IEG4180 Project 2
NetProbe Documentation

GUAN Hao
05569511
hguan5@ie.cuhk.edu.hk

March 1, 2008

1 Introduction

In this project, I write both a server and a client for the NetProbe.

The NetProbe Server will listen for incoming connections roiginating from a NetProbe Client, receiving the operating parameters and transmit packets. It is a concurrent server which is implemented using select-base I/O multiplexing.

The NetProbe Client is based on the last version in Project 1 and Message-Driven I/O mode is added in this version. User can choose either *TCP* or *UDP* protocol and either *Blocking* or *Message-Driven* I/O mode to receive the packets. The same as the last version, operating parameters can be set can the transmission statistics will be updated according to the sepecified refresh interval.

2 Program Structure

The program is a MFC application which is designed with the Object-Oriented concept. Following is the list of classes implemented in this project:

2.1 NetProbe Server

There is only one class in the server side.

- `class NetProbeServer`, the main class of the NetProbe Server. This class will initialize the TCP and UDP sockets, use `select()` to detect the protocol of the incomming connection and start new threads to send packets. Following is the prototype of this class.

```
1 class NetProbeServer{
  private:
3         struct sockaddr_in *TCP_Addr;
          struct sockaddr_in *UDP_Addr;
```

```
5           struct sockaddr_in *createSockAddr(char *host, int port
                );
            SOCKET tcpfd, udpfd;
7
public:
9           NetProbeServer(const char *tcp_h, int tcp_p, const char
                *udp_h, int udp_p);
            int TCPReady(void);
11          int UDPReady(void);
            int detectProtocol(void);
13          static DWORD WINAPI threadTCPSend(LPVOID lpInstance);
            static DWORD WINAPI threadUDPSend(LPVOID lpInstance);
15  };
```

2.2 NetProbe Client

- `class CNetProbeClinetApp`, the main class of this program. This class will create a instance of `class CNetProbeDlg` and show the dialog. Some `winsock` initialize functions are also invoked here.
- `class CNetProbeClientDlg`, this class control the main dialog of the application.
- `class NetProbe`, all the socket operation is implemented in this class. Following shows the prototype of `class NetProbe`:

```
1   class NetProbe
    {
3   private:
            struct sockaddr_in *Server_Addr;
5           struct sockaddr_in *createSockAddr(char *host, int port
                );
            SOCKET Sockfd;
7           CNetProbeClientDlg *theDlg;
            double bytesTransferred;
9           int packetsTransferred;
            int maxPacketNum;
11          ES_FlashTimer timer;
            int status;
13          int PacketSize;
            int SendingRate;
15          int NumPackets;

17
    public:
19          NetProbe(CNetProbeClientDlg *dialog, char *host, int
                port);
            CWinThread *wThread;
21          void stop();
            static DWORD WINAPI threadUpdateUI(LPVOID lpInstance);
23          int TCPConnect(LPVOID lpInstance);
```

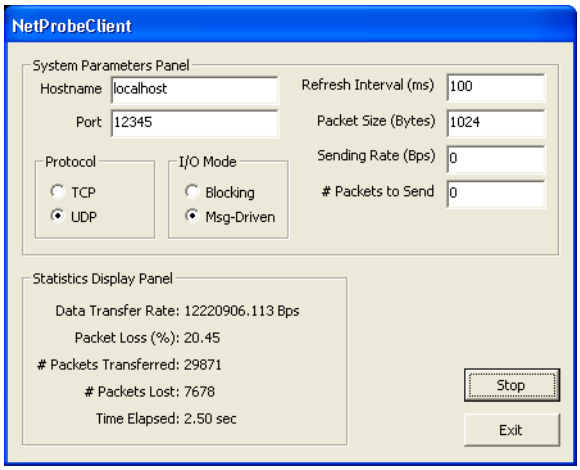

Figure 1: Statistics shown on the dialog.

```
        static DWORD WINAPI threadTCPReceive(LPVOID lpInstance)
            ;
25      int UDPConnect(LPVOID lpInstance);
        static DWORD WINAPI threadUDPReceive(LPVOID lpInstance)
            ;
27      int MsgDrivenReady();
        void OnRead();
29      void OnClose();

31  };
```

3 GUI Design

There is no GUI for the NetProbe Server and for NetProbe Client a similar GUI with the last version is used. The main dialog is designed as in Figure 1. User can input parameters and choose protocol on this dialog.After *Send* or *Receive* button is clicked, the caption of the button will turn to *Stop*. During transmission, the statistics will be shown on the bottom of the dialog.