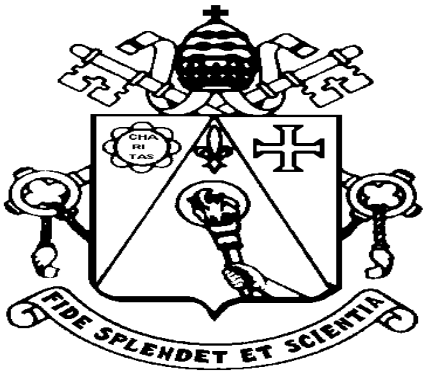
Pontifícia Universidade Católica de Campinas
Faculdade de Análise de Sistemas



PUC
CAMPINAS
PONTIFÍCIA UNIVERSIDADE CATÓLICA

Máquina de Estados TCP e Aplicações em Rede

por
Guilherme Cestarolli Seleguim

Estrutura e Recuperação da Informação II
Professor Francisco F. Primo

Maio-2003

LEARN THE RULES
SO YOU WILL KNOW HOW TO BREAK THEM PROPERLY.

DALAI LAMA

**Dedicado à todos aqueles que lutam e estão aguardando,
para um dia ficarem no topo e serem somados.**

.

Agradecimentos:

À Germano Cestarolli (em memória) e Orlando Seleguim (em memória).

À Nádia Bernuci dos Santos, Felipe Barbi (PUC, Finamax), Márcio Chagas (Siemens).

À Nicolo du Cesare Cestarolli e Família, pela empreitada e bravura nas idas de 1888.

Allo Spettabile Síndaco della Comune di Canda (Rovigo-Italia): Carretta Maurizio Mosè, Distinti Saluti !

Aos Respeitáveis Senhores da SIEMENS TUSA Power Transmission and Distribution e de seu Setor de Tecnologia e Desenvolvimento.

Aos Respeitáveis Senhores da Scope Systems Sistemas Corporativos Ltda., e à todos seus Desenvolvedores que diariamente me ajudam no trabalho.

Ao professor Francisco F. Primo, pela oportunidade.

Sumário

Resumo

O objetivo deste trabalho é apresentar a Máquina de Estados do protocolo TCP, desenvolvendo um aplicativo em C++ para ilustrar seu funcionamento. No decorrer do trabalho utilizaremos o conceito de conexão do protocolo TCP para que possamos, ao final, desenvolver e implementar um Port Scanner em Delphi 5. Para desenvolver o Port Scanner, fazemos uma breve apresentação de alguns conceitos sobre rede. Os utilitários ipconfig, netstat e telnet também são abordados no escopo deste trabalho. O software apresentado neste trabalho foi desenvolvido sob a Licença Pública Geral GNU, conforme publicada pela Free Software Foundation. Mais detalhes no item Licença do Sumário. Este trabalho foi desenvolvido somente para fins acadêmicos e o autor não se responsabiliza pelo mal uso do software aqui apresentado.

Palavras-Chave: Máquina de Estados, TCP, Redes de Computadores, C++, Delphi, Port Scanner.

Abstract

This essay intends to introduce the TCP Protocol StateMachine developing an C++ program to ilustrate TCP's states. We will use TCP's connection concepts to develop a Port Scanner in Delphi 5. To develop the Port Scanner, we will introduce some network concepts. The programs: ipconfig, netstat and telnet are also studied in this essay scope. The software we are going to show was developed under the GNU License from Free Software Foundation. The software here shown was made for educational purposes only and the author is not responsible for a bad usage of this software.

Key-Words: StateMachine, C++, Delphi, Port Scanner, Computer Networks, TCP.

Riassunto

Questa attività sia mostrare la Macchina di Stato di Protocolo TCP. Incluso anche spiegazione per utilizzazione di programmas: ipconfig, netstat e telnet. Uno Port Scanner in Delphi 5 è anche incluso questa attività. Il autore no è risponsabile per male uso dello programma. Software sviluppato sotto Licenza GNU di Software Libero.

Parola-Chiave: Macchina di Statos, C++, Delphi, Port Scanner, Rete di Computers, TCP.

Lista de Figuras

Introdução

A suíte de protocolos TCP/IP permite que computadores dos mais diferentes tipos, tamanhos e fabricantes se comuniquem. Isto é bastante surpreendente porque o TCP/IP superou em muito suas estimativas originais. O que começou no fim da década de 60 como um projeto de pesquisa governamental, na década de 90 tornou-se a forma mais amplamente usada para comunicação entre computadores.

O desenvolvimento do protocolo TCP/IP começou em 1969 com o projeto ARPANET, da Agência de Projetos e Pesquisas Avançadas do Departamento de Defesa dos EUA – DARPA).

O objetivo desse projeto era o desenvolvimento de uma rede que interligasse os computadores do governo americano utilizando diferentes sistemas operacionais. Essa rede deveria ser descentralizada e mesmo que um dos computadores dessa rede fosse destruído num eventual ataque militar, os demais continuariam a funcionar normalmente, graças à um mecanismo de rotas alternativas.

Algum tempo depois desse início com finalidade militar, a *National Science Foundation* criou uma rede semelhante para interconectar instituições de pesquisa e universidades, utilizando os mesmo protocolos da rede ARPANET.

Desses projetos surgiu o protocolo TCP/IP, que serviu como alicerce para a construção da rede que hoje conhecemos como Internet. À partir de 1993 a Internet ficou disponível para uso comercial e está em constante crescimento.

RFCs

Todos os padrões oficiais da comunidade Internet são publicados como um *Request for Comment*, ou *RFC*. Adicionalmente existem muitos RFCs que não são dos padrões oficiais, mas são publicados para propósitos informacionais. O tamanho de um RFC pode ir de 1 página para quase 200. Cada RFC é identificado por um número, tal como RFC 1122, sendo que quanto maiores forem os números, mais novos serão os RFCs.

Informações adicionais podem ser obtidas nos seguintes sites:
Repositório das RFCs oficiais e definitivas: http://www.rfc-editor.org/rfc.html

Informações sobre RFCs: http://www.faqs.org/rfcs/

O endereço IP

Todos as interfaces que estão numa rede, necessitam ter um endereço de identificação único, ou *endereço Internet*, também conhecido como *endereço IP*. Estes endereços são números formados por 32 bits. Ao contrário de usar um endereço como 1, 2, 3, existe uma estrutura para o endereço IP, como mostrado na figura 1. Estes endereços de 32 bits são normalmente escritos como 4 números decimais, um para cada byte do endereço. O endereço IP é definido nos RFCs 791, 1122 e 1812.

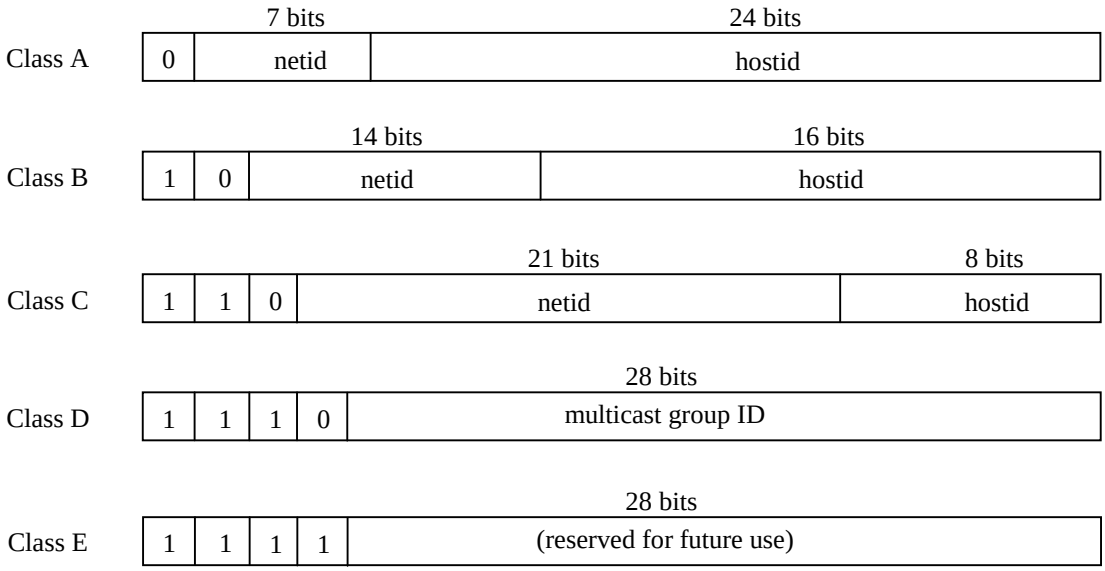


Figura 1 As cinco diferente classes do endereço IP.

Class	Range
A	**0**.0.0.0 to **127**.255.255.255
B	**128**.0.0.0 to **191**.255.255.255
C	**192**.0.0.0 to **223**.255.255.255
D	**224**.0.0.0 to **239**.255.255.255
E	**240**.0.0.0 to **255**.255.255.255

Figura 2 Ranges para as classes de endereço IP.

Se você estiver fora de uma rede, em casa por exemplo, seu endereço IP será o 127.0.0.1, comumente chamado de endereço de "*loopback*", pois quando usado faz referências à própria máquina. No Windows, existe um utilitário chamado **ipconfig** que exibe a configuração do protocolo TCP/IP. Se digitado sem nenhum parâmetro, exibe os valores de

endereço IP, máscara de sub-rede e gateway padrão para cada placa de rede instalada. Aqui vai o escopo do utilitário ipconfig:

ipconfig [/? | /all | /release [adaptador] | /renew [adaptador]]

/all Exibe informações detalhadas de IP para as placas de rede instaladas.

/release Libera o endereço IP obtido para uma placa de rede através de um servidor DHCP. Se a placa de rede não for especificada, libera os endereços IP obtidos para todas as placas de rede do computador.

/renew Renova um endereço IP obtido para uma placa de rede através de um servidor DHCP. Se a placa de rede não for especificada, renova os endereços IP obtidos para todas as placas de rede instaladas no computador.

adaptador Especifica uma placa de rede na renovação ou liberação de um endereço IP obtido através de um servidor DHCP. Para saber os nomes associados às placas de rede, utilize o comando ipconfig sem parâmetros.

Abaixo temos um exemplo:

Figura 3 ipconfig: sem conexão à Internet.

PPP adaptador NdisWan3 significa que eu tenho um modem que é identificado por NdisWan3, mas percebe-se que não estou conectado à Internet pois estou com o endereço 0.0.0.0 para Endereço IP. Abaixo temos um exemplo da utilização do ipconfig e podemos notar que já possuo um endereço IP associado ao meu modem.

Figura 4 ipconfig: endereço IP preenchido.

Modelo Cliente-Servidor

A maioria dos programas para rede são escritos assumindo que um lado é cliente e o outro é o servidor. O propósito da aplicação é para que o servidor provenha algum serviço definido para o cliente.

Podemos caracterizar servidores em duas classes: *interative* e *concurrent*. Um servidor *interative* interage pelos seguintes passos:

I1. Espera o pedido de um cliente chegar.

I2. Processa o pedido do cliente.

I3. Envia a resposta de volta ao cliente que a solicitou.

I4. Volta ao passo I1.

O problema com um servidor *interative* é quando o passo I2 demora. Durante este tempo nenhum outro pedido de cliente é processado.

Um servidor *concurrent*, por outro lado, executa os seguintes passos:

C1. Espera o pedido de um cliente chegar.

C2. Inicia um novo servidor para manipular o pedido deste cliente. Este pode envolver a criação de um novo processo, tarefa ou thread, dependendo do que o sistema operacional subordinado suporta. Este passo é feito dependendo do sistema operacional.

Este novo servidor que é iniciado manipula o pedido inteiro de um cliente. Quando ele completa o processamento, o novo servidor é encerrado.

C3. Volta ao passo C1.

Um cliente é um computador que solicita uma informação de um outro computador chamado servidor. Quando você digita no Internet Explorer o site: http://www.cestarolli.kit.net, você está solicitando ao computador que mantém o site que lhe envie sua página principal. Este é um processo cliente-servidor. Geralmente os clientes solicitam as informações e os servidores retornam as informações processadas para os clientes.

Números de Portas

Servidores são normalmente conhecidos por suas portas em operação. Por exemplo, toda implementação de TCP/IP que provê um servidor FTP provê este serviço na porta TCP 21. Todo servidor Telnet opera na porta TCP 23. Toda implementação de TFTP (*Trivial File Transfer Protocol*) opera na porta UDP 69. Estes serviços que podem ser providos por qualquer implementação de TCP/IP tem suas portas conhecidas (*well-known ports*) entre 1 e 1023. As portas conhecidas são gerenciadas pela *IANA – Internet Assigned Numbers Authority*.

Um cliente usualmente não se preocupa com qual porta ele irá usar. Tudo o que ele tem que ter certeza é que o número de porta que for usar seja único em seu *host*. As portas em clientes são classificadas como de vida curta pois existem somente enquanto o usuário que as está usando precisa de seus serviços. Servidores normalmente às mantém enquanto estão em pé (ativos).

As portas são identificadas entre 0 e 65535 e cada porta identifica um serviço que roda em servidores. Um computador servidor pode conter vários serviços rodando como correio eletrônico, transferência de arquivos e servidor web. Um servidor Web tem a porta 80 aberta. Se com um port scanner identificarmos que a porta 80 está aberta, provavelmente este computador é um servidor Web. Os números das portas padronizadas são definidos no RFC 1700.

Sockets

Um socket identifica uma conexão entre dois computadores. Através de sockets, aplicações em computadores distintos podem trocar informações, formando um canal de comunicação. Para um socket ser definido são necessárias as seguintes informações:

- Endereço IP do servidor
- Porta onde se encontra o serviço no servidor

- Endereço IP do cliente
- Porta do cliente para a comunicação com o servidor

O termo socket apareceu na especificação original do TCP (RFC 793).

O Protocolo TCP

O protocolo TCP é um serviço de entrega de pacotes que garante a entrega e a integridade do pacote e funciona baseado na *conexão lógica* entre dois computadores. Antes de transmitir os dados, o protocolo TCP estabelece uma conexão entre os dois computadores num processo chamado “three-way handshake”.

O termo *orientado à conexão* significa que as duas aplicações usando TCP precisam estabelecer uma conexão antes de trocarem dados. A analogia típica é a de uma ligação telefônica, espera-se o outro lado atender o telefone e dizer “alô” e aí perguntar quem está falando.

O TCP provê confiabilidade pelo seguinte:

- Os dados da aplicação são quebrados no que o TCP considera como sendo os pedaços mais adequados em tamanho pra transmití-los. A unidade de informação passada do TCP para o IP é chamada de segmento e o TCP decide qual o tamanho do segmento.
- Quando o TCP envia um segmento ele mantém um temporizador, que aguarda o reconhecimento da recepção do segmento pela outra parte. Se um reconhecimento não for recebido em um determinado tempo, o segmento é retransmitido.
- Quando o TCP recebe dados do outro lado da conexão ele envia um reconhecimento de recebimento. Este reconhecimento não é enviado imediatamente, mas é normalmente atrasado uma fração de segundos.
- O TCP mantém um *checksum* (um tipo de verificação) no seu cabeçalho e nos dados. O propósito é verificar qualquer modificação de dados no trânsito dos pacotes. Se um segmento chega com um *checksum* inválido, o TCP descarta o pacote e não reconhece o seu recebimento (espera que ocorra um *timeout* no remetente e o pacote seja retransmitido).
- Os segmentos TCP podem chegar fora da ordem no destinatário. O destinatário então reorganiza os dados se necessário, passando os dados recebidos na ordem para a aplicação.
- Um datagrama IP pode chegar duplicado então o TCP deve descartar dados duplicados.

- TCP também provê controle de fluxo. O cliente e o servidor tem uma quantidade finita de espaço em *buffer*. O destinatário só permite que seja enviada uma certa quantidade de dados se ele tiver *buffer* suficiente para recebê-los. Isto previne que um *host* rápido pegue todo o buffer de um *host* lento.

  O TCP é definido no RFC 793.

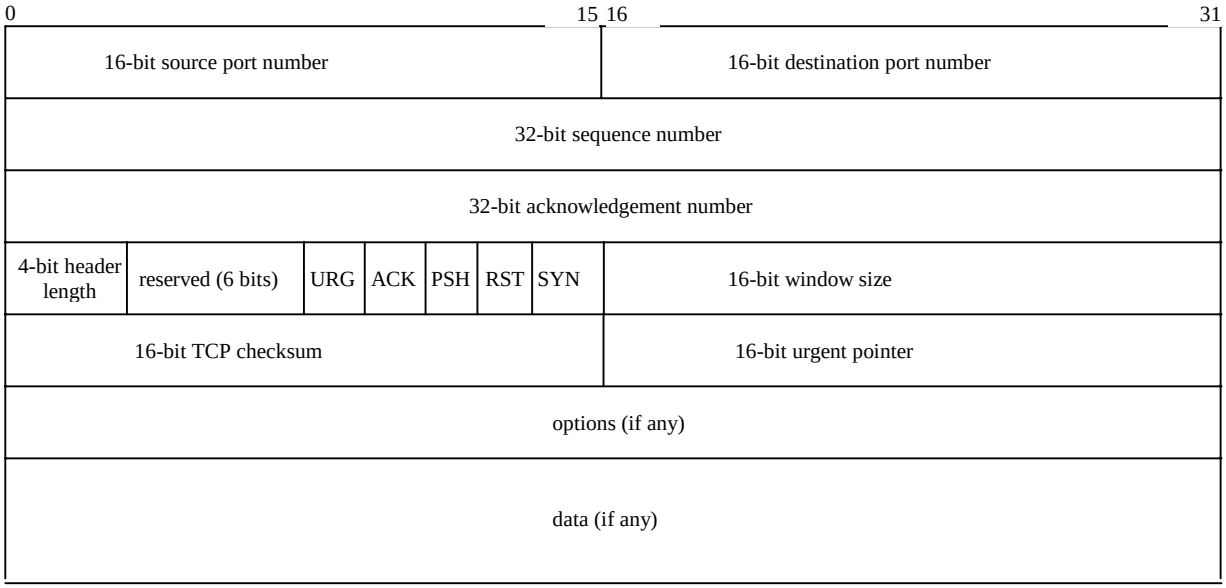


Figura 5 O cabeçalho TCP.

URG	O ponteiro urgente (*urgent pointer*) é valido.
ACK	O número de reconhecimento (acknowledgment number) é válido.
PSH	O recebedor deve enviar estes dados para a aplicação o mais rápido possível.
RST	Reinicia a conexão.
SYN	Sincroniza os números de sequência para iniciar uma conexão.
FIN	O remetente acabou de enviar dados.

Figura 6 Os possíveis flags do cabeçalho TCP.

O Estabelecimento de Conexão: Three-Way Handshake

O estabelecimento de uma conexão via TCP passa pelos seguintes passos:

1. O cliente envia um pacote SYN para o servidor especificando o número da porta à qual deseja se conectar e o número de sequência inicial (*initial sequence number-ISN*), por exemplo o número 1415531521.

2. O servidor responde com um pacote SYN contendo o seu número de sequência inicial. O servidor também reconhece o SYN do cliente enviando um pacote ACK contendo o número de sequência inicial do cliente mais 1.

3. O cliente precisa reconhecer o SYN que veio do servidor enviando um ACK para ele contendo o número de sequência inicial que ele enviou mais 1.

Estes três passos completam o estabelecimento de uma conexão.

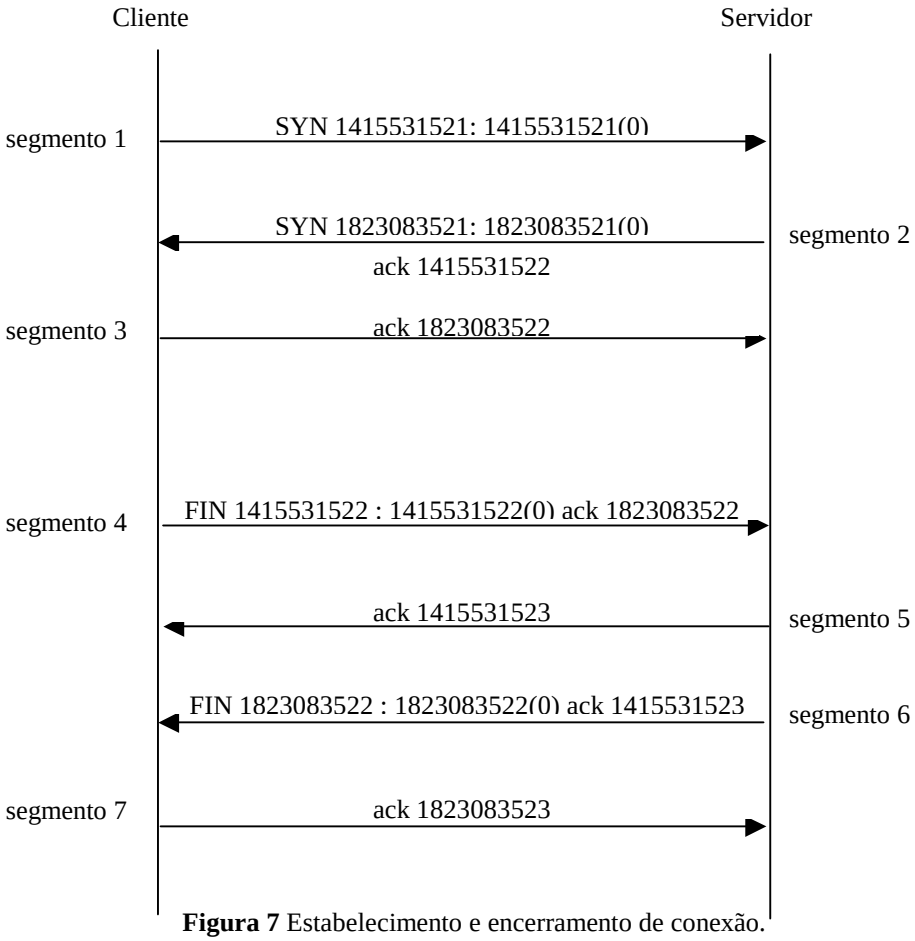


Figura 7 Estabelecimento e encerramento de conexão.

Protocolo de Encerramento de Conexão

Enquanto leva-se três segmentos para estabelecer uma conexão, leva-se quatro para terminá-la. De acordo com a figura acima, o segmento 4 inicia o encerramento da conexão. Isto faz com que o cliente envie um FIN para o servidor, fechando o fluxo de dados entre o cliente e o servidor.

Quando o servidor recebe o FIN, ele devolve um ACK do número de sequência recebido mais um (segmento 5). Neste ponto o servidor TCP também entrega, ou passa, um fim-de-arquivo (*end-of-file* EOF) para a aplicação. O servidor então encerra sua conexão, acarretando o envio de um FIN para o cliente (segmento 6), o qual o cliente precisa reconhecer (enviar um ACK de volta ao servidor) incrementando o número de sequência recebido em um (segmento 7).

A Máquina de Estados do Protocolo TCP

À seguir é apresentada a Máquina de Estados do protocolo TCP. Note que a representação está feita implementando a interação tanto do lado cliente quanto do lado servidor ao mesmo tempo.

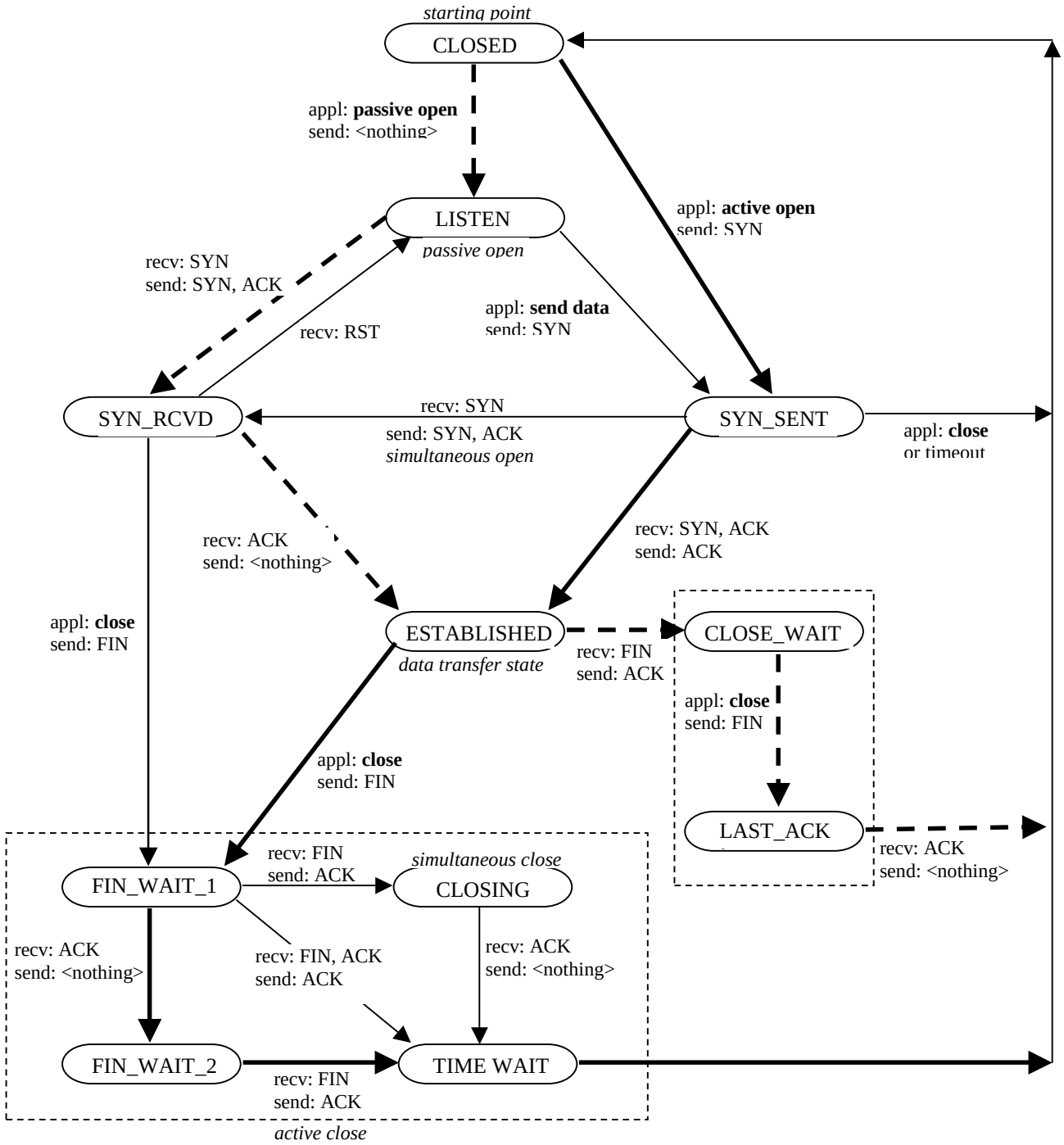


Figura 8 Máquina de Estados do Protocolo TCP.

As setas em negrito representam as transições de estados normais para o cliente. As setas pontilhadas em negrito representam as transições normais para o servidor. Appl indica transição de estado quando a aplicação executa uma operação.

A Máquina de Estados do Cliente

Separamos a máquina de estados do cliente para fazermos uma aplicação em C++ que ilustre seu funcionamento. Utilizamos para tanto um grafo e uma matriz de incidências.

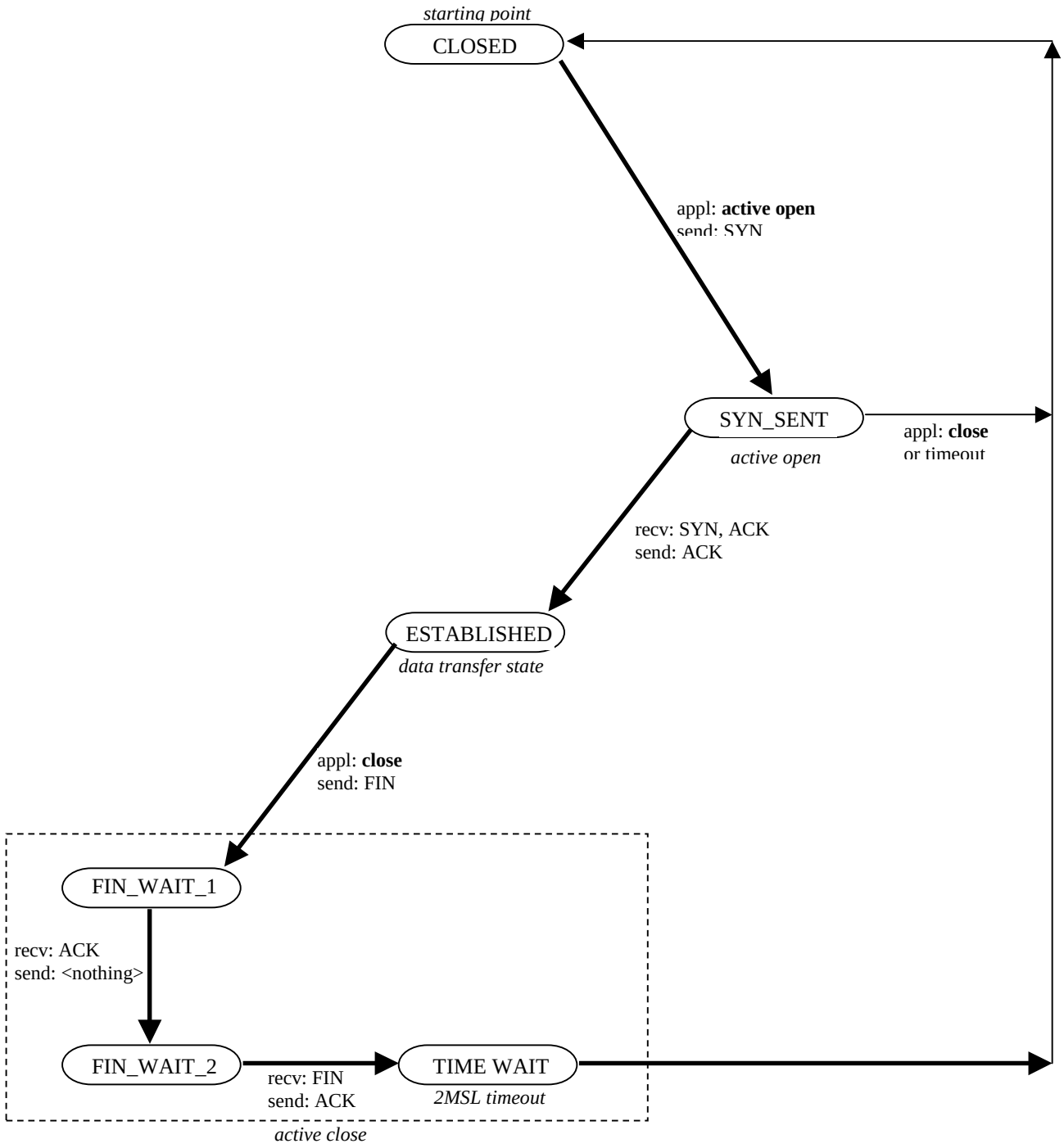


Figura 9 A máquina de estados de um cliente TCP.

Estados	Significados
E0	CLOSED
E1	SYN_SENT
E2	ESTABLISHED
E3	FIN_WAIT_1
E4	FIN_WAIT_2
E5	TIME_WAIT

Figura 10 Estados para o Grafo do Cliente TCP.

Eventos	Significados
ActiveOpen	Solicita conexão à um servidor. Envia um SYN ao servidor.
Close	A aplicação solicitou esta operação.
TimeOut	Excedido o tempo limite de espera.
SYN-ACK	Recebeu um SYN e um ACK do servidor e lhe devolve um ACK.
ACK	Recebeu um ACK do servidor. Não devolve nada.
FIN	Recebeu um FIN do servidor. Devolve um ACK.
2MSL	2 milisegundos de tempo de espera.

Figura 11 Eventos para o Grafo do Cliente TCP.

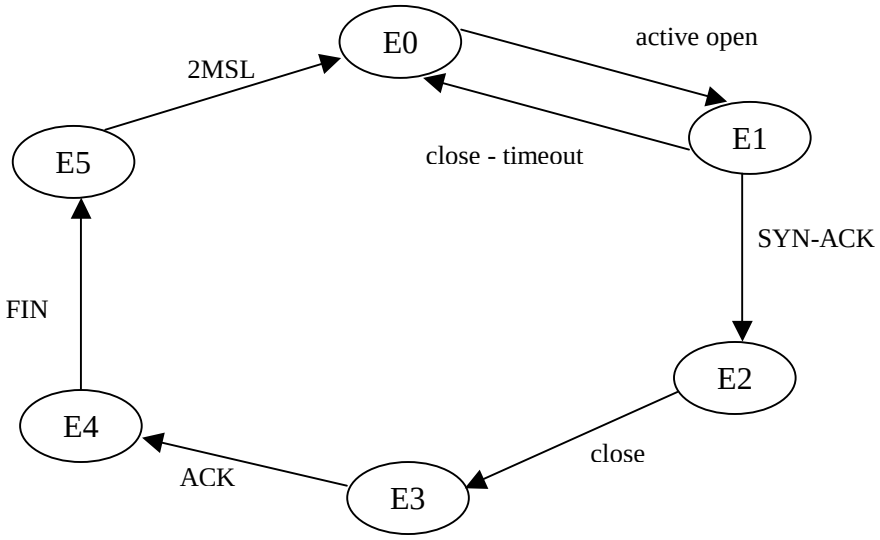


Figura 12 Grafo para o Cliente TCP.

	activeopen	close	timeout	SYN-ACK	ACK	FIN	2MSL
E0	E1	E0	E0	E0	E0	E0	E0
E1	E1	E0	E0	E2	E1	E1	E1
E2	E2	E3	E2	E2	E2	E2	E2
E3	E3	E3	E3	E3	E4	E3	E3
E4	E4	E4	E4	E4	E4	E5	E4
E5	E5	E5	E5	E5	E5	E5	E0

Figura 13 Matriz de Incidências (Matriz de Transição de Estados) para o Cliente TCP.

A aplicação em C++ da Máquina de Estados do Cliente TCP

```
/*
PUC-Campinas
Guilherme Cestarolli Seleguim


Maio-2003

     PT: Exemplo da Máquina de Estados de um Cliente TCP
     EN: Example of the TCP Client StateMachine
     IT: Esempio della Macchina di Stato di uno Cliente TCP
*/

#include "iostream.h"
#include "stdio.h"
#include "string.h"
#include "stdlib.h"

typedef int matriz[6][7];
char evento0[] = "activeopen";
char evento1[] = "close";
char evento2[] = "timeout";
char evento3[] = "synack";
char evento4[] = "ack";
char evento5[] = "fin";
char evento6[] = "2msl";

int le_codifica_entrada ()
{
     char entrada[10];

     cout<<"Entre com um evento: ";
     cin>>entrada;

     if (strcmp(evento0, entrada) == 0)
          return 0;
     else if (strcmp(evento1, entrada) == 0)
          return 1;
     else if (strcmp(evento2, entrada) == 0)
          return 2;
     else if (strcmp(evento3, entrada) == 0)
          return 3;
     else if (strcmp(evento4, entrada) == 0)
          return 4;
     else if (strcmp(evento5, entrada) == 0)
          return 5;
     else if (strcmp(evento6, entrada) == 0)
          return 6;
     else
          {    cout<<endl<<"Evento invalido, abortando..."<<endl;

               exit(0);
          }
}

void showbanner()
```

```
{
     cout<<"Maquina de Estados de Cliente TCP"<<endl;
     cout<<"Pontificia Universidade Catolica de Campinas -
PUCC"<<endl;
     cout<<"Estrutura e Recuperacao da Informacao II"<<endl;
     cout<<"Professor Francisco F. Primo 'Xicao'"<<endl;
     cout<<"Analise de Sistemas - Maio/2003"<<endl;
     cout<<"Guilherme Cestarolli Seleguim"<<endl<<endl;
     cout<<"Eventos possiveis"<<endl;
     cout<<"activeopen - closed - timeout - synack - ack - fin -
2msl"<<endl<<endl;
     cout<<"Estado Inicial: CLOSED"<<endl<<endl;
}

void main()
{
     matriz TCP={{1,0,0,0,0,0,0},
                 {1,0,0,2,1,1,1},
                 {2,3,2,2,2,2,2},
                 {3,3,3,3,4,3,3},
                 {4,4,4,4,4,5,4},
                 {5,5,5,5,5,5,0}};

     int entrada,estado = 0;

     showbanner();

     while(true)
     {
          entrada = le_codifica_entrada();
          estado = TCP[estado][entrada];

          switch (estado)
          {
               case 0: cout<<"Estado atual: CLOSED"<<endl;
                    break;
               case 1: cout<<"Estado atual: SYN_SENT"<<endl;
                    break;
               case 2: cout<<"Estado atual: ESTABLISHED"<<endl;
                    break;
          case 3: cout<<"Estado atual: FIN_WAIT_1"<<endl;
                    break;
               case 4: cout<<"Estado atual: FIN_WAIT_2"<<endl;
                    break;
               case 5: cout<<"Estado atual: TIME_WAIT"<<endl;
                    break;
          }
     }
}
```

A Máquina de Estados do Servidor

Agora é a vez de separarmos a máquina de estados do servidor para fazermos uma aplicação em C++. Utilizamos também um grafo e uma matriz de incidências.

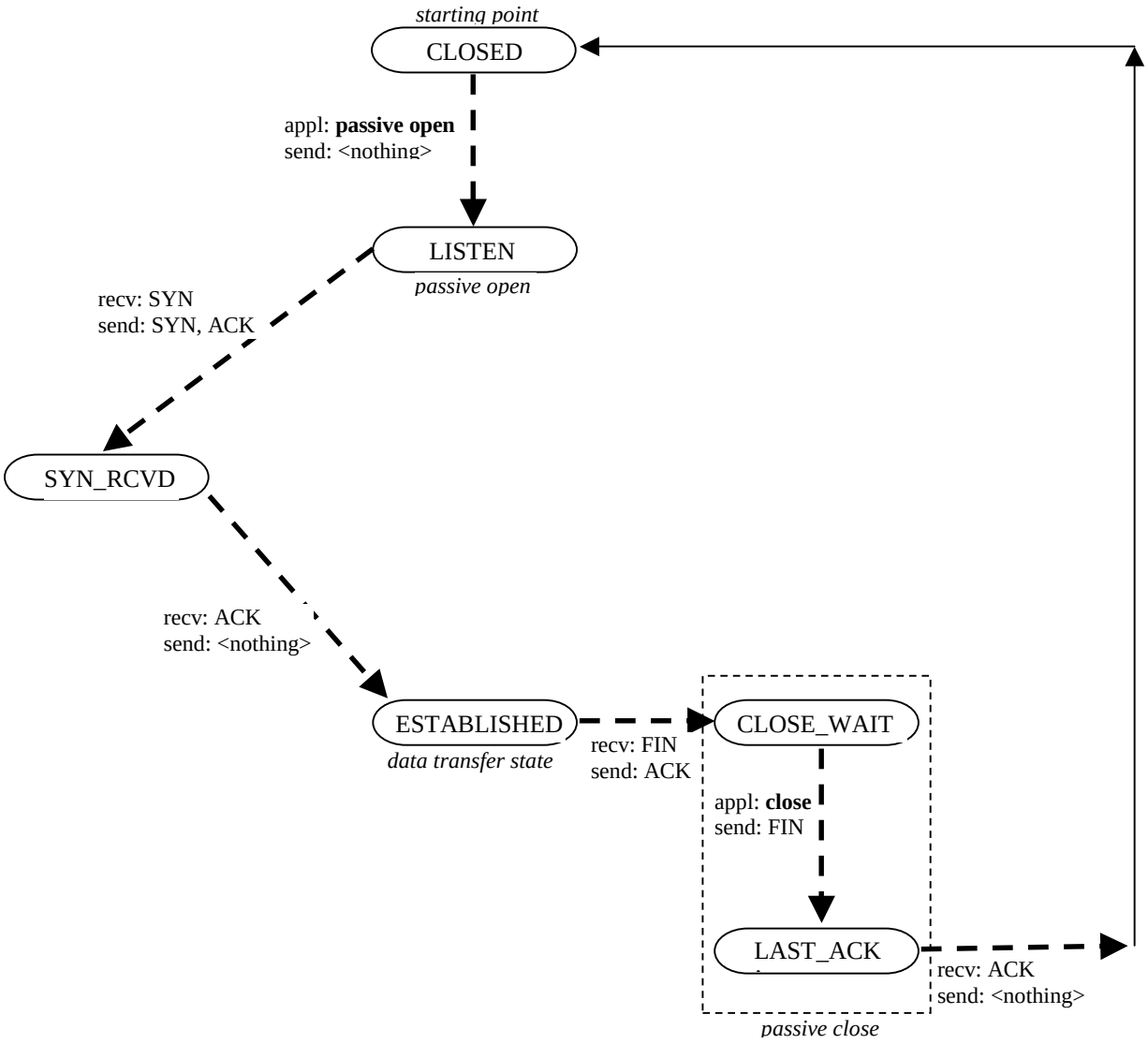


Figura 14 A máquina de estados do Servidor TCP.

Estados	Significados
E0	CLOSED
E1	LISTEN
E2	SYN_RCVD
E3	ESTABLISHED
E4	CLOSE_WAIT
E5	LAST_ACK

Figura 15 Estados para o Grafo do Servidor TCP.

Eventos	Significados
PassiveOpen	A aplicação solicitou esta operação.
SYN	Recebeu um SYN do cliente e devolve um SYN-ACK
ACK	Recebeu um ACK do cliente. Não devolve nada.
FIN	Recebeu um FIN do cliente e lhe devolve um ACK.
Close	A aplicação solicitou esta operação. Devolve um FIN.

Figura 16 Eventos para o Grafo do Servidor TCP.

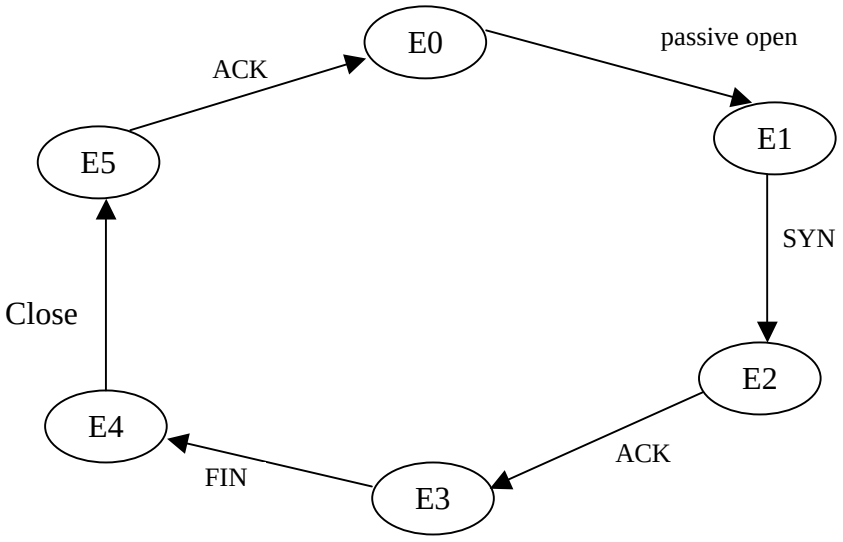


Figura 17 Grafo para o Servidor TCP.

	passiveopen	SYN	ACK	FIN	close
E0	E1	E0	E0	E0	E0
E1	E1	E2	E1	E1	E1
E2	E2	E2	E3	E2	E2
E3	E3	E3	E3	E4	E3
E4	E4	E4	E4	E4	E5
E5	E5	E5	E0	E5	E5

Figura 18 Matriz de Incidências (Matriz de Transição de Estados) para o Servidor TCP.

A aplicação em C++ da Máquina de Estados do Servidor TCP

```
/*
PUC-Campinas
Guilherme Cestarolli Seleguim


Maio-2003

     PT: Exemplo da Máquina de Estados de um Servidor TCP
     EN: Example of the TCP Server StateMachine
     IT: Esempio della Macchina di Stato di uno Servitore TCP
*/

#include "iostream.h"
#include "stdio.h"
#include "stdlib.h"
#include "string.h"

typedef int matriz[6][5];
char evento0[] = "passiveopen";
char evento1[] = "syn";
char evento2[] = "ack";
char evento3[] = "fin";
char evento4[] = "close";

int le_codifica_entrada ()
{
     char entrada[11];

     cout<<"Entre com um evento: ";
     cin>>entrada;

     if (strcmp(evento0, entrada) == 0)
          return 0;
     else if (strcmp(evento1, entrada) == 0)
          return 1;
     else if (strcmp(evento2, entrada) == 0)
          return 2;
     else if (strcmp(evento3, entrada) == 0)
          return 3;
     else if (strcmp(evento4, entrada) == 0)
          return 4;
     else
          {
               cout<<endl<<"Evento invalido, abortando..."<<endl;

               exit(0);
          }
}

void showbanner()
{
     cout<<"Maquina de Estados de Cliente TCP"<<endl;
     cout<<"Pontificia Universidade Catolica de Campinas -
PUCC"<<endl;
     cout<<"Estrutura e Recuperacao da Informacao II"<<endl;
```

```
      cout<<"Professor Francisco F. Primo 'Xicao'"<<endl;
      cout<<"Analise de Sistemas - Maio/2003"<<endl;
      cout<<"Guilherme Cestarolli Seleguim"<<endl<<endl;
      cout<<"Eventos possiveis"<<endl;
      cout<<"passiveopen - syn - ack - fin - close"<<endl<<endl;
      cout<<"Estado Inicial: CLOSED"<<endl<<endl;
}

void main()
{
      matriz TCP={{1,0,0,0,0},
                  {1,2,1,1,1},
                  {2,2,3,2,2},
                  {3,3,3,4,3},
                  {4,4,4,4,5},
                  {5,5,0,5,5}};

      int entrada,estado = 0;

      showbanner();

      while(true)
      {
            entrada = le_codifica_entrada();
            estado = TCP[estado][entrada];

            switch (estado)
            {
                  case 0: cout<<"Estado atual: CLOSED"<<endl;
                        break;
                  case 1: cout<<"Estado atual: LISTEN"<<endl;
                        break;
                  case 2: cout<<"Estado atual: SYN_RCVD"<<endl;
                        break;
            case 3: cout<<"Estado atual: ESTABLISHED"<<endl;
                        break;
                  case 4: cout<<"Estado atual: CLOSE_WAIT"<<endl;
                        break;
                  case 5: cout<<"Estado atual: LAST_ACK"<<endl;
                        break;
            }
      }
}
```

Nota: Em ambos os casos (cliente e servidor), se o estado receber um evento que não seja o esperado, ele permanecerá no estado em que se encontra.

Netstat

O Netstat (Network Status) é um utilitário do Windows para exibir o estado das conexões relacionadas aos protocolos IP, TCP e UDP. Aqui vai o escopo para o uso do utilitário netstat:

netstat [-a] [-e] [-n] [-s] [-p protocolo] [-r] [intervalo]

-a Exibe conexões atuais e as portas em estado LISTEN – aguardando conexões.

-e Exibe estatísticas Ethernet.

-n Exibe no formato numérico endereços IP e portas.

-s Informações agrupadas por protocolo.

-p Exibe somente as conexões para um dos protocolos – TCP, UDP e IP.

-r Exibe o conteúdo da tabela de roteamento.

intervalo Especifica o tempo em segundos de pausa para reexibir as informações selecionadas. Ctrl+C para interromper.

Telnet

Telnet é um utilitário para realizar conexões remotas. Com ele podemos nos conectar em um servidor e interagir com ele como se estivessemos digitando no próprio teclado do servidor. Para abrir o Telnet no Windows vá em Iniciar, Executar, digite telnet e pressione o botão OK.

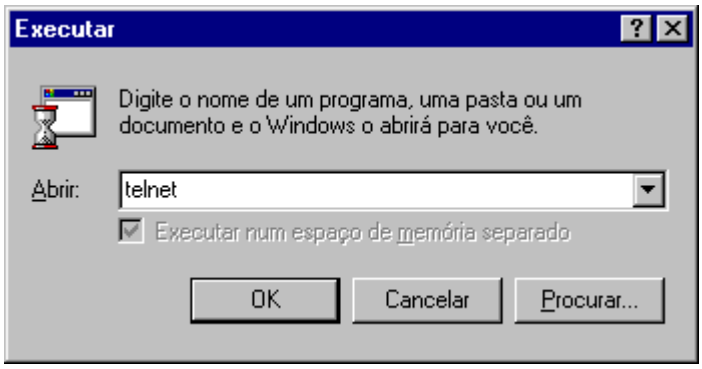


Figura 19 Executando o telnet.

A seguinte tela do Telnet aparecerá:

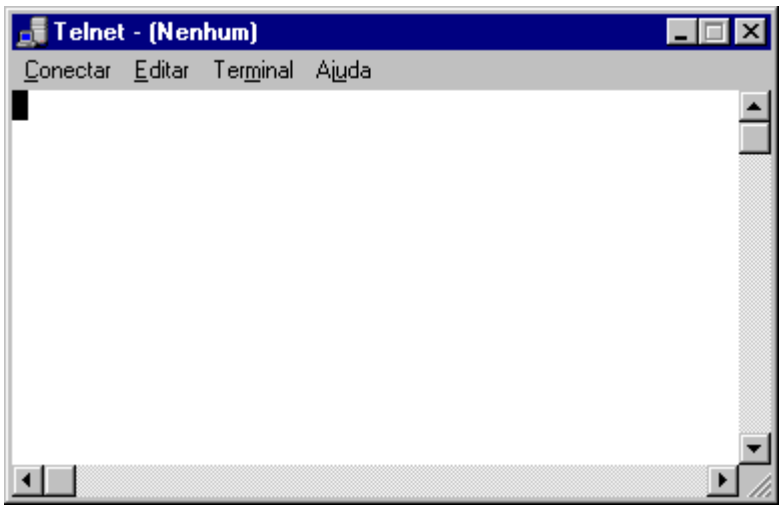


Figura 20 Tela inicial do telnet

Vá agora para a opção do menu Conectar, Sistema Remoto

Figura 21 Conectando à um servidor.

Em nome do host, digite o endereço IP ou nome do computador ao qual deseja se conectar e também a respectiva porta. No exemplo abaixo estou solicitando uma conexão ao host www.ig.com.br na sua porta 80, que á a porta padrão usada para ser um servidor Web.

Figura 22 Especificações do servidor.

Este utilitário de Telnet que acompanha o Windows pode ser usado para vários fins, como por exemplo se conectar a um computador que seja servidor de transferência de arquivos (FTP) e assim enviar e baixar arquivos. Ou também se conectado à um servidor de envio de e-mail (SMTP), podemos enviar e-mails. Assim como podemos enviar e-mails, podemos recebê-los também por Telnet, conectando-se à um servidor pop3, mas como o Telnet só mostra dados em formato texto, alguns e-mails ficariam não legíveis.

Protocolos para envio e recebimento de e-mail

E-mails grandes, principalmente os não solicitados costumam ser um problema. Quando você mais precisa acessar sua caixa postal, surge aquela mensagem de 5 MB que congestiona todo o acesso. Agora você não precisa mais passar por isso. Basta ter alguns conhecimentos técnicos sobre servidores POP3 e você mesmo poderá acessar o provedor para apagar as mensagens grandes.

Além disso é possível usar esta técnica com os servidores SMTP para o envio de mensagens. Se você souber usar estes recursos corretamente, poderá até mesmo ocultar o remetente de um e-mail.

Conexão direta no POP3

1- Abra o telnet e clique em Conectar->Sistema Remoto.

2- No campo Nome do Host, entre com o endereço do servidor POP3. Geralmente este endereço é algo como **pop3.provedor.com.br** ou um IP, como **200.245.211.10**

3- Fique muito atento ao campo Porta, pois é aí que está a jogada. Ao invés de Telnet, digite o número 110. Em seguida, clique em conectar e aguarde alguns instantes para que a conexão seja estabelecida.

4- Efetue o logon no servidor. Para fazer isso, digite o comando user, seguido pelo seu nome de acesso ao POP3, por exemplo, **user guilherme**. Se o nome do usuário existir, o servidor irá apresentar uma mensagem positiva, como um OK.

5- Entre com a senha usando o comando pass. Neste caso a sintaxe é: **pass senha**.

6- Digite o comando **list**. Uma relação de todas as mensagens disponíveis na sua caixa de e-mail aparecerá. Lembre-se que cada mensagem é indicada apenas por um número, que fica localizado do lado esquerdo. O número da direita é o tamanho em bytes da mensagem.

Existem alguns outros comandos que podem ser utilizados como:

stat: mostra as estatísticas da sua conta POP3. É muito pareciso com o list, porém bem mais simples. Pode ser uma boa idéia para contas que possuem muitas mensagens.

retr: usado para ler mensagens no próprio programa telnet. Neste caso é importante lembrar que os e-mails permanecem disponíveis após a leitura.

dele: é através dele que as mensagens podem ser apagadas.

help: mostra uma lista dos comandos disponíveis no servidor POP3. Por questões de segurança, a maioria dos provedores prefere desabilitar este comando.

quit: termina a conexão com o servidor.

Servidores SMTP

Se você achou interessante acessar o POP3 direto no servidor, espere até descobrir como funciona o protocolo SMTP. Você pode mandar até mesmo um e-mail sem identificação de remetente para outra pessoa.

Para fazer isso a ferramenta é a mesma: um programa telnet. A única diferença é a porta, que agora passa a ser a 25.

1- Conecte-se ao servidor SMTP: smtp.provedor.com.br, porta 25.

2- Ao contrário do POP3 não é necessário efetuar logon. Porém, a maioria dos provedores possuem firewall para impedir o acesso externo, ou seja, não adianta tentar enviar e-mails usando SMTP de um outro provedor pois não dará certo.

3- Para enviar uma mensagem basta digitar os seguintes comandos:

helo
mail from:usuario@provedor.com.br
rcpt to:destinatario@provedor.com.br
data
subject: título
mensagem aqui
.

O comando **mail from:** indica o remetente da mensagem. O comando **rcpt to:** indica qual será o destinatário da mensagem. Por fim o comando **data** permite que você comece a escrever a sua mensagem. Para finalizar digite apenas um ponto (**.**).

Embora este seja um método muito interessante, procure evitar fazer brincadeiras usando esta técnica. Os provedores sempre criam logs das conexões, de forma que seu endereço IP fique armazenado no caso de algum problema ocorrer.

Desenvolvendo um programa servidor em Delphi 5

Vamos fazer agora um pequeno programa em Delphi 5 para assimilar a idéia do funcionamento de um software servidor. Vamos utilizar o componente TServerSocket para contruir nosso programa.

1. Comece uma nova aplicação no Delphi;
2. Altere o nome do formulário para frmMain;
3. Adicione ao formulário um componente TServerSocket, que se encontra na palheta Internet.
4. Configura a propriedade Name para MyServer;
5. Configure a propriedade Port de MyServer para 10000 e a propriedade Active para True;
6. Em seguida execute a aplicação. O layout do formulário deve ficar parecido com o da figura abaixo:

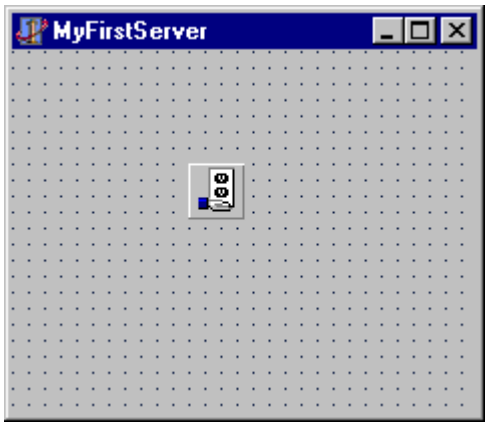


Figura 23 MyFirstServer.

O que fizemos agora foi configurar a porta em que o nosso programa servidor ficará aguardando para receber conexões. Quando executamos a aplicação, o componente TServerSocket estará ativo, pois mudamos a propriedade Active para True em tempo de projeto.

Vamos usar o utilitário netstat ainda com o programa rodando, portanto, no prompt de comando digite a seguinte linha:

netstat –an –p tcp | more

Podemos verificar com esse comando que a porta 10000 da coluna Endereço Local está em modo LISTENING por causa do nosso componente TServerSocket que está ativo.

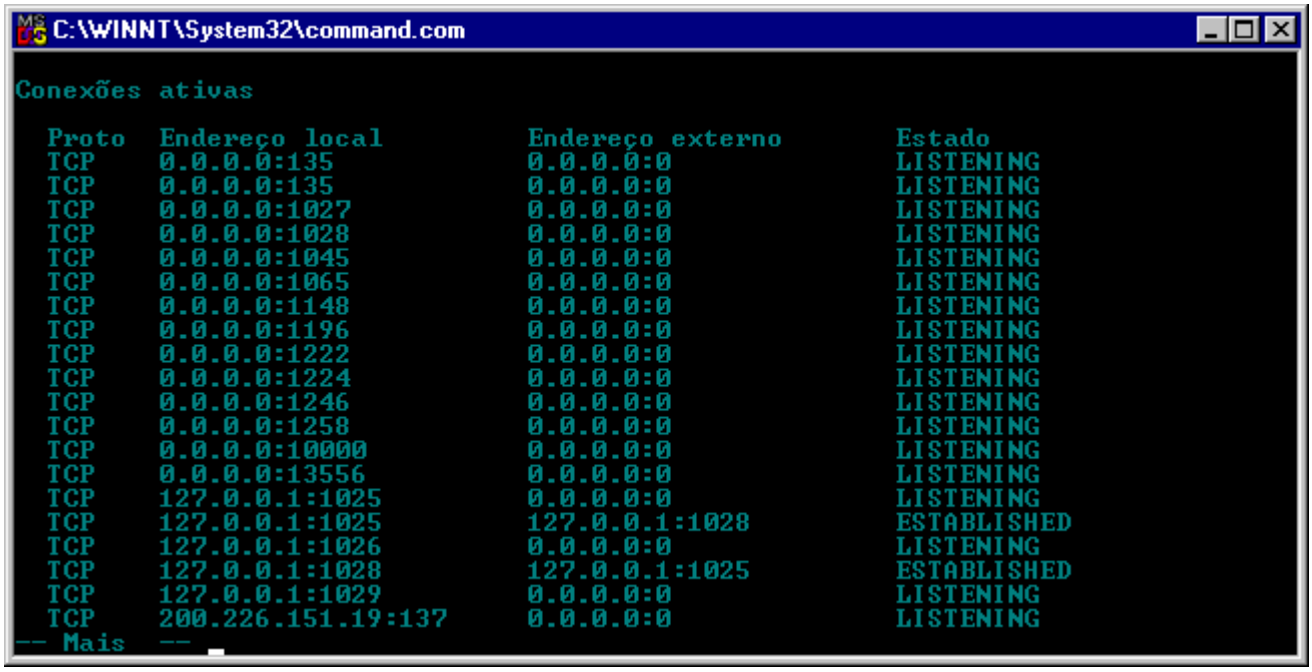


Figura 24 Usando o netstat.

Concluímos então nosso primeiro programa servidor. Mas serve apenas para ilustrar a programação básica de um programa servidor, mas falta ainda adicionar algumas linhas de código para testarmos nosso programa com o telnet como sendo um cliente.

Ainda em nosso programa servidor, faça as seguintes alterações:

1. Adicione um TMemo ao formulário e mude sua propriedade Name para mInfo;
2. Altere a propriedade Align de mInfo para alClient;
3. Altere a propriedade ScrollBars de mInfo para ssVertical;
4. No evento FormShow de frmMain adicione as seguintes linhas de código:

```
procedure TfrmMain.FormShow(Sender: TObject);
begin
  mInfo.Clear;
```

```
  mInfo.Lines.Add('LISTENING ON PORT: ' +
IntToStr(MyServer.Port));
end;
```

5. Agora adicione o código abaixo no event ClientConnect do componente MyServer:

```
procedure TfrmMain.MyServerClientConnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  mInfo.Lines.Add(Socket.RemoteHost + ' conectado.');
  Socket.SendText('Você está conectado ao servidor.');
end;
```

Este evento será gerado sempre que um cliente se conectar ao nosso programa servidor, e , quando isto ocorrer, o servidor irá enviar uma mensagem através do método SendText.

6. Adicione o seguinte código ao evento ClientDisconnect do componente MyServer:

```
procedure TfrmMain.MyServerClientDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  mInfo.Lines.Add(Socket.RemoteHost + ' desconectado.');
end;
```

Este evento será gerado sempre que um cliente se desconectar de nosso servidor. O layout do formulário deverá ficar parecido com o seguinte:

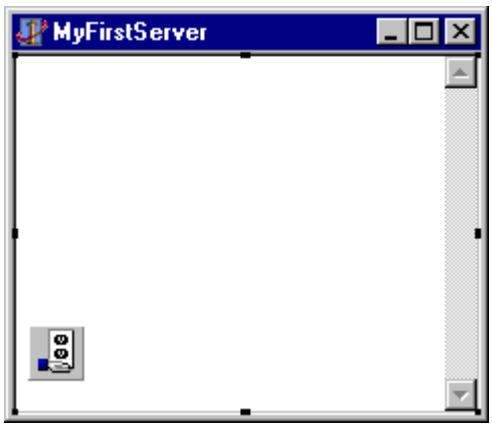


Figura 25 Layout de MyFirstServer modificado.

Agora execute a aplicação. Se executar o utilitário **netstat** novamente, iremos notar que a porta 10000 estará em modo LISTENING, porque deixamos o Active do componente MyServer como True. Vamos testar a nossa pequena aplicação com o programa telnet. Para isso vá em Inciar, Executar, digite telnet na caixa de texto e clique em abrir.

Vá ao menu Conectar e coloque o IP 127.0.0.1 (máquina local) em Nome do Host e a porta 10000 em Porta e clique no botão Conectar

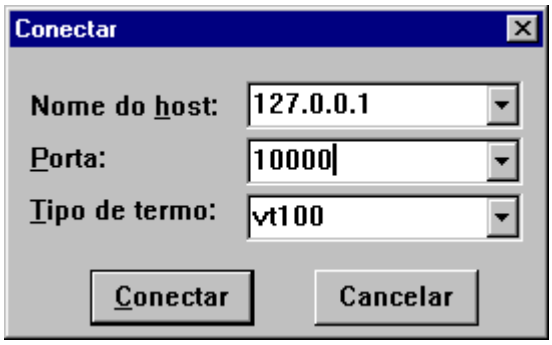


Figura 26 Conectando-se ao nosso servidor com telnet.

Ao se conectar, a mensagem “Você está conectado ao servidor”, conforme programamos, deverá aparecer na tela do telnet, conforme a figura 27:

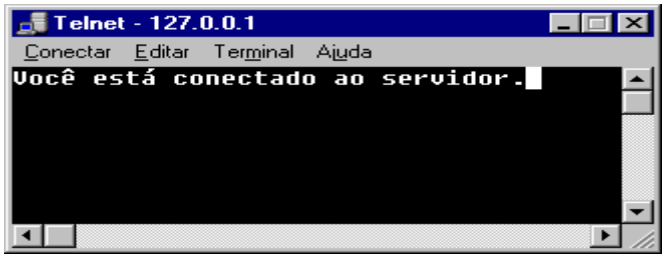


Figura 27 Conectado ao servidor.

Desenvolvendo um Servidor em Visual Basic

Para aqueles que têm uma richa pessoal com Delphi aqui vai o desenvolvimento de um servidor em Visual Basic. Abra o VB e escolha um novo projeto Standard EXE.

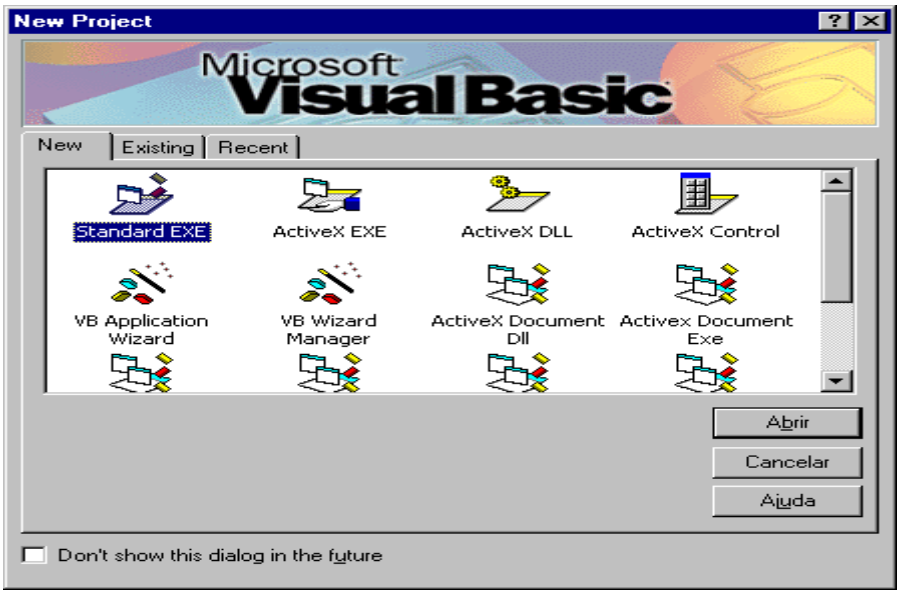


Figura 28 New Project VB.

Altere as seguintes propriedades do formulário:

- Name: frmServer;
- Caption: MyFirstVBServer;
- BorderStyle: 1-Fixed Single;
- MaxButton: False.

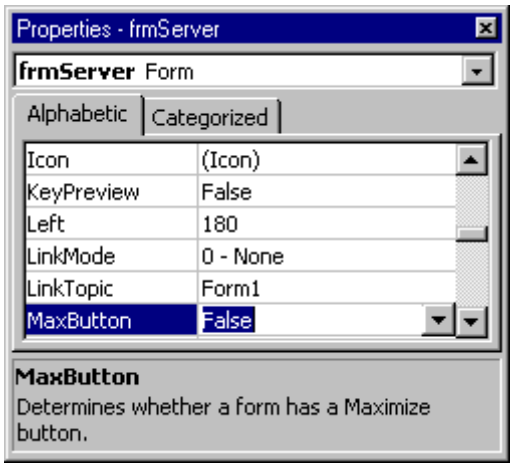


Figura 29 frmServer Properties.

Agora altere o nome do projeto para MyServer:

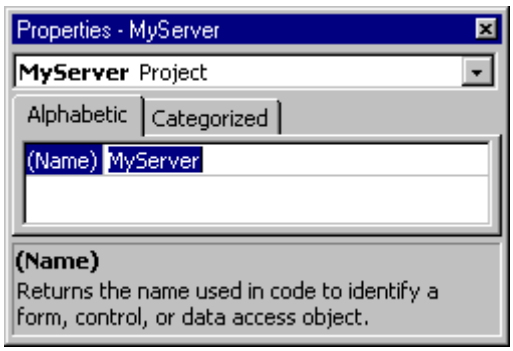


Figura 30 Alterando o nome do Projeto.

Salve o projeto com o nome de MyServer.vbp e o formulário como frmServer.frm. Agora vamos colocar nosso componente para rede no VB, o Microsoft Winsock Control 6.0. Para isso pressione Ctrl+T para acessar a tela de adicionamento de componentes e marque o Microsoft Winsock Control 6.0.

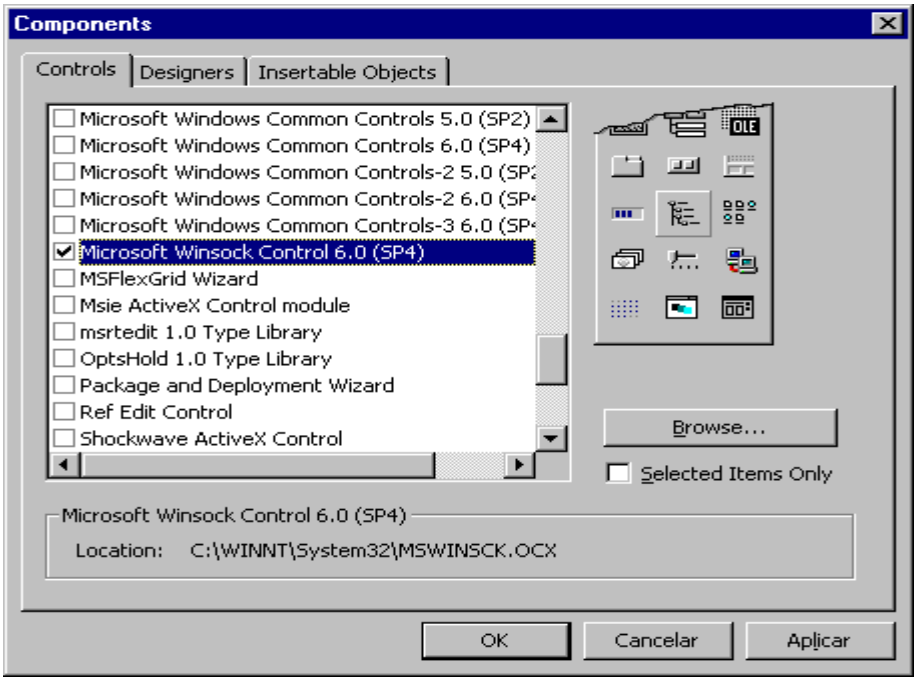


Figura 31 Adicionando Microsoft Winsock Control 6.0.

Clique no menu: View->Toolbox para chamar a seguinte janela:

Figura 32 A janela Toolbox.

Clique duas vezes sobre o componente Winsock, ele então será adicionado ao nosso formulário. Mude as propriedades deste componente conforme mostrado na figura abaixo:

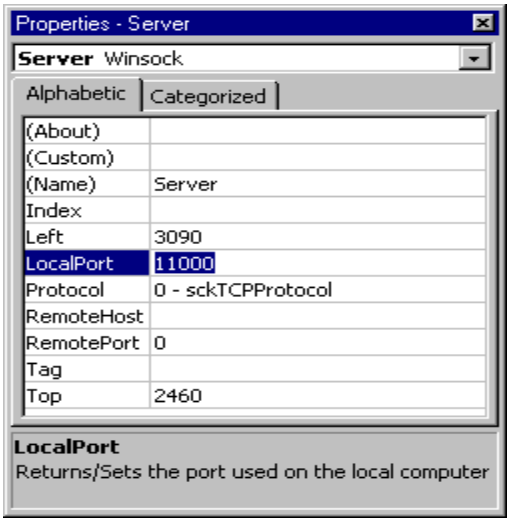


Figura 33 Server Properties.

Adicione um TextBox ao formulário e altere suas propriedades conforme à seguir:

- Name: txtInfo;
- Multline: True;
- Scrollbars: 2 – Vertical.

O formulário deve ficar parecido com o abaixo:

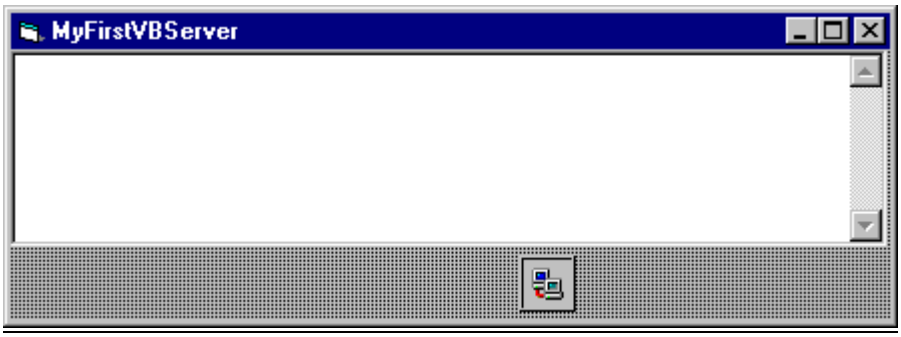


Figura 34 MyFirstVBServer.

Codificando o Servidor em Visual Basic

Agora passaremos para a codificação do servidor. No evento form_load do adicione o seguinte código:

```
Private Sub Form_Load()
    Server.Listen
    txtInfo.Text = "Listen..."
End Sub
```

Observer que o código Server.Listen coloca o programa em modo de escuta, i.e., aguardando conexões na porta 11000, previamente determinada em design time por sua propriedade.

Vamos agora programar nosso componente Server. Adicione o seguinte código em seu evento Error, assim, um erro que ocorrer será adicionado ao txtInfo:

```
Private Sub Server_Error(ByVal Number As Integer, Description As
String, ByVal Scode As Long, ByVal Source As String, ByVal HelpFile
As String, ByVal HelpContext As Long, CancelDisplay As Boolean)
    txtInfo.Text = txtInfo.Text + "Erro no servidor: " & Number & "-
" & Description
End Sub
```

Todo vez que alguém se conectar ao nosso servidor ele registrará seu endereço IP no txtInfo.

Adicione o seguinte código no evento DataArrival:

```
Private Sub Server_DataArrival(ByVal bytesTotal As Long)
    Dim Msg As String
    Server.GetData Msg, vbString
    txtInfo.Text = txtInfo.Text & "Mensagem de:" &
Server.RemoteHostIP & " -> " & Msg & vbCrLf
```

```
End Sub
```

Adicione o seguinte código no eveto ConnectionRequest:

```
Private Sub Server_ConnectionRequest(ByVal requestID As Long)
    Server.Close
    Server.Accept requestID
    txtInfo.Text = txtInfo.Text + "Cliente conectado: " &
Server.RemoteHostIP & vbCrLf
    Server.SendData "Você está conectado ao servidor em Visual
Basic"
End Sub
```

Pronto, agora podemos testar o nosso servidor em Visual Basic utlizando o telnet também !

Port Scanner

Um Port Scanner é um software capaz de identificar quais as portas estão em uso em um computador numa rede. É utilizado por administradores de redes e de sistemas mas também é muito utilizado no *underground* da Internet por *Hackers* e *Crackers*. O Port Scanner que será desenvolvido verifica apenas portas TCP.

Desenvolvendo a Interface

Aqui começaremos o desenvolvimento do Port Scanner porém nenhum código será implementado, somente serão adicionados e configurados os componenetes visuais.

Para começarmos, crie uma nova aplicação em Delphi e salve o projeto. Dê o nome uMain para a unit e PortScanner para o projeto.

1. Altere as propriedades do formulário conforme é mostrado à seguir:

- Width = 466
- Height = 348
- BorderIcons = [biSystemMenu,biMinimize]
- BorderStyle = bsSingle
- Caption = VisyBOT Port Scanner

- Position = poDesktopCenter
- Name = frmMain

2. Insira um componente Panel e altere as seguintes propriedades:

- Name = pnlName
- Width = 441
- Height = 41
- BevelWidth = 2
- Top = 7
- Left = 8

3. Insira dentro de pnlName duas Labels e altera as seguintes propriedades:

Label1

- Name = lblName
- Caption = VisyBOT Port Scanner
- Font.Style = [fsBold]

Label2

- Name = lblVersion
- Caption = Internet version 1.0 [Compilation 01]
- Font.Style = [fsBold]

4. Insira um componente PageControl e altere as seguintes propriedades:

- Name = pgcMain
- Width = 441
- Height = 249
- Top = 48
- Left = 8

5. Insira três componentes TabSheet em seu PageControl e altere as propriedades mostradas à seguir para os objetos:

TabSheet1

- Name = tsDefinition
- Caption = Definition

TabSheet2

- Name = tsPortScan
- Caption = Port Scanner

TabSheet3

- Name = tsAbout
- Caption = About

6. Em tsDefinition insira um Panel e altere as seguintes propriedades:

- Name = pnlData
- Align = alClient
- Caption = (em branco)
- Color = clWhite

7. Insira dentro de pnlData 6 Labels, 2 Edits, 2 SpinEdits, 1 Gauge, 1 Memo, 1 Animate e 2 Buttons e altere as propriedades desses componentes conforme segue:

Label1

- Name = lblHostName
- Caption = Host Name:
- Top = 8
- Left = 56

Label2

- Name = lblTimeOut
- Caption = Time Out:
- Top = 8
- Left = 280

Label3

- Name = lblDomain
- Caption = Host's Domain
- Top = 48
- Left = 56
- Font.Style = [fsBold]

Label4

- Name = lblStartPort
- Caption = Start Port:
- Top = 64
- Left = 56

Label5

- Name = lblFinalPort
- Caption = Final Port:
- Top = 64
- Left = 120

Label6

- Name = lblProgress
- Caption = Progress:
- Top = 64
- Left = 280

Edit1

- Name = edtHostName
- Text = 127.0.0.1
- Top = 24
- Left = 56
- Width = 209
- Color = clBtnFace
- Font.Color = clNavy

Edit2

- Name = edtTimeOut
- Text = 4000
- Top = 24
- Left = 280
- Width = 57
- Color = clBtnFace
- Font.Color = clNavy

SpinEdit1

- Name = sedtStartPort
- Value = 1
- Top = 80
- Left = 56
- Width = 57
- Color = clBtnFace
- Font.Color = clNavy

SpinEdit2

- Name = sedtFinalPort
- Value = 65535
- Top = 80
- Left = 120
- Width = 57
- Color = clBtnFace
- Font.Color = clNavy

Button1

- Name = btnVerify
- Top = 11
- Left = 342
- Caption = &Verify

Button2

- Name = btnAbort
- Top = 35
- Left = 342
- Caption = &Abort

Gauge1

- Name = gBar
- Top = 80
- Left = 280
- Width = 113
- Color = clSilver
- ForeColor = clWhite
- Kind = gkHorizontalBar

Memo1

- Name = mInfo
- Top = 112
- Left = 8
- Width = 420
- Font.Color = clNavy
- ReadOnly = True
- Lines = ::Application started

Animate1

- Name = AnimaDef
- Top = 0
- Left = 0
- CommonAVI = aviFindComputer

8. Em tsPortScanner insira um Panel e altere as seguintes propriedades:

- Name = pnlPortScan
- Align = alClient

- Caption = (em branco)
- Color = clWhite

9. Insira dentro de pnlPortScan 2 Labels, 1 ListView, e 1 Animate e altere as seguintes propriedades que seguem abaixo:

Animate1

- Name = AnimaScan
- Top = 0
- Left = 0
- CommonAVI = aviFindComputer

Label1

- Name = lblTimeRemaining
- Top = 8
- Left = 56
- Caption = Time remaining in seconds :

Label2

- Name = lblTime
- Top = 8
- Left = 56
- Caption = not available

ListView1

- Name = lvPortScan
- Top = 42
- Left = 0
- ViewStyle = vsReport
- Columns[0].Caption = Service
- Columns[0].Width = 95
- Columns[1].Caption = Port
- Columns[1].Width = 40
- Columns[0].Caption = Description

- Columns[0].Width = 180
- Columns[0].Caption = Proto
- Columns[0].Width = 40
- Columns[0].Caption = Status
- Columns[0].Width = 55

9. tsAbout não tem nenhuma aplicação efetiva no desenvolvimento, deixo à cargo do leitor usar ou não esta tabsheet. Insira em frmMain um StatusBar e altere as seguintes propriedades:

- Name = sBar
- Align = alBottom
- Panels[0].Text = Ready
- Panels[0].Width = 60
- Panels[0].Text = Application started
- Panels[0].Width = 120
- Panels[0].Text = Not processing
- Panels[0].Width = 50

10. Agora iremos inserir em frmMain o componente que fará o trabalho pesado em nossa aplicação, o Powersock. Apenas altere sua propriedade Name para Scan.

O form principal já está pronto. Agora adicione um novo form ao projeto e nomeio como frmData e salve-o com o nome de uData na mesma pasta onde foi salvo o arquivo de projeto. Este form guardará informações sobre as portas, que respectivamente falando são: serviço, número da porta, e comentário sobre a porta, além de guardar também os domínios existentes para sites na Internet.

As informações serão guardadas em Memos comuns, para não termos que utilizar tabelas e nem banco de dados, poupando o uso do BDE. Isso vai aumentar um pouco o tamanho do programa, mas se quisermos rodar nossa aplicação em um computador que não tenha Delphi, não vamos precisar instalar os 4 Megas da dll do BDE para bancos de dados. Após o design, frmData deverá ficar com a seguinte aparência:

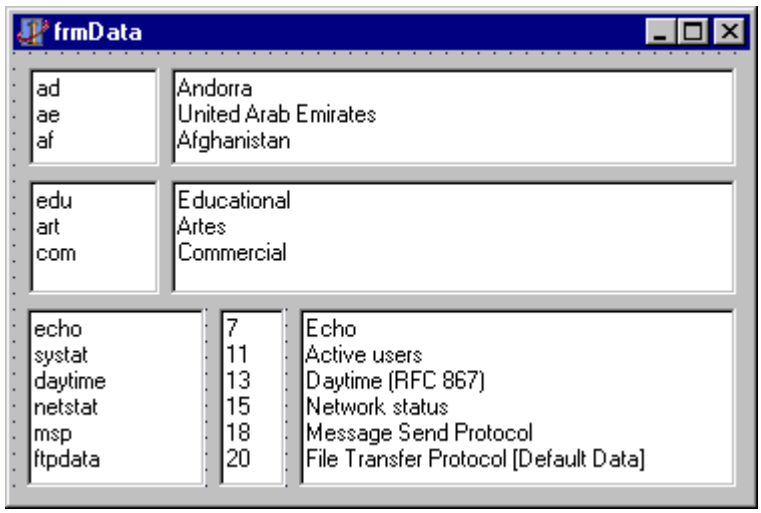


Figura 35 frmData.

1. Insira 7 Memos em frmData e configure suas propriedades conforme à seguir:

Memo1

- Name = mExt
- Lines = (começando de “ad” e sempre com uma sigla por linha do memo sem tirar da ordem)

ad	bj	cu	ga	ie	lc	mu	pf	sm	tw
ae	bm	cv	gd	il	li	mv	pg	sn	tz
af	bn	cx	ge	in	lk	mw	ph	so	ua
ag	bo	cy	gh	iq	lr	mx	pk	sr	ug
al	br	cz	gi	ir	ls	my	pl	st	uk
am	bs	de	gl	is	lt	mz	pt	su	us
an	bt	dj	gp	it	lu	na	pr	sv	uy
ao	bw	dk	gq	jm	lv	nc	py	sy	uz
aq	by	dm	gf	jo	ly	ne	qa	sz	va
ar	bz	do	gm	jp	ma	nf	ro	tc	vc
at	ca	dz	gn	ke	mc	ng	ru	td	ve
au	cf	ec	gr	kg	md	ni	rw	tg	vg
aw	cg	ee	gt	kh	mg	nl	sa	th	vi
az	ch	eg	gu	ki	mh	no	sb	tj	vn
ba	ci	es	gw	km	ml	np	sc	tk	vu
bb	ck	et	gy	kp	mn	nr	sd	tm	wf
bd	cl	fi	hk	kr	mo	nt	se	tn	ws
be	cm	fj	hn	kw	mp	nu	sg	to	ye
bf	cn	fk	hr	ky	mq	nz	sh	tp	yu
bg	co	fm	ht	kz	mr	om	si	tr	za
bh	cr	fo	hu	la	ms	pa	sk	tt	zm
bi	cs	fr	id	lb	mt	pe	sl	tv	zr
									zw

Figura 36 Domínios de países.

Memo2

- Name = mExtName
- Lines = (começando de “Andorra” e sempre com um país por linha do memo sem tirar da ordem)

Andorra	Guiana Francesa	Nauru
United Arab Emirates	Gambia	Zona Neutra
Afghanistan	Guinea	Niue
Antigua and Barbuda	Greece	New Zeland
Albania	Guatemala	Oman
Armenia	Guam (US)	Panama
Netherland Antilles	Guinea-Bissau	Peru
Angola	Guyana	French Polynesia
Antarctica	Hong Kong	Papua-New Guinea
Argentina	Honduras	Filipines
Austria	Croatia	Pakistan
Australia	Haiti	Poland
Aruba	Hungary	Portugal
Azerbaidjan	Indonesia	Porto Rico (US)
Bosnia-Herzegovina	Ireland	Paraguai
Barbados	Israel	Qatar
Bangladesh	India	Romania
Belgium	Iraq	Russian Federation
Burkina Fasso	Iran	Ruanda
Bulgaria	Iceland	Saudi Arabia
Bahrein	Italy	Salomon Islands
Burundi	Jamaica	Seychelles Islands
Benin	Jordan	Sudan
Bermuda	Japan	Sweden
Brunei	Kenya	Singapura
Bolivia	Kyrgystan	Saint Helena
Brazil	Cambodia	Slovenia
Bahamas	Kiribati	Slovak Republic
Bhutan	Comoros Islands	Sierra Leoa
Botswana	North Korea	San Marino
Belarus	South Korea	Senegal
Belize	Kuwait	Somalia
Canada	Cayman Islands	Suriname
Central African Republic	Kazachstan	Saint Tome and Principe
Congo	Laos	Former USSR
Switzerland	Lebanon	El Salvador
Ivory Coast	Saint Lucia	Syria
Cook Islands	Liechtenstein	Swaziland
Chile	Sri Lanka	Turks and Caicos Islands
Cameroon	Liberia	Chade
China	Lesotho	Togo
Colombia	Lithuania	Tailand
Costa Rica	Luxembourg	Tadjikistan
Tchecoslováquia	Latvia	Tokelau Islands
Cuba	Libya	Turkmenistan
Cape Verde	Marrocos	Tunisia
Christmas Island	Mônaco	Tonga
Chipre	Moldova	Timor
Czech Republic	Madagascar	Turkey
Germany	Marshall Islands	Trinidad e Tobago
Djibouti	Mali	Tuvalu

Denmark	Mongolia	Taiwan
Dominica	Macau	Tanzania
Dominican Republic	Marianas of Norte Islands	Ukraine
Algeria	Martinica (FR)	Uganda
Ecuador	Mauritania	United Kingdom
Estonia	Montserrat	United States
Egypt	Malta	Uruguai
Spain	Mauritius	Uzbekistan
Ethiopia	Maldivas	Vatican City
Finland	Malawi	São Vicente e Granadinas
Fiji	México	Venezuela
Falkland Island (Maldivas)	Malaysia	Virgin Islands (UK)
Micronesia	Mozambique	Virgin Islands (US)
Faroe Islands	Namibia	Vietnan
France	New Caledonia (FR)	Vanuatu
Gabon	Niger	Wallis and Futuna Islands
Grenada	Norfolk Islands	Western Samoa
Georgia	Nigeria	Yemen
Ghana	Nicaragua	Yugoslavia
Gibraltar	Netherlands	South Africa
Greenland	Norway	Zambia
Guadalupe (FR)	Nepal	Zaire
Equatorial Guinea		Zimbabwe

Figura 37 Nomes dos países referentes aos domínios.

Já deu para entender como vai funcionar nosso banco de dados ? É muito simples. Cada linha do memo mExt faz referência à uma linha do memo mExtName. mExt contém os nomes de domínios e mExtName contém os nomes dos países desses domínios, por exemplo: **fr** é um domínio da França, **pt** é de Portugal, assim como **br** é do Brasil

Memo3

- Name = mType
- Lines = (começando de “edu” e sempre com uma sigla por linha do memo sem tirar da ordem)

edu	bio
art	cng
com	cnt
esp	ecn
g12	eng
gov	eti
ind	fot
inf	fst
mil	jor
net	lel
org	med
psi	ntr
rec	odo

tmp	ppg
tur	pro
tv	psc
etc	slg
adm	vet
adv	zlg
arq	nom

Figura 38 Tipo do domínio.

Memo4

- Name = mTypeName
- Lines = (começando de "edu" e sempre com uma sigla por linha do memo sem tirar da ordem)

Educational	Biólogos
Artes	Cenógrafos
Commercial	Contadores
Esporte	Economistas
Ensino de Ensino de 1º e 2º Grau	Engenheiros
Government	Tecnologia da Informação
Indústria	Fotógrafos
Meios de Informação	Fisioterapeutas
Military	Jornalistas
Telecommunication	Leiloeiros
Non-Profit Organization	Médicos
Provedor de Internet	Nutricionistas
Entretenimento	Dentistas
Evento Temporário	Propaganda e Marketing
Turismo	Professores
Radiodifusão	Psicólogos
Não Categorizado	Sociólogos
Administradores	Veterinários
Advogados	Zoólogos
Arquitetos	Pessoas Físicas

Figura 39 Nomes dos tipos de domínio.

Memo5

- Name = mServiceTCP
- Lines = (começando de "echo" e sempre com uma palavra por linha do memo sem tirar da ordem)

echo	link
systat	supdup
daytime	hostnames
netstat	iso-tsap
msp	x400
ftpdata	x400-snd

ftp	pop/pop2
telnet	pop3
smtp	sunrpc
time	auth
name	sftp
whois	path/uucp-path
domain	nntp
bootps	ntp
bootpc	nbname
tftp	nbsession
gopher	nbdatagram
rje	snmp
finger	snmp-trap
http	nfs

Figura 40 Tipos de serviços TCP.

Memo6

- Name = mPortsTCP
- Lines = (começando de "7" e sempre com um número por linha do memo sem tirar da ordem)

7	87
11	95
13	101
15	102
18	103
20	104
21	109
23	110
25	111
37	113
42	115
43	117
53	119
67	123
68	137
69	139
70	148
77	161
79	162
80	2049

Figura 41 Portas TCP.

Memo7

- Name = mCommentsTCP
- Lines = (começando de "7" e sempre com um número por linha do memo sem tirar da ordem)

Echo	Link TTY
Active users	SUPDUP Protocol
Daytime (RFC 867)	SRI-NIC Host Name Server
Network status	ISSO-TSAP
Message Send Protocol	E-Mail Service X.400
File Transfer Protocol [Default Data]	E-Mail Send Service X.400
File Transfer Protocol [Control]	postoffice Post Office Protocol
Telnet	postoffice Post Office Protocol Version 3
Simple Mail Transfer Protocol	SUN Remote Procedure Call
timserver Time	Authentication Service
Host Name Server	Secure File Transfer Protocol (SFTP)
whois Who Is (NIC)	UUCP Path Service
Domain Name Server (DNS)	Network News Transfer Protocol (USENET)
DHCP | Bootstrap Protocol Server	Network Time Protocol
DHCP | Bootstrap Protocol Client	NetBIOS Name
Trivial File Transfer (TFTP)	Session Service for NetBIOS
Gopher	NetBIOS Datagram
RJE Remote Job Entry	SNMP
Finger	SNMP Interceptation
World Wide Web HTTP	Network File System

Figura 42 Significado dos serviços TCP.

Pronto. A questão do design de nossa aplicação está pronta, o próximo passo será partir para a codificação de nosso Port Scanner.

Programando

Esta é a parte em que ensinamos o programa a fazer realmente o port scanner. Vamos começar pelas variáveis globais que iremos utilizar na aplicação. Vá até a cláusula **var** de frmMain e adicione as seguintes linhas:

```
bAbort: Boolean = False;
iStartPort, iFinalPort: Integer;
iTimeOut, iTotalTime: Integer;
```

A variável bAbort irá controlar o processamento. Se for True, o processo será abortado. iStartPort e iFinalPort irão guardar a porta inicial que irá começar o port scanner e a porta final onde será encerrado o port scanner, que serão digitados em sedtStartPort e sedtFinalPort respectivamente. iTimeOut guardará o valor digitado em edtTimeOut, que por padrão deixaremos como 4000 milisegundos (4 segundos). iTotalTime guardará o tempo total restante para finalizar o port scanner.

edtHostNameChange

Adicione agora o seguinte código no evento OnChange de edtHostName:

```
procedure TfrmMain.edtHostNameChange(Sender: TObject);
var
  i, PosPonto: Integer;
  Server, Ext1, Ext2, Inverso: String;
begin
  //Verifica se edtHostName está vazio
  //e habilita ou desabilita o btnVerify
  if(edtHostName.Text <> '')then
    btnVerify.Enabled := True
  else
    btnVerify.Enabled := False;

  //Limpa lblDomain
  lblDomain.Caption:= '';

  //Se edtHostName conter mais que 4 caracteres
  //entra na rotina para tentar identificar o domínio
  if(length(edtHostName.Text)>4)then
  begin
    //Var Server recebe o conteúdo de edtHostName
    //para manipulação
    Server:= edtHostName.Text;

    //Var Inverso recebe o último caractere da var Server
    //e entra em looping até que seja completo com o inverso
    //da var server. Ex: server  = www.ig.com.br
    //                   inverso = rb.moc.gi.www
    Inverso:= Copy(Server, length(Server),1);
    for i:=1 to length(Server)-1 do
      Inverso:= Inverso + Copy(Server, length(Server)-i,1);

    //Verifica em que posição está o ponto
    PosPonto:= Pos('.', Inverso);

    //Domínio sem país. Ex: www.cnn.com, www.hgt.mil
    if(PosPonto=4)then
```

```
        begin
          //Ext2 recebe a extensão invertendo-a para o
          //lado normal
          Ext2:= Copy(Inverso, 3, 1);
          Ext2:= Ext2 + Copy(Inverso, 2, 1);
          Ext2:= Ext2 + Copy(Inverso, 1, 1);

          //Entra no looping até achar Ext2 em uma linha
          //de mType no frmData. Se achar pega nome da Ext2
          //em mTYpeName e coloca em lblDomain
          for i:=0 to 39 do
            if(Ext2=frmData.mType.Lines[i])then
            begin
              lblDomain.Caption:= frmData.mTypeName.Lines[i];
              Break;
            end;
        end

        //Dominio de país, podendo ter sequência
        else if(PosPonto=3)then
        begin
          //Pega a extensão do país
          Ext1:= Copy(Inverso, 2, 1);
          Ext1:= Ext1 + Copy(Inverso, 1, 1);
          Delete(Inverso, 1, PosPonto);

          //Entra no looping até achar Ext1 em uma linha
          //de mExt no frmData. Se achar pega nome da Ext1
          //em mExtName e coloca em lblDomain
          for i:=0 to 220 do
            if(Ext1=frmData.mExt.Lines[i])then
            begin
              lblDomain.Caption:= frmData.mExtName.Lines[i];
              Break;
            end;

          //Pega onde está o ponto
          PosPonto:= Pos('.', Inverso);
          //Se estiver na posição 4 então tem sequência
```

```
            if(PosPonto=4)then
            begin
              //Ext2 recebe a extensão invertendo-a para o
              //lado normal
              Ext2:= Copy(Inverso, 3, 1);
              Ext2:= Ext2 + Copy(Inverso, 2, 1);
              Ext2:= Ext2 + Copy(Inverso, 1, 1);
              //Entra no looping até achar Ext2 em uma linha
              //de mType no frmData. Se achar pega nome da Ext2
              //em mTYpeName e coloca em lblDomain
              for i:=0 to 39 do
                if(Ext2=frmData.mType.Lines[i])then
                begin
                  lblDomain.Caption:=  lblDomain.Caption  +  '  -  '  +
frmData.mTypeName.Lines[i];
                  Break;
                end;
            end;
          end;
        end;
      end;
```

É um código bastante grande mas na prático serve para uma coisa: mostrar o domínio de um site que queira ser verificado pelo port scanner. Compile o programa e digite em edtHostName: www.site.gov.de, obteremos o seguinte resultado em lblDomain mostrado abaixo:

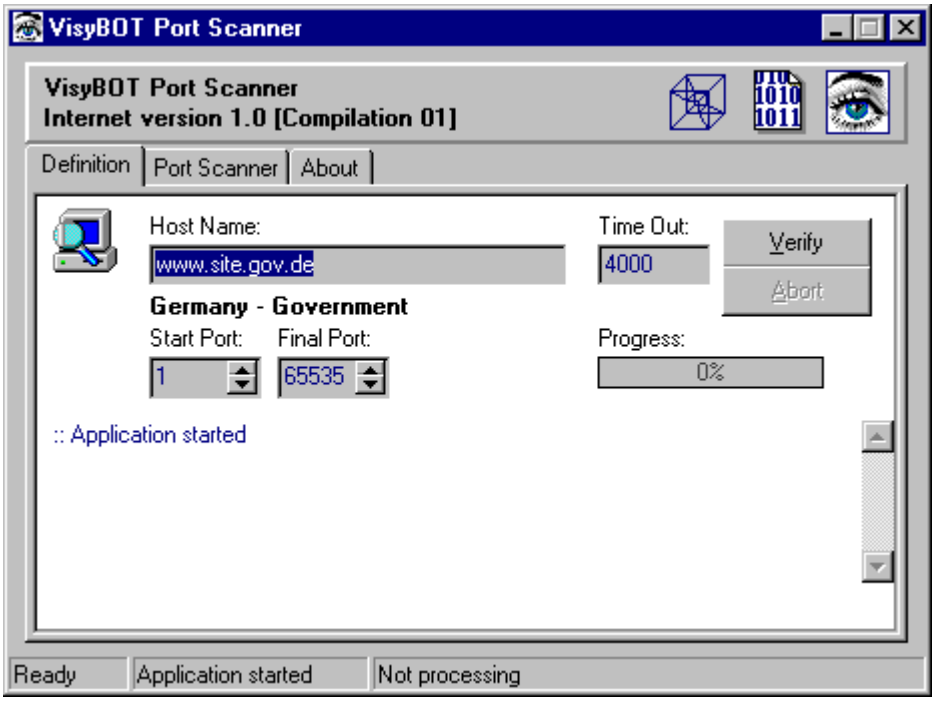


Figura 43 VisyBOT Port Scanner.

O domínio **gov** siginifica que é um site governamental, e o domínio **de** significa que é da Alemanha, por isso o resultado **Germany – Government** em lblDomain.

edtTimeOutChange

O valor de edtTimeOut será adicionado à propriedade TimeOut de Powersock Scan. Este é o tempo que Scan irá aguardar antes de lançar um erro quando estiver processando na rede.

sedtStartPortChange

No valor de entrada de sedtStartPort temos que ficar atentos para não deixar que o usuário digite um valor de porta maior que o valor da porta final em sedtFinalPort.

Adicione o seguinte código no evento OnChange de sedtStartPort:

```
      procedure TfrmMain.sedtStartPortChange(Sender: TObject);
      begin
        if(sedtStartPort.Text = '')then
          sedtStartPort.SetFocus
        else if(StrToInt(sedtStartPort.Text) >
StrToInt(sedtFinalPort.Text))then
          sedtStartPort.Text := '1';
```

```
      end;
```

sedtFinalPortChange

No valor de entrada de sedtFinalPort temos que ficar atentos para não deixar que o usuário digite um valor de porta menor que o valor da porta inicial em sedtStartPort.

Adicione o seguinte código no evento OnChange de sedtFinalPort:

```
      procedure TfrmMain.sedtFinalPortChange(Sender: TObject);
      begin
        if(sedtFinalPort.Text = '')then
          sedtFinalPort.SetFocus

        else if(StrToInt(sedtFinalPort.Text) <
StrToInt(sedtStartPort.Text))then
          sedtFinalPort.Text := '65535';
      end;
```

btnVerifyClick

Insira o seguinte código no evento OnClick de btnVerify:

```
      procedure TfrmMain.btnVerifyClick(Sender: TObject);
      begin
        //Limpa os ítems do port scanner
        lvPortScan.Items.Clear;

        //Zera a barra de progresso
        gBar.Progress := 0;

        //Desabilita os campos
        edtHostName.Enabled := False;
        edtTimeOut.Enabled := False;
        sedtStartPort.Enabled := False;
        sedtFinalPort.Enabled := False;
        btnVerify.Enabled := False;

        //Habilita btnAbort
```

```
      btnAbort.Enabled := True;

      //Guarda nas var's os valores das portas
      iStartPort := StrToInt(sedtStartPort.Text);
      iFinalPort := StrToInt(sedtFinalPort.Text);

      //Ativa os animates
      AnimaDef.Active := True;
      AnimaScan.Active := True;

      //Guarda o time out
      iTimeOut := StrToInt(edtTimeOut.Text);

      //Calcula o tempo total para o scan
      if((iFinalPort - iStartPort) > 0)then
      begin
        iTotalTime := (iFinalPort - iStartPort) * iTimeOut;
        lblTime.Caption := IntToStr(iTotalTime div 1000);
      end
      else
        lblTime.Caption := 'not available';

      //valor máximo do gauge é igual ao iTotalTime
      gBar.MaxValue := iTotalTime;

      //Atualiza o memo mInfo e a barra de status sBar
      mInfo.Lines.Add('Port scanner started');
      sBar.Panels[0].Text := 'Working...';
      sBar.Panels[1].Text := 'Progress: 0%';
      sBar.Panels[2].Text := 'Scanning process';

      //Configura o Powersock para começar o port scanner
      Scan.TimeOut := iTimeOut;
      Scan.Host := edtHostName.Text;
      Scan.Port := iStartPort;
      Scan.Connect;
    end;
```

btnVerify irá dar o início ao nosso port scanner, observe as seguintes linhas:

```
        //Configura o Powersock para começar o port scanner
        Scan.TimeOut := iTimeOut;
        Scan.Host := edtHostName.Text;
        Scan.Port := iStartPort;
        Scan.Connect;
```

Aqui é configurado o componente Powersock. Em sua propriedade TimeOut adicionamos o valor de iTimeOut. Em sua propriedade Host adicionamos o nome ou IP do computador que foi adicionado em edtHostName o qual queremos “scannear”. Em sua propriedade Port adicionamos o valor de iStartPort, e, finalmente, ativamos o método Connect, para que o componente tente se conectar ao host remoto e iniciar nosso port scanner.

Agora iremos adicionar 4 procedures privadas em nossa aplicação. Insira as seguintes linhas embaixo da cláusula **private**:

```
    procedure AppException(Sender: TObject; E: Exception);
    procedure Aborted;
    procedure ScanPorts;
    procedure ScanCompleted;
```

A procedure AppException será chamada sempre que um erro ocorrer, ou seja, irá capturar todos os erros do programa. A procedure Aborted será chamada se o processo de verificação das portas for abortado. A procedure ScanPorts será chamada sampre que formos scannear uma nova porta. E a procedure ScanCompleted, como o próprio nome diz, será chamada quando o port scanner estiver completo.

Vamos para a codificação desses procedimentos. Adicione-as após a cláusula implementation.

AppException

```
    procedure TfrmMain.AppException(Sender: TObject; E: Exception);
    begin
      mInfo.Lines.Add('Error in the class system ' + E.ClassName +
' .:. ' + E.Message);
```

```
end;
```

ScanCompleted

```
procedure TfrmMain.ScanCompleted;
begin
  bAbort:= False;
  edtHostName.Enabled := True;
  edtTimeOut.Enabled := True;
  sedtStartPort.Enabled := True;
  sedtFinalPort.Enabled := True;
  btnVerify.Enabled := True;
  btnAbort.Enabled := False;
  AnimaDef.Active:= False;
  AnimaScan.Active:= False;
  tsDefinition.Visible:= True;
  pgcMain.ActivePageIndex:= 0;
  edtHostName.Setfocus;
  edtHostName.SelectAll;
  mInfo.Lines.Add('Scanner completed');
  sBar.Panels[0].Text := 'Ready';
  sBar.Panels[1].Text := 'Scanner completed';
  sBar.Panels[2].Text:= 'Not processing';
end;
```

ScanPorts

```
procedure TfrmMain.ScanPorts;
var
  iTimeInSec: Integer;
begin
  Application.ProcessMessages;

  //Atualiza tempo restante
  iTotalTime := iTotalTime - iTimeOut;
  iTimeInSec := iTotalTime div 1000;
  lblTime.Caption := IntToStr(iTimeInSec);

  //Atualiza o progresso
```

```
        gBar.AddProgress(iTimeOut);
        sBar.Panels[1].Text := 'Progress: ' +
IntToStr(gBar.PercentDone) + '%';

        Scan.Port := iStartPort;
        Scan.Connect;
      end;
```

Aborted

```
      procedure TfrmMain.Aborted;
      begin
        bAbort:= False;
        edtHostName.Enabled := True;
        edtTimeOut.Enabled := True;
        sedtStartPort.Enabled := True;
        sedtFinalPort.Enabled := True;
        btnVerify.Enabled := True;
        btnAbort.Enabled := False;
        AnimaDef.Active:= False;
        AnimaScan.Active:= False;
        tsDefinition.Visible:= True;
        pgcMain.ActivePageIndex:= 0;
        edtHostName.Setfocus;
        edtHostName.SelectAll;
        sBar.Panels[0].Text := 'Ready';
        sBar.Panels[1].Text := 'Aborted';
        sBar.Panels[2].Text:= 'Not processing';
      end;
```

Agora iremos codificar os eventos de nosso componente Powersock Scan. Adicione o seguinte código no evento OnConnect:

ScanConnect

```
      procedure TfrmMain.ScanConnect(Sender: TObject);
      var
        ListScan: TListItem;
        i: Integer;
```

```
        bFoundInMemo: Boolean;
      begin
        //Inicializa como false antes de verificar
        //se a porta existe no memo
        bFoundInMemo := False;

        //Verifica se o scan foi abortado
        Application.ProcessMessages;
        if(bAbort=True)then
        begin
          Aborted;
          Exit;
        end;

        //Verifica se a porta atual existe no memo
        for i := 0 to frmData.mPortsTCP.Lines.Count - 1 do
        begin
          if(iStartPort = StrToInt(frmData.mPortsTCP.Lines[i]))then
          begin
            bFoundInMemo := True;
            with lvPortScan do
            begin
              ListScan := Items.Add;
              ListScan.Caption := frmData.mServiceTCP.Lines[i];
              ListScan.SubItems.Add(frmData.mPortsTCP.Lines[i]);
              ListScan.SubItems.Add(frmData.mCommentsTCP.Lines[i]);
              ListScan.SubItems.Add('TCP');
              ListScan.SubItems.Add('OPEN');
              ListScan.MakeVisible(false); //follow the scan as it
scrolls
            end;
          end;
        end;

        //Se não achou a porta no memo
        //adicionar na list sem descrição
        if(bFoundInMemo = False)then
        begin
          with lvPortScan do
```

```
          begin
            ListScan := Items.Add;
            ListScan.Caption := 'Unknown';
            ListScan.SubItems.Add(IntToStr(iStartPort));
            ListScan.SubItems.Add('Unknown');
            ListScan.SubItems.Add('TCP');
            ListScan.SubItems.Add('OPEN');
            ListScan.MakeVisible(false); //follow the scan as it
scrolls
          end;
        end;

        //Encerra a conexão
        Scan.Disconnect;

        //Se não terminou o scan
        //incrementa iStartPort e
        //roda o ScanPorts de novo
        if(iStartPort < iFinalPort)then
        begin
          iStartPort := iStartPort + 1;
          ScanPorts;
        end
        //Se terminou, roda ScanCompleted
        else
          ScanCompleted;
      end;
```

Adicione o seguinte código no evento OnConnectionFailed:

ScanConnectionFailed

```
      procedure TfrmMain.ScanConnectionFailed(Sender: TObject);
      var
        ListScan: TListItem;
        i: Integer;
        bFoundInMemo: Boolean;
      begin
        //Inicializa como false antes de verificar
```

```
        //se a porta existe no memo
        bFoundInMemo := False;

        //Verifica se o scan foi abortado
        Application.ProcessMessages;
        if(bAbort=True)then
          begin
            Aborted;
            Exit;
          end;

        //Verifica se a porta atual existe no memo
        for i := 0 to frmData.mPortsTCP.Lines.Count - 1 do
        begin
          if(iStartPort = StrToInt(frmData.mPortsTCP.Lines[i]))then
          begin
            bFoundInMemo := True;
            with lvPortScan do
            begin
              ListScan := Items.Add;
              ListScan.Caption := frmData.mServiceTCP.Lines[i];
              ListScan.SubItems.Add(frmData.mPortsTCP.Lines[i]);
              ListScan.SubItems.Add(frmData.mCommentsTCP.Lines[i]);
              ListScan.SubItems.Add('TCP');
              ListScan.SubItems.Add('CLOSED');
              ListScan.MakeVisible(false); //follow the scan as it
scrolls
            end;
          end;
        end;

        //Se não achou a porta no memo
        //adicionar na list sem descrição
        if(bFoundInMemo = False)then
        begin
          with lvPortScan do
          begin
            ListScan := Items.Add;
            ListScan.Caption := 'Unknown';
```

```
            ListScan.SubItems.Add(IntToStr(iStartPort));
            ListScan.SubItems.Add('Unknown');
            ListScan.SubItems.Add('TCP');
            ListScan.SubItems.Add('CLOSED');
            ListScan.MakeVisible(false); //follow the scan as it
scrolls
          end;
        end;

        //Encerra a conexão
        Scan.Disconnect;

        //Se não terminou o scan
        //incrementa iStartPort e
        //roda o ScanPorts de novo
        if(iStartPort < iFinalPort)then
        begin
          iStartPort := iStartPort + 1;
          ScanPorts;
        end
        //Se terminou, roda ScanCompleted
        else
          ScanCompleted;
      end;
```

Adicione o seguinte código no evento OnInvalidHost:

ScanInvalidHost

```
      procedure TfrmMain.ScanInvalidHost(var Handled: Boolean);
      begin
        mInfo.Lines.Add('Invalid host, application aborted process');
        Aborted;
      end;
```

Não podemos esquecer de adicionar o seguinte código no evento OnCreate de frmMain, para que todos os erros que forem gerados no decorrer do programa sejam desviados para a procedure AppException:

TfrmMain.FormCreate

```
procedure TfrmMain.FormCreate(Sender: TObject);
begin
  Application.OnException := AppException;
end;
```

Pronto! Agora é só compilar e testar o VisyBOT!

Licença

Anexo I - Sockets – Uma Analogia.

Tradução de “Windows Sockets: A Quick And Dirty Primer”, by Jim Frost.
Disponível em: http://world.std.com/~jimf/papers/sockets/winsock.html

Introdução

Atualmente, onde as conexões entre redes de computadores crescem cada vez mais, os programadores se encontram na necessidade de escreverem programas que se comuniquem entre as redes. Mas a parte mais difícil não está em escrever o código, mas sim em entender o conceito por trás das redes. Essa introdução pretende fornecer a teoria necessária para que o programador consiga desenvolver aplicativos em rede.

Que diabos é um Socket ? (Uma analogia)

À uns 50 anos atrás, a ARPA, *Advanced Research Projects Agency of the Department of Defense*, assinou com a Universidade da California em Berkeley a responsibilidade de construir um sistema operacional que pudesse ser usado como uma plataforma padrão para suportar a ARPAnet, que foi o antecessor da atual Internet.

Berkeley, que já era bem conhecida por seu trabalho no UNIX, adicionou uma nova interface à esse sistema operacional para suportar a comunicação em rede. Essa interface é comumente conhecida como *Berkeley Sockets Interface* (Interface de Sockets de Berkeley) e é a base para quase todos os protocolos de interface de rede TCP/IP, incluindo os Sockets do Windows, comumente conhecido como WinSock.

Um Socket é muito parecido com um telefone, é o ponto final de um canal de comunicação de duas vias. Conectando-se dois sockets juntos, você pode passar dados entre processos, mesmo processos rodando em computadores diferentes, uma vez tendo feito uma ligação de seu telefone para o de outra pessoa.

A analogia ao telefone é muito boa, e será usada repetidamente para descrever os procedimentos de um socket, mas diferente do telefone existe uma distinção na terminologia entre programas que aceitam conexões e aqueles que fazem as conexões. Um Servidor é um programa que espera conexões e presumidamente provê algum serviço para outros programas. Ao contrário, um cliente é um programa que se conecta à um servidor, usualmente para pedir que este lhe faça alguma coisa. É importante lembrar que não é o computador que distingue qual é o cliente e qual é o servidor, mas o jeito que os programas usam os sockets. A maioria das pessoas acredita que “Servidor” significa Mainframe (computador de grande porte) e “Cliente” significa PC (Computador Pessoal). Isto não é necessariamente o caso e tem gerado muita confusão pois computadores pessoais freqüentemente trabalham como ambos, cliente e servidor, simultaneamente.

A Lista Telefônica da Internet (Resolução de Endereços)

Como com o telefone, todo socket tem um único endereço feito de duas partes diferentes.

A primeira parte é o **Endereço IP**, que é um número de quatro dígitos escritos na forma decimal separados por pontos (Ex: 200.231.13.13), que especifica à qual computador você quer conversar. Cada computador na Internet tem ao menos um endereço IP.

A segunda parte é o **Número da Porta**, permitindo que o computador possa falar com vários ao mesmo tempo, parecido com ramais em telefones de escritórios. Uma aplicação pode escolher qual número de porta usar (obviamente, alguns números são reservados) ou solicitar uma porta randomicamente quando associando um endereço à um socket.

Números são difíceis de lembrar, particularmente quando você tem que negociar com eles numa determinada base. Como é com o telefone, um serviço de pesquisa é oferecido, para que você possa se lembrar de um nome simples, por exemplo www.sockets.com.br, melhor do que um conjunto de dígitos como 192.168.0.17. Em sockets, freqüentemente são usadas funções que pegam o nome do computador e retornam seu endereço IP, assim como as que pegam o endereço IP e retornam o nome do computador.

Anexo II – Links.

PUC-Campinas	**http://www.puc-campinas.edu.br**
JAVA	**http://java.sun.com**
Borland	**http://www.borland.com**
Conectiva Linux	**http://www.conectiva.com.br**
Modulo Security	**http://www.modulo.com.br**
N-Stalker	**http://www.nstalker.com.br**
Microsoft Corp.	**http://www.microsoft.com.br**
Oracle Corp.	**http://www.oracle.com**
Red Hat	**http://www.redhat.com**
Unicamp Security	**http://www.security.unicamp.br**
Scope Systems	**http://www.scopesystems.com.br**
Siemens	**http://www.siemens.com.br**

Bibliografia

STEVENS, W. Richard. **TCP/IP Illustrated: the protocols**. Massachusetts, USA. Addison-Wesley Publishing Company, 2000.

CESTAROLLI, Guilherme Seleguim. **Perigos do Mundo Virtual**. Trabalho Análise de Sistemas – PUC-Campinas – 2002.

NORTHCUTT, Stephen. **Como detectar invasão em rede – um guia para analistas**. Rio de Janeiro, RJ. Editora Ciência Moderna Ltda., 2000.

PALMA-PRATES, Luciano-Rubens. **TCP/IP Guia de Consulta Rápida**. São Paulo, SP. Novatec Editora Ltda., 2000.

VEIGA, Roberto G. A. **Windows Script Host – Guia de Consulta Rápida**. São Paulo, SP. Novatec Editora Ltda., 2000.

DÉCIO, Décio Jr. **HTTP – Guia de Consulta Rápida**. São Paulo, SP. Novatec Editora Ltda., 2001.

FERNANDES, André. **Delphi 5 Básico e Avançado**. Editora Book Express Ltda., 1999.

CRISTOVÃO, Leandro. **Administração Remota em Delphi**. Florianópolis, SC. Visual Books Ltda., 2000.

CRISTOVÃO, Leandro. **Registry com Delphi**. Florianópolis, SC. Visual Books Ltda., 2000.

RIBEIRO, Sebastião Elivaldo. **Criando Componentes no Delphi**. Florianópolis, SC. Visual Books Ltda., 1999.

BURN, G.B. **Por Dentro do Registro do Windows NT**. São Paulo, SP. Editora Market Books do Brasil Ltda., 1999.

PC MASTER, Revista de Informática (Ano 3, Nº 9)**. Por dentro do e-mail**. São Paulo, SP. Editora Europa Ltda., 2000.

PRIMO, Francisco F. **Estrutura e Recuperação da Informação – Anotações de Aula.**. Campinas, SP., 2003.

BROWN, Steve. **Visual Basic 6-Bíblia do Programador**. São Paulo, SP. Editora Berkeley, 1999.