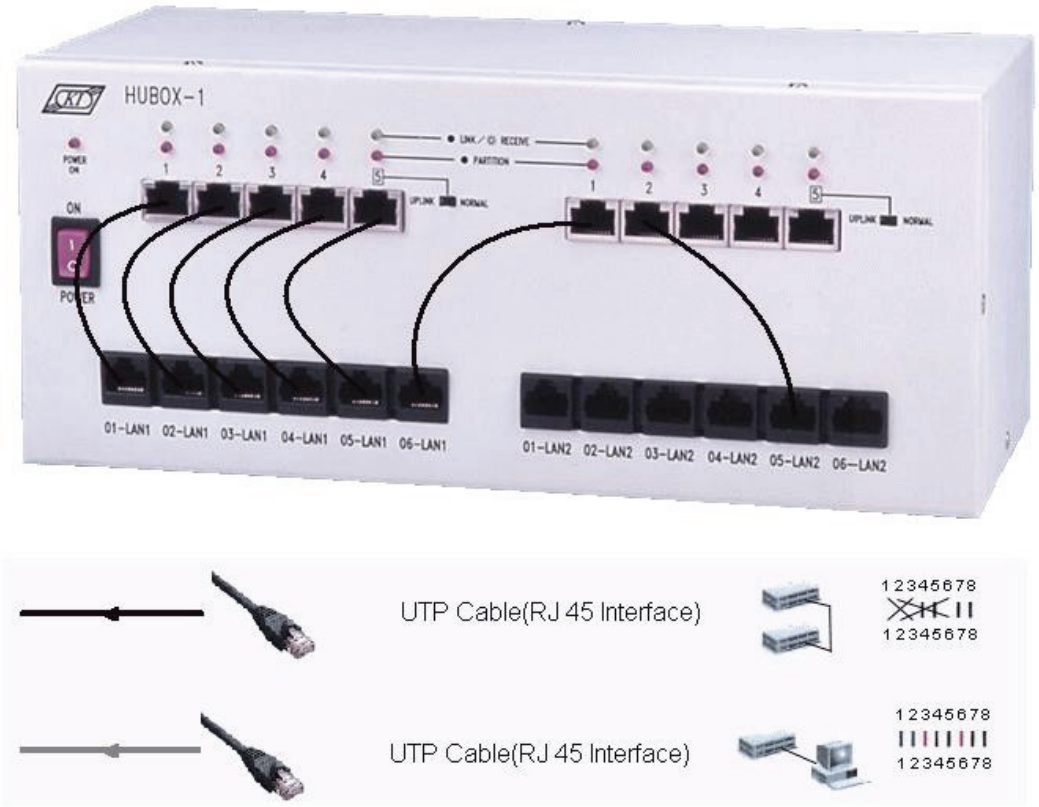


Figure 25.1

IP Alias Application

A. Setup

2. Execute **XCLIENT.BAT** to open the KCodes Network Explorer for ITS window.
3. Open the Network Configuration dialog box by selecting **Network Configuration** from the Tool menu.

ITS1

4. Type "**192.168.1.1**" into IP Address of Interface 1 and enter "**255.255.255.0**" into Subnet Mask of Interface 1. In the Routing Table, enter "**192.168.1.253**" into Gateway and "**0.0.0.0**" into Destination and Mask. Set **Host** and click the **Set & Close** button. (See Figure 25.2.)

ITS2

5. Type "**192.168.2.1**" into IP Address of Interface 1 and enter "**255.255.255.0**" into Subnet Mask of Interface 1. In the Routing Table, enter "**192.168.2.253**" into Gateway and "**0.0.0.0**" into Destination and Mask. Set **Host** and click the **Set & Close** button.

ITS3

6. Type "**192.168.3.1**" into IP Address of Interface 1 and enter "**255.255.255.0**" into Subnet Mask of Interface 1. In the Routing Table, enter "**192.168.3.253**" into Gateway and "**0.0.0.0**" into Destination and Mask. Set **Host** and click the **Set & Close** button.

ITS4 (IP Alias)

7. Type "**192.168.253.253**" into IP Address of Interface 1 and enter "**255.255.255.0**" into Subnet Mask of Interface 1. In the Routing Table, enter "**192.168.253.254**" into Gateway and "**0.0.0.0**" into Destination and Mask. Set **Gateway** and click the **Set & Close** button.

ITS5 (Gateway)

8. Type "**192.168.253.254**" into IP Address of Interface 1 and enter "**192.168.4.253**" into IP Address of Interface 2. Enter "**255.255.255.0**" into Subnet Mask of Interface 1 and Interface 2. In the Routing Table, enter "**192.168.253.253**" into Routing Table. Destination and mask are "**0.0.0.0**". Set **Gateway** and click the **Set & Close** button.

ITS6

9. Type "**192.168.4.1**" into IP Address of Interface 1 and enter "**255.255.255.0**" into Subnet Mask of Interface 1. In the Routing Table, enter "**192.168.4.253**" into Gateway and "**0.0.0.0**" into Destination and Mask. Set **Host** and click the **Set & Close** button. (Because we set the different subnet IP address for each ITS, ITS1 and ITS2 seem to be in the different subnet even they are in the same physical network.)

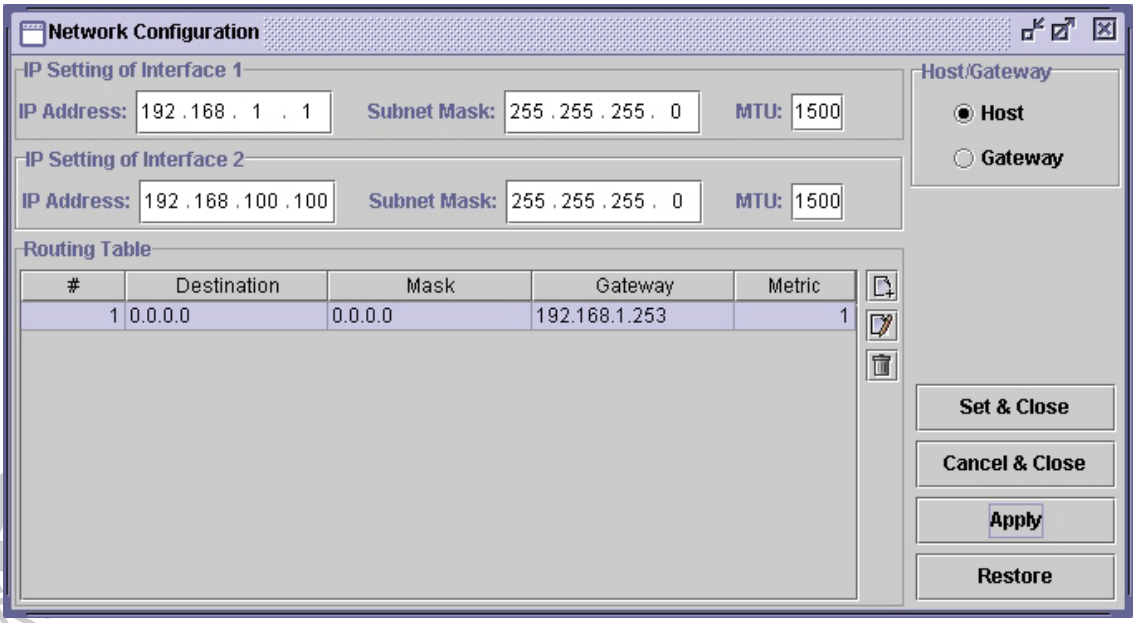


Figure 25.2

B. IP Aliasing

ITS4

10. Open the Network Message Browser window. Check **Listening On**.
11. Open the MDDL Editor by selecting **MDDL Reactor Panel** from the Reactor menu.
12. Click the **Load** button. Open the file C: \XClient \Data \Mddl \Tutorial \Ex25 \IPAlias.mddl, and click the **Upld** button.

ITS 1, 2, 3, 5 and 6

13. Open the Network Message Browser window. Check **Listening On**.
14. Referring to the Ex4 (Figure 4.1), ITS1 sends ICMP Echo Request to ITS6. Figure 25.3 shows that ITS1 sends ICMP Echo Request and receives ICMP Echo Reply back. Figure 25.4 shows that ITS4 (IP Alias) receives the ICMP Echo Request and routes to ITS6. Figure 25.5 shows that ITS6 receives ICMP Echo Request and sends ICMP Echo Reply.

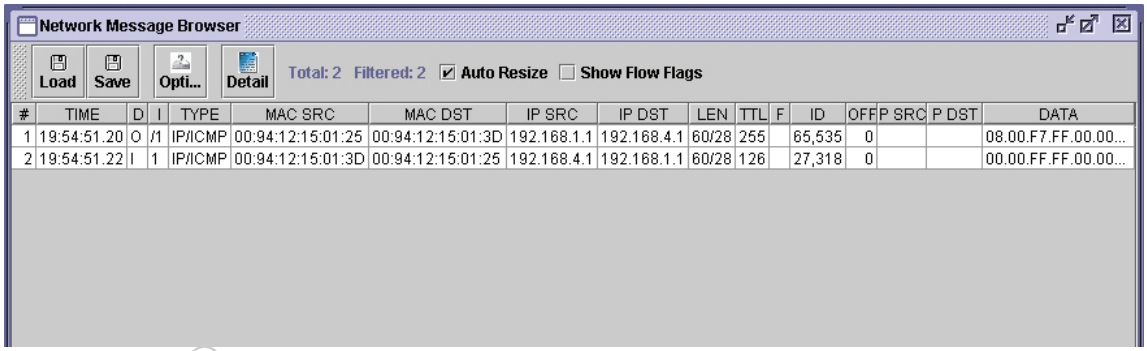


Figure 25.3

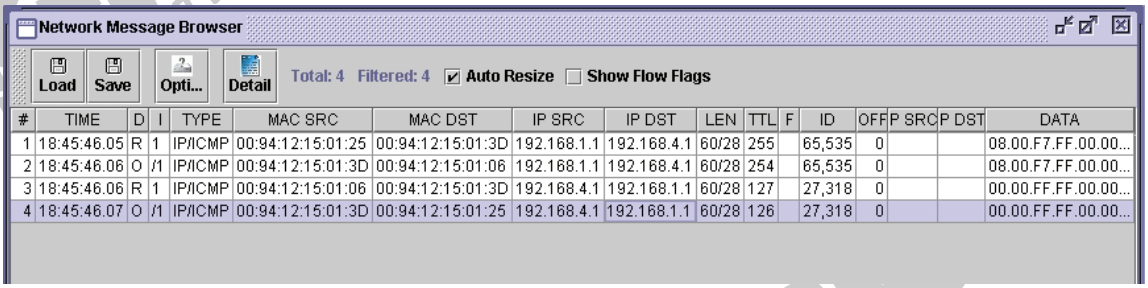


Figure 25.4

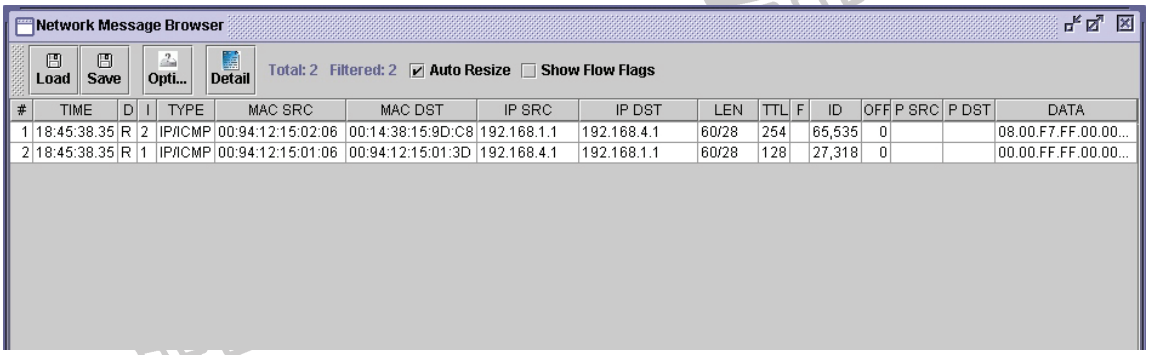


Figure 25.5

DISCUSSION

1. Try to send ICMP Echo Request from ITS1 to ITS2 or ITS3 and observe what will happen. Can ITS1 send ICMP Echo Request to ITS2 or ITS3 directly?

REACTOR PROGRAM

1. **IPAlias.mddl**

```
// IP Alias
VAR1[0,3]       ={192, 168, 1, 253}          ; // ITS 1
VAR1[4,7]       ={192, 168, 2, 253}          ; // ITS 2
VAR1[8,11]      ={192, 168, 3, 253}          ; // ITS 3

VAR2[0]          = 1                           ; // 192.168.253.253
VAR2[1]          = 2                           ;
VAR2[2,5]       = {192, 168, 253, 254}       ; //default gateway

ETHER_IN_HANDLER VAR2[0]
{

IF(S.ETHER_TYPE==CNST_ETHER_TYPE_ARP&&S.ETHER_ARP_OP==CNST_ETHER_ARP_OP_A
RPREQ )
    {

IF(S.ETHER_ARP_IPADDRTRGT==VAR1[0,3]||S.ETHER_ARP_IPADDRTRGT==VAR1[4,7]||S.ETHER_
ARP_IPADDRTRGT==VAR1[8,11])
        {
            SEND_OUT_ETHER_FROM_INTERFACE VAR2[0] WITH_DATA
            {
                T                              = S                                ,
                T.ETHER_MACADDRDST     = S.ETHER_MACADDRSRC      ,
                T.ETHER_MACADDRSRC     = MYMAC(VAR2[0])             ,
                T.ETHER_ARP_OP             = CNST_ETHER_ARP_OP_ARPREPLY,
                T.ETHER_ARP_MACADDRSNDR = MYMAC(VAR2[0])                ,
                T.ETHER_ARP_IPADDRSNDR    = S.ETHER_ARP_IPADDRTRGT      ,
                T.ETHER_ARP_MACADDRTRGT = S.ETHER_ARP_MACADDRSNDR     ,
                T.ETHER_ARP_IPADDRTRGT    = S.ETHER_ARP_IPADDRSNDR
            }
            DISCARD_MESSAGE;
        }
    }
}

IP_ROUTING_HANDLER
{

IF(!ISMYIPADDR(S.IP_ADDRDST)&&S.IP_ADDRDST!=VAR1[0,3]&&S.IP_ADDRDST!=VAR1[4,7]&&S.I
P_ADDRDST!=VAR1[8,11])
    {
     IF(S.IP_TTL>1)
        {
            IF((S.IP_ADDRDST.[0,2]== MYIPADDR(VAR2[0]).[0,2])||(S.IP_ADDRDST.[0,2] ==
               VAR1[0,3].[0,2]) ||(S.IP_ADDRDST.[0,2]==VAR1[4,7].[0,2])||(S.IP_ADDRDST.[0,2] ==
               VAR1[8,11].[0,2]))

            {
                SEND_OUT_IP_FROM_INTERFACE VAR2[0] WITH_NEXTHUB S.IP_ADDRDST
                WITH_DATA
                {
                    T                              = S                                ,
                    T.IP_TTL                   = S.IP_TTL - 1                ,
                    T.IP_HEADERCHKSUM = {0, 0}                        ,
                    T.IP_HEADERCHKSUM = CHECKSUM(T.IP_HEADER)
                }
```

```
            DISCARD_MESSAGE;
        }

        ELSE
        {

            IF(S.IP_ADDRDST.[0,2]== MYIPADDR(VAR2[1]).[0,2])
            {
                SEND_OUT_IP_FROM_INTERFACE VAR2[1] WITH_NEXTHUB S.IP_ADDRDST
                WITH_DATA
                {
                    T                       = S                     ,
                    T.IP_TTL                = S.IP_TTL - 1          ,
                    T.IP_HEADERCHKSUM = {0, 0}                  ,
                    T.IP_HEADERCHKSUM = CHECKSUM(T.IP_HEADER)
                }
                DISCARD_MESSAGE;

            }

            ELSE      //default gateway
            {
                SEND_OUT_IP_FROM_INTERFACE VAR2[0] WITH_NEXTHUB VAR2[2,5]
                WITH_DATA
                {
                    T                       = S                     ,
                    T.IP_TTL                = S.IP_TTL - 1          ,
                    T.IP_HEADERCHKSUM = {0, 0}                  ,
                    T.IP_HEADERCHKSUM = CHECKSUM(T.IP_HEADER)
                }
                DISCARD_MESSAGE;
            }
        }
    }
}

IP_RECEIVED_HANDLER
{
    IF
    (
        S.IP_PROT                   == CNST_IP_PROT_ICMP            &&
        S.IP_DATA.ICMP_TYPE_CODE == CNST_ICMP_TYPE_CODE_ECHOREQ
    )
    {

IF(ISMYIPADDR(S.IP_ADDRDST)||S.IP_ADDRDST==VAR1[0,3]||S.IP_ADDRDST==VAR1[4,7]||S.IP_A
DDRDST==VAR1[8,11])
        {
            IF((S.IP_ADDRSRC.[0,2]== MYIPADDR(VAR2[0]).[0,2])||(S.IP_ADDRSRC.[0,2] ==
            VAR1[0,3].[0,2]) ||(S.IP_ADDRSRC.[0,2]==VAR1[4,7].[0,2])||(S.IP_ADDRSRC.[0,2] ==
            VAR1[8,11].[0,2]))

            {
                SEND_OUT_IP_FROM_INTERFACE VAR2[0] WITH_NEXTHUB S.IP_ADDRSRC
                WITH_DATA
                {
                    T                       = S                                     ,
                    T.IP_TTL                = S.IP_TTL - 1                          ,
                    T.IP_ADDRSRC            = S.IP_ADDRDST                          ,
```

```
                    T.IP_ADDRDST                = S.IP_ADDRSRC                  ,
                    T.IP_DATA.ICMP_TYPE_CODE =
                    CNST_ICMP_TYPE_CODE_ECHOREPLY                               ,
                    T.IP_DATA.ICMP_CHKSUM       = {0x00, 0x00}                  ,
                    T.IP_DATA.ICMP_CHKSUM       = CHECKSUM(T[20,])              ,
                    T.IP_HEADERCHKSUM           = {0, 0}                        ,
                    T.IP_HEADERCHKSUM           = CHECKSUM(T.IP_HEADER)
                }
                DISCARD_MESSAGE;
            }
            ELSE
            {
                IF(S.IP_ADDRSRC.[0,2]== MYIPADDR(VAR2[1]).[0,2])
                {
                    SEND_OUT_IP_FROM_INTERFACE VAR2[1] WITH_NEXTHUB S.IP_ADDRSRC
                    WITH_DATA
                    {
                        T                           = S                         ,
                        T.IP_TTL                    = S.IP_TTL - 1              ,
                        T.IP_ADDRSRC                = S.IP_ADDRDST          ,
                        T.IP_ADDRDST                = S.IP_ADDRSRC          ,
                        T.IP_DATA.ICMP_TYPE_CODE    =
                        CNST_ICMP_TYPE_CODE_ECHOREPLY ,
                        T.IP_DATA.ICMP_CHKSUM       = {0x00, 0x00}                  ,
                        T.IP_DATA.ICMP_CHKSUM       = CHECKSUM(T[20,])          ,
                        T.IP_HEADERCHKSUM           = {0, 0}                    ,
                        T.IP_HEADERCHKSUM           = CHECKSUM(T.IP_HEADER)
                    }
                    DISCARD_MESSAGE;
                }
            }
        }
    }
}
```