UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE SOFTWARE PARA MONITORAÇÃO DO CONTEÚDO DAS MENSAGENS EM UMA REDE TCP/IP

Aluno: Fernando Hilgenstieler

Orientador: Francisco Adell Péricas
Membros da banca: Sérgio Stringari
Mauro M. Mattos

Sumário

- Introdução
- A Internet e sua História
- A arquitetura TCP/IP
- Protocolos da arquitetura TCP/IP
- Segurança em Redes de Computadores
- Políticas e procedimentos de segurança
- Algumas Técnicas de invasão e defesa
- Desenvolvimento do Trabalho
- Conclusões

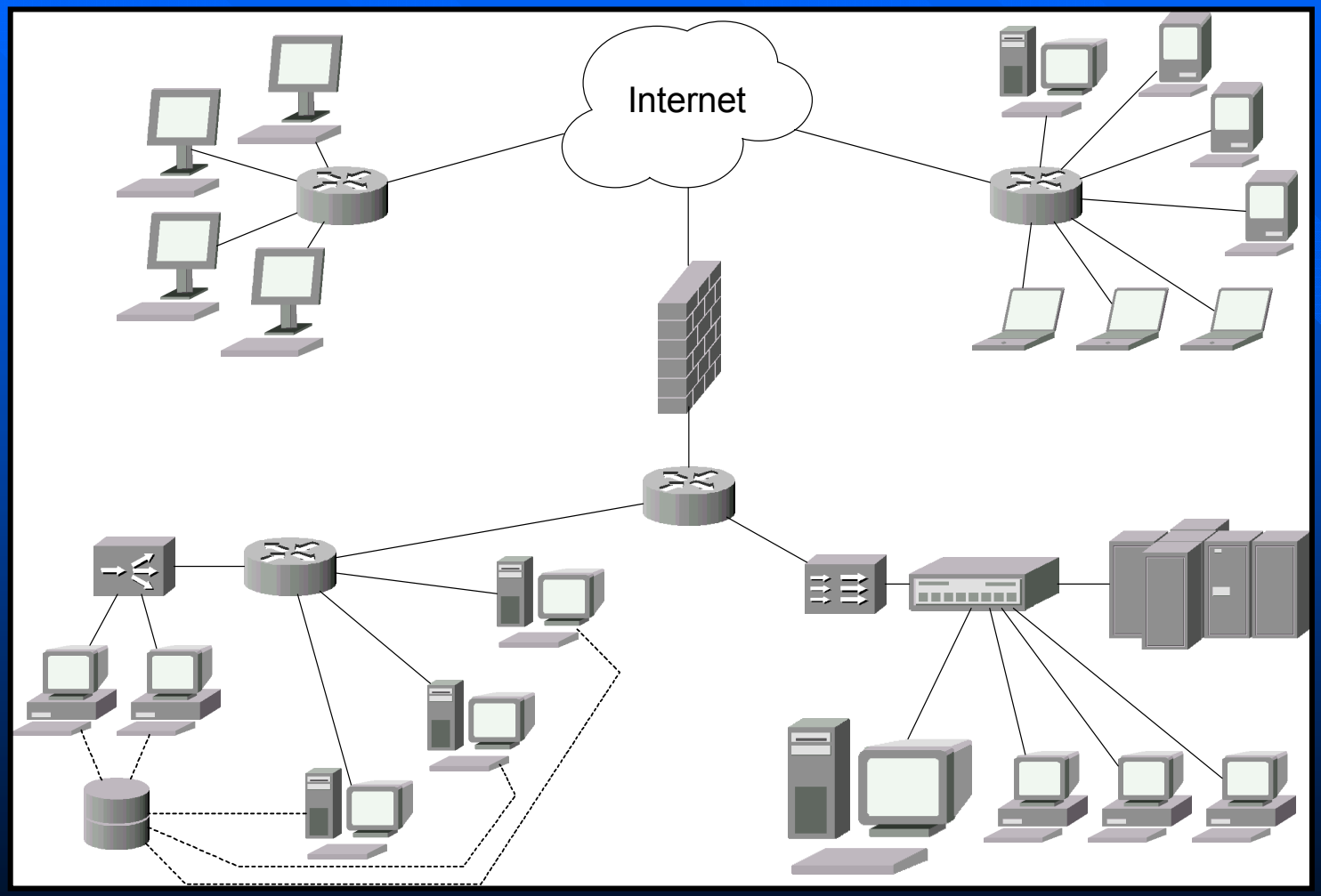
Introdução

Este trabalho apresenta a especificação e implementação de um protótipo de software para monitoração do protocolo HTTP da camada de aplicação em um computador conectado a uma rede *Ethernet*. Para este protótipo foi realizado um estudo sobre a arquitetura TCP/IP e sobre segurança de redes de computadores. O protótipo foi desenvolvido para o ambiente Windows

Objetivos

- interceptar e interpretar pacotes TCP/IP
- monitorar dados contidos nas mensagens: camada de rede, de transporte e de aplicação
- identificar situações suspeitas
- armazenar as informações monitoradas

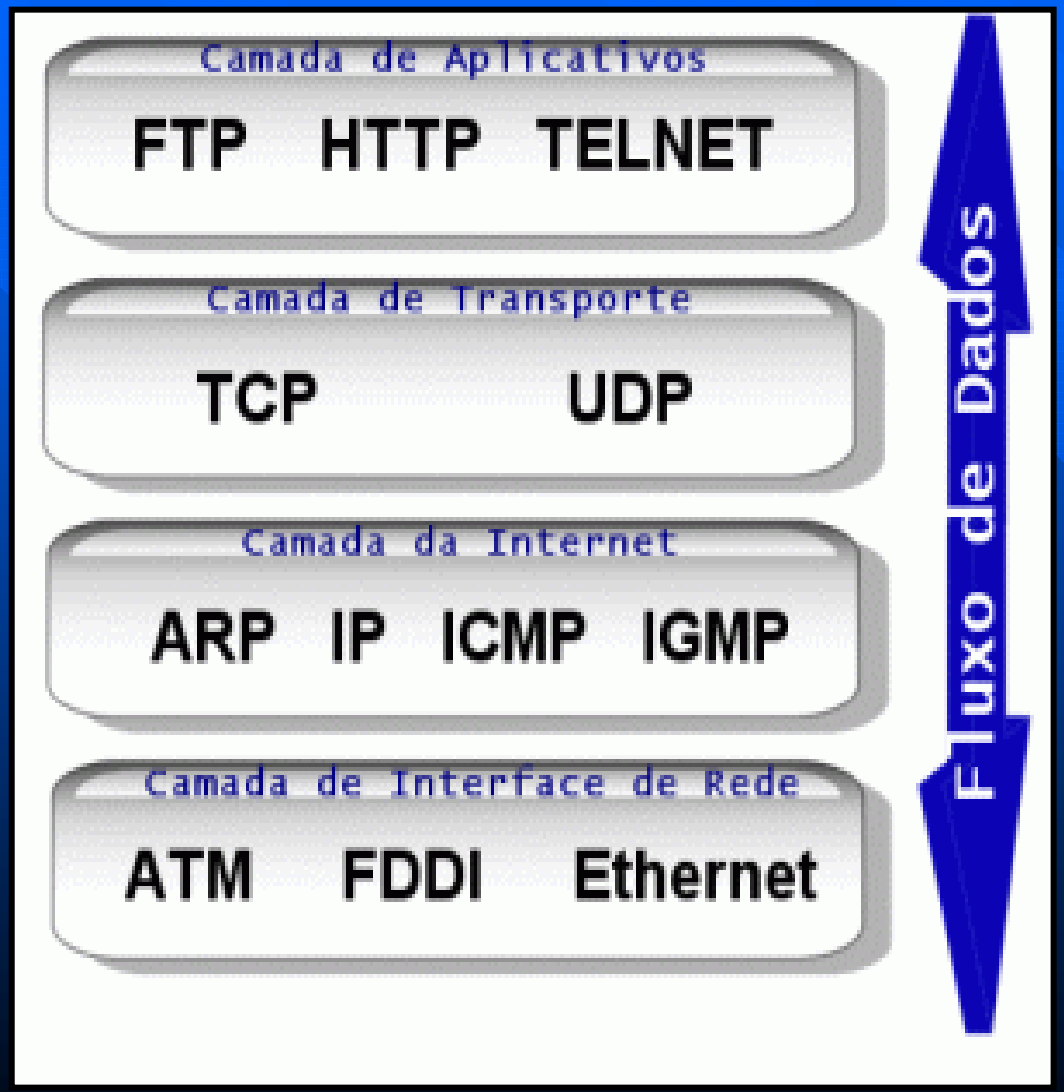
A Internet e sua História



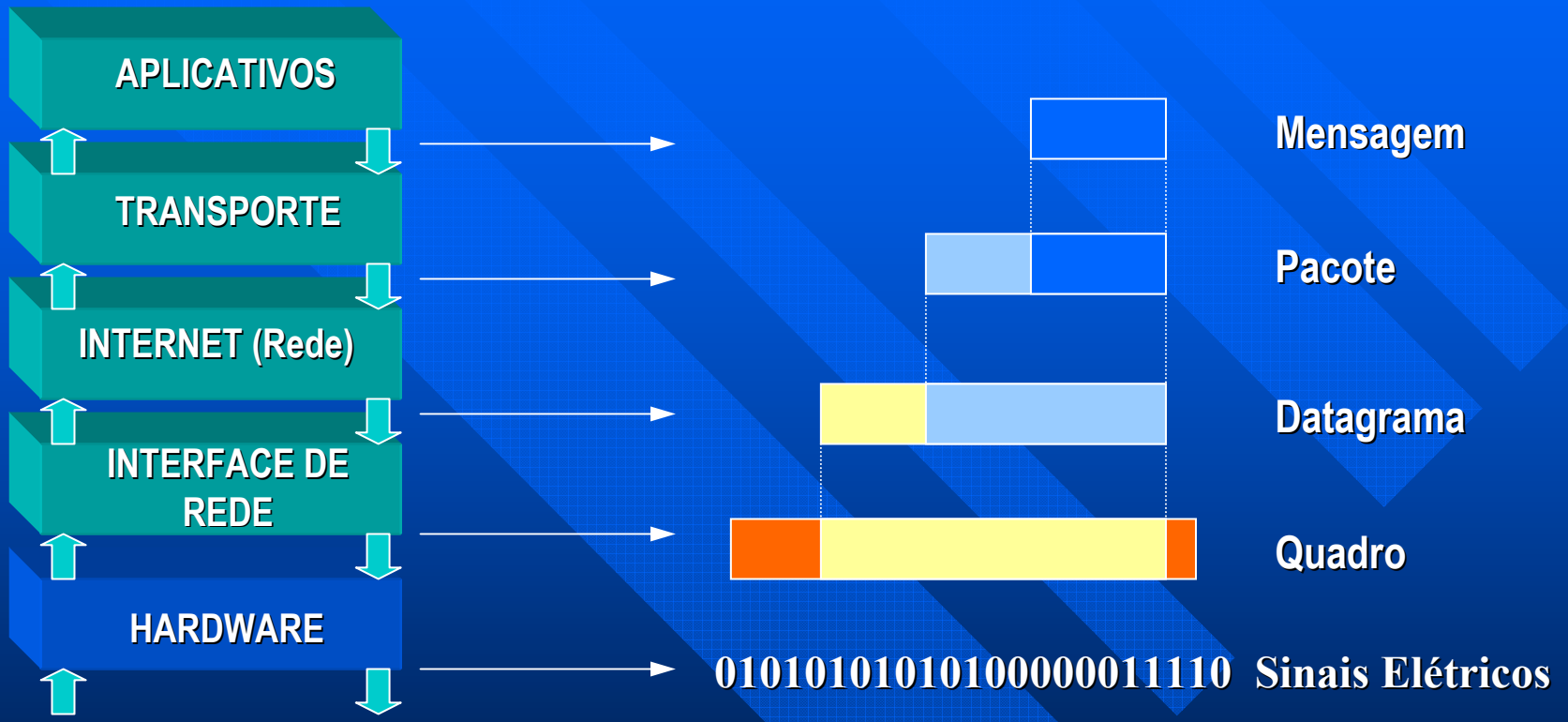
A estrutura da Internet

- *Backbones*: espinha dorsal das redes
- Provedor de acesso: conecta-se diretamente aos *Backbones*
- Usuários finais: conectados através de uma rede local ou uma conexão discada ao provedor de acesso

Protocolos da arquitetura TCP/IP



Modelo de Divisão em Camadas TCP/IP



Protocolo HTTP

- É um protocolo do nível de aplicação que possui objetividade e rapidez necessárias para suportar sistemas de informação distribuídos, cooperativos e de hipermídia

- Obedece ao paradigma de cliente/servidor: um cliente estabelece uma conexão com um servidor e envia um pedido ao servidor, o qual o analisa e responde. A conexão deve ser estabelecida antes de cada pedido de cliente e encerrada após a resposta

Protocolos da camada de Transporte

- TCP
  - É um protocolo orientado à conexão (sessão)
  - Verifica a existência de erros em cada pacote que recebe para evitar a corrupção dos dados
  - Confirma a entrega dos pacotes
- UDP
  - Não é um protocolo confiável
  - Não é baseado em sessão
  - Não confirma os pacotes enviados

Protocolo IP

- Designado para o uso em sistemas de computadores interconectados (redes), utilizando troca de pacotes
- Fornece transmissão de blocos de dados chamados de datagramas, entre origem e destino
- Origem e destino são *hosts* identificados por um endereço de tamanho fixo

Segurança em Redes de Computadores

Por mais seguro que um sistema operacional seja, as atitudes de um administrador sem treinamento formal ou até mesmo informal podem transformá-lo em uma fortaleza de portas escancaradas, pronto para receber todo tipo de ataque e invasões. Segurança de rede é uma preocupação que deve ser compartilhada por todos, inclusive pelo usuário final

Políticas e procedimentos de segurança

- Políticas de segurança com boas informações são o fundamento para um programa de segurança de informações eficaz
- Incorporar objetivos de segurança para sistemas ou aplicações específicas que definem como os sistemas deverão ser operados sob determinadas circunstâncias
- Procedimentos de resposta a incidentes, por exemplo, especificam o que usuários, administradores de sistema e gerentes devem fazer se alguém descobrir algum indício de um incidente relacionado a segurança

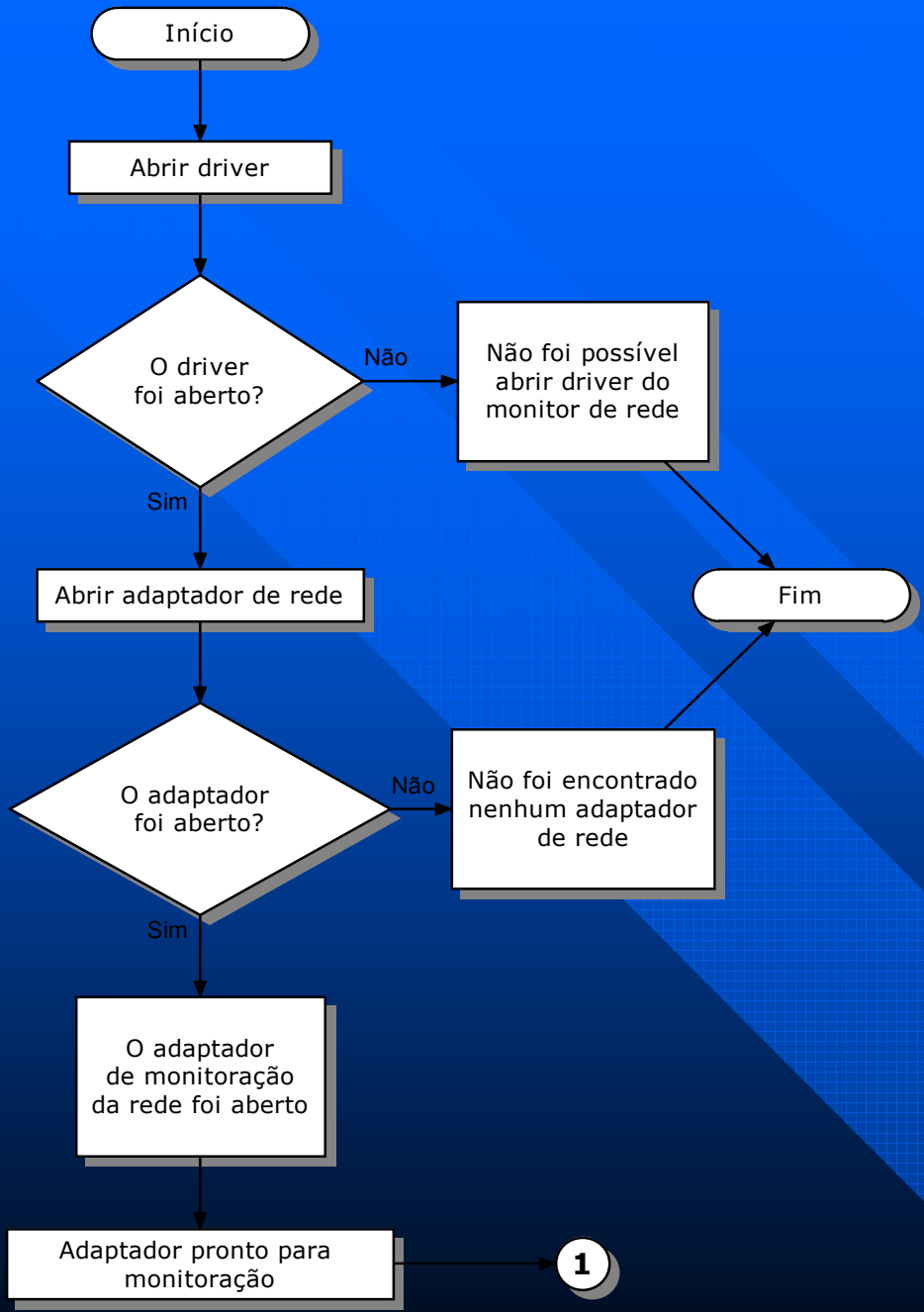
Algumas Técnicas de invasão e defesa

- Invasão
  - Cavalos de Tróia
  - DoS (*Denial of Service*)
- Defesa
  - *Firewalls*
  - Criptografia

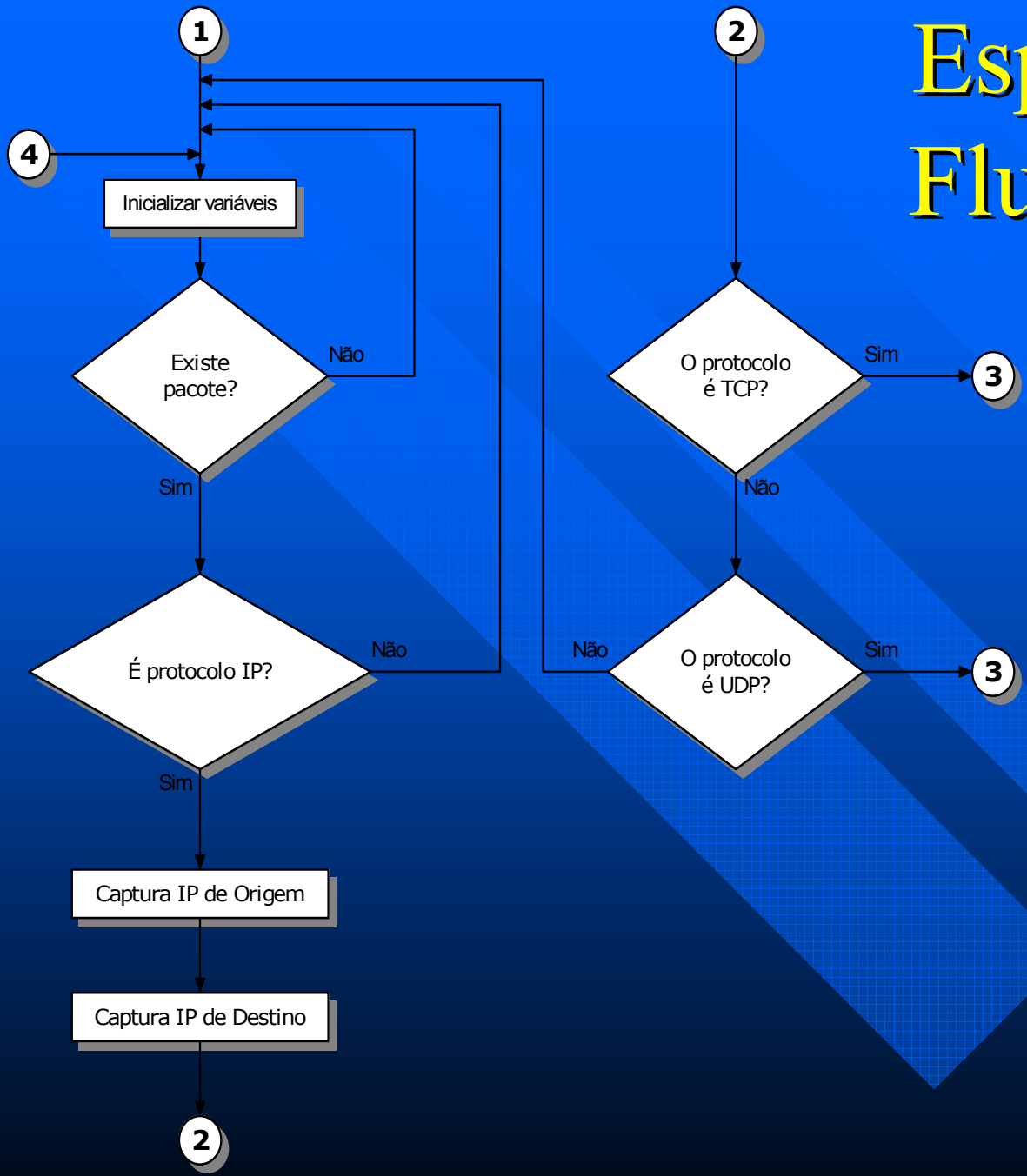
Requisitos principais do problema a ser trabalhado

- operar no ambiente (sistema operacional) Windows
- localizar o adaptador de rede
- iniciar a captura de pacotes de uma rede *Ethernet*
- monitorar os protocolos da arquitetura TCP/IP
- identificar o procotolo TCP e/ou UDP
- capturar e analisar os pacotes que trafegam pela camada de aplicação
- filtrar por um determinado *Host* (opcional)
- possibilitar a mudança dos filtros durante a execução do monitor
- no decorrer da execução, mostrar o conteúdo capturado ao usuário
- permitir que os dados capturados fossem salvos

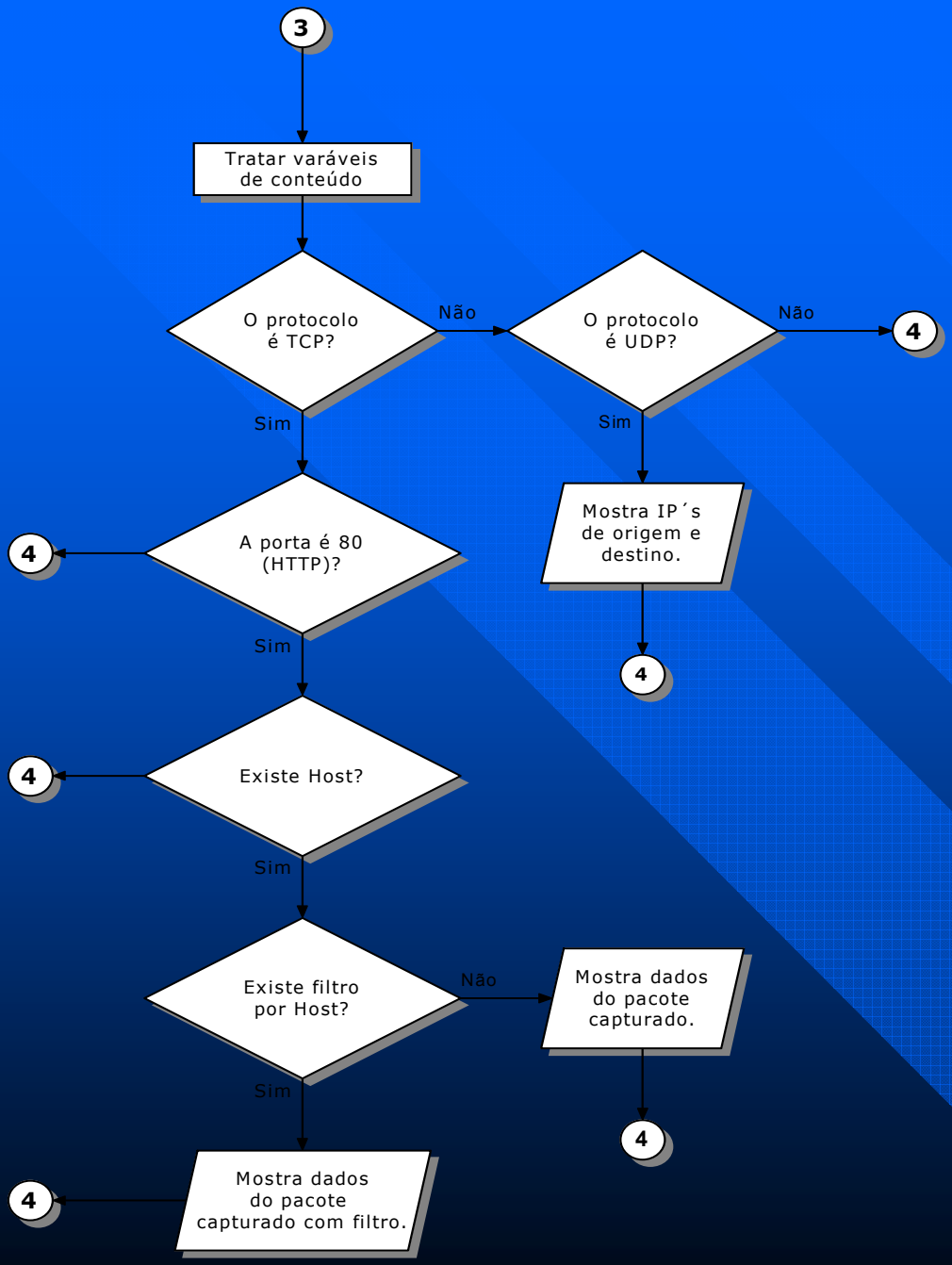
Especificação Fluxograma 1



Especificação Fluxograma 2



Especificação Fluxograma 3



Técnicas e ferramentas utilizadas

- Microsoft Windows
- Borland C++ Builder 6.0: ambiente de desenvolvimento
- WinPcap: biblioteca de acesso a interfaces de rede
- TJesNetMonitor: componente que tem funções encapsuladas da WinPcap
- EDGE Diagrammer: construção de fluxogramas

Implementação

Estabelecer conexão com o adaptador de rede

```
__fastcall Tfrm_Monitor::Tfrm_Monitor(TComponent* Owner): TForm(Owner)
{
  Parado = true;
  Iniciou = false;
  bb_Iniciar->Enabled = true;
  bb_Parar->Enabled = false;
  bb_Limpar->Enabled = true;
  AtualizaTempo->Interval = REFRESH_INTERVAL_S * 1000;
  if (!MonitorAplicacao->Open()
    BarraStatus->SimpleText = "O Monitor de Aplicação não pode ser inicializado!";
  else if (!MonitorAplicacao->SelectAdapter(0))
    BarraStatus->SimpleText = "Adaptador de rede não pode ser selecionado!"
  else
          edt_Adaptador->Text = MonitorAplicacao->get_adaptername(0);
       }
```

Implementação

Captura de pacotes e filtros IP, TCP e UDP

```
void __fastcall Tfrm_Monitor::CapturaPacote(TObject *Sender) {
  BYTE *ptr;
  BYTE *ipAddr;
  ETHERNET_FRAME *ePtr;
  char IP_Origem[20], IP_Destino[20];
  char EndOrigem[25], EndDestino[25];
  char PortaOrigem[6], PortaDestino[6];
  TJesNetPacket Pacote;
  while (MonitorAplicacao->get_packetcount() > 0) {
    short int etherHdrLen, ipHdrLen, tcpHdrLen, udpHdrLen;
    MonitorAplicacao->GetNextPacket(&Pacote);
    ptr = Pacote.get_pPacket();
    ePtr = (ETHERNET_FRAME *) ptr;
    if (XCHG(ePtr->FrameType) == 0x0800) // IP { {
        etherHdrLen = 14;
        IP_HEADER *ipPtr = (IP_HEADER *) &ptr[etherHdrLen];
        ipHdrLen = (ipPtr->x & (BYTE) 0x0f) << 2;
        ipAddr = (BYTE *) &nipper->src;
        sprintf(IP_Origem, "%d.%d.%d.%d", ipAddr[0], ipAddr[1], ipAddr[2],
                           ipAddr[3]);
        ipAddr = (BYTE *) &ipPtr->dest;
        sprintf(IP_Destino, "%d.%d.%d.%d", ipAddr[0], ipAddr[1], ipAddr[2],
                           ipAddr[3]);
```

Implementação

Captura de pacotes e filtros IP, TCP e UDP

```
if (ipPtr->protocol == 0x06) // TCP
        {
          TCP *pTCP = (TCP *) &ptr[etherHdrLen + ipHdrLen];
          tcpHdrLen = TCP_HdrLen(pTCP);
          sprintf(EndOrigem, "%s:%d", IP_Origem, TCP_SrcPort(pTCP));
          sprintf(PortaOrigem, "%d", TCP_SrcPort(pTCP));
          sprintf(EndDestino, "%s:%d", IP_Destino, TCP_DstPort(pTCP));
          sprintf(PortaDestino, "%d", TCP_DstPort(pTCP));
          frm_Monitor->Append(ipPtr->protocol, EndOrigem, EndDestino,
                              PortaOrigem, PortaDestino,
                              &ptr[etherHdrLen + ipHdrLen + tcpHdrLen]); }
       else if (ipPtr->protocol == 0x11) // UDP
       {
          UDP_HEADER *pUDP = (UDP_HEADER *) &ptr[etherHdrLen + ipHdrLen];
          udpHdrLen = 8;
          sprintf(EndOrigem, "%s:%d", IP_Origem, XCHG(pUDP->src_port));
          sprintf(PortaOrigem, "%d", XCHG(pUDP->src_port));
          sprintf(EndDestino, "%s:%d", IP_Destino, XCHG(pUDP->dest_port));
          sprintf(PortaDestino, "%d", XCHG(pUDP->dest_port));
          frm_Monitor->Append(ipPtr->protocol, EndOrigem, EndDestino,
                              PortaOrigem, PortaDestino,
                              &ptr[etherHdrLen + ipHdrLen + udpHdrLen]);
       } } } } }
```

Implementação

Visualização do conteúdo capturado

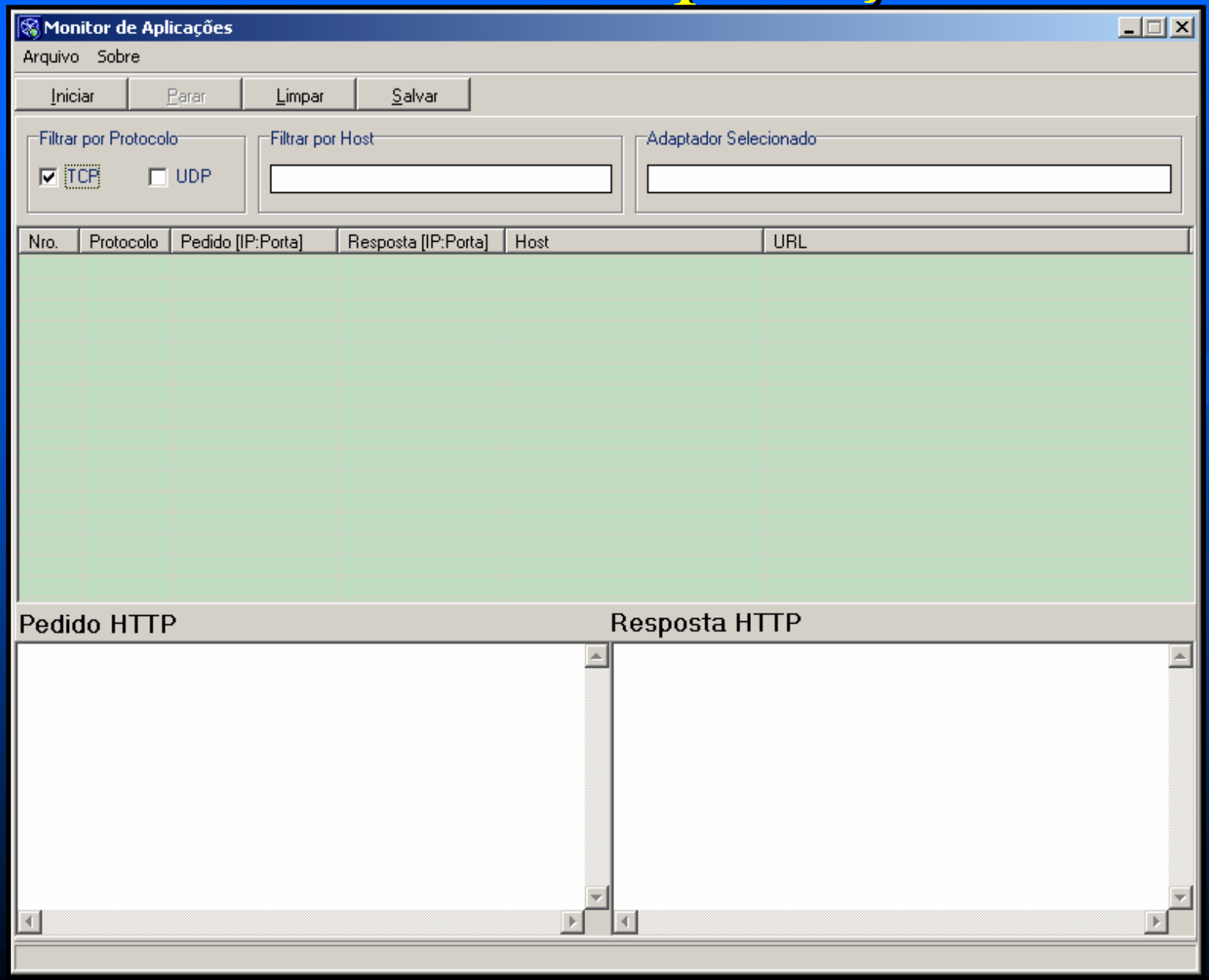
```
switch (prot) {
case 0x06:
  if (!frm_Monitor->cb_TCP->Checked)
     return;
  if ((PortaDestino == 80) && (HOST > "")) {
     if ((Filtro) || (edt_FiltrarHost->Text == "")) {
       frm_Monitor->lv_Protocolos->Items->Add();
       aux = frm_Monitor->lv_Protocolos->Items->Count - 1;
       frm_Monitor->lv_Protocolos->Items->Item[aux]->Caption = aux;
       frm_Monitor->lv_Protocolos->Items->Item[aux]->SubItems->Add("TCP");
       frm_Monitor->lv_Protocolos->Items->Item[aux]->SubItems->Add(EndOrigem);
       frm_Monitor->lv_Protocolos->Items->Item[aux]->SubItems->Add(EndDestino);
       frm_Monitor->lv_Protocolos->Items->Item[aux]->SubItems->Add(HOST);
       frm_Monitor->lv_Protocolos->Items->Item[aux]->SubItems->Add(GET);
       frm_Monitor->lv_Protocolos->Items->Item[aux]->Update(); } } break;
```

Implementação

Visualização do conteúdo capturado

```
if (len) {
   if ((Filtro) || (edt_FiltrarHost->Text == "") || ((auxPR == "HTTP") &&
       (StrPos(NomFiltro.c_str(), edt_FiltrarHost->Text.c_str())))) {
      if (auxPR == "GET ") {
        frm_Monitor->memo_Origem->Lines->Append("------> " + AnsiString(aux));
        frm_Monitor->memo_Origem->Lines->Append(asAux);
      else if (auxPR == "HTTP")
        frm_Monitor->memo_Destino->Lines->Append("-------------------------");
        frm_Monitor->memo_Destino->Lines->Append(asAux); } } }
Filtro = false; }
```

Monitor de Aplicações



Conclusões

- O protótipo desenvolvido neste trabalho demonstrou ser uma ferramenta de monitoração eficaz para redes *Ethernet*, utilizando a estrutura da arquitetura TCP/IP
- Foi possível aplicar na prática, conceitos que até então eram vistos apenas em livros
- A utilização da biblioteca *WinPcap*, que tem como principal objetivo, acessar diretamente o adaptador rede, sem a intervenção do sistema operacional, tornando abstrato o acesso ao meio
- A utilização do componente *TJesNetMonitor*, que tem algumas funcionalidades da *WinPcap*, encapsuladas, tornando mais fácil o desenvolvimento do protótipo

Extensões

- Implementar os filtros para os outros protocolos da camada de aplicação, como: SMTP de correio eletrônico, FTP de transferência de arquivos
- Um sistema que consiga filtrar protocolos seguros, como SSL
- Um sistema de bloqueios a páginas de internet ou endereços de e-mail, com base nas informações monitoras
- A continuidade deste trabalho, visando a criação de listas de filtros, controles mais amplos sobre as camadas referentes à arquitetura TCP/IP