INE5414 - Redes de Computadores I
Implementação de um Protocolo da Camada de Transporte

Daniel Ricardo dos Santos

João Paulo Pizani Flor

16 de novembro de 2008

Resumo

O trabalho apresenta os conceitos principais das camadas de rede, transporte e aplicação do modelo OSI, necessários ao desenvolvimento de um protocolo da camada de transporte. É apresentado, também, o detalhamento da implementação deste protocolo.

Capítulo 1

Introdução

A motivação para a realização deste trabalho é ajudar a fixar o entendimento dos conceitos da disciplina de Redes de Computadores I, em especial o funcionamento de cada uma das camadas da pilha de protocolos OSI, assim como a relação entre as camadas.

O objetivo do trabalho é implementar um protocolo da camada de transporte, baseado no exemplo de implementação que está disponível em [3].

A camada de transporte desenvolvida nesse trabalho tem características *sui generis*, e não implementa o protocolo TCP nem UDP. Dessa forma, não é possível usar aplicativos normais da Internet (browser, email) para testar o funcionamento da camada de transporte desenvolvida.

Desenvolvemos então um pequeno protocolo de aplicação, orientado a conexão, que utiliza os serviços da camada de transporte desenvolvida para comunicar-se com sua aplicação parceira.

O trabalho apresenta, primeiramente, os principais conceitos das camadas de rede, transporte e aplicação do modelo OSI e em seguida é detalhado o desenvolvimento do protocolo. Ao final, está anexado o código-fonte completo do trabalho, com cada seção (rede,transporte e aplicação).

Capítulo 2

Camada de Rede

A camada de rede é a terceira do modelo OSI. Ela utiliza os serviços da camada de enlace de dados para responder às requisições da camada de transporte.

A camada de rede é a primeira na pilha de protocolos a prover uma concexão ponto-a-ponto de forma transparente para quem a usa (as camadas superiores). A camada de enlace tem a responsabilidade única de transmitir dados de uma ponta a outra de um canal físico, se preocupando com questões como correção de erros e detecção de colisões. Já a camada de rede é a camada responsável pela entrega dos pacotes da origem ao destino, incluindo o roteamento entre hosts intermediários. Responsável, também, por propiciar os meios necessários para transmissão de dados de tamanho variado via uma ou mais redes, mantendo a Qualidade de Serviço e executando as funções de controle de erros.

Existem algumas informações que devem ser conhecidas pela camada de rede, para que a mesma consiga levar os pacotes da origem ao destino, entre elas:

- O protocolo de rede é orientado a conexão ou não?

- Quais são os endereços globais?
- Como se encaminha uma mensagem? (Dado um endereço global de destino, para qual de meus vizinhos emcaminho a mensagem que recebi?)

As redes podem ser orientadas a conexão e funcionarem por circuito virtual. Assim não é necessário que os endereços de origem e destino sejam transmitidos em cada pacote. Outra vantagem de se usar protocolos de rede orientados à conexão é que se pode garantir parâmetros de atraso máximo, garantia essa muito importante para a transmissão de dados multimídia, por exemplo. Um exemplo de protocolo de rede orientado a conexão é o IPX [2], usados nas redes Novell Netware.

Uma camada de rede não orientada a conexão funciona por comutação de pacotes, isto é, os pacotes não seguem uma rota fixa da origem ao destino. Assim cada pacote deve conter o endereço completo de origem e destino para poder ser roteado. Numa transmissão os pacotes de rede que correspondem a uma mesma mensagem de transporte, por exemplo, podem ser roteados por países e continentes diferentes. Uma grande vantagem de se usar uma camada de rede não orientada a conexão é que ela distribuir melhor a carga no canal físico, colaborando com menor congestionamento em momentos de grande utilização. Um exemplo de protocolo de redes não orientado a conexão é o IP [1], utilizado na Internet.

Um endereço global é um endereço único que identifica cada membro da rede. Na rede Internet esse endereço é chamado *endereço IP*, em analogia ao nome do protocolo em si. O endereço IP tem 32 bits, e é composto de duas partes: O endereço da sub-rede e o endereço do host.

Geralmente se relaciona a terceira camada do modelo OSI (camada de rede) à camada de Internet da pilha de protocolos TCP/IP. Porém, apesar

de a Internet ser a rede de computadores mais popular, existem muitos outros modelos tão interessantes quanto, como por exemplo as redes ATM.

Capítulo 3

Camada de Transporte

A camada de transporte é a quarta camada segundo o modelo OSI. Ela utiliza os serviços da camada de rede (conexão ponto-a-ponto entre hosts) para atender as requisições dos protocolos de aplicação.

A camada de transporte tem um papel importantíssimo dentre todas as camadas da pilha OSI. Ela é a primeira camada que estabelece uma conexão confiável entre dois processos de aplicação. Para um programa aplicativo (processo), que usa a interface da camada de transporte, a conexão de rede pode ser vista como um *tubo* direto, em que bytes são inseridos e após algum tempo magicamente aparecem no outro lado. O protocolo de transporte é responsável por estabelecer e encerrar as conexões entre processos que desejam comunicar-se, e faz a multiplexação dessas conexões que utilizam o mesmo NSAP (*ponto de acesso ao serviço de rede*).

Nessa camada os dados que chegam da camada de sessão são divididos em unidades menores, para que possam ser mais facilmente enviados pela camada de rede. É sua responsabilidade fazer coom que as unidades cheguem ao destino corretamente e em ordem.

O software do protocolo de transporte geralmente roda como parte do

núcleo do sistema operacional, porém é a última camada (em geral) que roda no S.O. As camadas de apresentação e aplicação geralmente são implementadas como bibliotecas a nível de usuário. Os processos que desejam comunicar-se utilizando HTTP, por exemplo, utilizam as bibliotecas de HTTP para *interpretar* os dados que obtiveram através da conexão de transporte.

Para suportar as diversas conexões de transporte simultâneamente, o software que implementa a camada de transporte tem várias alternativas:

- Caso haja apenas uma conexão de rede disponível, a camada de transporte realiza multiplexação no tempo dos pacotes de transporte sobre a conexão de rede.

- Se houver várias conexões de rede disponíveis, a camada de transporte pode fornecer um ponto de acesso à rede exclusivo a uma conexão de transporte com maiores requerimentos de banda.

- No caso em que houver mais de uma conexão de rede para cada conexão de transporte, pode ser feito um escalonamento round-robin entre as conexões de rede disponíveis, para diminuir o congestionamento nas sub-redes utilizadas.

A camada de transporte tem uma especial importância por separar as camadas mais baixas das mais altas, permitindo que se aprimorem tecnologias tanto nas camadas superiores quanto nas inferiores de forma independente.

Capítulo 4

Camada de Aplicação

É a camada de mais alto nível tanto no modelo OSI quanto no modelo TCP/IP. Apresenta uma interface direta com os aplicativos, provendo a eles seus serviços e enviando requisições a camada de apresentação no modelo OSI. No modelo TCP/IP a camada de aplicação se comunica diretamente com a camada de transporte e corresponde as camadas de aplicação, apresentação e sessão do modelo OSI.

Exemplos de protocolos da camada de aplicação:

- BitTorrent
- DHCP - Dynamic Host Configuration Protocol
- DNS - Domain Name System
- FTP - File Transfer Protocol
- HTTP - HyperText Transfer Protocol
- IMAP - Internet Message Acces Protocol

- NFS - Network File System

A camada de aplicação é responsável por dar *significado* aos bytes recebidos e enviados pela conexão de transporte. Os programas aplicativos (navegadores, players multimídia, clientes de IM, etc.) geralmente usam de bibliotecas de programação, desenvolvidas em várias linguagens, para fornecer ao usuário sua funcionalidade.

A interface entre os programas de aplicação e os protocolos da camada de transporte é feita usando-se *sockets*. Um socket pode ser visto como um fluxo (ou *stream*) bidirecional de dados, que é seguro e confiável. As operações que se pode realizar sobre um socket são muito similares às operações que se realizam sobre arquivos. As operações BIND, LISTEN, CONNECT, READ e WRITE são a base da interface de um socket.

É através dessa interface que a camada de aplicação envia e recebe suas mensagens de forma transparente, ou seja, sem tem nenhum conhecimento das tecnologias empregadas nas camadas inferiores.

Capítulo 5

Desenvolvimento do protocolo

5.1 Camada de Rede

É a camada com a implementação mais simples neste trabalho, pois serve apenas de ponte duas camadas de transporte. A camada de rede recebe os Packets de uma camada de transporte e os envia ao destino utoilizando o método sendPacket, que chama o método fromNet da camada de transporte destino.

5.2 Camada de Transporte

Para o desenvolvimento do protocolo foi escolhida a linguagem Java. O modelo de protocolo de camada de transporte apresentado em [1] foi então "traduzido"para a linguagem escolhida e foram completadas as funções que faltavam: sleep, que deixa a camada inativa esperando uma conexão, wakeUp, que torna a camada ativa novamente, toNet, que envia os dados à camada de rede e fromNet, que recebe os dados da camada de rede. A camada de transporte recebe da camada de aplicação os dados que serão

enviados e os envia à camada de rede, por isso são necessárias as referências entre as camadas. Na camada de transporte há um array que armazena e organiza as conexões da camada, representadas pela classe Conn. Os pacotes, representados pela classe Packet são enviados pelo método toNet à camada de rede, que os repassará à camada de transporte do destino, utilizando o método fromNet.

5.3 Camada de Aplicação

A camada de aplicação é repsentada por dois aplicativos trocando mensagens. Ambos devem se conectar e depois de estabelecida a conexão uma mensagem digitada em um aplicativo é enviada ao outro. O envio é feito através da camada de transporte, já que as aplicações possuem referêncais a essa camada, utilizando o método send.

5.4 Validação, Verificação e Testes

A validação e a verificação foram feitas utilizando-se uma máquinas de estados que representava o protocolo a ser implementado e, após a implementação, foram realizados testes nas camadas, verificando-se o funcionamento dos aplicativos.

Capítulo 6

Conclusões e comentários

O trabalho foi de extrema importância para a disciplina e atingiu seu principal objetivo, o de clarear os conceitos de camadas do modelo OSI e desenvolvimento de protocolos. Mesmo sendo apenas uma simulação e, por isso, trazendo consigo limitações, foi possível entender como funciona, na prática, o que foi visto durante o semestre.

Capítulo 7

Anexo: Código fonte comentado

7.1 Camada de rede

```
package layers;

public class NetworkLayer {

  // Conhece as camadas de rede da origem e do destino
  public TransportLayer tOrigin;
  public TransportLayer tDestination;

  public void sendPacket(TransportLayer sender, Packet packet) {

    TransportLayer destination = null;

    // Decide para quem enviar o pacote
    if (sender == tOrigin)
      destination = tDestination;

    else
      destination = tOrigin;

    //Envia o pacote a camada de transporte escolhida
```

```
    destination.fromNet(packet);

  }

}
```

7.2 Camada de transporte

```
package layers;

public class TransportLayer {

  final int MaxConn = 32;
  final int MaxMsgSize = 8192;
  final int MaxPktSize = 512;
  final int TimeOut = 20;
  final int Cred = 1;
  final int OK = 0;
  final int ErrFull = -1;
  final int ErrReject = -2;
  final int ErrClosed = -3;
  final int LowErr = -3;
  int transportAddress;
  int listenAddress;
  int listenConn;
  private boolean wakeUp;
  String data[];
  Conn conn[];

  // Pacote que serah enviado
  Packet packet;

  // Camada de rede a ser utilizada
  NetworkLayer netLayer;

  // Camada de aplicacao a ser utilizada
  public ApplicationLayer app;

  // struct Conn
  class Conn {
```

```
  int localAddress;
  int remoteAddress;
  String state;
  String userBufAddr[];
  int byteCount;
  int clrReqReceived;
  int timer;
  int credits;
  final int buffSize = 100;

  public Conn() {
    state = "IDLE";
    userBufAddr = new String[100];
    for (int i = 0; i < 100; i++)
      userBufAddr[i] = "";

  }
}

public TransportLayer(NetworkLayer netLayer) {
  data = new String[32];
  wakeUp = true;
  conn = new Conn[32];
  for (int i = 0; i < 32; i++)
    conn[i] = new Conn();

  this.netLayer = netLayer;
}

//Fica inativa esperando wakeUp
void sleep() {
  for (wakeUp = false; !wakeUp;)
    ;
}

//Torna ativa novamente
void wakeUp() {
  wakeUp = true;
}
```

```
// Envia o pacote a camada de rede
void toNet(int cid, int q, int m, String pt, String data[], int bytes) {
  Packet packet = new Packet(cid, q, m, pt, data, bytes);
  netLayer.sendPacket(this, packet);
}

// Recebe um pacote da camada de rede
void fromNet(Packet packet) {
  packetArrival(packet, 1);

}

public int listen(int t) {
  int i = 1;
  int found = 0;
  for (i = 0; i < 32; i++) {
    if (!conn[i].state.equals("QUEUED") || conn[i].localAddress != t)
      continue;
    found = i;
    break;
  }

  if (found == 0) {
    listenAddress = t;
    sleep();
    i = listenConn;
  }
  conn[i].state = "ESTABLISHED";
  conn[i].timer = 0;
  listenConn = 0;
  toNet(i, 0, 0, "CALL_ACC", data, 0);
  return i;
}

public int connect(int l, int r) {
  data[0] = (new StringBuilder(String.valueOf(r))).toString();
  data[1] = (new StringBuilder(String.valueOf(l))).toString();
  int i;
  for (i = 31; !conn[i].state.equals("IDLE") && i > 0; i--)
    ;
  if (conn[i].state.equals("IDLE")) {
```

```
      Conn cptr = conn[i];
      cptr.localAddress = l;
      cptr.remoteAddress = r;
      cptr.state = "WAITING";
      cptr.clrReqReceived = 0;
      cptr.credits = 0;
      cptr.timer = 0;
      toNet(i, 0, 0, "CALL_REQ", data, 2);
      sleep();
      if (cptr.state.equals("ESTABLISHED"))
        return i;
      if (cptr.clrReqReceived != 0) {
        cptr.state = "IDLE";
        toNet(i, 0, 0, "CLEAR_CONF", data, 0);
        return -2;
      } else {
        return 0;
      }
    } else {
      return -1;
    }
  }

  public int send(int cid, String bufptr[], int bytes) {
    Conn cptr = conn[cid];
    if (cptr.state.equals("IDLE"))
      return -3;
    cptr.state = "SENDING";
    cptr.byteCount = 0;
    if (cptr.clrReqReceived == 0) {
      do {
        int count;
        int m;
        if (bytes - cptr.byteCount > 512) {
          count = 512;
          m = 1;
        } else {
          count = bytes - cptr.byteCount;
          m = 0;
        }
        for (int i = 0; i < count; i++) {
```

```
        data[i] = bufptr[cptr.byteCount + i];
        toNet(cid, 0, m, "DATA_PKT", data, count);
        cptr.byteCount = cptr.byteCount + count;
      }

    } while (cptr.byteCount < bytes);
    cptr.state = "ESTABLISHED";
    return 0;
  } else {
    cptr.state = "ESTABLISHED";
    return -3;
  }
}

int receive(int cid, String bufptr[], Integer bytes) {
  Conn cptr = conn[cid];
  if (cptr.clrReqReceived == 0) {
    cptr.state = "RECEIVING";
    cptr.userBufAddr = bufptr;
    cptr.byteCount = 0;
    data[0] = "CRED";
    data[1] = "1";
    toNet(cid, 1, 0, "CREDIT", data, 2);
    sleep();
    bytes = Integer.valueOf(cptr.byteCount);
  }
  cptr.state = "ESTABLISHED";
  return cptr.clrReqReceived != 1 ? 0 : -3;
}

public int disconnect(int cid) {
  Conn cptr = conn[cid];
  if (cptr.clrReqReceived == 1) {
    cptr.state = "IDLE";
    toNet(cid, 0, 0, "CLEAR_CONF", data, 0);
  } else {
    cptr.state = "DISCONNECT";
    toNet(cid, 0, 0, "CLEAR_REQ", data, 0);
  }
  return 0;
}
```

```
void packetArrival(Packet packet, Integer count) {
  Integer cid = Integer.valueOf(packet.getCid());
  String ptype = packet.getPt();
  String data[] = packet.getP();
  Conn cptr = conn[cid.intValue()];
  if (ptype.equals("CALL_REQ")) {
    cptr.localAddress = Integer.parseInt(data[0]);
    cptr.remoteAddress = Integer.parseInt(data[1]);
    if (cptr.localAddress == listenAddress) {
      listenConn = cid.intValue();
      cptr.state = "ESTABLISHED";
      wakeUp();
    } else {
      cptr.state = "QUEUED";
      cptr.timer = 20;
    }
    cptr.clrReqReceived = 0;
    cptr.credits = 0;
  } else if (ptype.equals("CALL_ACC")) {
    cptr.state = "ESTABLISHED";
    wakeUp();
  } else if (ptype.equals("CLEAR_REQ")) {
    cptr.clrReqReceived = 1;
    if (cptr.state.equals("ESTABLISHED")) {
      cptr.state = "IDLE";
      disconnect(cid.intValue());
    }
    if (cptr.state.equals("WAITING") || cptr.state.equals("RECEIVING")
        || cptr.state.equals("SENDING"))
      wakeUp();
  } else if (ptype.equals("CLEAR_CONF")) {
    cptr.state = "IDLE";
    System.out.println("Desconectei");
  } else if (ptype.equals("CREDIT")) {
    cptr.credits += Integer.parseInt(data[1]);
    if (cptr.state.equals("SENDING"))
      wakeUp();
  } else if (ptype.equals("DATA_PKT")) {
    for (int i = 0; i < count.intValue(); i++)
      cptr.userBufAddr[0] = data[i];
```

```
      cptr.byteCount += count.intValue();
      app.receiveMessage(cptr.userBufAddr[0]);
    }
  }

  void clock() {
    for (int i = 1; i < 32; i++) {
      Conn cptr = conn[i];
      if (cptr.timer > 0) {
        cptr.timer--;
        if (cptr.timer == 0) {
          cptr.state = "IDLE";
          toNet(i, 0, 0, "CLEAR_REQ", data, 0);
        }
      }
    }

  }

}
```

7.3 Pacote

```
package layers;

public class Packet {

  private int cid;
  private int q;
  private int m;
  private int bytes;
  private String pt;
  private String p[];

  public Packet(int cid, int q, int m, String pt, String p[], int bytes) {
    this.cid = cid;
    this.q = q;
    this.m = m;
    this.pt = pt;
```

```
    this.p = p;
    this.bytes = bytes;
  }

  public int getBytes() {
    return bytes;
  }

  public int getCid() {
    return cid;
  }

  public int getM() {
    return m;
  }

  public String[] getP() {
    return p;
  }

  public String getPt() {
    return pt;
  }

  public int getQ() {
    return q;
  }

}
```

7.4 Camada de aplicação

```
package layers;

import javax.swing.JFrame;

public abstract class ApplicationLayer extends JFrame {

  private static final long serialVersionUID = 1L;
```

```
  //id da Camada de Aplicacao
  private String id;

  //id da conexao utilizada
  private int cid;

  public ApplicationLayer(String id) {
    this.id = id;
  }

  public String getId() {
    return id;
  }

  public int getCid() {
    return cid;
  }

  public void setCid(int cid) {
    this.cid = cid;
  }

  //Fecha a conexao
  public abstract void closeConnection();

  //Recebe a mensagem que serah repassada
  public abstract void receiveMessage(String s);

}
```

7.5 Aplicativos

7.5.1 Aplicativo A

```
package application;

import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
```

```
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextArea;
import javax.swing.border.EtchedBorder;

import layers.ApplicationLayer;
import layers.TransportLayer;

public class ApplicationA extends ApplicationLayer {

  private JLabel labelMensagemAEnviar;
  private JLabel labelMensagemRecebida;
  private JPanel painel1;
  private JPanel painel2;
  private JPanel painel3;
  private JPanel painel4;
  private JPanel painelDosBotoes;
  private JTextArea campoMensagemAReceber;
  private JTextArea campoMensagemAEnviar;
  private JButton botaoEnviarDados;
  private JButton botaoConectar;

  private TransportLayer tLayer;

  private static final long serialVersionUID = 1L;

  class ButtonsActionListener implements ActionListener {
    class ThreadListener extends Thread {

      public void run() {
        if ("Conectar".equals(evt.getActionCommand())) {
          if (!connected) {
            setCid(tLayer.connect(1, 2));
            if (getCid() > 0) {
              System.out.println("Conexao estabelecida");
```

```
            connected = true;
          } else {
            System.out
                .println("Conexao nao estabelecida");
          }
        } else {
          int desconectou = tLayer.disconnect(getCid());
          if (desconectou == 0) {
            System.out.println("Conexao terminada");
            connected = false;
          } else {
            System.out.println("Conexao nao terminada");
          }
        }
      } else if ("EnviarDados".equals(evt.getActionCommand())) {
        String bytes[] = new String[1];
        bytes[0] = campoMensagemAEnviar.getText();
        int result = tLayer.send(getCid(), bytes, 1);
        if (result == 0) {
          campoMensagemAReceber.setText("");
          System.out.println("Enviado");
        } else {
          System.out.println("Erro");
        }
      }
    }

    private ActionEvent evt;
    final ButtonsActionListener botao;

    public ThreadListener(ActionEvent evt) {
      super();
      botao = ButtonsActionListener.this;
      this.evt = evt;
    }
  }

  public void actionPerformed(ActionEvent e) {
    ThreadListener thrListener = new ThreadListener(e);
    thrListener.start();
  }
```

```
  boolean connected;
  final ApplicationA cliente;

  ButtonsActionListener() {
    super();
    cliente = ApplicationA.this;
    connected = false;
  }
}

public ApplicationA(String id, TransportLayer camadaDeTransporte) {
  super(id);
  this.tLayer = camadaDeTransporte;
  initComponents();
  setVisible(true);
}

private void initComponents() {
  this.setTitle("Aplicativo A");
  painel1 = new JPanel();
  painel2 = new JPanel();
  labelMensagemAEnviar = new JLabel();
  campoMensagemAEnviar = new JTextArea();
  painel3 = new JPanel();
  labelMensagemRecebida = new JLabel();
  campoMensagemAReceber = new JTextArea();
  painel4 = new JPanel();
  painelDosBotoes = new JPanel();
  botaoConectar = new JButton("Conectar");
  botaoEnviarDados = new JButton("Enviar");

  botaoConectar.setActionCommand("Conectar");
  ButtonsActionListener buttonsActionListener = new ButtonsActionListener();
  botaoConectar.addActionListener(buttonsActionListener);
  botaoEnviarDados.setActionCommand("EnviarDados");
  botaoEnviarDados.addActionListener(buttonsActionListener);

  addWindowListener(new WindowAdapter() {

    public void windowClosing(WindowEvent evt) {
```

```
        System.exit(0);
      }

    });
    painel1.setLayout(new GridLayout(2, 2));
    painel2.setLayout(new BorderLayout());
    painel2.setBorder(new EtchedBorder());

    labelMensagemAEnviar.setText("Mensagem a enviar");
    painel2.add(labelMensagemAEnviar, "North");
    campoMensagemAEnviar.addKeyListener(new KeyAdapter() {

    });
    painel2.add(campoMensagemAEnviar, "South");
    painel1.add(painel2);
    painel3.setLayout(new BorderLayout());
    painel3.setBorder(new EtchedBorder());
    labelMensagemRecebida.setText("Mensagem recebida");
    painel3.add(labelMensagemRecebida, "Center");
    painel3.add(campoMensagemAReceber, "North");
    painel1.add(painel3);
    getContentPane().add(painel1, "Center");
    getContentPane().add(painel4, "North");
    painelDosBotoes.add(botaoConectar);
    painelDosBotoes.add(botaoEnviarDados);
    getContentPane().add(painelDosBotoes, "South");

    pack();
}

public void showMessage(String dado) {
    campoMensagemAReceber.append(dado);
}

public void receiveMessage(String texto) {
    campoMensagemAReceber.append(texto);
}

public void closeConnection() {
    System.out.println("Conexao terminada");
}
```

```
}
```

7.5.2 Aplicativo B

```
package application;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextArea;

import layers.ApplicationLayer;
import layers.TransportLayer;

public class ApplicationB extends ApplicationLayer {

  private static final long serialVersionUID = 1L;
  private JLabel labelMensagemAEnviar;
  private JLabel mensagemRecebida;
  private JPanel painel1;
  private JPanel painel2;
  private JPanel painel3;
  private JPanel painel4;
  private JPanel painelDosBotoes;
  private JTextArea campoMensagemAReceber;
  private JTextArea campoMensagemAEnviar;
  private JButton botaoEnviarDados, botaoConectar;

  private TransportLayer tLayer;

  public ApplicationB(String id, TransportLayer camadaDeTransporte) {
    super(id);
    this.tLayer = camadaDeTransporte;
    initComponents();
    setVisible(true);
```

```
}

class ButtonsActionListener implements ActionListener {
  boolean connected = false;

  class ThreadListener extends Thread {

    private ActionEvent evt;

    public ThreadListener(ActionEvent evt) {
      this.evt = evt;
    }

    public void run() {

      if ("EnviarDados".equals(evt.getActionCommand())) {

        String[] bytes = new String[1];
        bytes[0] = campoMensagemAEnviar.getText();
        int result = tLayer.send(getCid(), bytes, 1);

        if (result == 0) {
          campoMensagemAReceber.setText("");
          System.out.println("Enviado");
        } else {
          System.out.println("Erro");
        }

      } else if ("escutar".equals(evt.getActionCommand())) {

        if (!connected) {
          setCid(tLayer.listen(2));
          if (getCid() > 0) {

            System.out.println("Escutando");
            connected = true;
          } else {
            System.out.println("Erro");

          }
        } else {
```

```
            int desconectou = tLayer.disconnect(getCid());

            if (desconectou == 0) {
              System.out.println("Escuta terminada");
              connected = false;
            } else {
              System.out.println("Escuta nao terminada");
            }
          }

        }
      }

    };

    public void actionPerformed(ActionEvent e) {

      ThreadListener thrListener = new ThreadListener(e);
      thrListener.start();

    }
  }

  private void initComponents() {
    this.setLocation(100, 100);
    this.setTitle("Aplicativo B");
    painel1 = new JPanel();
    painel2 = new JPanel();
    labelMensagemAEnviar = new JLabel();
    campoMensagemAEnviar = new JTextArea();
    painel3 = new JPanel();
    mensagemRecebida = new JLabel();
    campoMensagemAReceber = new JTextArea();
    painel4 = new JPanel();
    painelDosBotoes = new JPanel();

    botaoConectar = new JButton("Conectar");
    botaoEnviarDados = new JButton("Enviar");

    ButtonsActionListener buttonsActionListener = new ButtonsActionListener();
```

```
botaoConectar.setActionCommand("escutar");
botaoConectar.addActionListener(buttonsActionListener);

botaoEnviarDados.setActionCommand("EnviarDados");
botaoEnviarDados.addActionListener(buttonsActionListener);

addWindowListener(new java.awt.event.WindowAdapter() {

});

painel1.setLayout(new java.awt.GridLayout(2, 2));

painel2.setLayout(new java.awt.BorderLayout());

painel2.setBorder(new javax.swing.border.EtchedBorder());
labelMensagemAEnviar.setText("Mensagem a enviar");
painel2.add(labelMensagemAEnviar, java.awt.BorderLayout.NORTH);

campoMensagemAEnviar.addKeyListener(new java.awt.event.KeyAdapter() {
});

painel2.add(campoMensagemAEnviar, java.awt.BorderLayout.SOUTH);

painel1.add(painel2);

painel3.setLayout(new java.awt.BorderLayout());

painel3.setBorder(new javax.swing.border.EtchedBorder());
mensagemRecebida.setText("Mensagem recebida");
painel3.add(mensagemRecebida, java.awt.BorderLayout.CENTER);

painel3.add(campoMensagemAReceber, java.awt.BorderLayout.NORTH);

painel1.add(painel3);

getContentPane().add(painel1, java.awt.BorderLayout.CENTER);

getContentPane().add(painel4, java.awt.BorderLayout.NORTH);

painelDosBotoes.add(botaoConectar);
painelDosBotoes.add(botaoEnviarDados);
```

```
    getContentPane().add(painelDosBotoes, java.awt.BorderLayout.SOUTH);

    pack();
  }

  public void receiveMessage(String texto) {
    campoMensagemAReceber.append(texto);
  }

  public void closeConnection() {
    System.out.println("Conexao Encerrada");
  }

  public void showMessage(String dado) {
    campoMensagemAReceber.append(dado);
  }

}
```

7.6 Main

```
package application;

import javax.swing.JOptionPane;

import layers.NetworkLayer;
import layers.TransportLayer;

public class Main {

  public static void main(String[] args) {

    //Criacao da camada de rede
    NetworkLayer netLayer = new NetworkLayer();

    //Criacao das camadas de transporte
    TransportLayer tOrigin = new TransportLayer(netLayer);
    TransportLayer tDestination = new TransportLayer(netLayer);
```

```
        //Passagem a camada de rede das referencias as camadas de transporte
        netLayer.tOrigin = tOrigin;
        netLayer.tDestination = tDestination;

        //Instrucoes para o usuario
        JOptionPane.showMessageDialog(null, "Conecte cada um dos aplicativos. \n Digite a
    mensagem a ser enviada no campo superior de um aplicativo. \n Envie. \n A mensagem
    chegarah no campo inferior do outro aplicativo", "Instrucoes", 1);

        //Criacao dos aplicativos
        ApplicationA appA = new ApplicationA("A", tOrigin);
        ApplicationB appB = new ApplicationB("B", tDestination);

        //Passagem as camadas de transporte das referencias as camadas de aplicacao
        tOrigin.app = appA;
        tDestination.app = appB;

    }

}
```

Referências Bibliográficas

[1] Internet protocol. http://en.wikipedia.org/w/index.php?title=Internet_Protocol&oldid=251988439. Link permanente p/ a revisão referenciada.

[2] Ipx/spx. http://en.wikipedia.org/w/index.php?title=IPX/SPX&oldid=238242922. Link permanente p/ a revisão referenciada.

[3] Andrew S. Tanenbaum. *Computer networks*. Prentice-Hall, Englewood Cliffs, N.J. :, 2nd ed. edition, 1989.