**Scanner de porturi pentru sistemele dintr-o retea**

**Scopul :**

**1. realizarea securitatii retelelor cu o aplicatie care are o arhitectura client server si care utilizeaza socluri de windows**

**2. crearea de servicii Windows.**

**1. Descrierea generala a aplicaţiei**

Aplicatia propriu – zisa este alcatuita din 3 module. Modulul principal este reprezentat de un Server care va fi instalat pe unul din sistemele din LAN.Prin conectarea la acest server al doilea modul denumit AdminTool va putea afisa informatiile cerute pentru toti clientii din retea. Clienti sunt denumiti toti utilizatorii care au instalat cel de-al treilea modul, modulul de Client. Acest modul poate fi instalat pe oricate sisteme din retea si la pornirea lui se creeaza o conexiune cu Serverul, conexiune prin care acesta poate obtine toate informatiile necesare.

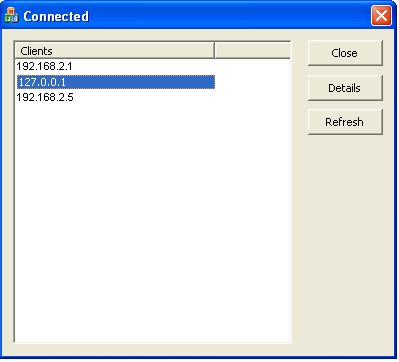
Modulele Client si Server vor rula ca servicii de Windows, numai modulul de Admin avand o interfata grafica (GUI – Graphical User Interface). Acest modul este dezvoltat pe arhitectura claselor MFC(Microsoft Foundation Classes). Această arhitectura permite dezvoltarea cu uşurinţă a aplicaţiilor tip MDI(Multiple Document Interface) dar şi accesul la tehnologiile Microsoft. Alegerea acestei librării este motivată şi de documentaţia existenta pe site-ului Microsoft legată de structura Windows-ului şi a claselor MFC.

O data pornit serverul administatorul de retea poate porni aplicatia Admin, deschizandu-i-se initial un dialog ce ii permite introducerea IP-ului sistemului pe care se afla Serverul.



**Fig 1. Dialog Conectare**

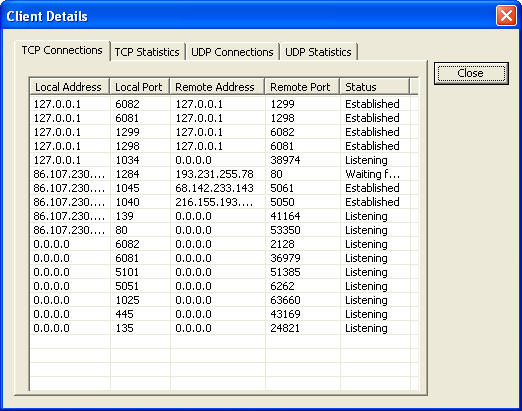
Furnizand un IP corect si prin apasarea butonului Connect modulul de Admin se conecteaza la Server si primeste toti Clientii care sunt deja conectati.



**Fig 2. Dialog principal**

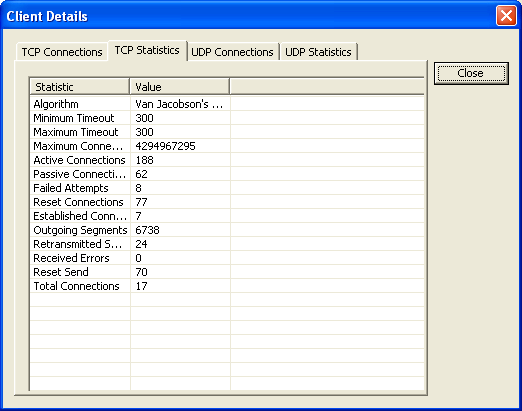
In continuare administratorul are posibilitatea selectarii unuia din clienti si prin apasarea butonului “Details” poate vizualiza intr-un nou dialog ce contine patru tab-uri conexiunile TCP, conexiunile UDP, precum si statisticile acestor protocoale.

Astfel primul tab va afisa, intr-o lista impartita in cinci coloane, conexiunile TCP deschise cu adresa locala, portul local, adresa remote, portul remote, si statusul conexiunii.



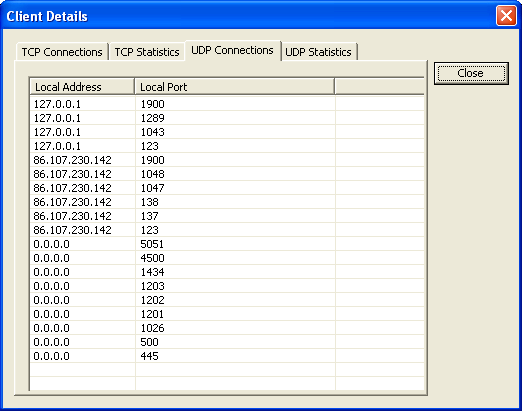
**Fig 3. Tab Conexiuni TCP**

In cel de-al doilea tab sunt afisate statisticile TCP, printre care algoritmul utilizat, timeout-urile minime si maxime sau pachetele primite si trimise.Asupra acestui aspect voi insista la prezentarea modului dechizitionare al datelor.



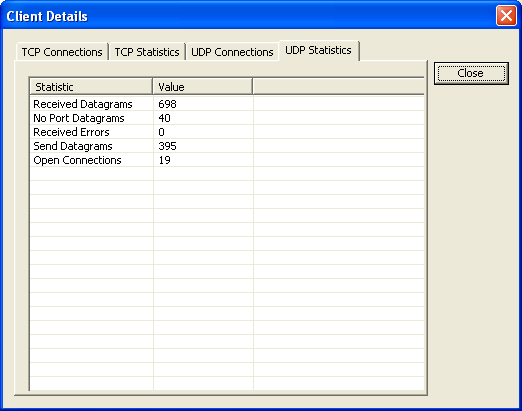
**Fig 4. Tab Statistici TCP**

Cel de-al treilea tab va contine, intr-o lista impartita pe doua coloane, conexiunile UDP, cu adresa locala, si portul local, UDP fiind un protocol orientat mesaj si nu orientat conexiune ca TCP.



**Fig 5. Tab Conexiuni UDP**

Ultimul tab contine statisticile UDP, printre care datagrame primite, datagrame trimise sau numarul conexiunilor deschise.



**Fig 6. Tab Statistici UDP**

Acest dialog poate fi inchis prin apasarea butonului Close oferind astfel posibilitatea administratorului de a alege un nou client pentru a-I vizualiz detaliile.

In dialogul principal prin apasarea butonului “Refresh” lista de clienti este golita si se face o cerere noua catre server de listare a clientilor conectati.

**2. Instalarea modulelor**

Modulul Admin este o aplicatie MFC bazata pe un dialog si nu are nevoie de o instalare prealabila fiind suficienta doar executarea, astfel ca voi insista numai pe instalarea Serverului si a Clientilor.

Pentru instalarea unui client este nevoie de introducerea unei noi chei in registrii cu IP-ul masinii pe care se afla serverul.Astfel se ofera posibilitatea ca acest client sa se poata conecta la alt server numai prin modificarea acestei chei. Cheia poate fi adaugata cu usurinta prin executarea unui fisier de tip .reg care contine liniile urmatoare:

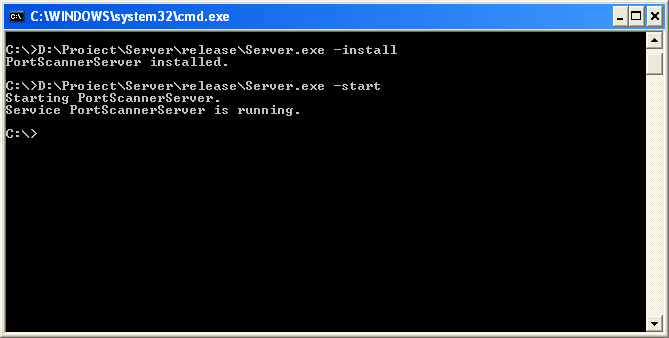
[HKEY\_LOCAL\_MACHINE\SOFTWARE\PortScannerClient]

[HKEY\_LOCAL\_MACHINE\SOFTWARE\PortScannerClient]

"Server"="192.168.2.1"

In urma executarii se va adauga cheia PortScannerClient ce va contine o valoare de tip string cu numele “server” si cu valoarea egala cu IP-ul Serverului.

In continuare instalarea Serverului si a Clientilor este foarte asemanatoare, pentru ca se vor instala ambele ca servicii si vor trebui pornite.Acest lucru se realizeaza prin rularea din Command Prompt a executabilului pe rand cu parametrii –install si –start.



**Fig 7. Instalarea modulelor**

**3. Comunicarea intre module**

Pentru a realiza comunicarea intre cele trei module ale aplicatiei, am implementat o clasa wrapper peste un handle de socket – SOCKET, suprascriind toate metodele necesare cum ar fi *Create, Bind, Connect, Listen, Send* sau *Receive.*Prin aceasta suprascriere nu am facut altceva decat sa apelez functiile de WINSOCK API. Se poate observa aces lucru de exemplu in metoda *Connect*

BOOL CAsyncSock::Connect( LPCTSTR lpszHostAddress, UINT nHostPort )

{

ASSERT( m\_hSocket != 0);

ASSERT(lpszHostAddress != NULL);

SOCKADDR\_IN sockAddr;

memset(&sockAddr,0,sizeof(sockAddr));

LPSTR lpszAscii = T2A((LPTSTR)lpszHostAddress);

sockAddr.sin\_family = AF\_INET;

sockAddr.sin\_addr.s\_addr = inet\_addr(lpszAscii);

if (sockAddr.sin\_addr.s\_addr == INADDR\_NONE)

{

LPHOSTENT lphost;

lphost = gethostbyname(lpszAscii);

if (lphost != NULL)

sockAddr.sin\_addr.s\_addr = ((LPIN\_ADDR)lphost->h\_addr)->s\_addr;

else

{

WSASetLastError(WSAEINVAL);

return FALSE;

}

}

sockAddr.sin\_port = htons((u\_short)nHostPort);

return ( SOCKET\_ERROR != connect(m\_hSocket, (SOCKADDR\*) &sockAddr,

sizeof(sockAddr) ) );

}

Fiecare din cele trei module contine cate o fereastra, vizibila numai la modulul de Admin, fereastra utilizata in prinderea si tratarea mesajelor socketilor. Astfel am evitat utilizarea socketilor blocanti, metoda ce ar fi ingreunat considerabil functionarea aplicatiei. Asignarea ferestrei drept destinatarul mesajelor socketilor se realizeaza prin apelarea functiei *WSAAsyncSelect*, suprascrisa drept AsyncSelect.

Antetul acestei functii este:

int WSAAsyncSelect(SOCKET *s*, HWND *hWnd*, unsigned int *wMsg*, long *lEvent*);

Parametrii:

s- handle al socketului ce urmeaza sa trimita mesajele catre fereastra

hWnd - handle-ul ferestrei destinatar al mesajelor.

wMsg – ID al mesajului ce urmeaza sa fie primit in urma primirii mesajului

de la socket.

lEvent – eventuri ale socketului ce va genera mesajul de mai sus.

Aceste eventuri pot fi spre exemplu FD\_READ, ce anunta ca socketul este gata sa citeasca date, sau FD\_CLOSE care anunta inchiderea socketului asociat.

Pentru modulul de Admin fereastra care primeste aceste mesaje este chiar fereastra principala a aplicatiei. La celelalte doua module a fost necesara crearea, in momentul pornirii serviciului, a unui nou thread pe care se construieste o noua ferestra invizibila. Acest thread se va opri in momentul opririi serviciului.

Astfel in momentul crearii ferestrei Serverului, se creaza si un socket de ascultare pentru clienti si unul de ascultare pentru Admin. In momentul in care un client sau un Admin va incerca conectarea la server se va crea un nou socket care se va lega cu cel ce a incercat conectarea si va fi retinut intr-o lista.

pSocketInfo = new CAsyncSock();

SOCKADDR\_IN sockAddr;

int sockAddrLen = sizeof(SOCKADDR\_IN);

if ( m\_pClientListening->Accept(pSocketInfo,(SOCKADDR\*)&sockAddr,

&sockAddrLen ) )

{

m\_ClientSocketList.AddTail(pSocketInfo);

pSocketInfo->AsyncSelect(WM\_SOCKET\_MESSAGE\_CLIENT, m\_hWnd,

FD\_READ | FD\_CLOSE);

}

O data stabilita legatura intre Server si Clienti sau intre Server si Admin comunicarea propriu zisa se realizeaza prin comenzi trimise in bufferul functiilor *Send* sub forma de siruri de caractere cu urmatoarea structura:

Lungime\_comandaCOMMAND=NUME\_COMANDA&PARAM1=val1&PARAM2=val2…

Unde:

Lungime\_comanda – lungimea sirului de caractere reprezentat pe 4 caractere

COMMAND – un identificator predefinit

NUME\_COMANDA – poate avea o serie de valori printre care

LISTUSERS – cere serverului trimiterea tuturor clientilor

GETDETAILS - cere serverului trimiterea detaliilor pentru

un client

“&”, “=” – separator predefinit

PARAM1 – numele primului parametru, cum ar fi numele clientului

pentru care se cer detaliile

val1 – valoare primului parametru.

La notificarea unui socket ca este gata sa primeasca un buffer, se vor citi primele patru caractere ce reprezinta lungimea comenzii primite, apoi se va citi un buffer de lungime egala cu valoarea citita anterior. Acest buffer este apoi parsat in vederea obtinerii numelui comenzii a parametrilor aferenti.

while(TRUE)

{

int bytesRecv = SOCKET\_ERROR;

char recvbuf[MAX\_BUFFER] = "";

bytesRecv = m\_pSocket->Receive(recvbuf, COMMAND\_SIZE, 0 );

if(bytesRecv == SOCKET\_ERROR)

break;

int bytesToRead = \_ttoi(recvbuf);

bytesRecv = m\_pSocket->Receive(recvbuf, bytesToRead, 0 );

ParseReceivedBuffer(CString(recvbuf));

}

Parsarea comenzii primite se realizeaza prin cautarea separatorilor predefiniti “&” si “=” precum si a identificatorului COMMAND. Pentru aceasta am utilizat o clasa ajutatoare CcommandTokenizer intr-o functie ce primeste ca parametri bufferul de parsat si o mapa de siruri de caractere in care cheie va fi numele parametrului si valoare chiar valoarea acestuia.

void CAtlSocketWindow::ParseTagValue(CString& strBuffer,

MapStringToString\* mapCommand)

{

CCommandTokenizer next(strBuffer);

CString token;

while ( !(token = next('&')).IsEmpty() )

{

CCommandTokenizer next(token);

CString name = next('=');

CString value = next('=');

mapCommand->SetAt(name, value);

}

}

**4. Achizitionarea datelor necesare la client**

O data rezolvata problema conectarii modulelor, a stabilirii unui protocol de comunciare si a unei formatari a comenzilor a mai ramas de implementat acizitionarea datelor necesare la nivel de client, adica listarea conexiunilor deschise si a statisticilor TCP si UDP.

Pentru aceasta am utilizat o serie de functii ce permit obtinerea informatiilor despre parametrii de retea ce se gasesc in Internet Protocol Helper (IP Helper Api). Acest Api face posibila obtinerea si modificarea parametrilor de retea dintr-un sistem. Functiile pot fi folosite in programarea C/C++, aplicatiile tipice fiind protocoale de rutare IP si agenti SNMP (Simple Network Management Protocol).De asemenea ,IP Helper contine mecanisme de notificare ce pot fi folosite pentru a anunta ca anumiti parametrii de retea s-au modificat.

IP Helper face posibil accesarea informatiilor legate de protocoalele de retea ce sunt folosite de catre Windows.

Astfel pentru a obtine conexiunile TCP ale unui sistem se utilizeaza functia *GetTcpTable* cu antetul.

DWORD GetTcpTable(PMIB\_TCPTABLE *pTcpTable*,DWORD *pdwSize*, BOOL *bOrder*)

Parametrii:

*pTcpTable [out ] –* Pointer catre un buffer ce primeste tabela de conexiuni TCP

ca pe o structura de tip MIB\_TCPTABLE (**MIB**

**=**Management Information Base)

*pdwSize* [in, out] – Pe intrare,specifica marimea buffer-ului pointat de catre

parametrul  *pTcpTable*

-Pe iesire ,daca bufferul nu este destul de mare pentru a

contine toata tabela de conexiuni,functia seteaza acest

parametru astfel incat sa incapa toata tabela

*bOrder [in] –* Specifica daca tabela va fi sau nu sortata .Daca acest parametru

este True, tabela este sortata dupa : Local IP address, Local Port, Remote IP address, remote port.

Valoarea returnata este NO\_ERROR daca nu a aparut nici o eroare, in caz contrar fiind returnat un cod specific de eroare.

Conexiunile UDP sunt achizitionate utilizand o functie asemanatoare, singura diferenta fiind primul parametru care va fi de tipul MIB\_UDPTABLE.

Statisticile Tcp se obtin prin apelarea functiei GetTcpStatistics ce va furniza un set de informtii referitoare la protocolul TCP:

Algoritmul specifica algoritmul de retransmisie folosit (Retransmisiion time-out RTO).Acesta poate fi unul din urmatorii algoritmi: Constant, RSRE MIL-STD 1778, Van Jacobson’s Algorithm sau altul.

Timeout-ul minim specifica timpul minim de asteptare intre retransmiteri, exprimat in milisecunde.

Timeout-ul maxim specifica timpul maxim de asteptare inter retransmiteri, exprimat in milisecunde.

Maximum connections specifica numarul maxim de conexiuni . In cazul in care valoarea este -1 ,atunci numarul maxim de conexiuni e generat dinamic.

Conexiuni Active specifica numarul de conexiuni active din system.Intr-o conexiune active, clientul initiaza o conexiune cu serverul.

Conexiuni Pasive specifica numarul de conexiuni pasive.Intr-o conexiune pasiva,serverul asculta o cerere de conectare a unui client.

Incercari esuate specifica numarul de conexiuni esuate din diverse motive.

Conexiuni Resetate specifica numarul de conexiuni ce au fost stabilite ,dar resetate la un moment dat.

Conexiuni Stabilite specifica numarul de conexiuni stabilite in sistem la momentul actual.

Segmente primite , segmente trimise (acest numar nu include si segmentele retransmise).

Segmente trimise cu flag de reset reprezinta numarul de segmente transmise pentru a reseta o conexiune, iar Conexiuni cumulate semnifica numarul total de conexiuni TCP din sistem (in orice stare ar fi ele).

Statisticile protocolului UDP se obtin cu o functie asemanatoare denumita GetUdpStatistics ce va umple o structura de tipul MIB\_UDPSTATS. Aceasta structura este de dimensiuni mult mai reduse decat cea referitoare la statisticile TCP din cauza caracterului orientat pe mesaj al protocolului UDP.

**5. Metode alternative de implementare**

O alta metoda ce ar fi putut fi utilizata pentru a realiza aceasta aplicatie ar fi fost utilizarea *performance counter-elor* cu ajutorul librariei Pdh. Aceasta librarie pune la dispozitia programatorului o serie de functii prin care un server se poate conecta la sistemele dintr-o retea si poate cere datele citite de anumite *performance counter.*

Avantajul principal ar fi ca s-ar evita implementarea layer-ului de socketi implicit toate problemele legate de comunicare, de aceasta ocupandu-se intern libraria.

Ca dezavantaj poate fi privit faptul ca pentru a se putea conecta serverul trebuie sa primeasca o serie de drepturi asupra sistemelor tinta, printre care drepturi totale asupra unor chei din registrii si asupra unor fisiere de sistem, fapt ce ar putea crea anumite probleme de securitate.

**6. Bibliografie**

* + - 1. ANDREW S. TANENBAUM, “Computer networks – Third Edition”
      2. WILLIAM J. BUCHANAN , BSc, CEng, PhD “Mastering Networks”
      3. http://www.cisco.com/web/learning/netacad
      4. JON POSTEL, „RFC: 793 Transmission Control Protocol, Protocol Specifications”
      5. JON POSTEL, „RFC 768 User Datagram Protocol, Protocol Specifications”
      6. <http://en.wikipedia.org>
      7. Lect.univ.dr. Carmen TIMOFTE; Prep.univ.drd. Radu CONSTANTINESCU; Prep.univ.drd. Iulian ILIE-NEMEDI , “[RETELE DE CALCULATOARE Caiet de seminar](http://www.ase.ro/biblioteca/carte2.asp?id=379&idb=11) “

# Mark Towfiq - “Windows Sockets An Open Interface for Network Programming under Microsoft Windows” Version 1.1 20 January 1993

* + - 1. <http://msdn2.microsoft.com/>
      2. Bob Quinne, Dave Shute, “Windows Sockets Network Programming”