

Exp 9: 路径 MTU 之发现

目的: 练习如何找出path MTU 的方法。

摘要: 本实验中先简单介绍 UDP 协议，并发送一个数据量较大的UDP报文让IP层进行切割，观察何谓 MTU。再透过MDDL的程序设计让学生了解「发现路径 MTU(Path MTU Discovery)算法」的运作规则。

时间: 3 hrs。

一、网络拓扑

A:

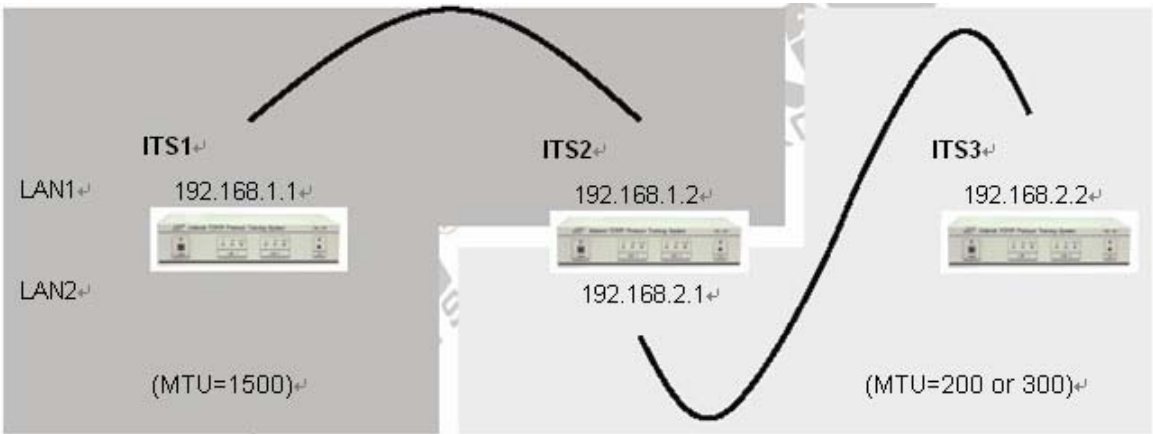


图9.1

B:

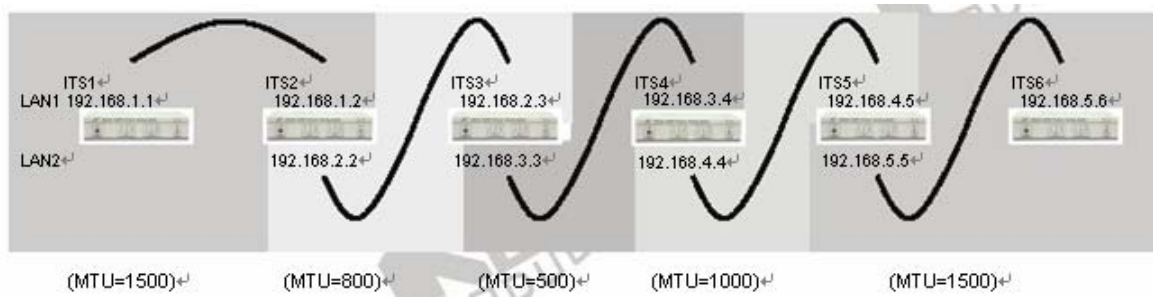


图9.2

二、技术背景

What is MTU?

MTU (Maximum Transfer Unit)是指IP报文的最大传输单位。当一IP报文欲传输跨越数个网络时，如果欲传送的报文大小比网络中最小的MTU还大，IP便会执行切割

(fragment- ation), 把报文分成较小的区块, 让每个小段比MTU小。此网络中的最小传输限制值就称为MTU而这段发现MTU的过程就称为发现路径MTU (Path MTU Discovery, documented in RFC 1191) 。

Path MTU Discovery的做法是传送一个不要切割(DF位被设定为1), 且大小为本身MTU的IP报文, 如果目的端或传递路径中的router发现此报文的大小超过本身的MTU且旗标DF又设定为不得切割, 则会丢弃此封包并传回一ICMP错误讯息“Destination Unreachable”(TYPE=3), 其内文意义为“Fragmentation needed and DF set”(code=4), 并其本身的MTU大小, 放置在下表中NEXT-HOP MTU字段里告知发送端。

0	8	16	31
TYPE (3)	CODE (4)	CHECKSUM	
UNUSED (0)		NEXT-HOP MTU	
INTERNET HEADER + 64 BITS OF ORIGINAL DATAGRAM DATA			
...			

表9.1

What is UDP?

UDP (User Datagram Protocol)在TCP/IP 协议里被定义在网络层(Internet Layer)之上, 在OSI模型中位于传输层(Transport Layer), 换言之, 一个UDP报文(UDP segment, 又称数据段)是被包含在 IP报文(IP datagram)里传递的, 如表9.2所示。UDP协议提供的是不可靠、以非连结为导向的数据传输模式, 它并不支持流量控制(flow control)、错误控制(error control)或是重新传送(retransmission)。然而, 有些应用软件仍会使用UDP协议传送与接受数据而不是TCP协议, 主要是因为UDP对于传输的控制较为简单, 特别是某些强调要在一定时间内送但较不在意错误率的传输上, 更进一步对UDP与TCP协议的讨论会在之后的实验章节说明。UDP协议格式与其主要字段说明如下:

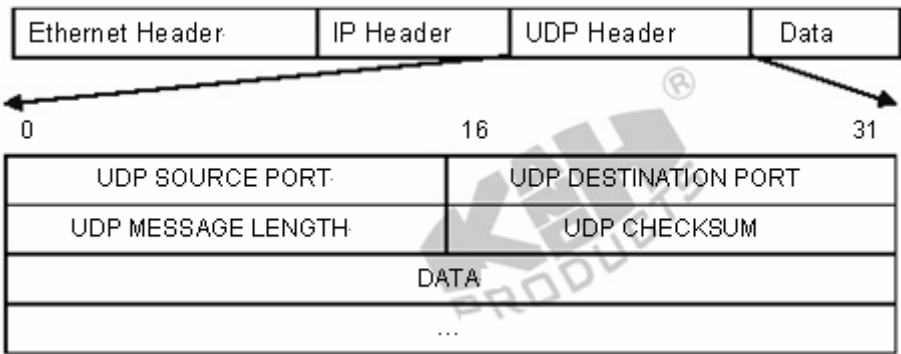


表9.2

SOURCE PORT (16 bits): 发送端连接端口号。此为选择性字段，如未使用则设为“00”。

DESTINATION PORT (16 bits): 目的端连接端口号。

LENGTH (16 bits): UDP标头(header)后的数据长度，最小值为 8。

CHECKSUM (16 bits): UDP校验码。当此字段被设为“0000”时，是代表此报文可以忽略校验码验证。UDP校验码与IP校验码的计算方式是一样的，但UDP报文在计算校验码时需要再加入一个12Bytes的虚拟标头(pseudo-header)做计算，其格式如表9.3所示。

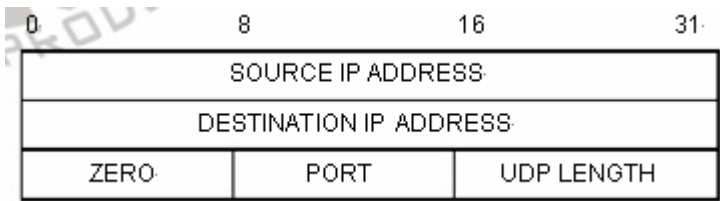


表9.3

SOURCE IP ADDRESS (32 bits): 发送端IP地址。

DESTINATION IP ADDRESS (32 bits): 目的端IP地址。

ZERO (8 bits): 此字段设为“00”，用来将UDP标头的长度补满为16 bits的倍数。

PROTO: 通讯协议代码，UDP为17。

UDP LENGTH: 此为UDP标头的长度，不包含Data与pseudo-header。

Algorithm of Path MTU Discovery:

- 1、找到一联机在不同网络但确定可以沟通的目的端计算机主机。
- 2、创建一个UDP报文，指定其目的IP地址及目的端连接端口(通常最好大于30000)，DF (do not fragmentation)位设为 1，并将此报文的大小(Data length)设定与该发送端网络接口的MTU大小相同，数据内容不限，然后将此报文传送出去。
- 3、等待一个带有错误訊息的ICMP封包传回，首先确认该ICMP封包是否是对应步骤1中发出去的UDP报文。
- 4、如果步骤3接获的ICMP讯息为“ICMP port unreachable”，则表示此UDP报文已成功绕送至目的端虽然出现通讯端口无法找到的错误讯息(通常大于30000的通讯端口为关闭且无特定程序使用) 但我们即可知道此报文的长度即为此段路径的MTU。，
- 5、如果步骤3接获的ICMP讯息为“Fragmentation needed”，则表示此UDP报文长度已经超过此段路径的MTU，我们必须将回应的ICMP报文内的Next-Hop MTU值抄录下来，重新再回到步骤2，但发出去的UDP报文MTU大小须要改为刚刚所抄录下来的

Next-Hop MTU值。

- 6、如果 Next-Hop MTU值小于68，则表示绕送路径中的节点(通常为router)不提供Path MTU Discovery(RFC 1911)，因为MTU最小值为68(RFC894)。我们需要将原报文长度减8再执行步骤2的动作，否则将抄录的MTU值填入欲传送的报文的长度栏并执行步骤2的动作。
- 7、如果步骤3的响应报文时间超过30秒未有响应，则改变一个目的端连接埠，重新再执行步骤2。
- 8、如果步骤7重试了数次都无法改善逾时问题，则表示目的端可能遇到一些未预期性错误 (例如：目的端设置的防火墙拦截了这些预连接未开放端口的封包)，我们可能需要更换一个目的端IP再重试一遍或关闭其对外的防火墙。

三、实验步骤

1、网络拓扑连线

- 1) 在Hubox上将网络连线如图9.3所示，此实验设计以3台ITS为一组。

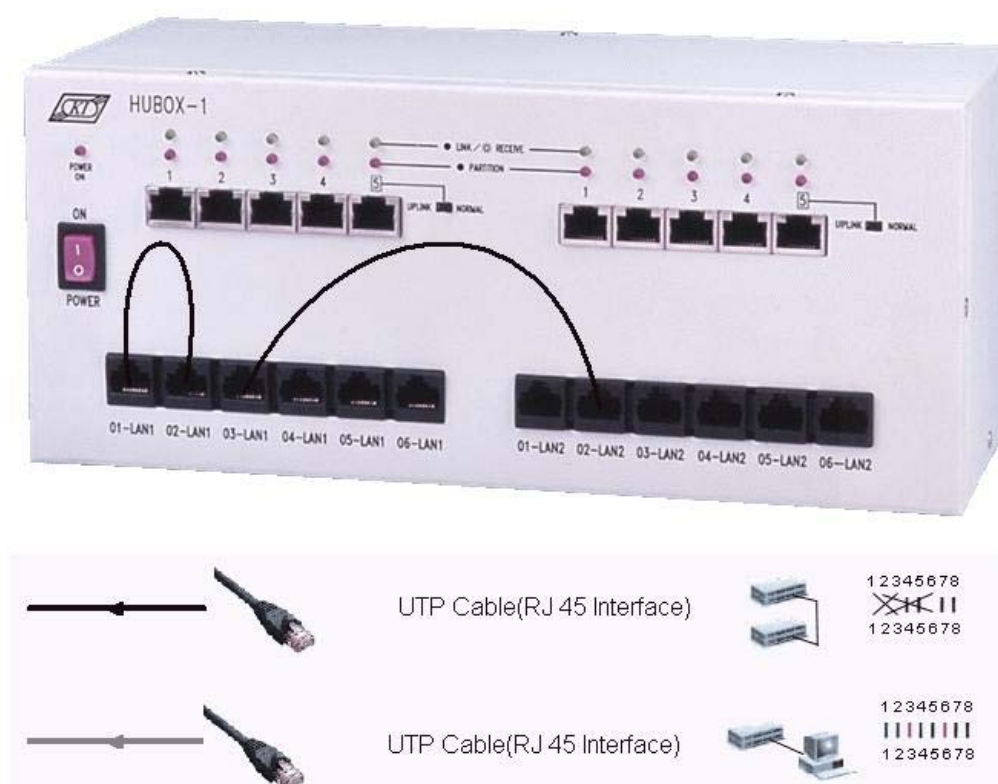


图 9.3

2、切割分组

A. 设定 Host 与 Gateway

- 2) 运行 **XCLIENT.BAT** 程序，打开 ITS 软件界面（KCodes Network Explorer）
- 3) 从 Tool menu 中选择 **Network Configuration** ，打开网络属性设置界面

ITS1 (Host)

- 4) 参照网络拓扑 A，输入 “**192.168.1.1**” 到 **IP Setting Address of Interface 1** 文本框中，并单击 **Add new routing entry** 按钮，见图 9.4。
- 5) 输入 “**192.168.2.0**” 到 Destination 文本框中，输入 “**255.255.255.0**” Mask 文本框中，并且输入 “**192.168.1.2**” 到 Gateway 文本框中(见图 9.5)，最后点击 **Update** 按钮。
- 6) 选择 **Host** 模式并且点击 **Set & Close** 按钮。

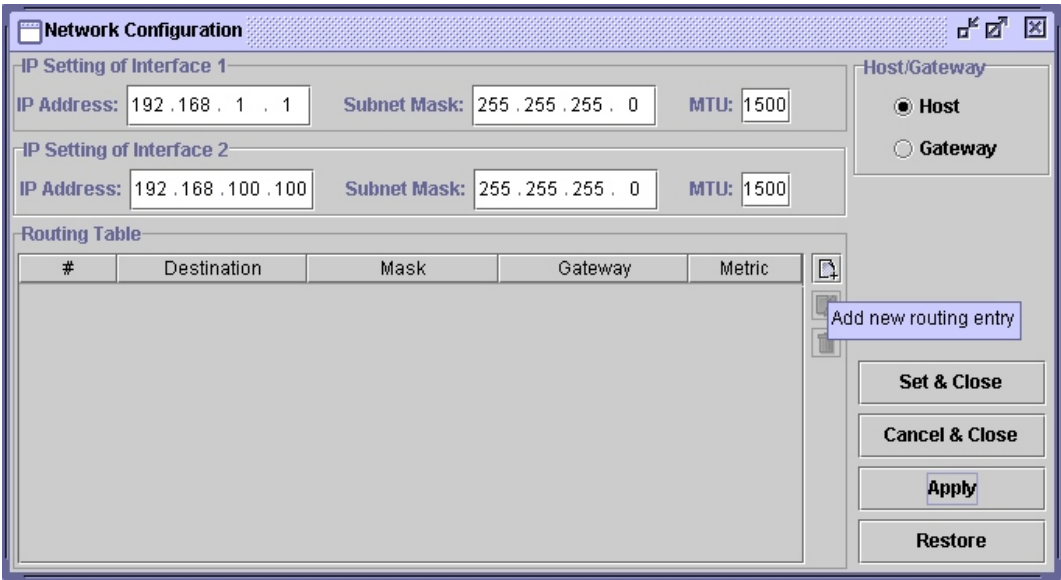


图 9.4

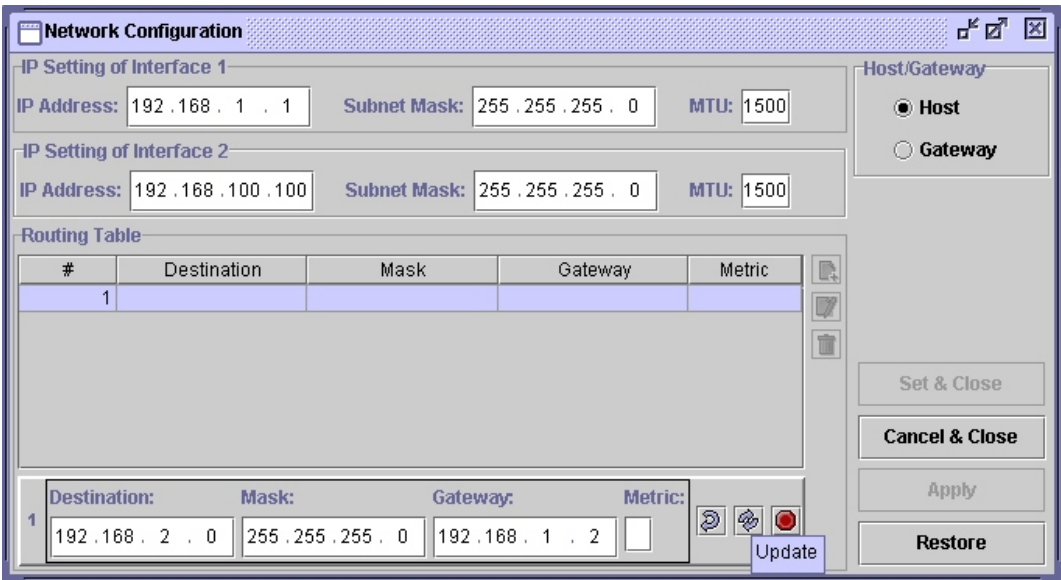


图 9.5

ITS3 (Host)

- 7) 参照网络拓扑 A, ITS 设置的步骤相同。先输入“192.168.2.2”到 **IP Setting Address of Interface 1** 文本框中, 并点击 **Add new routing entry** 按钮。
- 8) 输入“192.168.1.0”到 **Destination** 文本框, 输入“255.255.255.0”到 **Mask** 文本框, 并且输入“192.168.2.1”到 **Gateway** 文本框, 最后点击 **Update** 按钮。
- 9) 选择 **Host** 模式, 并且单击 **Set & Close** 按钮。

ITS2 (Gateway)

- 10) 参照网络拓扑 A, 输入“192.168.1.2”到 **IP Setting Address of Interface 1** 文本框中, 输入到“192.168.2.1” into **IP Setting Address of Interface 2** 文本框中 (见图 9.6)。
- 11) 选择 **Gateway** 模式, 并点击 **Set & Close** 按钮。

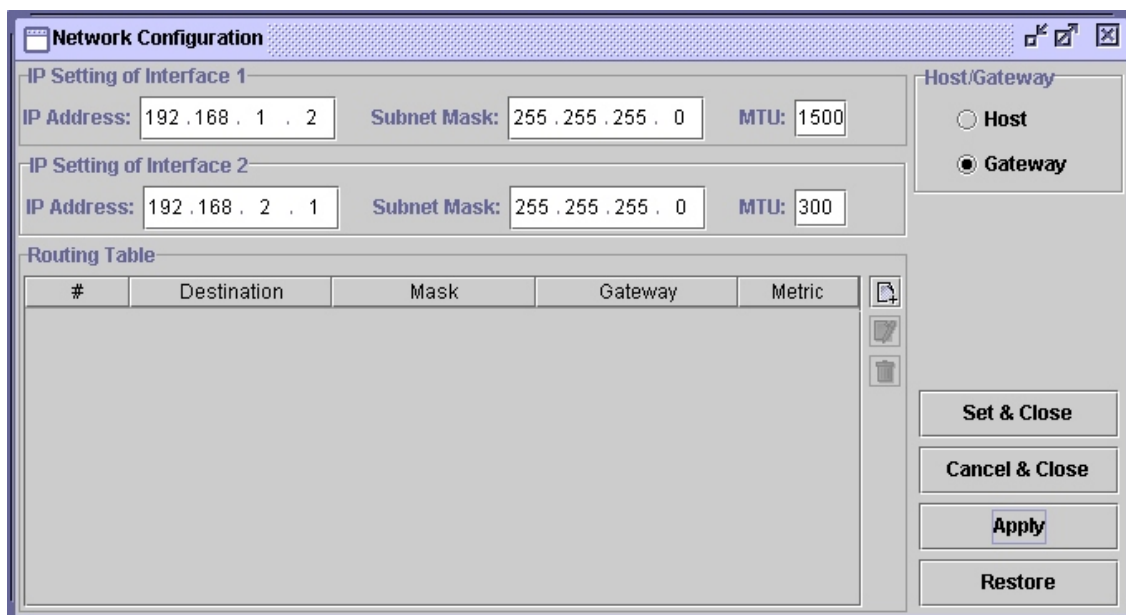


图 9.6

- 12) 完成基本设置后, 尝试发送 ICMP Echo Request 测试 ITS 与 ITS3 是否能正常通信。

B. 设定 MTU 并发送 IP 报文ITS2

- 13) 从 Tool menu 中选择 **Network Configuration**, 打开网络属性设置界面
- 14) 参照图 9.6, 在 interface 1 的 MTU 文本框中输入“1500”, interface 2 的 MTU 文本框中输入 “300”, 最后选择 **Gateway** 模式, 并且点击 **Set & Close** 按钮。
- 15) 从 Listen menu 中选择 **New Memorized Message Browser**, 打开网络信息浏览

器（Network Message Browser），勾选 **Listening On** 开始监听整个网络。

ITS3

- 16) 从 Tool menu 里选择 **Network Configuration**，打开网络属性配置界面。在 Interface 1 的 MTU 文本框中输入“**300**”，最后点击选择 **Host** 模式，并点击 **Set & Close** 按钮。
- 17) 从 Listen menu 中选择 **New Memorized Message Browser**，打开网络信息浏览器（Network Message Browser），勾选 **Listening On** 开始监听整个网络。

ITS1

- 18) 从 Tool menu 里选择 **Network Configuration**，打开网络属性配置界面。在 Interface 1 的 MTU 文本框中输入“**1500**”，最后点击选择 **Host** 模式，并点击 **Set & Close** 按钮。此外，从 send menu 中选择 **Send IP Packet**，打开 IP 报文发送界面。

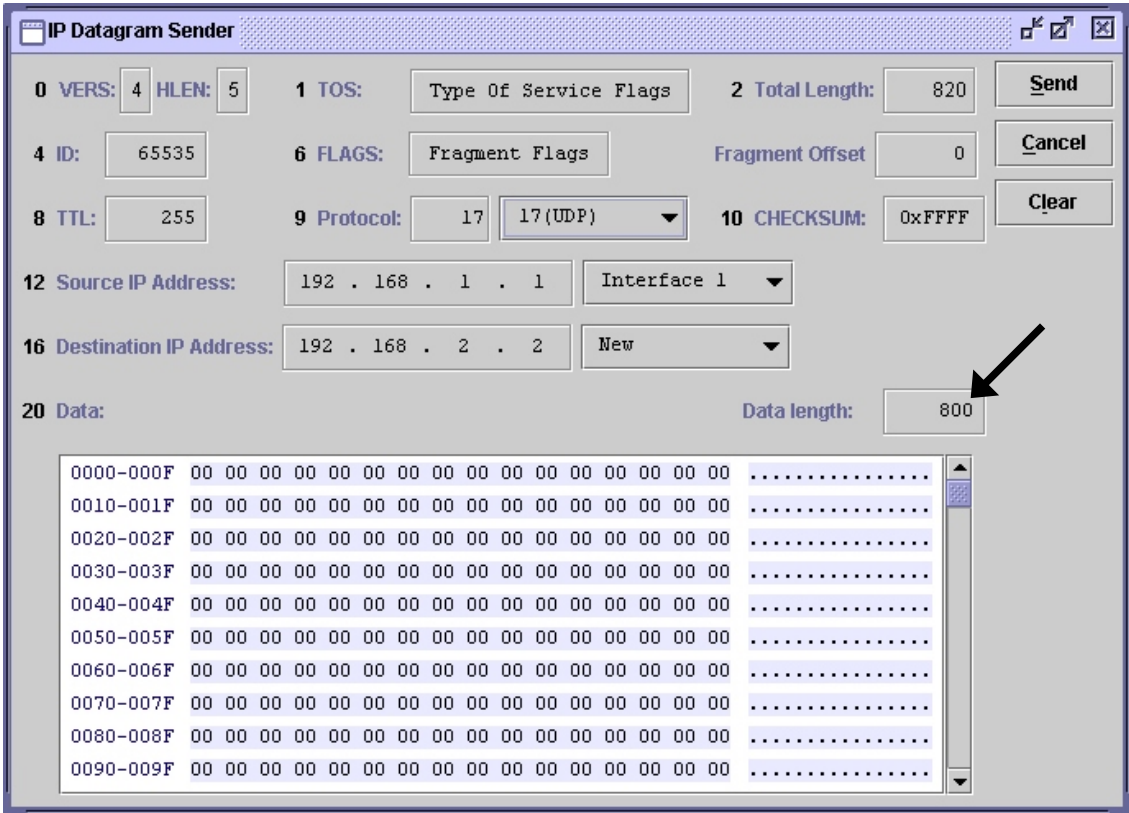
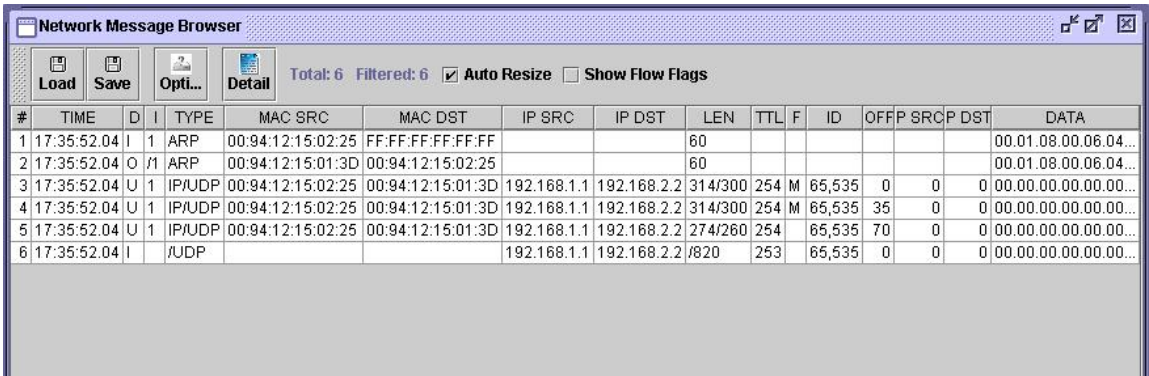


图 9.7

- 19) 输入 “**192.168.2.2**” 到 **Destination IP Address** 文本框中，并且在 **Data length** 中输入 “**800**”，如图 9.7 所示
- 20) 单击 **Send** 按钮。ITS1 会立即发送一个 800bytes 的 UDP 报文到 ITS3。当该报

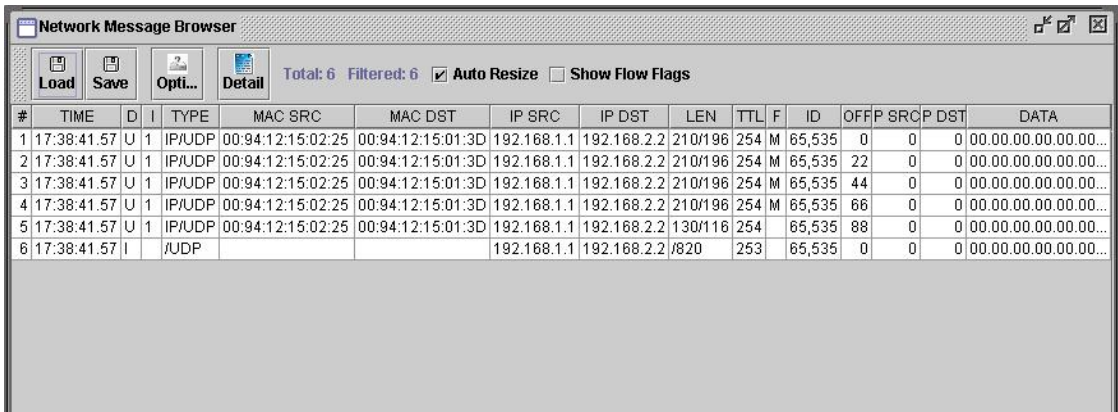
文通过 ITS2 时, 由于 MTU 大小只有 300, 所以该报文将被切割。图 9.8 为 ITS3 接收来自 ITS1, 并被切割后的报文。



#	TIME	D	I	TYPE	MAC SRC	MAC DST	IP SRC	IP DST	LEN	TTL	F	ID	OFFP	SRC	P	DST	DATA
1	17:35:52.04	I	1	ARP	00:94:12:15:02:25	FF:FF:FF:FF:FF:FF			60								00.01.08.00.06.04...
2	17:35:52.04	O	1	ARP	00:94:12:15:01:3D	00:94:12:15:02:25			60								00.01.08.00.06.04...
3	17:35:52.04	U	1	IP/UDP	00:94:12:15:02:25	00:94:12:15:01:3D	192.168.1.1	192.168.2.2	314/300	254	M	65,535	0	0	0	0	00.00.00.00.00.00...
4	17:35:52.04	U	1	IP/UDP	00:94:12:15:02:25	00:94:12:15:01:3D	192.168.1.1	192.168.2.2	314/300	254	M	65,535	35	0	0	0	00.00.00.00.00.00...
5	17:35:52.04	U	1	IP/UDP	00:94:12:15:02:25	00:94:12:15:01:3D	192.168.1.1	192.168.2.2	274/260	254		65,535	70	0	0	0	00.00.00.00.00.00...
6	17:35:52.04	I		/UDP			192.168.1.1	192.168.2.2	/820	253		65,535	0	0	0	0	00.00.00.00.00.00...

图 9.8

21) 将 ITS2 与 ITS3 的 MTU 值从 300 改为 200, 再重新做步骤 18 到 20。图 9.9 为 ITS3 接收来自 ITS1, 并被切割后的报文。



#	TIME	D	I	TYPE	MAC SRC	MAC DST	IP SRC	IP DST	LEN	TTL	F	ID	OFFP	SRC	P	DST	DATA
1	17:38:41.57	U	1	IP/UDP	00:94:12:15:02:25	00:94:12:15:01:3D	192.168.1.1	192.168.2.2	210/196	254	M	65,535	0	0	0	0	00.00.00.00.00.00...
2	17:38:41.57	U	1	IP/UDP	00:94:12:15:02:25	00:94:12:15:01:3D	192.168.1.1	192.168.2.2	210/196	254	M	65,535	22	0	0	0	00.00.00.00.00.00...
3	17:38:41.57	U	1	IP/UDP	00:94:12:15:02:25	00:94:12:15:01:3D	192.168.1.1	192.168.2.2	210/196	254	M	65,535	44	0	0	0	00.00.00.00.00.00...
4	17:38:41.57	U	1	IP/UDP	00:94:12:15:02:25	00:94:12:15:01:3D	192.168.1.1	192.168.2.2	210/196	254	M	65,535	66	0	0	0	00.00.00.00.00.00...
5	17:38:41.57	U	1	IP/UDP	00:94:12:15:02:25	00:94:12:15:01:3D	192.168.1.1	192.168.2.2	130/116	254		65,535	88	0	0	0	00.00.00.00.00.00...
6	17:38:41.57	I		/UDP			192.168.1.1	192.168.2.2	/820	253		65,535	0	0	0	0	00.00.00.00.00.00...

图 9.9

3、MTU的发现

1) 在 Hubox 上将网络连线如图 9.10 所示。

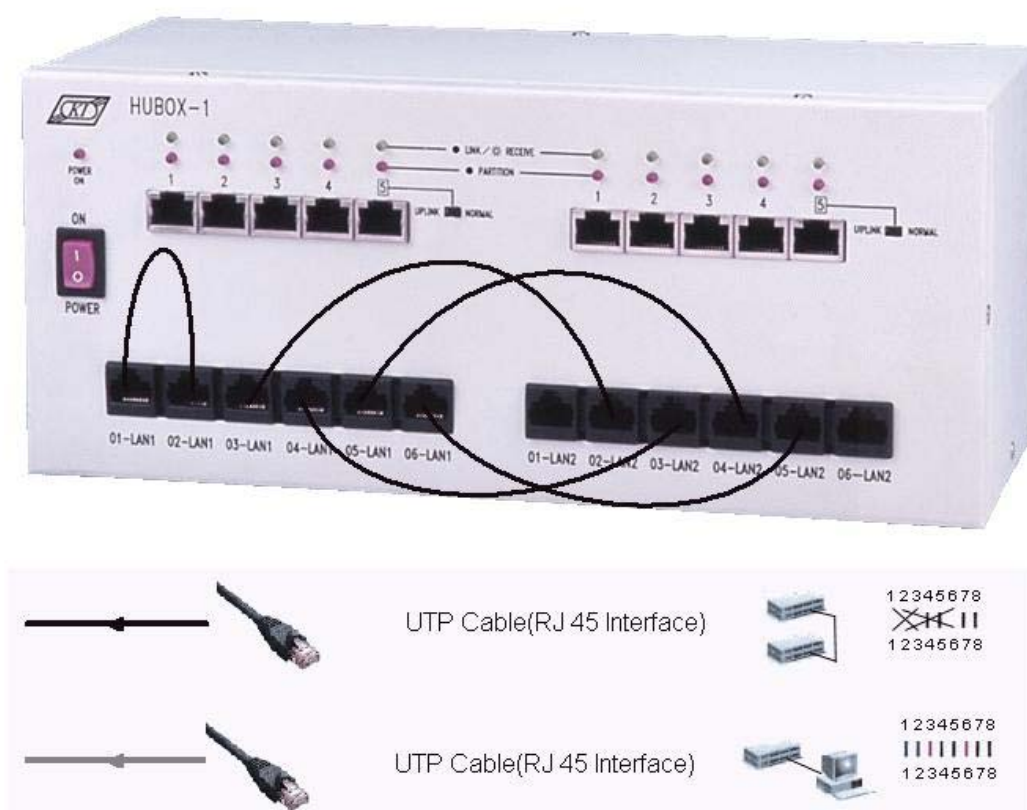


图 9.10

- 2) 参照网络拓扑 Band Exp 7, 设定每台 ITS 的路由表 (见表 7.1)。
- 3) 参照前一部分的实验步骤, 将 ITS2 的 Interface 2 和 ITS3 的 Interface 1 的 MTU 值设为“800”; 将 ITS3 的 Interface 2 和 ITS4 的 Interface 1 的 MTU 设为“500”; 将 ITS4 的 Interface 2 和 ITS5 的 Interface 1 的 MTU 设为 “1000”将 ITS1 的 Interface 1, ITS2 的 Interface 1, ITS5 的 Interface 2 和 ITS6 的 Interface 1 的 MTU 值设为 “1500”。
- 4) 打开网络信息浏览器 (Network Message Browser) 勾选 **Listening On**。

ITS1

- 5) 从 Send menu 里选择 **Send IP Packet** , 打开 **IP Datagram Sender** 界面。输入 “192.168.2.2” 到 **Destination IP Address** 文本框中, 并输入 “1000” 到 **Data length** 文本框中。在 **FLAGES** 选项中选择 **Don't Fragment** , 如图 9.11 所示。

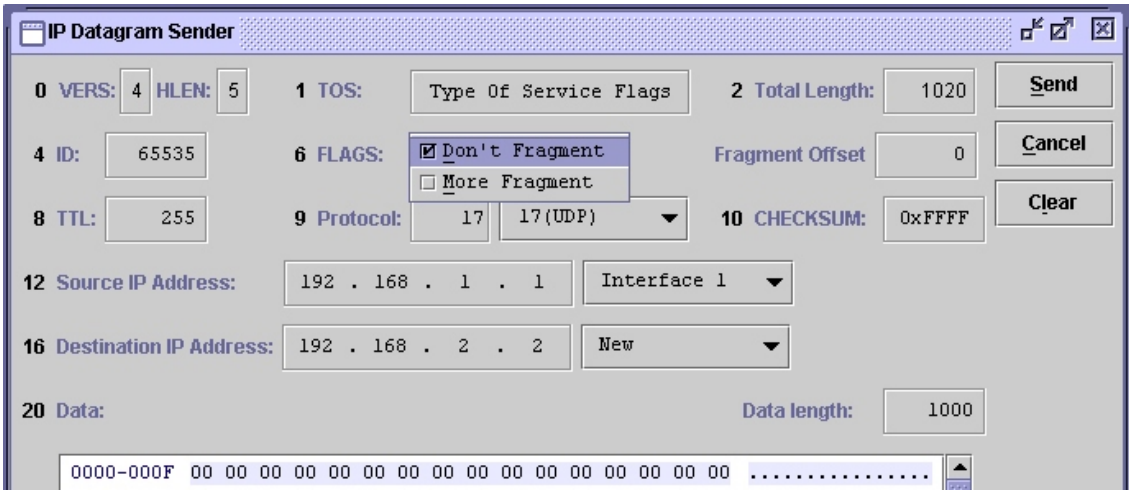


图 9.11

- 6) 最后点击 **Send** 按钮。您的 ITS1 将会发送一个报文给 ITS6。此报文的输入段长度为 1000 且为不可分割。
- 7) 当报文长度大小超过路由所规定的值时，路由会将报文丢弃并回报自己 MT 大小给发送端 (ITS1)。图 9.12 里，我们可以发现 ITS2 丢弃了 ITS1 发送来的报文，并告诉 ITS1 自己的 MTU 的大小。

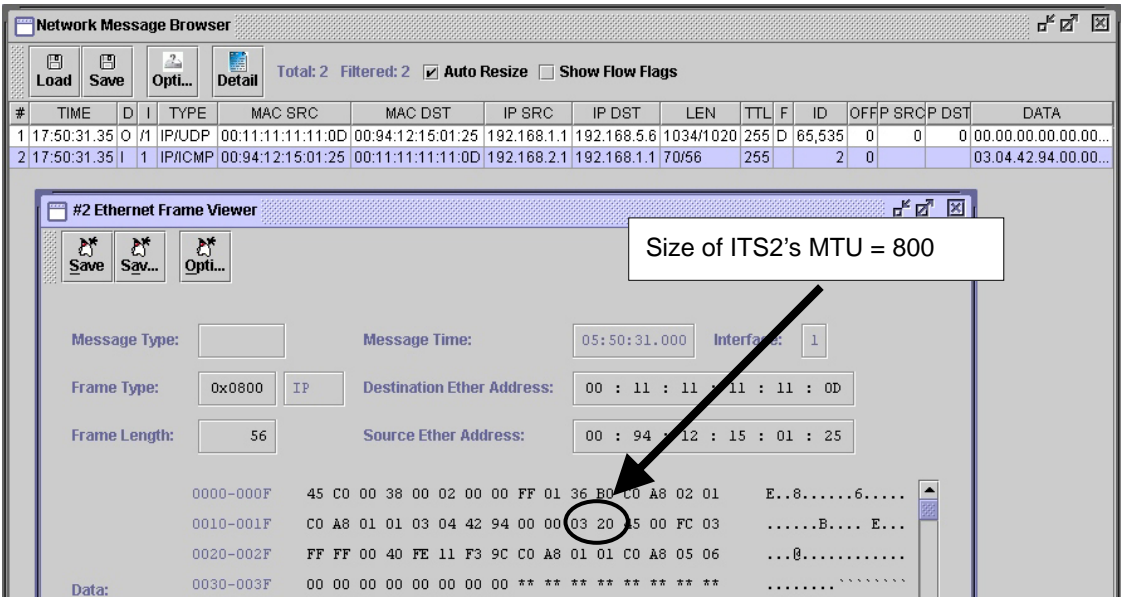


图 9.12

- 8) 重新发一个报文，并将 MTU 值改为路由所回报的 MTU 值。
- 9) 重复步骤 3 到步骤 8，一直到该报文能成功到达 ITS6 为止。

4、利用MDDL发现MTU

ITS1

10) 参考并延续前一部分的实验，在本实验中，我们将要调用一个 mddl 程序，让 ITS

自动发送报文并找到传输路径中的 MTU 值。首先，点击 **Load** 按钮，打开 RulePathMTUDiscovery.mddl 程序（路径为 C:\XClient\Data\Mddl\Tutorial\Ex09\RulePathMTUDiscovery.mddl），最后点击 **Upld** 按钮。

- 11) 从 **Batch** menu 中选择 **MDDL Batch Panel**，ITS 会打开一个名为 **MDDL Batch Editor** 的编辑平台。如图 9.13 所示，点击 **Load** 按钮，调用 StartPathMTUDiscovery.bmddl 程序（路径为 C:\XClient\Data\Mddl\Tutorial\Ex09\StartPathMTUDiscovery.bmddl），最后点击 **Exe** 按钮。

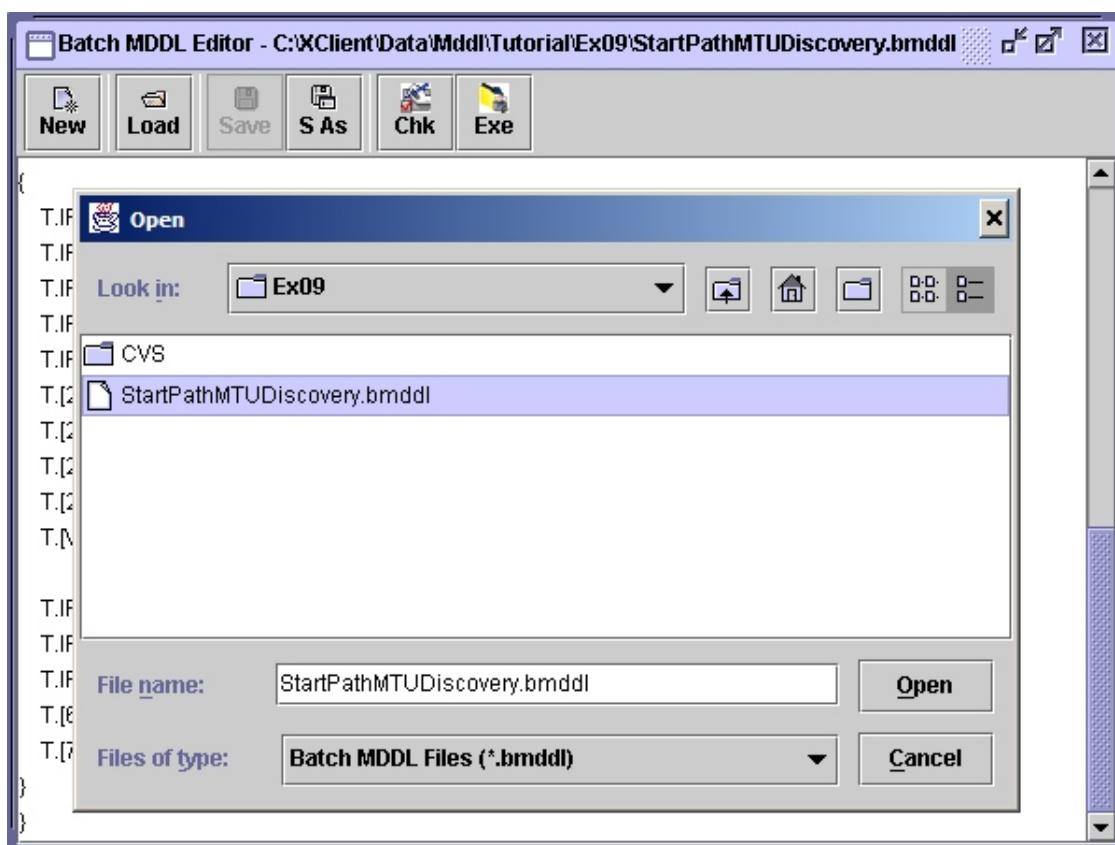


图 9.13

- 12) 当执行完 StartPathMTUDiscovery.bmddl 程序后，ITS1 将自动发送一个报文给 ITS6，并找到路径中的 MTU 值，如图 9.14 所示。

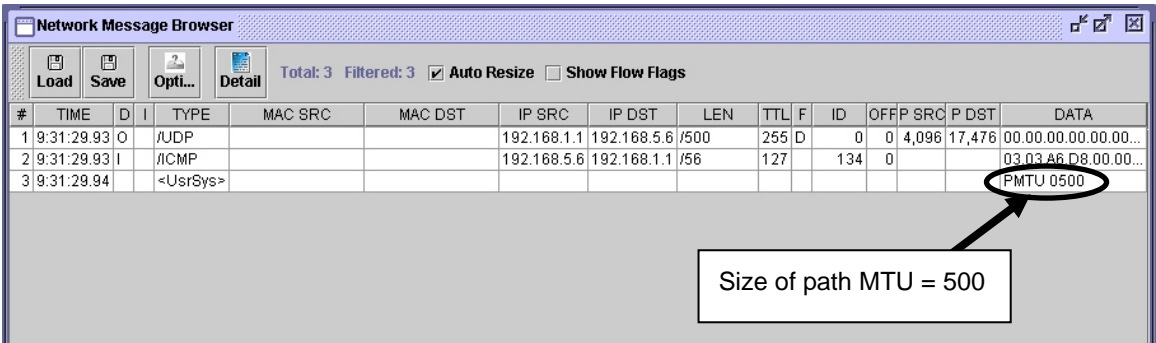


图 9.14

四、实验讨论

- 1、在网络拓扑A里，当你发送MTU=200的封包和MTU=300的封包时，它在传送时，最大的差异在哪？
- 2、在发现路径 MTU 这段实验中，什么状况会重复发生？我们可以很顺利的发送封包到目的端吗？试着找出这整段路径的bottleneck (path MTU)值并解释为什么。
- 3、如果传送时，被切割的封包掉了其中一个或是发生延迟，有没有解决的办法？
(提示：试着利用 ICMP协议或参考实验10)

REACTOR PROGRAM

1、RulePathMTUDiscovery.mddl

```
IP_RECEIVED_HANDLER
{
    IF(S.IP_PROT == CNST_IP_PROT_ICMP && S.[20, 21] == 0x0304W)
    {
        VAR1[4] = 0;
        IF(VAR1[0, 1]<68)
        {
            VAR1[4] = 255;                                     // Abort TRACEROUTE
        }
    ELSE
    {
        IF(VAR1[0, 1]<=S.[26,27]|| 68>S.[26,27])
        {
            VAR1[0, 1] = VAR1[0, 1]-8;
        }
    }
}
```

```

    }
ELSE
{
    VAR1[0, 1] = S.[26,27];
}
SEND_OUT_IP WITH_DATA
{
    T.IP_FLAGSOFF      = 0x4000W, // SET DON'T FRAGMENTATION BIT
    T.IP_TTL           = 255          ,
    T.IP_PROT          = CNST_IP_PROT_UDP ,
    T.IP_ADDRSRC       = VAR2[0, 3]      ,
    T.IP_ADDRDST       = VAR2[4, 7]      ,
    T.[20, 21]         = VAR2[8, 9]      ,
    T.[22, 23]         = VAR1[2, 3]      ,
    T.[24, 25] = VAR1[0, 1] - 20, // UDP length=MINUS IP HEADER LENGTH
    T.[26, 27]         = 0W             ,
    T.[VAR1[0, 1]-1]   = 0              ,
    T.IP_LEN           = LENGTH(T)      ,
    T.IP_HEADERCHKSUM  = {0, 0}         ,
    T.IP_HEADERCHKSUM  = CHECKSUM(T[0,19])
}
}

// DISCARD_MESSAGE;
}

ELSE IF(S.IP_PROT == CNST_IP_PROT_ICMP && S.[20, 21] == 0x0303W) // Port unreachable
{
    VAR1[4] = 255; // Stop TRACEROUTE

    GENERATE_USER_SYSMMSG WITH_DATA
    {
        TARGET = "PMTU ",

```

```

        T.[5]  = ((VAR1[0, 1])/1000)+0X30,
        T.[6]  = (((VAR1[0, 1])%1000)/100)+0X30,
        T.[7]  = (((VAR1[0, 1])%100)/10)+0X30,
        T.[8]  = ((VAR1[0, 1])%10)+0X30
    }

    DISCARD_MESSAGE;

}

}

TIMER_WITH_PERIOD 1000                                     // Period = 1 sec
{
    IF(VAR1[4] != 255)                                       // TRACEROUTE still run
    {
        VAR1[4] = VAR1[4] + 1;                               // Increment time counter
        IF(VAR1[4] >= 60)                                    // 60 secs after last IP send
        {
            VAR1[4] = 0;
            VAR1[2, 3] = VAR1[2, 3] + 1;                     // Increment test port
            IF(VAR1[2, 3] == 0W)                              // Cannot find port for test
            {
                VAR1[4] = 255;                                // Abort TRACEROUTE
            }
        }
        ELSE
        {
            SEND_OUT_IP WITH_DATA
            {
                T.IP_FLAGSOFF  = 0x4000W, // SET DON'T FRAGMENTATION BIT
                T.IP_TTL       = 255      ,
                T.IP_PROT      = CNST_IP_PROT_UDP ,
                T.IP_ADDRSRC   = VAR2[0, 3]      ,
                T.IP_ADDRDST   = VAR2[4, 7]      ,
            }
        }
    }
}

```



```

        T.[20, 21]      = VAR2[8, 9]          ,
        T.[22, 23]      = VAR1[2, 3]          ,
        T.[24, 25]      = VAR1[0, 1] - 20 , // UDP length=MINUS IP HEADER LENGTH
        T.[26, 27]      = 0W                  ,
        T.[VAR1[0, 1]-1] = 0                  ,
        T.IP_LEN        = LENGTH(T)           ,
        T.IP_HEADERCHKSUM = {0, 0}            ,
        T.IP_HEADERCHKSUM = CHECKSUM(T[0,19])

    }

}

}

}

```

2、StartPathMTUDiscovery.bmddl

```

// VARIABLE FOR ALGORITHM

VAR1[0, 1] = IPMTU(1); // IP LEN
VAR1[2, 3] = 0x4444W;  // DESTINATION PORT
VAR1[4]    = 0;        // TIME COUNTER

// CONSTANT DEFINATION

VAR2[0, 3] = MYIP(1); // SOURCE IP
VAR2[4, 7] = {192,168,5,6}; // DESTINATION IP
VAR2[8, 9] = 0x1000W; // SOURCE PORT

SEND_OUT_IP WITH_DATA
{
    T.IP_FLAGSOFF = 0x4000W , // SET DONT FRAGMENTATION BIT
    T.IP_TTL      = 255     ,
    T.IP_PROT     = CNST_IP_PROT_UDP ,

```

```
T.IP_ADDR SRC      = VAR2[0, 3]      ,
T.IP_ADDR DST      = VAR2[4, 7]      ,
T.[20, 21]         = VAR2[8, 9]      ,
T.[22, 23]         = VAR1[2, 3]      ,
T.[24, 25]         = VAR1[0, 1] - 20 , // UDP length=MINUS IP HEADER LENGTH
T.[26, 27]         = 0W              ,
T.[VAR1[0, 1]-1]   = 0              ,
T.IP_LEN           = LENGTH(T)      ,
T.IP_HEADERCHKSUM  = {0, 0}         ,
T.IP_HEADERCHKSUM  = CHECKSUM(T[0,19])

}

}
```