

Exp 22: NAT

目的：了解NAT的运作模式及如何执行。

摘要：本实验将介绍网络地址转换(NAT, Network Address Translation), 其主要功能是让使用者能通过私有IP 地址(private IP address)也能畅游因特网。另外实验中也借助MDDL程序语言, 解析NAT机制的作业流程及原理。

时间：4.5 hrs。

一、网络拓扑

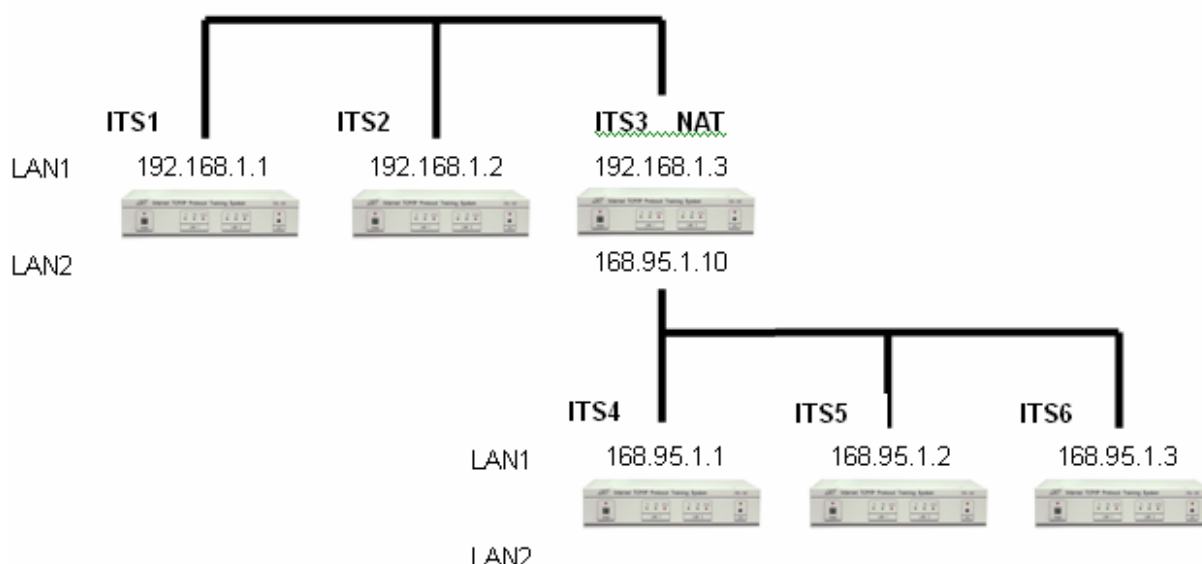


图 22.1

二、技术背景

由于目前的因特网在TCP/IP 协议的规划中, 公共IP地址(public IP address)严重不足, 于是私有IP 地址(private IP address)与公共IP 地址的转换机制就出现了, 也就是网络地址转换(NAT, Network Address Translation)。NAT可以让多台计算机只要有一个公共IP 地址, 就能够一起使用因特网。

在谈NAT之前, 我们要先了解在TCP联机中, 关于插槽组(socket pair)的观念。每一个TCP 联机主要是靠4 个值来维持通道: 来源端地址、来源端通讯端口、目的端地址、目的端通讯端口。其中来源端地址与来源端通讯端口为来源端插槽(source socket), 目的端地址与目的端通讯端口为目的端插槽(destination socket), 两者加起来成为一个插槽

组。而NAT能让这么多的计算机凭借一个公共IP地址对外沟通，其主要的原理，就是NAT server在每台计算机要与网际网络建立TCP联机时，更换其插槽。

Socket pair			
Socure socket		Destination socket	
Source address	Socure port	Destination address	Destination port

表22.1

当NAT server接获一NAT client的对外封包时，NAT server 会先进入NAT table中寻找对应的来源端插槽(即source address 和source port)，如果找不到就会在NAT table中新建一笔记录，记录中包含原本的来源端插槽与新的来源端插槽，新的来源端插槽里的来源端地址即为公共IP 地址，来源端通讯端口则由NAT决定。以下为主要流程：

- 从non-NAT port 接收到内部网络传来的封包
- 从NAT table 中查寻此封包的来源端插槽(socure socket)的资料
- 如果NAT table 有，将其来源端插槽更换后传送出去
- 如果没有，建立一组新记录，并将其来源端插槽更换后传送出去

反之，当NAT server 接获从外部如因特网传来的封包后，其主要流程如下：

- 从NAT port 接收到外部网络传来的封包
- 从NAT table 中查寻此封包的目的端插槽(destination socket)的数据
- 如果NAT table 有，将其目的端插槽更换后传给其对应的私有IP 地址
- 如果没有，此封包会被拒绝进入内部网络

在port mapping table 中每一笔记录都有其生命周期，每当接获client端的联机请求时，NAT server都会将生命周期的值重置。一旦生命周期届满NAT server就会将此记录从NAT table中删除，此作法的功用在于限制NAT table的无限扩张。生命周期的长度依各系统而不同，有的NAT系统是在TCP联机结束时就将NAT table中的记录删除，不过此种作法并不适用于UDP传输协议，因为它是以非连结为导向的传输方式。

许多TCP/IP协议组中较高层级的协议会将client端的IP地址信息记录在封包内，例如：当一个FTP的控制联机(control connetion)建立时，client端会告知FTP server其本身的IP地址与通讯端口号码，让FTP server可以连过来建立数据联机(data connetion)。这时候就比较麻烦了，NAT server不只是改插槽就好，还必须要看得懂这些封包的内容，并且更改此IP地址的信息，同时极有可能封包长度、序列号码(sequence number)和确认号码

(acknowledgenumber)也都必须更改。

总而言之，NAT支持大部份的TCP/IP协议，但并非全部。就像FTP一样，部份协议需要NAT server能够了解其协议格式或内容，又或者其协议需要让client端知道NAT，才能进行网络地址转译。

三、实验步骤

1、了解网络拓扑

1) 在Hubox 上将网络连接如图22.2所示。

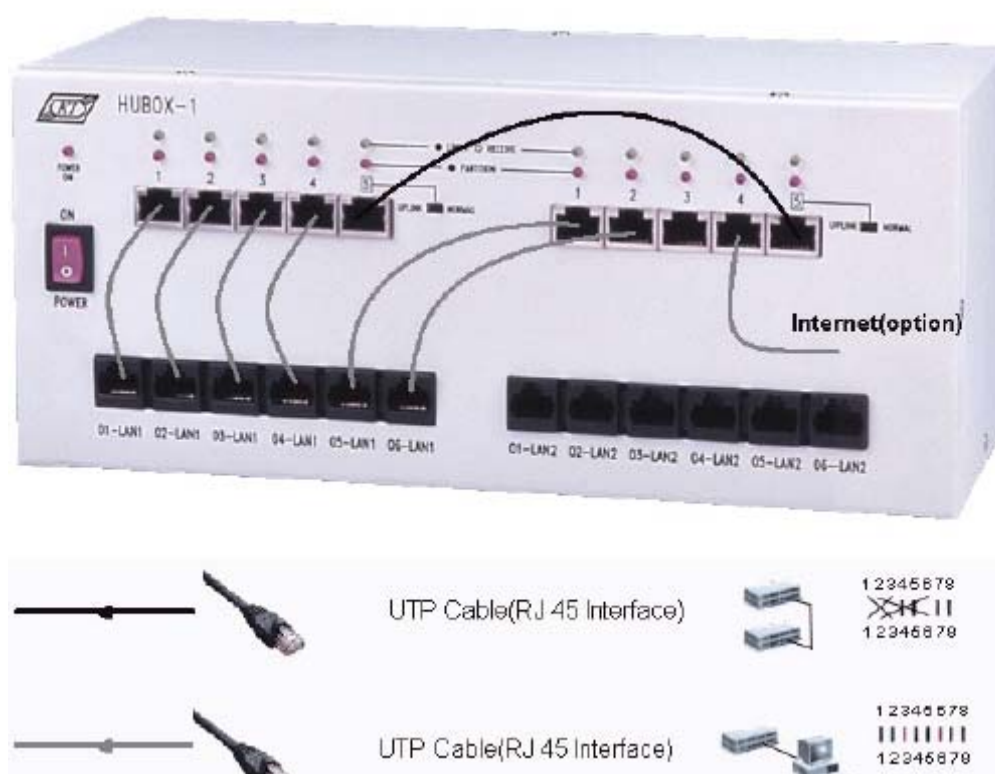


图 22.2

2、Transmission using NAT Table 实验

A. 初始设置:

- 2) 执行 **XCLIENT.BAT**，打开 ITS 应用软件 KCodes Network Explorer
- 3) 从 Tool 菜单部分打开“网络配置”对话框 (**Network Configuration**) 见图 22.3。

ITS1 设置如下:

- 4) 定义 Interface 1 的 IP 地址为“**192.168.1.1**”。在路由表部分 (Routing Table)，
“Destination ”and “Mask” 部分 输入 “**0.0.0.0**”； “Gateway” 部分 输入
“**192.168.1.3**”； 模式设定为“**Host**”，然后点击“**Set & Close**”按钮。

ITS2 设置如下:

- 5) 定义 Interface 1 的 IP 地址为“**192.168.1.2**”。在路由表部分 (Routing Table), “Destination ”and “Mask”部分输入“**0.0.0.0**”; “Gateway”部分输入“**192.168.1.3**”; 模式设定为“**Host**”, 然后点击“**Set & Close**”按钮。

ITS3 (NAT) 设置如下:

- 6) 定义 Interface 1 的 IP 地址为“**192.168.1.3**”, 定义 Interface2 的 IP 地址为“**168.95.1.10**” 模式设定为“**Gateway**”, 然后点击“**Set & Close**”按钮。

ITS4

- 7) 定义 Interface 1 的 IP 地址为“**168.95.1.1**”。在路由表部分 (Routing Table), “Destination ”and “Mask” 部分 输入 “**0.0.0.0**”; “Gateway” 部分 输入 “**168.95.1.10**”; 模式设定为“**Host**”, 然后点击“**Set & Close**”按钮。

ITS5

- 8) 定义 Interface 1 的 IP 地址为“**168.95.1.2**”。在路由表部分 (Routing Table), “Destination ”and “Mask” 部分 输入 “**0.0.0.0**”; “Gateway” 部分 输入 “**168.95.1.10**”; 模式设定为“**Host**”, 然后点击“**Set & Close**”按钮。

ITS6

- 9) 定义 Interface 1 的 IP 地址为“**168.95.1.3**”。在路由表部分 (Routing Table), “Destination ”and “Mask” 部分 输入 “**0.0.0.0**”; “Gateway” 部分 输入 “**168.95.1.10**”; 模式设定为“**Host**”, 然后点击“**Set & Close**”按钮。

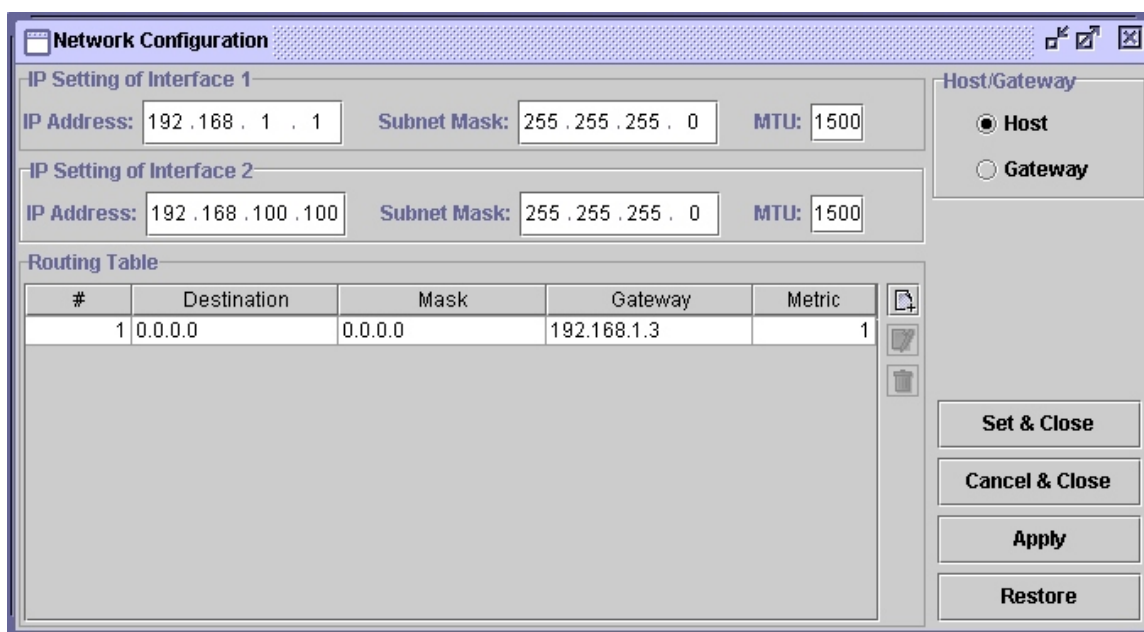


图 22.3

B. Transfer 实验:

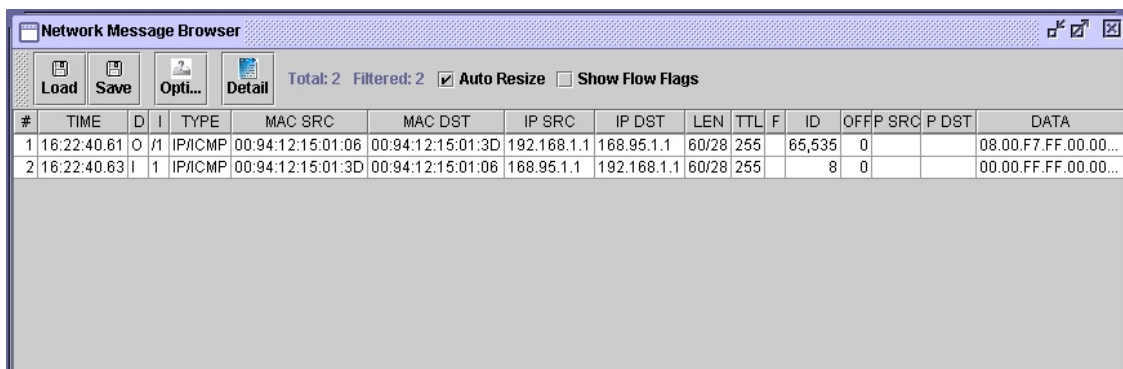
ITS 3 设置如下:

- 10) 打开网络封包浏览器 (Network Message Browser)。检查是否打开 **Listening**。
- 11) 打开 MDDL 编辑界面。
- 12) 点击 **Load** 按钮。调用 C:\XClient\Data\Mddl\Tutorial\Ex22\NATSetup.mddl, 然后点击 **Upld** 按钮。

ITS 1, 2, 4, 5 and 6 设置如下:

- 11) 打开网络封包浏览器 (Network Message Browser)。检查是否打开 **Listening**。
- 12) 根据实验 4 部分(见图 22.4), 由 ITS1 发送 ICMP Echo Request 封包至 ITS4。

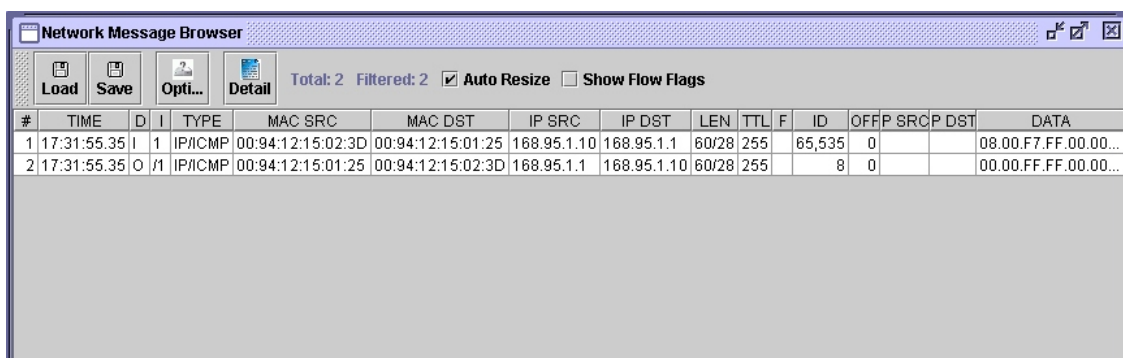
图 22.5 显示 ITS1 发送一个 ICMP Echo Request 封包给 ITS4 并且收到 ITS4 的 ICMP Echo Reply 封包。



#	TIME	D	I	TYPE	MAC SRC	MAC DST	IP SRC	IP DST	LEN	TTL	F	ID	OFFP	SRCP	P DST	DATA
1	16:22:40.61	O	/1	IP/ICMP	00:94:12:15:01:06	00:94:12:15:01:3D	192.168.1.1	168.95.1.1	60/28	255		65,535	0			08.00.F7.FF.00.00...
2	16:22:40.63	I	/1	IP/ICMP	00:94:12:15:01:3D	00:94:12:15:01:06	168.95.1.1	192.168.1.1	60/28	255		8	0			00.00.FF.FF.00.00...

图 22.4

图 22.4 显示 ITS4 接收到 ITS3 的 ICMP Echo Request 封包, 然后发送一个 ICMP Echo Reply 封包给 ITS3, 所以 ITS4 并不知道 ITS1 的 IP 地址。



#	TIME	D	I	TYPE	MAC SRC	MAC DST	IP SRC	IP DST	LEN	TTL	F	ID	OFFP	SRCP	P DST	DATA
1	17:31:55.35	I	/1	IP/ICMP	00:94:12:15:02:3D	00:94:12:15:01:25	168.95.1.10	168.95.1.1	60/28	255		65,535	0			08.00.F7.FF.00.00...
2	17:31:55.35	O	/1	IP/ICMP	00:94:12:15:01:25	00:94:12:15:02:3D	168.95.1.1	168.95.1.10	60/28	255		8	0			00.00.FF.FF.00.00...

图 22.5

图 22.6 显示 ITS3 (NAT) 在转送 IP 封包时转换了该封包的 IP 地址。

#	TIME	D	I	TYPE	MAC SRC	MAC DST	IP SRC	IP DST	LEN	TTL	F	ID	OFF	P SRC	P DST	DATA
1	16:22:50.10	I	1	IP/ICMP	00:94:12:15:01:06	00:94:12:15:01:3D	192.168.1.1	168.95.1.1	60/28	255		65,535	0			08.00.F7.FF.00.00...
2	16:22:50.11	O	2	IP/ICMP	00:94:12:15:02:3D	00:94:12:15:01:25	168.95.1.10	168.95.1.1	60/28	255		65,535	0			08.00.F7.FF.00.00...
3	16:22:50.11	I	2	IP/ICMP	00:94:12:15:01:25	00:94:12:15:02:3D	168.95.1.1	168.95.1.10	60/28	255		8	0			00.00.FF.FF.00.00...
4	16:22:50.12	O	1	IP/ICMP	00:94:12:15:01:3D	00:94:12:15:01:06	168.95.1.1	192.168.1.1	60/28	255		8	0			00.00.FF.FF.00.00...

图 22.6

四、实验讨论

- 1、试着从任一ITS 发送报文给其它ITS，观察其现象并讨论什么是「封包伪装(IP Masquerading)」？对网络来说会有什么影响？

提示：「封包伪装」是一种现象，如果从ITS 4、5和6的角度来看，不管是ITS1、2或3，发出来的封包，都会变成由NAT server发出。

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_ADDR SRC                = 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_ADDR DST            ;

        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_ADDR SRC,
        T[5,8]=S.IP_ADDR DST
    }
    SEND_OUT_IP WITH_DATA
    {
        T                                = S                                ,
        T.IP_ADDR SRC                    = VAR2[0,3]                        ,
        T.IP_LEN                         = LENGTH(T)                        ,
        T.IP_HEADERCHKSUM               = {0, 0}                            ,
        T.IP_HEADERCHKSUM               = CHECKSUM(T[0,19])
    }
}
}
ELSE // IP arriving
{
    LOOK_FOR_ONE_ELEMENT_IN_POOL 20 WITH_CONDITION
    (S.IP_PROT==PE[0]&&S.IP_ADDR SRC==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;

}
```