Centro de Informática
Universidade Federal da Paraíba

Disciplina: Redes sem fio
Professor: Fernando Menezes

Relatório do projeto

Simulador de redes sem fio

Jaqueline Donin Noleto - 20160144455
Johan Kevin Estevão de Freitas - 20170171741
Vanessa Bonifácio e Silva - 11423466

1. Introdução

Neste trabalho, será apresentado de forma simples e objetiva a implementação de uma rede sem fio do tipo Ad-Hoc utilizando a linguagem Python, tendo como objetivo mostrar o tráfego de pacotes saindo da origem até o destino passando por nós intermediários utilizando os conhecimentos e conceitos adquiridos nas aulas presenciais de Redes sem fio, junto a pesquisas auxiliares.

2. Camada de rede

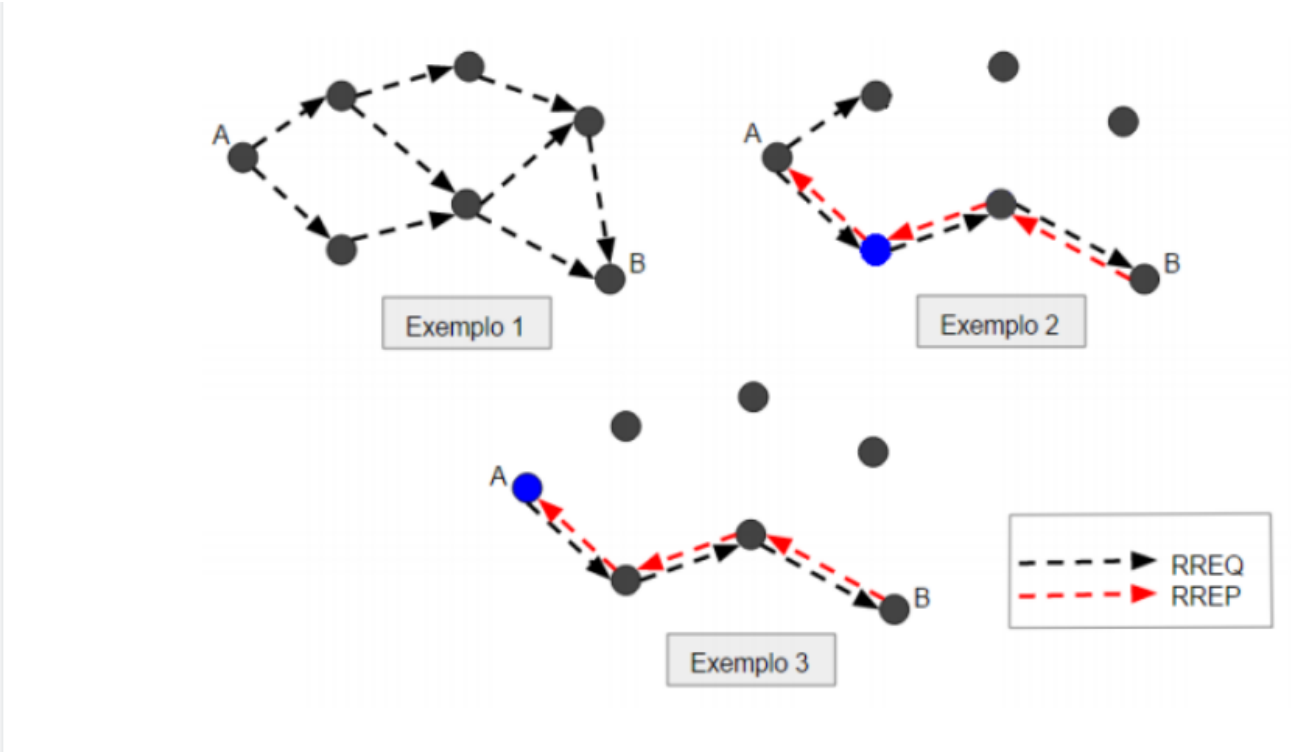
A camada de rede é responsável por prover conectividade ao computador e selecionar caminhos
para que os pacotes de dados possam trafegar. Isso é feito através do roteamento. Dentre os
vários protocolos de roteamentos, já estudados neste semestre, o protocolo escolhido para
implementação foi o DSR (Dynamic Source Routing), a razão da escolha foi pelo fato de que
a principal característica deste protocolo é possuir o melhor desempenho em ambientes onde
a velocidade de mobilidade dos nós é baixa, que é o caso mostrado na implementação.

2.1 O protocolo Dynamic Source Routing (DSR)

O Dynamic Source Routing é um algoritmo de roteamento para RSSF baseado em um método
conhecido como roteamento por fonte. É um protocolo reativo. Sendo assim, gera um anúncio
de rotas apenas quando demandado. Suas etapas de roteamento são descritas nas subseções a seguir.

2.1.1 Descoberta de Rotas

O processo de descoberta de rotas inicia-se quando um nó origem analisa se
em sua cache existe alguma rota salva que forneça uma comunicação entre ele
e o nó de destino. Caso não possua uma rota salva o mesmo emite, em broadcast,
um sinal de RREQ para os nós vizinhos.
Caso algum nó roteador possua informações em cache da rota até o destino, ele envia ao
nó origem uma mensagem contendo o endereço de todos os nós até o destino e o processo
de busca por rotas é então finalizado, dando início à transmissão de dados; caso contrário,
o mesmo salva seu endereço na tabela de roteamento e retransmite o sinal broadcast de
requerimento de rotas até que seja encontrado um roteador com informações sobre a rota
ao destino ou o próprio nó destino receba o pacote RREQ.
A Figura 1 ilustra exemplos das três possíveis situações que podem vir a ocorrer em uma
etapa de descoberta de rotas de um protocolo DSR. O Exemplo 1 ilustra o caso em que o nó "A"
a necessita de efetuar uma transmissão até o nó B, porém nenhum dos nós possui dados
salvos em suas tabelas de roteamento que ligue uma rota entre o nó A e B.
O caso mostrado no Exemplo 2 mostra um exemplo em que um nó roteador (azul), ao receber
um RREQ e analisar sua tabela de roteamento, percebe que possui uma rota válida entre
o nó A e nó B. Feito isto todo o processo de descoberta de rotas é encerrado e a
transmissão é iniciada.
No caso representado no Exemplo 3, o próprio nó A possui em sua cache a tabela de
roteamento com todos os endereços do nós que criam uma rota entre ele e o nó destino B.
Neste caso, a etapa de descoberta de rotas não efetua nenhuma transmissão em broadcast
e já inicia a transmissão de dados.



2.1.2. Manutenção de Rotas

No que diz respeito à manutenção das rotas no DSR o procedimento é feito de modo que,
o nó de origem transmite um sinal de verificação (HELLO) através da rota até o destino
e aguarda o recebimento de um pacote de dados de confirmação. Caso algum nó não responda
ao sinal de verificação isto é considerado erro na comunicação. O erro é reportado ao nó
fonte através de um pacote ERROR transmitido para nó fonte contendo o endereço do nó que
identificou o erro e o nó que não respondeu ao sinal HELLO, fazendo com que se inicie um
novo processo de descoberta de rotas.

3. Camada de enlace

A camada física junto com a camada de enlace forma a tecnologia de transmissão.
A função das tecnologias de transmissão é movimentar os dados. A real movimentação
dos sinais é feita na camada física, mas para transformar esses sinais em dados,
saber quando se pode transmití-los, por quanto tempo se pode transmitir e receber dados,
como identificar de quem e para quem o pacote deve ser encaminhado, como cada pacote deve
percorrer um determinado caminho. Tudo isso é feito na camada de enlace.
Dentre vários protocolos de acesso ao meio, o escolhido para implementação desse projeto
foi o CSMA/CA, por ser teoricamente simples e se adequar ao roteamento mostrado nessa implementação.

3.1 O protocolo Carrier Sense Multiple Access com collision avoidance (CSMA/CA)

O CSMA/CA é um protocolo da subcamada MAC para transmissão de operadoras em redes 802.11.
Ao contrário do CSMA / CD (Detecção de colisão / acesso múltiplo do sentido da operadora),
que trata das transmissões após uma colisão, o CSMA / CA age para evitar colisões antes que
elas aconteçam.

No CSMA / CA, assim que um nó recebe um pacote a ser enviado, ele verifica se o canal está
disponível (nenhum outro nó está transmitindo no momento). Se o canal estiver livre, o pacote
será enviado. Se o canal não estiver livre, o nó aguardará um período de tempo escolhido aleatoriamente
e, em seguida, verificará novamente se o canal está disponível. Esse período de tempo é
chamado de fator backoff e é contado por um contador de backoff. Se o canal estiver livre quando
o contador de backoff chegar a zero, o nó transmite o pacote. Se o canal não estiver livre
quando o contador de backoff chegar a zero, o fator de backoff será definido novamente e o
processo será repetido.

4.0 Trabalho final da disciplina de Redes sem fio

O objetivo do trabalho é implementar uma rede sem fio, utilizando os protocolos descritos
anteriormente, através da simulação das camadas de rede, enlace e física.

Imports

In [1]:

```
import random
import math as math
```

Variaveis globais

In [2]:

```
nos = []
pacotes = []
indicesParaEnvio = []
indicesParaVer = []
proximoAEnviar = []
```

Estrutura do projeto

1. Criação das classes para simulação da rede sem fio:

1. Cabecalhos: definição do cabeçalho de acordo com a camada (enlace ou rede)

2. Rota: construtor com as variáveis destino e sequência para o destino

3. Pacote: construtor com as variáveis identificador, lista de cabeçalhos das camadas, mensagem e duração.

1. Método para adicionar cabeçalho do enlace a lista de cabeçalhos do pacote
2. Método para retornar um pacote criado
3. Método para exibir a mensagem do pacote
4. Método para obter o cabeçário da camada de enlace
5. Método para obter o cabeçário da camada de rede
6. Método para atualizar a lista de sequência do cabeçário

4. Camada Fisica: simulador da camada física, construtor com ...

1. Método para verificar os vizinhos
2. Método para enviar pacotes
3. Método para receber pacotes

5. Camada de enlace: simulador da camada de enlace, construtor com backoff, acesso ao meio, cama física, lista de pacotes lidos.

1. Método para enviar pacotes
2. Método para receber pacotes
3. Método para detectar o meio
4. Método para adicionar pacote a camada de enlace

6. Host: define um nó da rede sem fio

1. Método para criar pacote

7. Camada de rede: simulador da camada de rede, construtor com camada de enlace, lista de pacotes, lista de pacotes de requisição, lista de destinos não salvos, lista de rotas.

1. Método para enviar pacote de Reposta da rota
2. Método para enviar pacote de requisição da rota
3. Método para receber pacote
4. Método para criar um pacote e adicionar cabeçalho de rede
5. Método para enviar pacotes

2. Função main

1. Instanciação dos Nós da rede
2. Definição do tempo de execução da rede
3. Envio e recebimento de pacotes

Implementações

In [3]:

```
class Cabecalho:

    def __init__(self, camada, macOrigem, macDestino, numero, requesicao, sequen
Num, sequenList):

        #Cabeçario para a camada de enlace
        if(camada == "ENLACE"):
            self._camada = "ENLACE"
            self._macOrigem = macOrigem
            self._macDestino = macDestino
            self._numero = numero

        #Cabeçario para a camada de rede
        if( camada == "REDE"):
            self._camada = "REDE"
            self._macDestino = macDestino
            self._requesicao = requesicao
            self._sequenNum = sequenNum
            self._sequenList = sequenList
```

In [4]:

```
class Rota:

    def __init__(self, destino, sequencia):

        self._destino = destino
        self._sequence = sequencia
```

In [5]:

```
import math as math
#Verifica se um nó alcança outro
def alcance(centroX, centroY, raio, x, y):
    #calculo de distância usando a fórmula de distância entre dois pontos
    distancia = math.sqrt((centroX - x) ** 2 + (centroY - y) ** 2)
    #se alcancar retorna verdadeiro
    if(distancia <= raio):
        return True
    #caso contrario retorna falso
    else:
        return False
```

In [6]:

```
class Pacote:

    def __init__(self, mensagem, duracao):
        self._id = -1
        self._dados = mensagem
        self._duracao = duracao
        self._cabecalhos = []

    #Retorna um pacote criado
    @staticmethod
    def criaPacote(mensage, duracao):
        return Package(mensage, duracao)

    #Obtem cabeçalho da camada de rede
    def getCabecalhoRede(self):
        for cabecalho in self._cabecalhos:
            if(cabecalho._camada == "REDE"):
                return cabecalho

    #Obtem cabeçalho da camada de enlace
    def getCabecalhoEnlace(self):
        for cabecalho in self._cabecalhos:
            if(cabecalho._camada == "ENLACE"):
                return cabecalho

    #Adiciona novos cabeçalhos na lista
    def addCabecalho(self, cabecalho):
        self._cabecalhos.append(cabecalho)

    #Exibe as informações do pacote
    def ExibeInfoPacote(self):
        print("Dados: ", self._dados)

    def atualizaSequen(self, sequencia):
        for cabecalho in self._cabecalhos:
            if(cabecalho._camada == "REDE"):
                cabecalho._sequenList = sequencia
```

In [7]:

```
class CamadaFisica:

    def __init__(self, x, y, id, tamanho):
        self._x = x
        self._y = y
        self._id = id
        self._tamanho = tamanho
        self._vizinhos = []
        self._pacotesEnviados = []
        self._pacotesRecebidos = []
        self._pacotesSalvos = []

    #Descobre nós vizihos e add na lista
    def encontraVizinhos(self):
        for no in nos:
            #se o nó não for ele mesmo e se puder alcançá-lo
            if((no._id != self._id) and (alcance(self._x, self._y, self._tamanho
, no._camadaRede._camadaEnlace._camadaFisica._x, no._camadaRede._camadaEnlace._c
amadaFisica._y))):
                # se o não estiver na lista de vizinhos
                if(no not in self._vizinhos):
                    #add a lista de vizinhos
                    self._vizinhos.append(no)

    #Recebe pacote
    def recebePacote(self, pacote):
        #Add a lista global
        indicesParaVer.append(self._id)
        #Add a lista local de recebidos
        self._pacotesRecebidos.append(pacote)


    #Envia pacote
    def enviaPacote(self):
        #Encontra e add os vizinhos na lista
        self.encontraVizinhos()
        #Percorre vizinhos
        for no in self._vizinhos:
            #Envia pacote aos vizinhos
            no._camadaRede._camadaEnlace._camadaFisica.recebePacote(self._pacote
sEnviados[0])
        #Add o pacote na lista de salvos
        self._pacotesSalvos.append(self._pacotesEnviados.pop(0))
```

In [8]:

```
class CamadaEnlace:

    def __init__(self, camadaFisica):
        self._backoff = 0
        self._mediumAcess = True
        self._camadaFisica = camadaFisica
        self._pacotesLidos = []

    #Envia os pacotes para outros nós
    def enviaPacote(self):
        self._acessoAoMeio = self.acessoAoMeio()
        #Verifica se o meio está livre
        if(self._acessoAoMeio == True):
            #Vefifica se a lista de pacotes para envio da camada da física está
 vazia
            if(self._camadaFisica._pacotesEnviados != []):
                #Verifica se o no não está em backoff
                if(self._backoff == 0):
                    #Envia pacote pela camada fisica
                    self._camadaFisica.enviaPacote()
                #Caso contrário
                else:
                    #Add o no na lista global de proximo a enviar
                    proximoAEnviar.append(self._camadaFisica._id)
                    #Subtrai backoff
                    self._backoff = (self._backoff - 1)

        #Caso o meio esteja ocupado o host entrará em backoff
        else:
            #Vefifica se a lista de pacotes para envio da camada da física está
 vazia
            if(self._camadaFisica._pacotesEnviados != []):
                #Verifica se o no não está em backoff
                if(self._backoff == 0):
                    #coloca no em backoff
                    self._backoff = randint(1, 8)
                    #Exibe backoff
                    self.exibeBackoff(self._camadaFisica._id, self._backoff)
                    #Add o no na lista global de proximo a envir
                    proximoAEnviar.append(self._camadaFisica._id)

    #Exibe backoff
    def exibeBackoff(self, id, backoff):
        print("Id\n", id,": No entrou em Backoff, valor: ", backoff)

    #Pacotes recebidos
    def recebePacote(self):
        #Verifica se mais de uma pacote chegou no instante, detecta colisão
        if(len(self._camadaFisica._pacotesRecebidos) > 1):
            self._camadaFisica._pacotesRecebidos.clear()
            self.exibeColisao(self._camadaFisica._id)
        #Caso não haja colisão
        else:
            #Verifica se existe pacote para receber
            if(len(self._camadaFisica._pacotesRecebidos) == 1):
                pacote = self._camadaFisica._pacotesRecebidos.pop(0)
                cabecalho = pacote.getCabecalhoEnlace()
                #Verifica se o pacote é para aquele no
                if(cabecalho._macDestino == self._camadaFisica._id):
```

```
                    #Add pacote a lista de lidos
                    self._pacotesLidos.append(pacote)
                elif(cabecalho._macDestino  == -1):
                    #Add pacote a lista de lidos
                    self._pacotesLidos.append(pacote)

    #Exibe colisão
    def exibeColisao(self, id):
        print("Houve Colisão neste no \nID: ", id)

    #Detecta o meio
    def acessoAoMeio(self):
        if(self._camadaFisica._pacotesRecebidos == []):
            return True
        else:
            return False

    #Add pacote na camada fisica
    def addPacote(self, pacote, macDestino):
        cabecalho = Cabecalho("ENLACE", self._camadaFisica._id, macDestino, 0, -
1, -1, -1)
        pacote.addCabecalho(cabecalho)
        self._camadaFisica._pacotesEnviados.append(pacote)
```

In [9]:

```
class No:

    def __init__(self, id, tamanho, x, y):
        self._id = id
        #Adds a si mesmo na lista global de nós
        nos.append(self)
        self._camadaRede = CamadaRede(CamadaEnlace(CamadaFisica(x, y, id, tamanh
o)))
    #Criar pacote
    def criarPacote(self, duracao, macDestino, mensagem):
        self._camadaRede.addPacote(macDestino, mensagem, duracao)
        self.exibePacote(self._camadaRede._camadaEnlace._camadaFisica._id, macDe
stino)
    #Exibe pacote criado
    def exibePacote(self, macDestino, id):
        print("Pacote criado \n Destino:", macDestino, "\n ID: ", id)
```

In [10]:

```
class CamadaRede:

    def __init__(self, camadaEnlace):
        self._camadaEnlace = camadaEnlace
        self._listaPacotes = []
        self._listaRREQS = []
        self._ListasRotasEspera = []
        self._rotas = []


    #Envia pacote de resposta de rota
    def enviaRREP(self, macDestino, sequencia, rota):
        #Cria um pacote e inseri o cabeçalho da camada de rede
        cabecalho = Cabecalho("REDE", self._camadaEnlace._camadaFisica._id, macD
estino, -1, 1, -1, sequencia)
        pacote = Pacote(rota, 1)
        pacote.addCabecalho(cabecalho)
        msg = "Enviando um RREP com destino para"
        #exibe pacote criado
        self.exibePacote(msg,  self._camadaEnlace._camadaFisisca._id, macDestino
)
        #Define a rota requisitada
        for indice,mac in enumerate(cabecalho._sequenList):
            if(mac == self._camadaEnlace._camadaFisica._id):
                proximoDestino = cabecalho._sequenList[indice+1]
                proximoPacote = pacote
                self._camadaEnlace.addPacote(proximoPacote, proximoDestino)
                break


    #Manda o pacote de requisição de rota
    def enviaRREQ(self, macDestino):
        #Inicializa uma sequencia e coloca o seu ID como primeiro
        sequencia = []
        sequencia.append(self._camadaEnlace._camadaFisica._id)
        sequenNum = random.randint(1,128733788)
        self._listaRREQS.append(sequenNum)
        #Cria um pacote e inseri o cabeçalho da camada de rede
        cabecalho = Cabecalho("REDE", self._camadaEnlace._camadaFisica._id, macD
estino, -1, 0, sequenNum, sequencia)
        pacote = Pacote("", 1)
        pacote.addCabecalho(cabecalho)
        msg = "Enviando um RREQ "
        #exibe pacote criado
        self.exibePacote(msg, self._camadaEnlace._camadaFisica._id, None)
        #add a camada de enlace o pacote
        self._camadaEnlace.addPacote(pacote, -1)

    #Recebe e trata o pacote recebido na camada de rede
    def recebePacote(self):
        #Chama a função de tratar pacote recebido na camada de enlace
        self._camadaEnlace.recebePacote()
        #Verifica se tem pacotes recebidos
        if(self._camadaEnlace._pacotesLidos != []):
            pacote = self._camadaEnlace._pacotesLidos.pop(0)
            cabecalho = pacote.getCabecalhoRede()
            #Se o pacote for recebido for de dados
            if(cabecalho._requesicao == -1):
                #Verifica se o pacote é para aquele no
```

```
                if(cabecalho._macDestino == self._camadaEnlace._camadaFisica._id
):
                    msg = "Pacote de dados: "
                    self.exibePacote(msg, self._camadaEnlace._camadaFisica._id,
pacote._dados)
                else:
                    msg = "Chegada de pacote de dados mas não é pra mim "
                    self.exibePacote(msg, self._camadaEnlace._camadaFisica._id,
0)
                    msg_2 = "Enviando pacote de dados para o nó seguinte"
                    self.exibePacote(msg_2, self._camadaEnlace._camadaFisica._id
, 0)
                    for indice,mac in enumerate(pacote._cabecalhos[0]._sequenLis
t):
                        if(mac == self._camadaEnlace._camadaFisica._id):
                            proximoDestino = cabecalho._sequenList[indice-1]
                            break
                    pacote._cabecalhos.pop(1)
                    self._camadaEnlace.addPacote(pacote, proximoDestino)
                    indicesParaEnvio.append(self._camadaEnlace._camadaFisica._id
)

            #Se o pacote for recebido for um RREQ
            elif(cabecalho._requesicao == 0):
                msg = "Chegada de pacote RREQ"
                self.exibePacote(msg, self._camadaEnlace._camadaFisica._id, cabe
calho._sequenNum)
                #Verifica se esse RREQ já foi recebido pelo nó
                if(not cabecalho._sequenNum in self._listaRREQS):
                    self._listaRREQS.append(cabecalho._sequenNum)
                    cabecalho._sequenList.append(self._camadaEnlace._camadaFisic
a._id)

                    #Verifica se o RREQ é para o nó
                    if(cabecalho._macDestino == self._camadaEnlace._camadaFisica
._id):
                        msg = "Eu sou o destino do RREQ"
                        self.exibePacote(msg, self._camadaEnlace._camadaFisica._
id, 0)
                        rota = cabecalho._sequenList
                        macDestino = rota[0]
                        sequenParaFonte = rota
                        sequenParaFonte.reverse()
                        self.enviaRREP(macDestino,sequenParaDestino, rota)
                        indicesParaEnvio.append(self._camadaEnlace._camadaFisica
._id)

                    else:
                        print("ID", self._camadaEnlace._camadaFisica._id, ": Eu
 não sou o destino do RREQ")
                        self._camadaEnlace.addPacote(pacote, -1)
                        indicesParaEnvio.append(self._camadaEnlace._camadaFisica
._id)

                else:
                    msg = "Ja tenho esse RREQ"
                    self.exibePacote(msg, self._camadaEnlace._camadaFisica._id,
cabecalho._sequenNum)
            #Se o pacote for recebido for um RREP
            elif(cabecalho._requesicao == 1):
                destino = cabecalho._macDestino
```

```
                msg = "Chegada de pacote RREP: "
                self.exibePacote(msg, self._camadaEnlace._camadaFisica._id, cabe
calho._sequenList)
                #Verifica se o RREP é para o nó
                if(destino == self._camadaEnlace._camadaFisica._id):
                    msg = "Eu sou o destino do RREP"
                    self.exibePacote(msg, self._camadaEnlace._camadaFisica._id,
0)
                    msg_2 = "Enviando dados"
                    self.exibePacote(msg, self._camadaEnlace._camadaFisica._id,
0)
                    sequenciaParaDestino = pacote._dado
                    rota = Rota(cabecalho._sequenList[0],sequenciaParaDestino)
                    self._rotas.append(rota)
                    indicesParaEnvio.append(self._camadaEnlace._camadaFisica._id
)

                else:
                    msg = "Eu não sou o destino do RREP"
                    self.exibePacote(msg, self._camadaEnlace._camadaFisica._id,
0)
                    for indice,mac in enumerate(cabecalho._sequenList):
                        if(mac == self._camadaEnlace._camadaFisica._id):
                            proximoDestino = cabecalho._sequenList[indice+1]
                            proximoPacote = pacote
                            pacote._cabecalhos.pop(1)
                            self._camadaEnlace.addPacote(proximoPacote, proximoD
estino)
                            indicesParaEnvio.append(self._camadaEnlace._camadaFi
sica._id)
                            break


    #Cria um pacote novo e adicionar o cabeçalho de rede
    def addPacote(self, macDestino, mensage, tempo):
        pacote = Pacote(mensage, tempo)
        cabecalho = Cabecalho("REDE",self._camadaEnlace._camadaFisica._id, macDe
stino, -1, -1, -1, None)
        pacote.addCabecalho(cabecalho)
        self._listaPacotes.append(pacote)

    #Envia os pacotes da camada de rede
    def enviaPacote(self):
        #Verifica pacotes a serem enviados
        if(self._listaPacotes != []):
            pacote = self._listaPacotes[0]
            cabecalho = pacote.getCabecalhoRede()
            sequencia = None

            for rota in self._rotas:
                if(rota._destino == pacote._cabecalhos[0]._macDestino):
                    sequencia = rota._sequencia
                    if (pacote._cabecalhos[0]._macDestino in self._ListasRotasEs
pera):
                        self._ListasRotasEspera.remove(pacote._cabecalhos[0]._ma
cDestino)
            #Verifica se a rota para o destino é conhecida
            if(sequencia != None):
                pacote.atualizaSequencia(sequencia)
                self._listaPacotes.pop(0)
```

```
                for indice,mac in enumerate(pacote._cabecalho[0]._sequenList):
                    if(mac == self._camadaEnlace._camadaFisica._id):
                        proximoDestino = cabecalho._sequenList[indice-1]
                        break

                self._camadaEnlace.addPacote(pacote, proximoDestino)
                indicesParaEnvio.append(self._camadaEnlace._camadaFisica._id)

            elif(not cabecalho._macDestino in self._ListasRotasEspera):

                self._ListasRotasEspera.append(pacote._cabecalhos[0]._macDestino
)
                self.enviaRREQ(pacote._cabecalhos[0]._macDestino)


        #Chama a função de enviar pacotes da camada de enlace
        self._camadaEnlace.enviaPacote()

    def exibePacote(self, mensagem,  id, macDestino):
        print(mensagem," ", macDestino, "\nID: ", id)
```

In [17]:

```
#Main
#Instaciando os hosts
Jaque = No(0,2,1,1)
Johan = No(1,2,2,1)
Vanessa = No(2,2,3,1)
Fernando = No(3,2,4,1)

#Loop do tempo
for i in range(20):

    print("")
    print("Tempo: ", i)
    print("")

    #Loop para percorrer
    for no in nos:

        #Numero aleatorio entre 0 e 100 para probabilidade de criação de pacotes
        rand = random.randint(0, 100)

        #Numero aleatorio entre 0 e a quantidade de hosts, para escolher um para
 enviar
        enviar = random.randint(0, len(nos)-1)

        #Teste se rand é menor que 3, gerando assim uma probabilidade de 0.3
        if(rand < 3):

                #Se o ID do destino for diferente do dele mesmo, adiciona o paco
te no host
                if(enviar != (no._camadaRede._camadaEnlace._camadaFisica._id)):
                        no.criarPacote(enviar, "Oi, Tudo bem?", 1)
                        indicesParaEnvio.append(no._camadaRede._camadaEnlace._ca
madaFisica._id)


    if(proximoAEnviar != []):
        for i in proximoAEnviar:
                indicesParaEnvio.append(i)
    del proximoAEnviar[:]

    #Existe algum nó querendo receber, recebe
    for j in indicesParaVer:
            nos[j]._camadaRede.recebePacote()
    del indicesParaVer[:]

    #Existe algum nó querendo transmitir, transmite naquele instante se posssíve
l
    for i in indicesParaEnvio:
            nos[i]._camadaRede.enviaPacote()
    del indicesParaEnvio[:]
```

```
Tempo:  0

Chegada de pacote RREQ   93653523
ID:  0
Ja tenho esse RREQ   93653523
ID:  0
Chegada de pacote RREQ   93653523
ID:  2
Ja tenho esse RREQ   93653523
ID:  2
Chegada de pacote RREQ   93653523
ID:  3
Ja tenho esse RREQ   93653523
ID:  3

Tempo:  1

Pacote criado
 Destino: 3
 ID:  Oi, Tudo bem?
Houve Colisão neste no
ID:  1
Houve Colisão neste no
ID:  2

Tempo:  2


Tempo:  3


Tempo:  4


Tempo:  5

Pacote criado
 Destino: 3
 ID:  Oi, Tudo bem?
Pacote criado
 Destino: 1
 ID:  Oi, Tudo bem?

Tempo:  6


Tempo:  7


Tempo:  8


Tempo:  9

Pacote criado
 Destino: 0
 ID:  Oi, Tudo bem?

Tempo:  10
```

```
Tempo:  11

Pacote criado
 Destino: 0
 ID:  Oi, Tudo bem?

Tempo:  12


Tempo:  13


Tempo:  14


Tempo:  15

Pacote criado
 Destino: 1
 ID:  Oi, Tudo bem?

Tempo:  16


Tempo:  17

Pacote criado
 Destino: 1
 ID:  Oi, Tudo bem?
Pacote criado
 Destino: 0
 ID:  Oi, Tudo bem?

Tempo:  18


Tempo:  19

Pacote criado
 Destino: 3
 ID:  Oi, Tudo bem?
Pacote criado
 Destino: 1
 ID:  Oi, Tudo bem?
```