UNIVERSIDADE DECAXIAS DO SUL
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

Relatório sobre o NoseV6

Sniffer para redes IPv6

Álisson R. Teles

Leonardo Gobi

Caxias do Sul, 07 de Julho de 2015

Introdução

O relatório que segue é o resultado de proposta estabelecida pela professora Maria de Fatima Webber do Prado Lima para projeto final da disciplina de Redes de Computadores II - INF0220A. Nele são apresentados os principais algoritmos desenvolvidos pelo grupo para resolver as problemáticas estabelecidas:

Funcionalidade 1: Interface da aplicação;

Funcionalidade 2: Estatística sobre os tipos de endereços

Funcionalidade 3: Estatística sobre a classe de tráfego

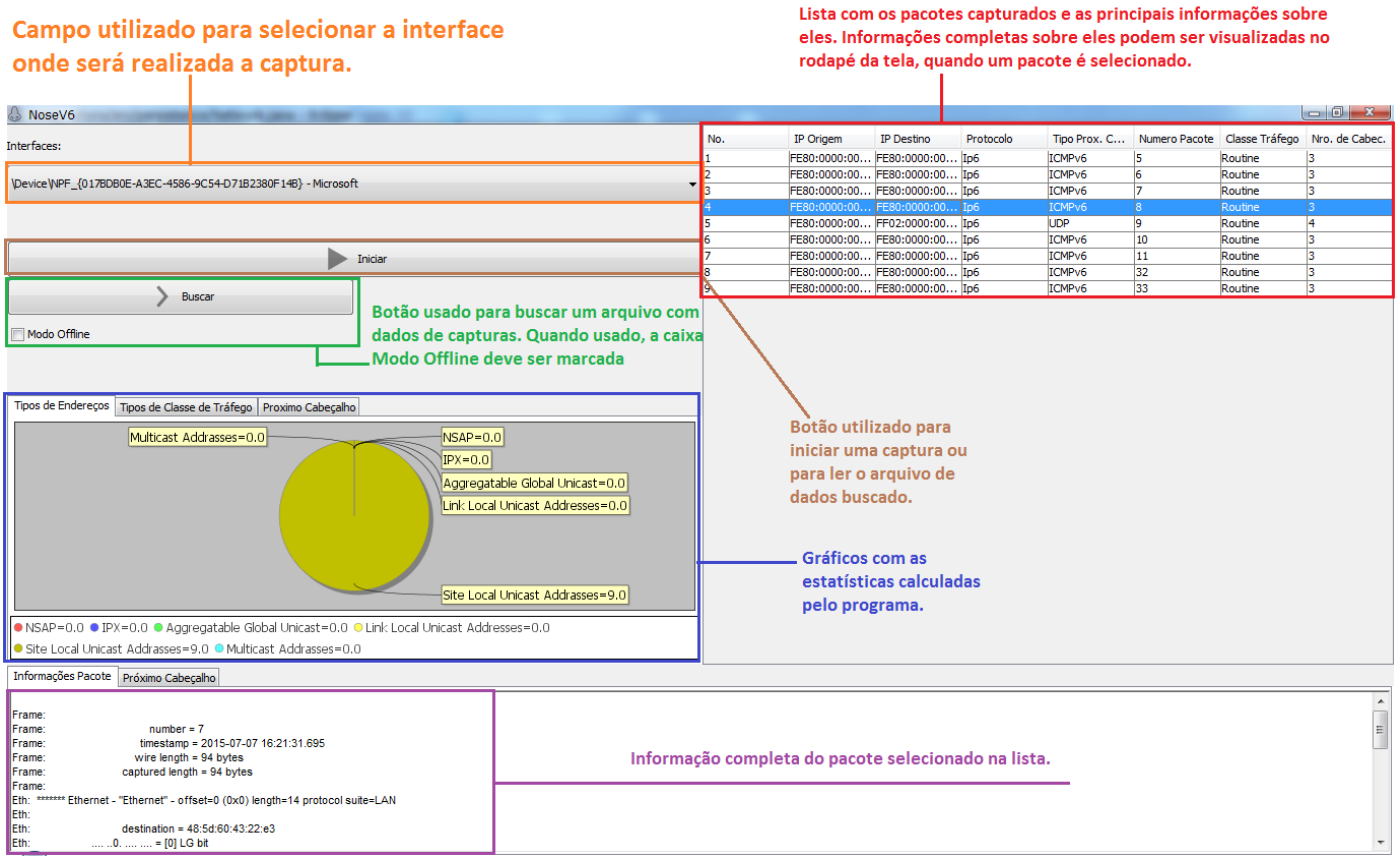
Funcionalidade 4: Tabela dos Fluxos

Funcionalidade 5: Estatística sobre o Próximo Cabeçalho

Funcionalidade 6: Duas estatísticas sobre o Próximo Cabeçalho

Funcionalidade 1: Interface da aplicação

Abaixo um resumo sobre a interface gráfica do software:



Funcionalidade 2: Estatística sobre os tipos de endereços

Fontes de pesquisa:

https://en.wikipedia.org/wiki/Reserved_IP_addresses
https://www.ultratools.com/tools/ipv6CIDRToRange

Para esta funcionalidade, analisamos os endereços IP e conseguimos diferenciá-los pela range de IPs que cada um representa.

Convertendo os valores hexadecimais para decimal, do menor valor até o maior valor da respectiva range, conseguimos realizar comparações com o endereço a ser analisado e identificar a qual tipo de endereço ele pertence.

Para um endereço ser NSAP, (200:: - 21ff:ffff:ffff:ffff:ffff:ffff:ffff:ffff), em valor decimal, deve estar entre 512 e 8703.

Para ser IPX, deve estar entre 1024 e 1535, Aggregatable Global Unicast entre 8192 e 16383, Link Local Unicast Adresses entre 65152 e 65215, Site Local Unicast Addresses entre 65216 e 65279 e para ser Multicast Adresses deve estar entre 65280 e 65535.

Algoritmo:

Endereço é convertido para uma string.
Variável decimal guarda número até o primeiro ":".

```
Se (decimal entre 512 e 8703){
  retorna tipo =  1
senao
  Se (decimal entre 1024 e 1535)
    retorna tipo = 2
  senao
    Se (decimal entre 8192 e 16383)
      retorna tipo = 3
    senao
      Se (decimal entre 65152 e 65215)
        retorna tipo = 4
      senao
        Se (decimal entre 65216 e 65279)
          retorna tipo = 5
        senao
          Se (decimal entre 65280 e 65535)
            retorna tipo = 6
```

Código desenvolvido:

```
public int estatisticaEndereco (PcapPacket packet){
        Ip6 ip6 = new Ip6();
        int tipo = 0;

        if  (packet.hasHeader(ip6)) {
            String ipSource = FormatUtils.ip(ip6.source());
            char[] endereco = ipSource.toCharArray();
            int index = 0;
            for (int i = 0; i< endereco.length; i++){

                if (endereco[i] == ':'){
                    index = i;
                    break;
                }
            }
            String comparacao = ipSource.substring(0, index);
            int decimal = Integer.parseInt(comparacao, 16);
            if (decimal >= 512 && decimal <= 8703){
                tipo =  1;
            }
            else{
                if (decimal >= 1024 && decimal <= 1535){
                    tipo =  2;
                }
                else{
                    if (decimal >= 8192 && decimal <= 16383){
                        tipo =  3;
                    }
                    else{
                        if (decimal >= 65152 && decimal <= 65215){
                            tipo =  4;
                        }
                        else{
                            if (decimal >= 65216 && decimal <= 65279){
                                tipo =  5;
                            }
                            else{
                                if (decimal >= 65280 && decimal <= 65535){
                                    tipo =  6;
                                }
                            }
                        }
                    }
                }
            }
        }
        return tipo;
    }
```

O resultado da estatística pode ser visualizado na interface gráfica do programa, sendo representada através de um gráfico com o nome de "Tipos de Endereços".

Funcionalidade 3: Estatística sobre a classe de tráfego

Para esta funcionalidade utilizamos o 8 bits referentes a Classe de Tráfego que estão presentes no cabeçalho IPv6. Para realizar a identificação da classe, utilizamos somente os 3 bits mais significativos, pois estão são os responsáveis por identificá-la. Os 2 bits menos significativos representam o controle de congestionamento, e em nossa análise ambos os tipos são considerados. O resultado da estatística pode ser visualizado na interface gráfica do programa, sendo representado através de um gráfico com o nome de "Tipos de Classe de Tráfego".

Como os 3 primeiros bits correspondem ao tipo de classe, analisamos da seguinte maneira:

CLASSE	BINÁRIO	DECIMAL	
0	.000.00000	0	Iníco classe 0
	.000.11111	31	Fim classe 0
1	.001.00000	32	Iníco classe 1
	.001.11111	63	Fim classe 1
2	.010.00000	64	Iníco classe 2
	.010.11111	95	Fim classe 2
3	.011.00000	96	Iníco classe 3
	.011.11111	127	Fim classe 3
4	.100.00000	128	Iníco classe 4
	.100.11111	159	Fim classe 4
5	.101.00000	160	Iníco classe 5
	.101.11111	191	Fim classe 5
6	.110.00000	192	Iníco classe 6
	.110.11111	223	Fim classe 6
7	.111.00000	224	Iníco classe 7
	.111.11111	255	Fim classe 7

Algoritmo:

classeTrafego = Pegamos do cabeçalho IPv6 o ultimo nibble do primeiro byte e primeiro nyble do segundo byte, formando o campo de Classe de Tráfego. Após, tratamos os nibbles separadamente, limpando os bits desnecessários, e posicionando os bits corretamente através de shifts. Realizamos um OR entre os 2 nibbles, guardando assim o valor da classe na variável classeTrafego para futura comparação.

Na sequencia, é realizado a comparação para ver em qual classe o pacote se encaixa:

```
Se (classeTrafego entre 0 e 31)
  tipo = 0 (Routine)
Senao
  Se (classeTrafego entre 32 e 63)
    tipo = 1 (Priority)
  Senao
    Se (classeTrafego entre 64 e 95)
      tipo = 2 (Immediate)
    Senao
      Se (classeTrafego entre 96 e 127)
        tipo = 3 (Flash)
      Senao
        Se (classeTrafego entre 128 e 159)
          tipo = 4 (Flash-override)
        Senao
          Se (classeTrafego entre 160 e 191)
            tipo = 5 (Critical)
          Senao
            Se (classeTrafego entre 192 e 223)
              tipo = 6 (Internet)
            Senao
              Se (classeTrafego entre 224 e 255)
                tipo = 7 (Network)
```

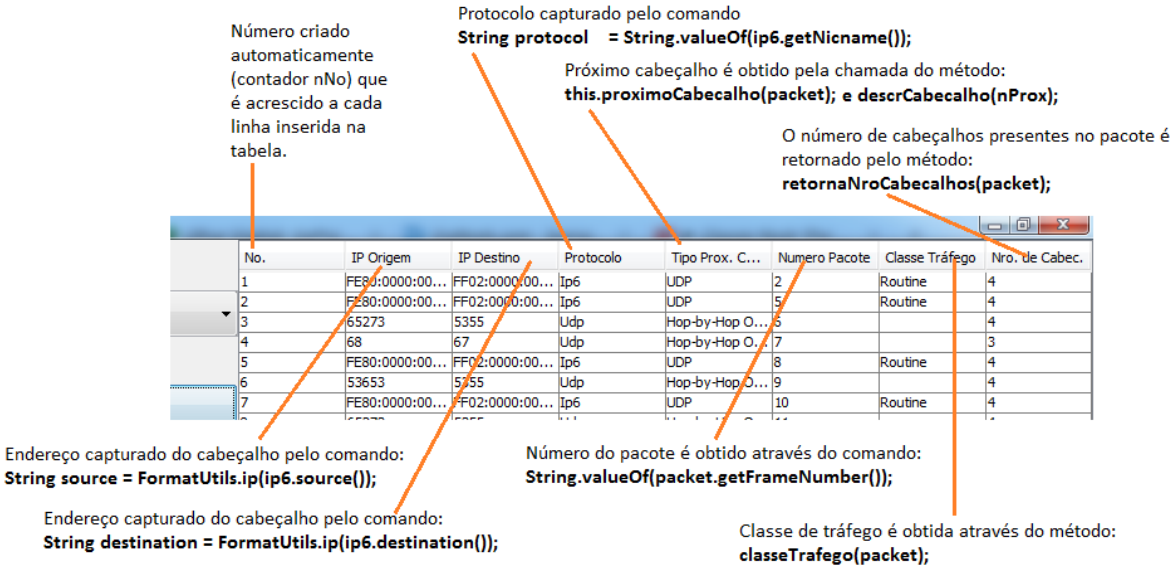
Código desenvolvido:

```
public int classeTrafego (PcapPacket packet){
    int classeTrafego = -1;
    int tipo = -1;

    Ip6 ip6 = new Ip6();
    if  (packet.hasHeader(ip6)) {
        int nParte1 = 0;
        int nParte2 = 0;

        byte[] header = ip6.getHeader();

        nParte1 = header[0];
        nParte1 = (nParte1 & Integer.parseInt("11110000", 2)) >> 4;
        nParte2 = header[1];
        nParte2 = (nParte2 & Integer.parseInt("00001111", 2)) << 4;

        classeTrafego = nParte1 | nParte2;

        if (classeTrafego>=0 && classeTrafego<=31){
            tipo = 0;
        }
        else{
            if (classeTrafego>=32 && classeTrafego<=63){
                tipo = 1;
            }
            else{
                if (classeTrafego>=64 && classeTrafego<=95){
                    tipo = 2;
                }
                else{
                    if (classeTrafego>=96 && classeTrafego<=127){
                        tipo = 3;
                    }
                    else{
                        if (classeTrafego>=128 && classeTrafego<=159){
                            tipo = 4;
                        }
                        else{
                            if (classeTrafego>=160 && classeTrafego<=191){
                                tipo = 5;

                            }
                            else{
                                if (classeTrafego>=192 && classeTrafego<=223){
                                    tipo = 6;
                                }
                                else{
                                    if (classeTrafego>=224 && classeTrafego<=255){
                                        tipo = 7;
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
    return tipo;
}
```

Funcionalidade 4: Tabela dos Fluxos

Em nossa tabela são exibidos os principais dados da captura.

Como os dados solicitados por esta funcionalidade são obtidos somente através de comandos nativos do jnetpcap, abaixo segue uma imagem de como estes dados são capturados.

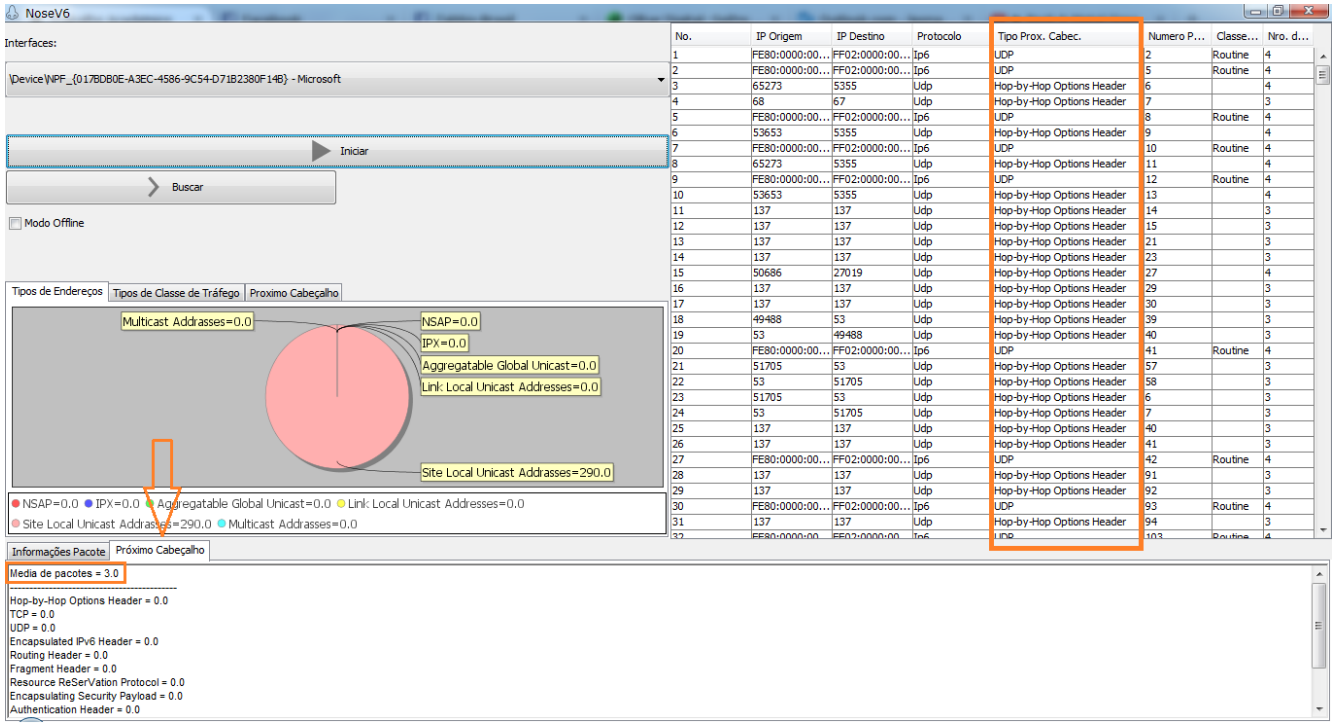


Como os dados foram obtidos através de métodos do jnetpcap, um algoritmo para isso não fica claro, sendo mais simples e objetivo para entendimento o próprio código desenvolvido e comentado:

```
if (packet.hasHeader(ip6)) {
    contaIp6++;
    // -- captura dadps de origem e destino
    String source      = FormatUtils.ip(ip6.source());
    String destination = FormatUtils.ip(ip6 .destination());
    String protocol    = String.valueOf(ip6.getNicname());
    int    nProx       = this.proximoCabecalho(packet);
    String nextHeader  = descrCabecalho(nProx);
    String number      = String.valueOf(packet.getFrameNumber());
    String cTipTraf    = "";
    int nroCab = retornaNroCabecalhos(packet);
    totalNroCab = totalNroCab + nroCab;
    // -- pega dados para estatistica 1
    int nEstatitica = this.estatisticaEndereco(packet);
    // -- atuaiza dados de graficos
    Nodo nodoAux = listG1.get(nEstatitica);
    nodoAux.setnValor(new Double(nodoAux.getnValor() + 1));
    listG1.set(nEstatitica, nodoAux);
    PieDataset dataset1 = g1.criaDataSet(listG1);
    PiePlot plot = (PiePlot) g1.getChart().getPlot();
    plot.setDataset(dataset1);
    int nClassTraf = this.classeTrafego(packet);
    // -- atuaiza dados de graficos
    Nodo nodoAux2 = listG2.get(nClassTraf);
    nodoAux2.setnValor(new Double(nodoAux2.getnValor() + 1));
    listG2.set(nClassTraf, nodoAux2);
    PieDataset dataset2 = g2.criaDataSet(listG2);
    PiePlot plot2 = (PiePlot) g2.getChart().getPlot();
    plot2.setDataset(dataset2);
    cTipTraf = nodoAux2.getcTitulo();
    // -- atuaiza dados de graficos
    int indice = this.proximoCabecalho(packet);
    int posicao = converteIndice(indice);
    Nodo nodoAux3 = listG3.get(posicao);
    nodoAux3.setnValor(new Double(nodoAux3.getnValor() + 1));
    //listG3.set(posicao, nodoAux3);
    PieDataset dataset3 = g3.criaDataSet(listG3);
    PiePlot plot3 = (PiePlot) g3.getChart().getPlot();
    plot3.setDataset(dataset3);
    // -- atualiza tabela
    DefaultTableModel model = (DefaultTableModel) table.getModel();
    model.addRow(new Object[]{nNo++, source, destination, protocol,nextHeader, number
    ,cTipTraf,nroCab});
    // -- preenche textarea com informacoes do pacote
    this.listheader.add(packet.toString());
}

Daqui para baixo é UDP, usado somente para teste, então não é necessário exibir...
```

Funcionalidade 5: Estatística sobre o Próximo Cabeçalho

Na funcionalidade 5, dividimos o código em duas partes. Na primeira parte, referente ao tipo do próximo cabeçalho, colocamos o resultado ao lado de cada pacotes capturado, lá na tabela, por ser uma informação individual de cada um. Na segunda parte, onde é exibido a média de quantidade de cabeçalhos, colocamos o resultado na aba "Próximo Cabeçalho", na parte inferior da tela. Abaixo uma imagem com o posicionamento das informações.



<u>Algoritmo para obter a informação do próximo cabeçalho:</u>

nProximo = 0
Se (pacote tem cabeçalho IPv6)
  vetor de bytes header = cabeçalho IPv6
  nProximo = header[6] *(header na posição 6 contém o byte com o valor do tipo do próximo cabeçalho)*
Retorna nProximo

Após, outra função é chamada para converter o valor encontrado em uma string.

descrCabecalho (valor)
  Caso valor = 0 retorna "Hop-by-Hop Options Header"
  Caso valor = 6 retorna "TCP"
  Caso valor = 17 retorna "UDP"
  Caso valor 41 retorna "Encapsulated IPv6 Header"
  Caso valor = 43 retorna "Routing Header"
  Caso valor = 44 retorna "Fragment Header"

Caso valor = 46 retorna "Resource ReSerVation Protocol"
Caso valor = 50 retorna "Encapsulating Security Payload"
Caso valor = 51 retorna "Authentication Header"
Caso valor = 58 retorna "ICMPv6"
Caso valor = 59 retorna "No next header"
Caso valor = 60 retorna "Destination Options Header";

Codigo desenvolvido:

```
public int proximoCabecalho(PcapPacket packet){
    int nProximo = 0;

    Ip6 ip6 = new Ip6();
    if  (packet.hasHeader(ip6)) {
        byte[] header = ip6.getHeader();
        nProximo = header[6];
    }

    return nProximo;

}
```

```
//Recebe um valor de cabeçalho e converte em uma string
public String descrCabecalho (int valor){

    String descr = null;

    switch (valor){
    case 0:
        descr = "Hop-by-Hop Options Header";
        break;
    case 6:
        descr = "TCP";
        break;
    case 17:
        descr = "UDP";
        break;
    case 41:
        descr = "Encapsulated IPv6 Header";
        break;
    case 43:
        descr = "Routing Header";
        break;
    case 44:
        descr = "Fragment Header";
        break;
    case 46:
        descr = "Resource ReSerVation Protocol";
        break;
    case 50:
        descr = "Encapsulating Security Payload";
        break;
    case 51:
        descr = "Authentication Header";
        break;
    case 58:
        descr = "ICMPv6";
        break;
    case 59:
        descr = "No next header";
        break;
    case 60:
        descr = "Destination Options Header";
        break;
    }
    return descr;
}
```

Algoritmo para obter a média de cabeçalhos nos pacotes:

A cada pacote com cabeçalho IPv6 lido, uma variável contadora chamada "contaIp6" é acrescida em uma unidade. Assim, teremos a quantidade total de pacotes IPv6 lidos.
Após isso, uma variável chamada "totalNroCab", que possui o total de cabeçalhos lidos em todos os pacotes, é acrescida com o número de cabeçalhos do pacote atual.

Por último, uma variável chamada "mediaPct" recebe a divisão de "totalNroCab" por "contaIp6", resultando na média esperada.

Isso então é escrito em tela, no local mencionado anteriormente.

```
this.textEstat.setText(this.escreveEst());
mediaPct = totalNroCab/contaIp6;
System.out.println("Media pacotes = "+mediaPct);
```

Funcionalidade 6: Duas estatísticas sobre o Próximo Cabeçalho

Nesta funcionalidade, como já havíamos feito todas as verificações para os tipos de próximos cabeçalhos, implementamos estatísticas para todos os tipos de próximos cabeçalhos.

Estas estatísticas podem ser vistas tanto pela aba "Próximo Cabeçalho" na parte inferior da interface (modo texto), tanto quanto pelo gráfico que é plotado na interface com o nome de "Próximo Cabeçalho".

Descrição do algoritmo:

Uma lista é criada para armazenar os dados do gráfico. Nesta lista, cada posição da mesma, representa um tipo de próximo cabeçalho. Então a cada pacote lido, se o mesmo contiver cabeçalho IPv6, pegamos seu campo Próximo Cabeçalho(com uma função usada em funcionalidades anteriores), acessamos a posição da lista com este tipo e acrescentamos uma unidade ao contador. Como isso é feito dinamicamente e a cada pacote capturado, no momento em que mandamos plotar o gráfico, o mesmo já estará atualizado e com estatísticas de todos os tipos.

Código implementado:

```
// -- atuaiza dados de graficos

int indice = this.proximoCabecalho(packet);
int posicao = converteIndice(indice);

Nodo nodoAux3 = listG3.get(posicao);
nodoAux3.setnValor(new Double(nodoAux3.getnValor() + 1));

//listG3.set(posicao, nodoAux3);
PieDataset dataset3 = g3.criaDataSet(listG3);

PiePlot plot3 = (PiePlot) g3.getChart().getPlot();
plot3.setDataset(dataset3);
```