

Exp 15: TCPvsUDP

目的：用自行设计的流量控制规则来测试与比较TCP与UDP处理网络流量的效能。

摘要：以TCP通讯协议为基础，常见的应用程序如：TELNET、FTP。因为TCP提供的可信任的连接方式，所以应用程序不需考虑封包遗失、封包错误或封包排序的问题，但TCP不一定是某些应用程序的最佳协议，例如：需及时数据交换的影音传输应用程序，使用TCP协定反而会效果不佳。实验中透过ITS里的GUI接口工具FileTransfer，分别在有网络壅塞的状况下，传送同一个档案，比较TCP与UDP的特性。

时间：4.5 小时。

一、网络拓扑

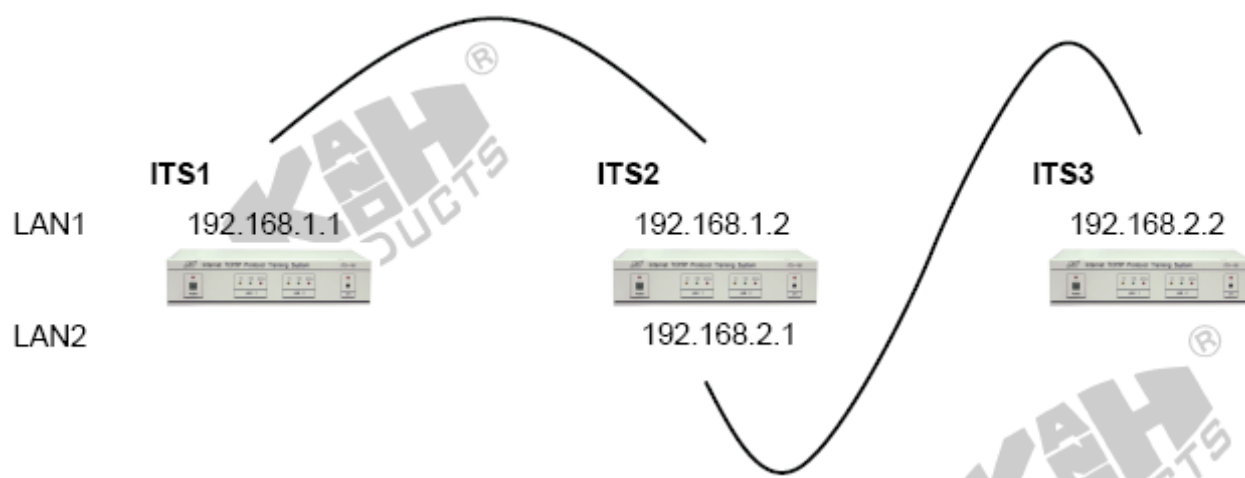


图 15.1

二、技术背景

在先前的实验里，我们都把焦点放在数据链路层(Data Link Layer)、网络层(Network Layer)及传输层(Transport Layer)，然而这些层级都是建构在应用层(Application Layer)之下。在一般的应用程序当中，我们称连结网络的能力为网络应用程序接口API，而且一般来说，我们只能选择TCP、UDP 或是两者一起。

以大多数的运用来说，TCP已经能够满足使用上的需求，也因此TCP大部份都被设计用在可以容易存取的API及完善的流量控制中。在使用TCP时，应用程序可以把焦点放在其主

要的任务上，不太需要去担心网络的传输是否发生问题、数据的正确性与否、数据的顺序是否正确、封包掉了是否会重送...等等一大堆问题。不过，在一些特殊领域的用途上，TCP就不见得是一个很有效率的选择，事实上TCP并不能满足一些应用程序的需求，像是实时的影音数据传输。像这些应用，通常会选择UDP来做这些特殊需求的数据传输。

为什么会有这样的结果呢？其实是因为两者之间对于数据传输有一些不同的地方。TCP以连接为导向(Connection Oriented)的协议，但UDP并不是，因此在一些需要传送和接收广播的应用上，UDP是比较好的选择，甚至可以说在这个状况下你只能选择UDP，在这里我们就不深入探讨了，只做一个简单的比较。

首先，TCP保证数据包能够安全的到达目的地，且顺序一定是对的。我们可以在实验14的TCP.mddl中去了解在数据传输时TCP如何保证IP datagram能够顺利地到达目的地，以及保证它的顺序是正确的。相较于TCP来说，UDP是一种非常简单的服务，它不保证封包的可靠度，同时也不保证中封包到达的顺序是否正确，但是从另外一个角度来看，就表示UDP比TCP来得有效率。因为UDP并没有重送机制(Retransmission)与封包确认(Acknowledgement)。对影像或是声音传输来说，封包是否能顺利传送到达，会比封包的顺序与内容是否正确还要来得重要许多。基本上，若是封包在传输的过程中掉了一些或是数据有部份错误，我们还可以接受，因为最多是画面不清晰而已，但是我们万一要是因为封包的顺序不对或是数据错了导致封包一直在重传，就将会造成通讯上的严重不良，可能连声音都没有，这就是为什么TCP无法满足于实时影音数据传输的原因。

我们可以另外举一个很好的例子，就是实时传输协议(RTP, Real-time Transport Protocol)，它的标准被定义在IETF RFC1889。RTP最常被使用在因特网上的多媒体传输。因为传输影像或是声音信息时，频宽是一个很重要的因素，所以封包传输时必须要有十分的有效率才行。

不过目前操作系统中往往只有建置TCP及UDP，使用者通常不能做其它的变动。如果使用者用TCP在因特网上传输多媒体信息，则表示使用者十分重视重送机制与封包确认。当然，也就表示使用者需要更大的频宽来处理资料的传送。

总而言之，TCP不利于多媒体信息的传输，所以只好使用UDP，但是记住，UDP并不保证数据可靠度，也无法确认封包到达的顺序无误。当然还有第三条路：就是两者都不选，自己开发自己的通讯协议！快试着用MDDL 发展一个吧。

三、实验步骤

1、拓扑结构

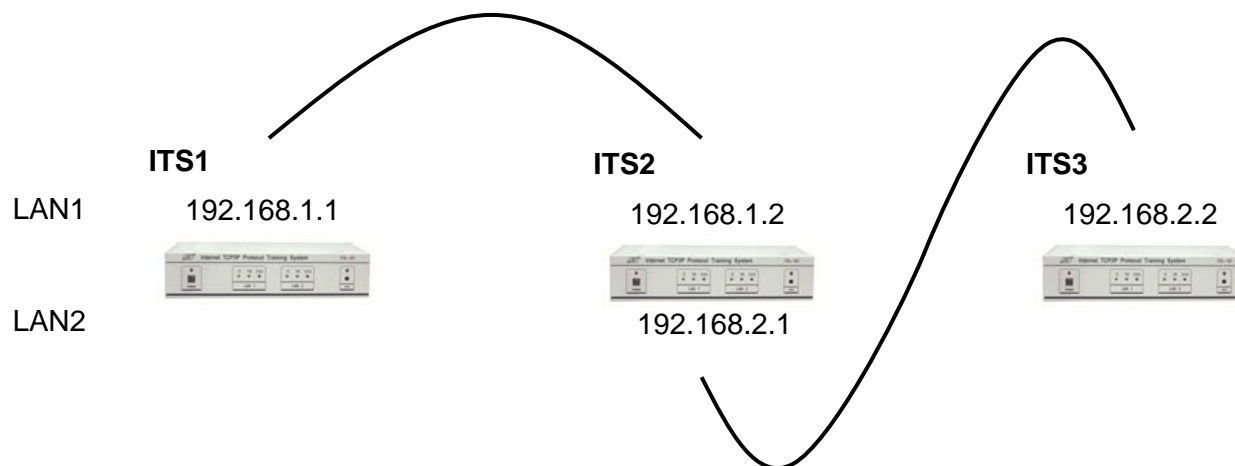


图 15.2

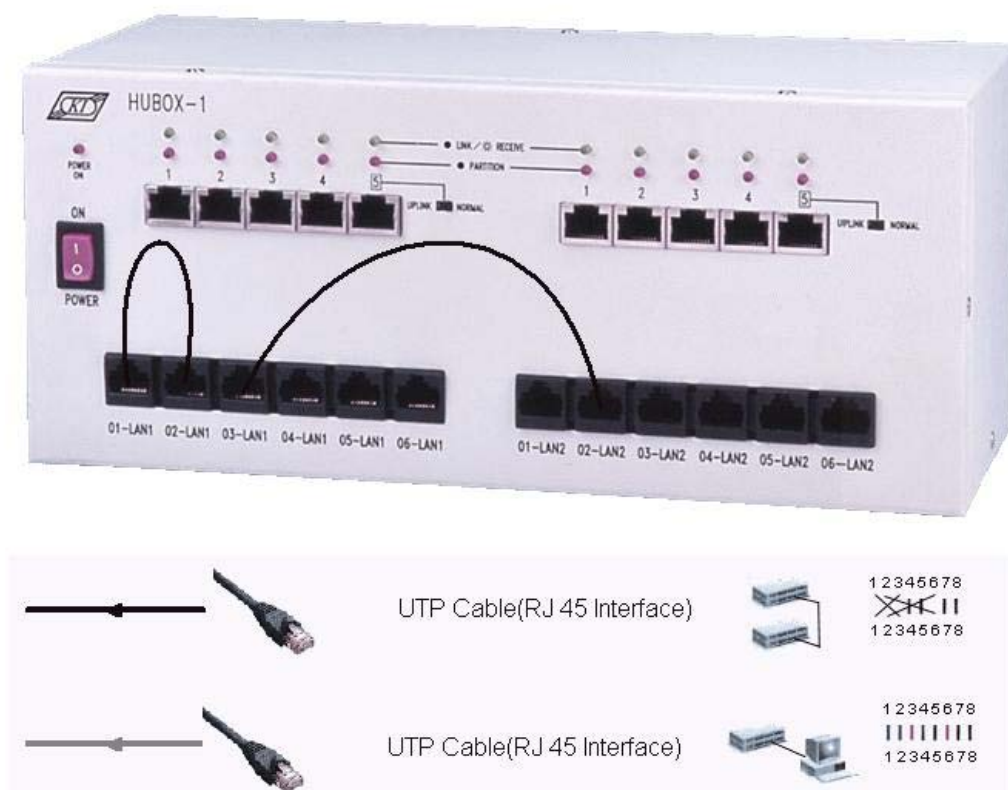


图 15.3

- 1) 执行 **XCLIENT.BAT**，打开 ITS 应用软件 KCodes Network Explorer。
- 2) 从主菜单打开 Network Configuration 设置界面。

ITS1 (Host)设置如下：

- 3) 根据拓扑结构。定义 Interface 1 的 IP 地址为 “**192.168.1.1**”，子网掩码设为 “**255.255.255.0**”MTU 设为“1500”。然后点击“**Add new routing entry**”按钮。(见图 15.4)
- 4) 定义 Destination 为 “**192.168.2.0**”，MASK 为 “**255.255.255.0**”，Gateway 为 “**192.168.1.2**”。最后点击 **Update** 按钮。(见图 15.5)
- 5) 模式选择“**Host**”，之后点击“**Set & Close**”按钮。

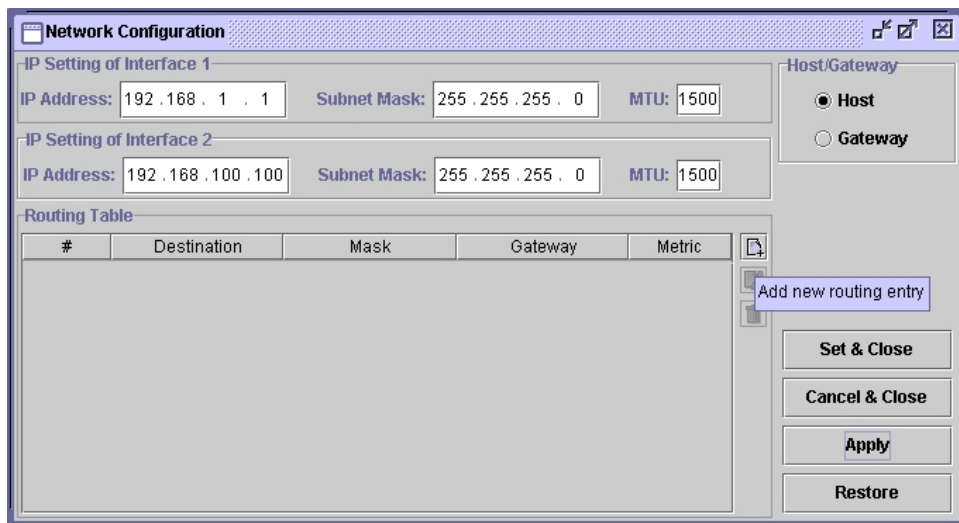


图 15.4

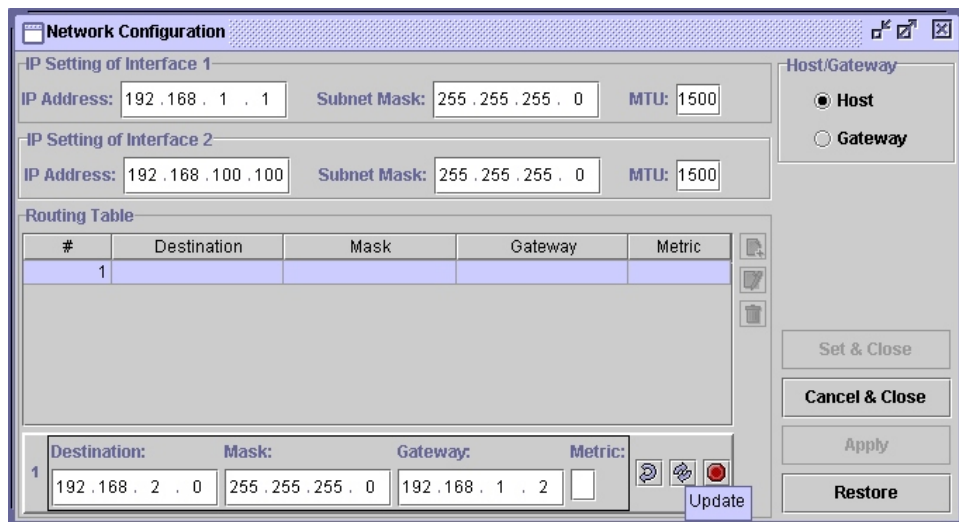


图 15.5

ITS3 (Host)设置如下:

- 6) 根据拓扑结构。定义 Interface 1 的 IP 地址为 “**192.168.2.2**”，子网掩码设为 “255.255.255.0” MTU 设为 “1500”。然后点击 “**Add new routing entry**” 按钮。
- 7) 定义 Destination 为 “**192.168.1.0**”，MASK 为 “**255.255.255.0**”，Gateway 为 “**192.168.2.1**”。最后点击 **Update** 按钮。
- 9) 模式选择 “**Host**”，之后点击 “**Set & Close**” 按钮。

ITS2 (Gateway) 设置如下:

- 10) 根据拓扑结构。定义 Interface 1 的 IP 地址为 “**192.168.1.2**”，并且定义 Interface 2 的 IP 地址为 “**192.168.2.1**”。(见图 15.6)
- 11) 模式选择 “**Gateway**” 之后点击 “**Set & Close**” 按钮。现在，我们已经设置好了路由表，下面可以开始实验。

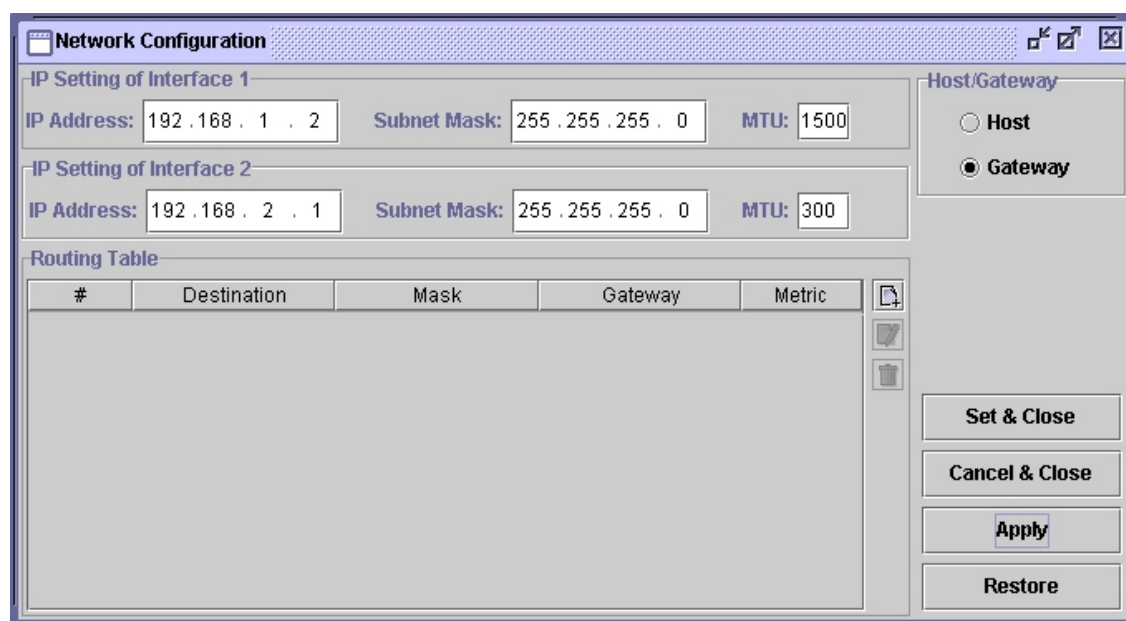


图 15.6

2、TCP 文件传输实验

ITS2 操作如下:

- 12) 打开网络封包浏览器 (Network Message Browser)。打开监听。
- 13) 打开 MDDL 编辑平台 **MDDL Reactor Panel**，在主菜单的 Reactor 菜单下。
- 14) 点击 **Load** 按钮。调用程序 C: \XClient \Data \Mddl \Tutorial \Ex15 \PktDelay15.mddl，最后点击 **Upld** 按钮。

ITS1 操作如下：

- 15) 在 Application 菜单下打开“**File Transfer**”对话框。
- 16) 选择“**System Default TCP**”, Source IP 输入 “**192.168.1.1**”, 并且 Source Port 选择 **HTTP (80)**t, 最后点击“**Listen**”按钮。(见图 15.7)

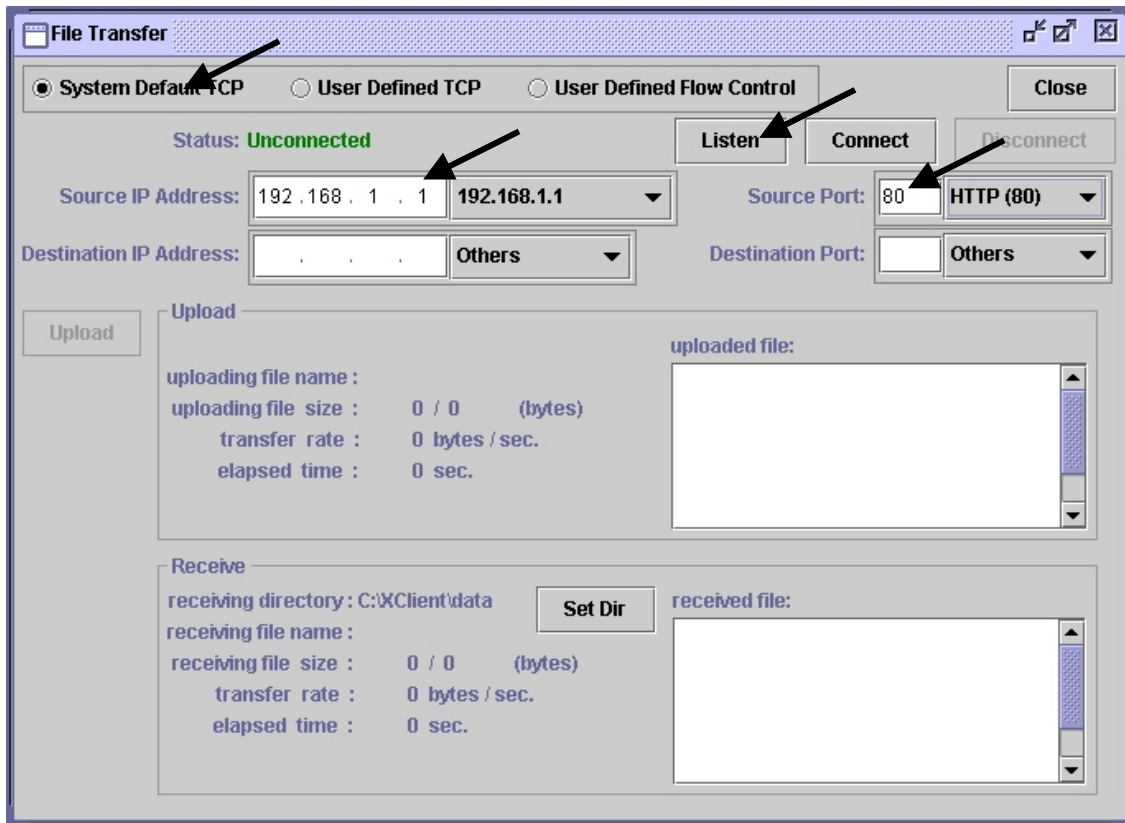


图 15.7

ITS3 操作如下：

- 17) 在 Application 菜单下打开“**File Transfer**”对话框。
- 18) 选择“**System Default TCP**”, Destination IP 输入 “**192.168.1.1**”, 并且 Destination Port.选择 **HTTP (80)** , 最后点击“**Connect**”按钮。
- 19) 在连接成功后,点击 **Upload** 按钮,并且打开范例文件 C:\XClient \Data \9.70.ini (10 KB in size), 见图 15.8 ITS3 将会发送这个文件给 ITS1。我们可以观察到传输的速率和总共传输的时间。

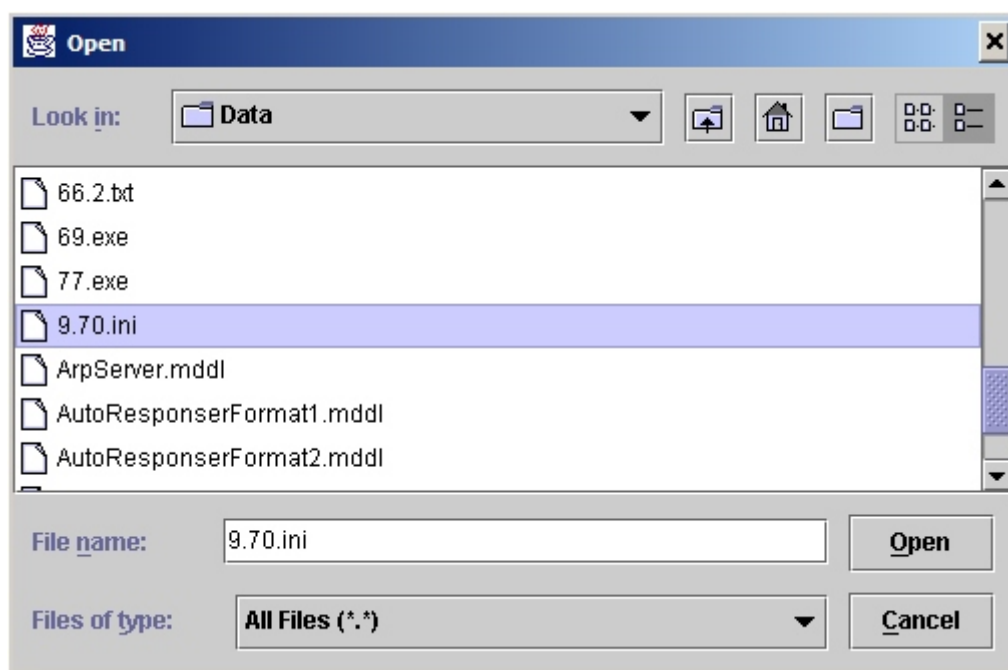


图 15.8

3、UDP 文件传输实验

ITS2 操作如下：

- 20) 打开网络封包浏览器（Network Message Browser）。 打开监听。
- 21) 打开 MDDL 编辑平台 **MDDL Reactor Panel**，在主菜单的 **Reactor** 菜单下。
- 22) 点击 **Load** 按钮。调用程序 C:\XClient\Data\Mddl\Tutorial\Ex15\PktDelay15.mddl, 最后点击 **Upld** 按钮。

ITS1 and ITS3 操作如下：

- 23) 点击“**Load**”按钮。打开文件 C:\XClient\Data\Mddl\Tutorial\Ex15\UFC.mddl, 然后点击“**Upld**”按钮。
- 24) 在 Application 菜单下打开“**File Transfer**”对话框., 见图 15.9。

ITS1 操作如下：

- 25) 选择“**User Defined Flow Control**”， Source IP 输入“**192.168.1.1**”， Source Port 输入 **HTTP (80)** 然后点击 **Listen** 按钮。



图 15.9 File Transfer dialog box

ITS3 操作如下:

- 26) 选择“**User Defined Flow Control**”, Destination IP 输入 “**192.168.1.1**”, Destination Port 输入 **HTTP (80)**。最后点击 **Connect** 按钮。
- 27) 在连接成功后,点击 **Upload** 按钮,并且打开范例文件 C:\XClient \Data \9.70.ini (10 KB in size)。ITS3 将会发送这个文件给 ITS1。我们可以比较 TCP 和 UDP 传输的速率和总共传输的时间。

四、实验讨论

- 1、试着在ITS 2没有载入PktDelay15.mddl的情况下再重新做上两段实验, 在整个传输时间上, UDP是否有比TCP快?
- 2、再试着将PktDelay15.mddl 所设的延迟时间分别改成0.2sec、0.5sec、1sec、1.5sec、2sec、3sec 和5sec, 比较一下传同一个档案时, TCP与UDP各自会花多少时间, 并讨论其差异。

REACTOR PROGRAMS**1、PktDelay15.mddl**

```

VAR1[0] = 0 ;

IP_ROUTING_HANDLER
{
    IF(ISMYIPADDR(S.IP_ADDRDST))
        RETURN;

    IF((S.IP_ADDRDST.[0,2] != MYIPADDR(2).[0,2]) && (S.IP_ADDRDST.[0,2] != MYIPADDR(1).[0,2]))
        //compare with netmask
        255.255.255.0
        RETURN;

    VAR1[0, 3] = RANDOM(10000L);

    IF(VAR1[0, 3] < 5000L)
    {
        IF(S.IP_TTL > 1)
        {
            SEND_OUT_IP WITH_DATA
            {
                T                = S                ,
                T.IP_TTL          = S.IP_TTL - 1
                T.IP_HEADERCHKSUM = {0, 0}            ,
                T.IP_HEADERCHKSUM = CHECKSUM(T.IP_HEADER)
            }
        }
    }

    else if(VAR1[0, 3] < 10000L)
    {
        ADD_TO_POOL 22 WITH_LIFETIME 20000 WITH_DATA
        {
            T.[0] = 10 ,
            T.[6, ] = SOURCE
        }
    }
}

```

10 x 100 ms = 1000 ms = 1 sec

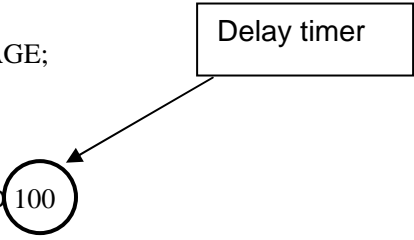
T.[0] = 10

```

    }
}
DISCARD_MESSAGE;
}

TIMER_WITH_PERIOD 100
{
    FOR_EVERY_ELEMENT_IN_POOL 22
    {
        PE[0] = PE[0] - 1;
        IF(PE[0] == 0)
        {
            SEND_OUT_IP WITH_DATA
            {
                TARGET          = PE[6, ],
                T.IP_TTL         = PE.IP_TTL - 1,
                T.IP_LEN         = LENGTH(T),
                T.IP_HEADERCHKSUM = {0, 0} ,
                T.IP_HEADERCHKSUM = CHECKSUM(T.IP_HEADER)
            }
            REMOVE_CURRENT_POOL_ELEMENT;
        }
    }
}

```



2、UFC.mddl

```

VAR6.UCB_SOCKET_ID      = 5          ;
VAR6.UCB_STATE          = CNST_UFC_STATE_CLOSED ;

SERVICE_UFC_OPEN
{
    IF( VAR6.UCB_STATE != CNST_UFC_STATE_CLOSED )
        RETURN;

    VAR6.UCB_STATE          = CNST_UFC_STATE_ESTABLISHED;

```

```

IF(PARA_IPADDR_SRC() == CNST_IP_ADDR_BROADCAST)
    VAR6.UCB_IP_ADDRSRC = MYIP(1);
ELSE
    VAR6.UCB_IP_ADDRSRC = PARA_IPADDR_SRC();

VAR6.UCB_IP_ADDRDST                = PARA_IPADDR_DST()      ;

IF(PARA_PORT_SRC() == 0W)
    VAR6.UCB_PORTSRC                = 49152W;
ELSE
    VAR6.UCB_PORTSRC                = PARA_PORT_SRC()        ;

VAR6.UCB_PORTDST                   = PARA_PORT_DST()        ;

RETVAL_SOCKET_ID    = VAR6.UCB_SOCKET_ID;
RETVAL_IPADDR_SRC   = VAR6.UCB_IP_ADDRSRC;
RETVAL_PORT_SRC     = VAR6.UCB_PORTSRC;
RETVAL_ERRORCODE    = CNST_UFC_NO_ERROR;
}

SERVICE_UFC_LISTEN
{
    IF( VAR6.UCB_STATE != CNST_UFC_STATE_CLOSED )
        RETURN;

    VAR6.UCB_STATE          = CNST_UFC_STATE_LISTEN ;
    VAR6.UCB_IP_ADDRSRC     = PARA_IPADDR_SRC()      ;
    VAR6.UCB_IP_ADDRDST     = PARA_IPADDR_DST()      ;
    VAR6.UCB_PORTSRC        = PARA_PORT_SRC()        ;
    VAR6.UCB_PORTDST        = PARA_PORT_DST()        ;

    WAIT_SIGNAL VAR6.UCB_SOCKET_ID;

    RETVAL_SOCKET_ID    = VAR6.UCB_SOCKET_ID;
    RETVAL_IPADDR_SRC   = VAR6.UCB_IP_ADDRSRC;
    RETVAL_IPADDR_DST   = VAR6.UCB_IP_ADDRDST;

```

```
    RETVAL_PORT_DST      = VAR6.UCB_PORTDST;
    RETVAL_ERRORCODE     = CNST_UFC_NO_ERROR;
}

SERVICE_UFC_CLOSE
{
    IF( VAR6.UCB_STATE != CNST_UFC_STATE_ESTABLISHED )
        RETURN;

    VAR6.UCB_STATE                = CNST_UFC_STATE_CLOSED;

    GENERATE_DISCONNECTED(1, VAR6.UCB_SOCKET_ID);
}

SERVICE_UFC_SEND
{
    VAR7.UFC_PSEUDO_IP_ADDR SRC      = VAR6.UCB_IP_ADDR SRC      ;
    VAR7.UFC_PSEUDO_IP_ADDR DST     = VAR6.UCB_IP_ADDR DST     ;
    VAR7.UFC_PSEUDO_ZERO              = 0                        ;
    VAR7.UFC_PSEUDO_PROT              = CNST_IP_PROT_UFC        ;
    VAR7.UFC_PSEUDO_LEN               = 8W                      ;

    VAR7.UFC_PSEUDO_DATA.UFC_PORTSRC = VAR6.UCB_PORTSRC        ;
    VAR7.UFC_PSEUDO_DATA.UFC_PORTDST = VAR6.UCB_PORTDST        ;
    VAR7.UFC_PSEUDO_DATA.UFC_LEN      = PARA_SOCKET_BUFFER_LEN() + 8;
    VAR7.UFC_PSEUDO_DATA.UFC_CHKSUM   = 0W                      ;

    VAR7.UFC_PSEUDO_DATA.UFC_DATA     = PARA_DATA()             ;

    VAR7.UFC_PSEUDO_DATA.UFC_CHKSUM   = CHECKSUM(VAR7[0, 12 +
                                                VAR7.UFC_PSEUDO_LEN +
                                                PARA_SOCKET_BUFFER_LEN() -
                                                1 ])    ;

    SEND_OUT_IP WITH_DATA
    {
```

```

        T.IP_PROT                = CNST_IP_PROT_UFC                ,
        T.IP_ADDRDST             = VAR6.UCB_IP_ADDRDST             ,
        T.IP_DATA                = VAR7.UFC_PSEUDO_DATA.[0,
                                                                    VAR7.UFC_PSEUDO_LEN +
                                                                    PARA_SOCKET_BUFFER_LEN() - 1]
    }

    RETVAL_DATA_LEN = PARA_SOCKET_BUFFER_LEN();
    RETVAL_DATA = PARA_DATA();
}

SERVICE_UFC_RECEIVE
{
    GENERATE_USER_MSG WITH_DATA
    {
        TARGET = "UFC_RECEIVE"
    }
    RETVAL_DATA_LEN = 0;
    FOR_EVERY_ELEMENT_IN_POOL 30
    {
        RETVAL_DATA[ RETVAL_DATA_LEN, ] = PE[0, ];
        RETVAL_DATA_LEN += LENGTH(PE);
        REMOVE_CURRENT_POOL_ELEMENT;
    }
}

IP_IN_HANDLER
{
    IF(S.IP_PROT!=CNST_IP_PROT_UFC)
        RETURN;

    DISCARD_MESSAGE;

    IF( VAR6.UCB_STATE == CNST_UFC_STATE_ESTABLISHED &&
        VAR6.UCB_IP_ADDRSRC == S.IP_ADDRDST && VAR6.UCB_IP_ADDRDST ==

```

```
S.IP_ADDRSRC && VAR6.UCB_PORTSRC == S.IP_DATA.UFC_PORTDST &&
VAR6.UCB_PORTDST == S.IP_DATA.UFC_PORTSRC )
{
    ADD_TO_POOL 30 WITH_DATA
    {
        TARGET = S.IP_DATA.UFC_DATA
    }

    GENERATE_SEND_BUFFER_PARAMETERS_CHANGED(VAR6.UCB_SOCKET_ID, 0,
    0, 0, 0);
}
ELSE IF( VAR6.UCB_STATE == CNST_UFC_STATE_LISTEN &&
( VAR6.UCB_IP_ADDRSRC == CNST_IP_ADDR_BROADCAST ||
VAR6.UCB_IP_ADDRSRC == S.IP_ADDRDST ) && ( VAR6.UCB_PORTSRC ==
S.IP_DATA.UFC_PORTDST ) )
{
    IF(VAR6.UCB_IP_ADDRSRC == CNST_IP_ADDR_BROADCAST)
        VAR6.UCB_IP_ADDRSRC = S.IP_ADDRDST;

    VAR6.UCB_IP_ADDRDST = S.IP_ADDRSRC;

    VAR6.UCB_PORTDST = S.IP_DATA.UFC_PORTSRC;

    VAR6.UCB_STATE = CNST_UFC_STATE_ESTABLISHED;

    WAKEUP_SIGNAL VAR6.UCB_SOCKET_ID;

    ADD_TO_POOL 30 WITH_DATA
    {
        TARGET = S.IP_DATA.UFC_DATA
    }

    GENERATE_SEND_BUFFER_PARAMETERS_CHANGED(VAR6.UCB_SOCKET_ID, 0,
    0, 0, 0);
}
}
```