UNILASALLE
CENTRO UNIVERSITÁRIO LA SALLE

CRISTIANE DE ÁVILA

ESTUDO DE PADRÕES DE MOBILIDADE REALÍSTICOS PARA REDES AD HOC

CANOAS, 2008

CRISTIANE DE ÁVILA

ESTUDO DE PADRÕES DE MOBILIDADE REALÍSTICOS PARA REDES AD HOC

Trabalho de conclusão apresentado para a banca examinadora do curso de Ciência da Computação do Centro Universitário La Salle, como exigência parcial para a obtenção do grau de Bacharel em Ciência da Computação, sob orientação do Prof. Me. Gustavo Passos Tourinho.

CANOAS, 2008

TERMO DE APROVAÇÃO

CRISTIANE DE ÁVILA

ESTUDO DE PADRÕES DE MOBILIDADE REALÍSTICOS PARA REDES AD HOC

Trabalho de conclusão aprovado como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação do Curso de Ciência da Computação do Centro Universitário La Salle - UNILASALLE, pela seguinte banca examinadora:

Prof. Me. Débora Nice Ferrari Barbosa

Unilasalle

Prof. Me. Gaspare Giuliano Elias Bruno

Unilasalle

Canoas, 24 de Junho de 2008.

DEDICATÓRIA

Dedicado esse trabalho de conclusão aos meus pais Paulo e Maria pelo eterno incentivo e que nos momentos mais difíceis dessa jornada, sempre dedicaram o seu amor incondicional. Ao meu esposo Alexsandro pela paciência durante todos os momentos. Aos meus irmãos e demais familiares pelo carinho. Aos amigos que participaram dessa caminhada e de alguma forma me deram apoio. Enfim a todas as pessoas que contribuíram para que esse trabalho fosse desenvolvido.

AGRADECIMENTOS

Agradeço ao meu orientador, Gustavo Passos Tourinho pelo apoio e dedicação na conclusão deste trabalho.

Aos professores Cristina Nunes e Gaspare Giuliano Elias Bruno que compartilharam de seu tempo e conhecimento para que esse TC fosse uma realidade.

Ao colega da Universidade PUC-RS, Alberto Bengoa, pela ajuda durante o desenvolvimento desse trabalho.

Também gostaria de registrar aqui os meus grandes amigos que também contribuíram nessa caminhada, Tonismar Regis Bernardo, Nelson Buis Sonntag, Sabrina Palharini Ziani e Virginia Mattos Thoma e a todos que ofereceram seu apoio contribuindo em favor desse trabalho.

RESUMO

Atualmente a área de computação móvel está em constante desenvolvimento, tanto na comunidade acadêmica quanto comercialmente. Através da computação móvel é possível acessar serviços independente da localização física do usuário, além de prover mobilidade. Entre os tipos de redes sem fio existentes está a rede ad hoc, onde os nodos se movimentam de forma independente e não dependem de um ponto de acesso para realizar qualquer comunicação, pois os mesmos trabalham como roteadores, distribuindo rotas para a rede e possibilitando a comunicação direta entre os mesmos. Porém um dos principais problemas encontrados nesse tipo de rede é manter e determinar as rotas, devido à mobilidade dos nós. A maioria dos estudos realizados referente à mobilidade utilizam modelos sintéticos, que não simulam ambientes reais, não refletindo o verdadeiro desempenho do protocolo. Este trabalho apresenta alguns modelos de mobilidade realísticos existentes e exibe uma análise comparativa entre os mesmos através de simulações usando o Network Simulation (NS).

PALAVRAS-CHAVE: Redes *Ad Hoc*, Modelos de Mobilidade, Geradores de Cenário, Protocolo AODV (*Ad hoc On-Demand Distance Vector)*

ABSTRACT

Now a days the development of mobile computing is still increase in both academics and enterprise segment. Using mobile computing we are able to use different kinds of services regardless of user physical location. Wireless networks can be divided into two different segments called ad hoc and infrastructure. Using ad hoc networks, users can move independently and they dont need an access point to be able to comunicate with eath other. Evey node acts like a router to neighbours sharing route information. A significant number of papers describer the use of synthetic model that do not simulate real environment, so can not represent the protocol performance. This work present a couple of realistics mobile model. Also, it shows an comparative analysis between them using an network simulator called NS.

KEYWORDS: Ad Hoc Networks, Models of Mobility, generators Scenario, AODV Protocol (Ad Hoc On-Demand Distance Vector)

LISTA DE ABREVIATURAS E SIGLAS

AODV	Ad hoc On-Demand Distance Vector
BLS	Bureau of Labor Statistics
CBR	Constant Bit Rate
DARPA	Defense Advanced Research Projects Agency
DSR	Dynamic Source Routing
GIS	Geographic Information System
IP	Internet Protocol
NAM	Network Animator
MPLS	Multiprotocol Label Switching
NS2	Network Simulator 2
NSF	National Science Foundation
OM	Obstacle Mobility Model
OTCL	Object-oriented Toll Command Language
RERR	Route Error
RREP	Route Reply
RREQ	Route Request
TCP	Trasmission Control Protocol
TIGER	Topologically Integrated Geographic Encoding and Referencing
UDEL	Udel Model
UDP	User Datagram Protocol
UPF	Urban Pedestrian Flow
VBR	Variable Bit Rate
WFQ	Weight Fair Queueing

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A utilização de redes móveis tem evoluído de forma expressiva nos últimos anos com o surgimento de equipamentos como notebooks, telefones celulares, palmtops, entre outros (TANEMBAUM, 1997). Como características deste tipo de rede, destacam-se o uso do ar como meio físico ao invés da estrutura de cabeamento tradicional, e a da comodidade de conectividade em qualquer local (que possua suporte), proporcionado pelo uso de canais de rádio freqüência (rádio difusão) ou infravermelho na transmissão dos dados (PEREIRA, 2004).

As redes sem fio são classificadas de duas formas: redes infra-estruturadas e redes *ad hoc*. Segundo Pinheiro (2005), nas redes infra-estruturadas os nodos se comunicam através de dispositivos com suporte a mobilidade, conhecidos como *Access Point*. Não é possível a comunicação direta entre os nodos.

Diferente da rede infra-estruturada, as redes *ad hoc* não dependem de um *Access Point* para realizar a comunicação entre os nodos, pois os mesmos trabalham como estações de trabalho e roteadores, distribuindo rotas para a rede e possibilitando a comunicação direta entre os mesmos (STALLINGS, 2005). Essas redes são formadas por múltiplos nodos que se comunicam via uma comunicação ponto a ponto.

Atualmente esse tipo de rede vem sendo bastante analisada por pesquisadores da área, pois devido à mobilidade dos nodos pode haver uma troca constante da topologia, o que ocasiona problemas como perda de conectividade entre os nodos. Neste sentido, muitos pesquisadores tentam desenvolver novos algoritmos para descoberta de roteamento dinâmico, a fim de se obter protocolos de comunicação eficientes.

O bom funcionamento da rede é influenciado por suas propriedades de conectividade. Segundo (LASSILA; HYYTI¨A; KOSKINEN, 2005), a conectividade

depende do alcance da transmissão, do número de nodos na área e da distribuição dos nodos na rede. Para este último aspecto, modelos de mobilidade que imitam o comportamento de uma rede real são utilizados tanto em estudos analíticos quanto em simulações. Alguns estudos analisam propriedades de conectividade considerando uma distribuição uniforme para os nodos. Entretanto, é muito importante examinar o impacto da mobilidade quando se analisam propriedades de conectividade da rede, pois assim é possível prever onde os nodos estarão localizados na área depois que o movimento ocorra.

O uso de modelos de mobilidade ajuda a demonstrar o comportamento dos nodos dentro da rede, representando um cenário real ou hipotético que contém entidades em movimento e esta mobilidade impacta diretamente no desempenho do protocolo utilizado (CAMP; BOLENG; DAVIES, 2002).

O principal objetivo deste trabalho é realizar uma análise mais detalhada de alguns modelos de mobilidade realísticos e em virtude disso, foi necessário um estudo aprofundado sobre os modelos de mobilidades propostos, e a análise será realizada por meio do levantamento das informações adquiridas nas simulações, onde é possível simular o movimento dos nós na rede, e através das métricas e parâmetros de comparação pré-definidos, o trabalho visa demonstrar um comparativo dos resultados obtidos entre os modelos de mobilidades realísticos estudados e auxiliar no conhecimento em redes *ad hoc* propiciando futuros projetos na área.

Este trabalho está organizado da seguinte forma: Na seção 2 são apresentados conceitos de redes sem fio, sua classificação e características. Na seção 3, é mostrado o protocolo de roteamento utilizado em redes *ad hoc* e a definição de qual será utilizado nas simulações. A seção 4 abrange um estudo sobre três modelos de mobilidade realísticos. Na seção 5 são descritos os trabalhos relacionados e uma breve explicação sobre cada um. Na seção 6 são apresentadas as ferramentas geradoras de cenário com suas definições. Na seção 7 é apresentado o NS-2 (*Network Simulator*). Além do simulador, também são citados alguns detalhes sobre os seus recursos, como o visualizador de imagens NAM e o arquivo *trace* que, através dele, consegue-se obter todas as informações referente às simulações geradas. Na seção 8, 9 e 10 é mostrado a metodologia de pesquisa e desenvolvimento, as limitações encontradas durante o desenvolvimento, além dos

resultados obtidos. Finalmente, em anexo, estão relacionados alguns exemplos de simulações geradas no NS-2.

2 REDES SEM FIO

As redes sem fio são uma das tecnologias de comunicação que vem crescendo e ganhando mais espaço em relação a sua utilização. O aumento da demanda se da devido ao surgimento de equipamentos como *notebooks*, telefones celulares, *palmtops*, entre outros (TANEMBAUM, 1997).

Como algumas das características deste tipo de rede, destacam-se o uso do ar como meio físico ao invés da estrutura de cabeamento tradicional, além da comodidade de conectividade em qualquer local (que possua suporte), proporcionado pelo uso de canais de rádio freqüência (rádio difusão) ou infravermelho na transmissão dos dados (PEREIRA, 2004).

Existem dois tipos de redes sem fio, a primeira é redes infra-estruturadas que segundo Pinheiro [Pinheiro 2005], os nodos se comunicam através de dispositivos com suporte a mobilidade, conhecidos como *Access Point*. Toda e qualquer comunicação realizada entre os dispositivos é através desse pondo, ou seja, o dispositivo móvel se comunica com um ponto de acesso e este se comunica com o outro dispositivo móvel, como mostra a Figura 1.

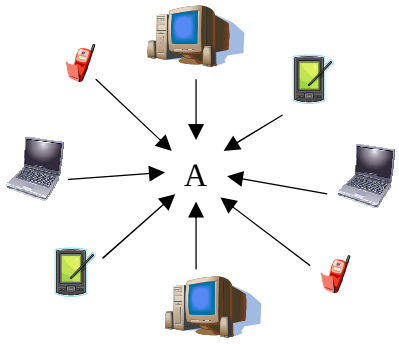

Figura 1 – Modo de Conexão
Fonte: Autoria Própria, 2008.

O segundo tipo de rede sem fio são as redes *ad hoc*, onde a rede não depende de um *Access Point* para realizar a comunicação entre os nodos, pois os mesmos trabalham como estações de trabalho e roteadores, distribuindo rotas para a rede e possibilitando a comunicação direta entre os mesmos (STALLINGS, 2005). Essas redes são formadas por múltiplos nodos que se comunicam via uma troca de mensagens ponto a ponto, como apresenta a Figura 2.

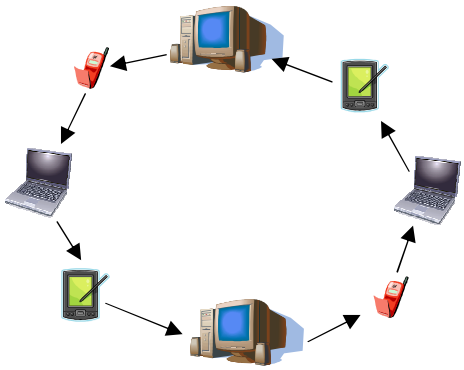

Figura 2 – Modo de Conexão *Ad Hoc*.
Fonte: Autoria Própria, 2008.

2.1 Redes Ad Hoc

Devido ao trabalho se referir a uma rede *ad hoc*, nesta seção será feita uma descrição mais detalhada desse tipo de rede.

As redes *ad hoc* são redes que não necessitam de ponto de acesso para se comunicar, pois os nodos conseguem trocar informações entre si diretamente ou através de nodos intermediários, pelo fato de que todos os nodos podem ser roteadores ou dispositivos simples na rede ao mesmo tempo (PINHEIRO, 2005).

Devido às características citadas anteriormente, esses tipos de redes são ideais para serem utilizadas em locais onde não existe a possibilidade de instalação de *Access point* (rede infra-estruturada), ou até mesmo uma rede fixa (cabeada), por exemplo, pessoas em um Campus, em conferências ou em *Shoppings* tendo que trocar informações.

Algumas vantagens encontradas na utilização desse tipo de rede em relação as redes infra-estruturadas são (CAMPOS, 2003):

- Rápida instalação, devido à rede não necessitar de uma infra-estrutura previamente montada, e a mesma pode ser estabelecida em qualquer local.

- Conectividade, pois os nodos conseguem se comunicar entre si (diretamente) desde que os mesmos estejam dentro da área de cobertura das ondas de rádio, enquanto que em uma rede infra-estruturada mesmo que os dispositivos estejam perto um do outro não existe a comunicação entre os mesmo, pois necessitam de um ponto de acesso para se comunicar.
- Tolerância a Falhas, pois a rede é dinamicamente atualizada caso um dos dispositivos móveis se desconectar da rede ou falhar.
- Topologia dinâmica, devido à locomoção do dispositivo móvel não ser de forma aleatória, a topologia se altera com bastante freqüência, sendo assim a mesma se torna dinâmica.
- Mobilidade, que é uma das vantagens primordiais com relação a redes fixas.

Algumas desvantagens encontradas são (CAMPOS, 2003):

- Localização, pois não existe a informação da localização física do nodo na rede como é possível obter nas redes fixas e infra-estruturadas.
- Pequena banda passante, devido aos canais de comunicação, que possuem menor banda passante de enlace em relação à rede fixas.
- Roteamento, devido a grande mobilidade dos nodos na rede, pois os dispositivos locomovem-se de forma aleatória, criando rotas válidas dentro de rotas inválidas, sendo necessário descobrir as rotas válidas novamente. E isso causa um grande impacto no desempenho da rede.

Vários protocolos foram desenvolvidos para as redes *ad hoc* para tentar resolver o problema do roteamento entre os nodos. Na próxima seção será abordado o protocolo que será utilizado durante a simulação para análise dos modelos de mobilidade realísticos.

3 PROTOCOLO DE ROTEAMENTO

Existem vários protocolos para redes *ad hoc*, desenvolvidos visando resolver os problemas de definição de rotas. Os protocolos são necessários para que os dispositivos consigam acessar uns aos outros.

Os protocolos podem ser divididos em dois tipos:

Reativos: onde as rotas são estabelecidas somente quando um dispositivo solicita a comunicação com o outro dispositivo, desta forma ocorre um baixo consumo de energia e de tráfego na rede. Quando um nodo quer se comunicar na rede é iniciado um processo de descoberta de rota, após a rota ser descoberta a mesma fica sendo mantida até que a comunicação não seja mais desejada ou o dispositivo tenha saído da rede (CAMPOS, 2003).

Pró-ativos: onde as rotas são estabelecidas no instante em que o nodo é ativado na rede, sendo que cada nodo deve manter uma tabela de roteamento e que a cada mudança na rede tal tabela seja atualizada e replicada aos nodos da rede para manter a integridade da informação. Este tipo de protocolo causa um grande consumo de energia e de tráfego na rede devido à atualização das tabelas (CAMPOS, 2003).

Dada a proposta do trabalho de comparar os modelos de mobilidade realísticos, o protocolo escolhido para realizar as simulações foi o AODV por ser reativo, ser um dos mais utilizados em pesquisas voltadas para a área de redes, além deste já estar disponível no simulador NS-2 que será utilizado no trabalho.

3.1 PROTOCOLO AODV

O protocolo de roteamento AODV (*Ad hoc On-Demand Distance Vector*) foi desenvolvido pelo IETF conforme a RFC 3561 (PERKINS; BELDING-ROYER, 2005) para atender desde redes pequenas até redes com muitos nodos móveis.

As rotas são estabelecidas somente quando um dispositivo solicita a comunicação com o outro dispositivo. Quando um nodo quer se comunicar na rede é iniciado um processo de descoberta de rota que funciona da seguinte maneira: o nó origem envia um pacote RREQ[1] (*Route Request*), via broadcast, para os nós vizinhos, e esses repassam a informação recebida para os seus outros vizinhos, e assim sucessivamente, até que o nó destino seja alcançado ou algum nó intermediário tenha a rota desejada (SILVA; OLIVEIRA, 2005).

Durante o processo de reenvio dos pacotes RREQ, os nós intermediários registram temporariamente em suas tabelas a origem destes pacotes. Desta forma, consegue-se obter um caminho reverso. Quando o RREQ alcança o próprio destino ou um nó intermediário que contenha a rota desejada, esses respondem enviando um pacote unicast RREP[2] (*Route Reply*) de volta para a origem da requisição (SILVA; OLIVEIRA, 2005). Este pacote é enviado pelo caminho reverso montado pelos nós intermediários no instante em que a mensagem RREQ estava sendo encaminhada. Durante o encaminhamento do pacote RREP, todos os nós intermediários incrementam o campo que corresponde à quantidade de saltos necessários para alcançar o nó destino.

A Figura 3 mostra o funcionamento do protocolo AODV onde o nodo móvel “1” requer comunicação com o nodo “8”.

---

1 RREQ – Mensagem de requisição de rota enviada pelo protocolo de roteamento AODV.

2 RREP – Mensagem de resposta a uma requisição de rota RREQ.

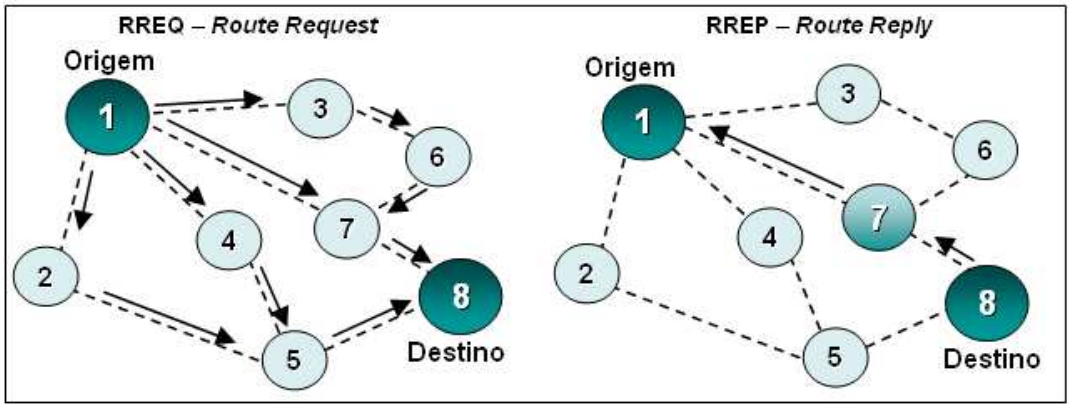

Figura 3 – Protocolo AODV.
Fonte: Silva e Oliveira, 2005.

Cada entrada de rota possui um temporizador que causa o seu apagamento, caso essa rota não seja usada em um espaço de tempo específico (SILVA; OLIVEIRA, 2005).

Com este protocolo, os nós móveis podem utilizar mensagens de *hello,* que são broadcasts locais que um nó envia aos seus vizinhos a fim de informar que ainda está ativo. Desta maneira, pode-se manter a conectividade.

Devido às redes móveis possuírem limitação de energia e grande mobilidade dos nós, pode haver quebra de conexão. Quando isso ocorre, mensagens do tipo RERR[3] (*Route Error*) são enviadas a todos os nós envolvidos contendo a informação da não existência de rotas disponíveis através deste nó específico (SILVA; OLIVEIRA, 2005).

[3] RRER – Mensagem de erro enviada pelo protocolo de roteamento AODV.

4 MODELOS DE MOBILIDADE

Os modelos de mobilidade podem ser divididos em sintéticos e realísticos. Conforme (PEREIRA, 2004), os modelos sintéticos podem não refletir corretamente o verdadeiro desempenho dos protocolos, pois os mesmos não utilizam cenários realistas para simulação e são baseados em processos aleatórios. Pessoas em um Campus, em conferências ou em Shoppings geralmente não se movem em direções aleatórias, mas normalmente selecionam um destino específico e seguem um caminho bem definido para chegar a esse destino. Como exemplo de modelos sintéticos tem-se: *Random Walk*, *Random Waypoint*, *Random Direction*, entre outros (CAMP; BOLENG; DAVIES, 2002), (THOMA; NUNES; 2006).

Na maioria dos casos, a seleção do caminho é influenciada por vias, calçadas e obstáculos, como por exemplo, em um Campus de uma universidade onde indivíduos geralmente utilizam caminhos que interconectam os prédios. Enquanto certos indivíduos podem desviar-se desses caminhos (por gramados, por exemplo), a maioria das pessoas se move ao longo dos caminhos existentes. Adicionalmente, os destinos normalmente não são aleatórios, mas são prédios, restaurantes, estacionamentos e outras localizações específicas dentro do campus.

Ao contrário dos modelos sintéticos, os modelos realísticos tentam simular ambientes próximos da realidade, ou seja, inserindo nos cenários obstáculos que, por exemplo, simulem uma cidade, como prédios, ruas, casas, vegetação, etc. Alguns padrões realísticos existentes são: UDEL (*Udel Model*) (KIM; SRIDHARA; BOHACEK, 2006), (KIM; BOHACEK, 2005), (SRIDHARA; BOHACEK, 2007), (SRIDHARA; KIM; BOHACEK, 2005), (BOHACEK et al. 2004), OM (*Obstacle Mobility Model*) (JARDOSH; et al. 2005), (JARDOSH; et al. 2006) e UPF (*Urban Pedestrian Flow*) (MAEDA et al., 2005), entre outros, como: (CAPKA; BOUTABA,

2005), (HELBING; FARKAS; VICSEK, 2000), (JUNGKEUN et al., 2006), (KYRIAKAKOS et al., 2002), (LEGENDRE et al., 2006) e (TUGCU, 2001).

Portanto, pesquisadores tem se esforçado para fornecer padrões de mobilidade para redes móveis considerados “realísticos” pelos mesmos, sendo que cada um modela de uma forma diferente a realidade. Como não existe na literatura um *survey* sobre os padrões de mobilidade realísticos, Nas próximas seções, são apresentados os três modelos de mobilidade realísticos que possuem suporte ao NS-2, e serão estudados neste trabalho.

4.1 Obstacle Mobility Model

O *Obstacle Mobility Model* foi proposto por (JARDOSH et al., 2006) e foi projetado para modelar o movimento dos nodos dentro de uma topologia real do mundo, podendo ser definido o tamanho, a posição e as formas dos obstáculos que serão inseridos no cenário a ser simulado. Esses obstáculos são incluídos no modelo para tornar mais realístico o movimento dos nodos móveis. Esses obstáculos podem modelar cenários como prédios, cidades, rodovias, etc.

Neste padrão de mobilidade, os nodos são distribuídos aleatoriamente por caminhos e movem na direção de um destino. Os destinos são selecionados aleatoriamente, isto é, a probabilidade com a qual o nodo seleciona um destino é independente da sua distância para o destino, e o caminho que ele irá seguir é selecionado de um conjunto de caminhos definidos por diagramas de Voronoi (OVERMARS, 2000).

O Diagrama de Voronoi é formado com base na seguinte regra: dado um conjunto de locais (pontos) no plano, associa-se a cada local (ponto) a região do plano que esteja mais próximo deste ponto. Com isso, é possível responder com eficiência a uma grande variedade de perguntas a respeito de proximidade, como por exemplo, qual local está mais próximo de um ponto dado, qual a maior região desocupada, qual é o vizinho mais próximo de um ponto, entre outras. A Figura 4 mostra um exemplo de Diagrama de Voronoi.

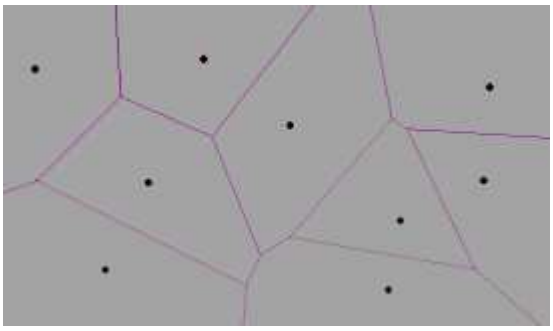

Figura 4 – Diagrama de Voronoi.
Fonte: http://pt.wikipedia.org/wiki/Diagrama_de_Voronoi

Este diagrama é pré-definido e estabelece caminhos de um lado para outro. Desta forma, a rota que possui o caminho mais curto entre cada nodo e seu destino é calculada e é usada para determinar o caminho que os nodos usarão. Quando o nodo alcança o destino, ele faz uma pausa, escolhe aleatoriamente outro destino, calcula a rota e retorna o movimento. A velocidade e o tempo de pausa são escolhidos ao acaso, a partir de uma distribuição do usuário.

Além disso, nesse padrão de movimentação, quando o nodo transmite, os obstáculos podem bloquear a propagação da transmissão naquela área, então a transmissão pode ser bloqueada completamente pelo obstáculo. Para modelar o bloqueio da propagação do sinal foram utilizados cones como ilustrado na figura 5, onde, por exemplo, o sinal do nodo i é obstruído na região definida pelos pontos i1, i2, i3 e i4, e o sinal do nodo j foi obstruído na região onde estão os pontos j1, j2, j3, e j4. Em cenários reais, edifícios e outros obstáculos são compostos de material diferentes e de espessuras variadas, sendo assim a propagação do sinal deve prever estes fatores.

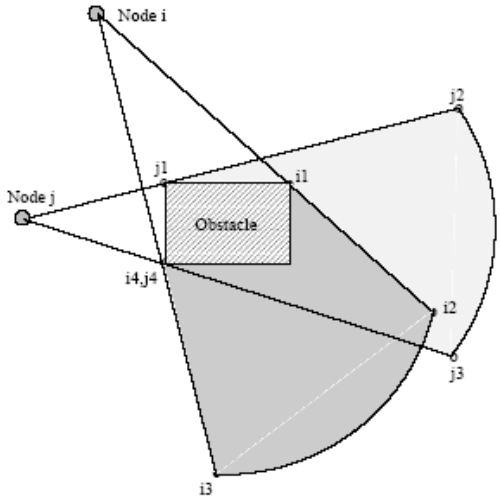

Figura 5 – Cones de Obstrução Nodos i e j.
Fonte: (OVERMARS, 2000)

Ao analisar essa forma de modelar a movimentação, pode-se observar que características aleatórias estão sendo utilizadas, como por exemplo, escolha do tempo de pausa e velocidade de movimentação, as quais são obtidas sem qualquer medida real. Contudo, para criar cenários de movimentação mais reais, esse modelo é estendido em (JARDOSH et al., 2005). As características adicionais incluídas no padrão de movimentação foram: seleção do destino a partir de uma distribuição exponencial, movimento para pontos de atração (nodos podem se mover em direção a certos pontos de interesse) e diminuição realística da intensidade do sinal. Contudo, neste novo padrão de movimentação, a escolha do caminho através da rota mais curta continua existindo, bem como outras características que não correspondem à realidade não foram alteradas.

4.2 Urban Pedestrian Flow

O objetivo do *Urban Pedestrian Flow* (MAEDA et al., 2005), é reproduzir o fluxo real de pedestres, classificados em grupos de interesse de acordo com o seu comportamento. Esse fluxo indica como os pedestres se movem através de pontos geográficos, isto é, indica a rota por onde eles devem seguir. De acordo com (MAEDA et al., 2005), esse fluxo pode ser identificado através de observações nas ruas feitas de pontos de observações fixos, usando câmeras web, voluntários, entre

outros. Polígonos representam obstáculos, como por exemplo, prédios, entradas de prédios, estações, terminais, *shopping centers*, entre outros.

Os pontos geográficos podem ser considerados específicos, onde os pedestres podem aparecer ou desaparecer. Assim os pontos específicos são considerados como pontos de origem ou de destino do fluxo de pedestres. Estes pontos incluem, por exemplo, entradas dos edifícios, das estações, dos terminais, dos *shoppings*, etc.

Devido às pessoas não se moverem de maneira aleatória, (MAEDA et al., 2005) classificou o comportamento das pessoas em grupos e, posteriormente, realizou alguns testes nesses grupos para gerar a rota por onde as pessoas passam, que é chamado de fluxo no modelo. Entretanto, como em alguns casos foi difícil estimar um comportamento padrão, o modelo também utiliza o algoritmo *Bellman-Ford* onde o cálculo da rota é realizado utilizando o trajeto mais curto entre os pontos origem e destino, sendo possível ainda à concatenação de dois ou mais caminhos mais curtos.

Para realizar as simulações do modelo proposto, (MAEDA et al., 2005) define um simulador chamado MobiREAL. Neste simulador dois cenários foram analisados: um deles foca no desempenho do protocolo DSR (*Dynamic Source Routing*), e outro analisa o desempenho de uma aplicação de disseminação de informações. Para verificar se a simulação é semelhante com a realidade, (MAEDA et al., 2005) compara a densidade de pedestres com a densidade obtida de situações reais. A densidade de pedestres foi obtida de fotos nas ruas usando câmeras digitais, que mostram estaticamente o comportamento dos pedestres. Além disso, os pontos de destino foram selecionados nas extremidades da área, foi calculado o caminho mais curto até os pontos e a velocidade foi escolhida aleatoriamente. Portanto, sua maior contribuição é um método que reproduz, em um cenário de simulação, o fluxo de pedestres a partir dos dados observados.

4.3 Udel Model

O *Udel Model* foi descrito em (KIM; SRIDHARA; BOHACEK. 2006), (KIM; BOHACEK, 2005), (SRIDHARA; BOHACEK, 2007) e (SRIDHARA; KIM; BOHACEK, 2005) para modelar redes *mesh* em ambientes urbanos. Os dados para modelar tais ambientes foram obtidos de mapas dimensionais de edifícios, fornecidos pelo GIS

(*Geographic Information System*)[4], além de indicações de estradas, rios, lagos, escala de endereços, limites de velocidades, etc, fornecidos pelo TIGER (*Topologically Integrated Geographic Encoding and Referencing*) (U.C, 2006) do departamento de Census dos Estados Unidos. O último é freqüentemente utilizado para simulações para redes *ad hoc* em veículos.

O modelo Udel possui uma hierarquia de três níveis: modelo de atividade, modelo de tarefa e modelo de agente. O modelo de atividade determina o tempo em que uma pessoa inicia e termina várias atividades, tais como trabalhar, comer, comprar ou receber serviços profissionais. Os dados usados para gerar o modelo de atividade foram obtidos de um estudo realizado pelo *US Bureau of Labor Statistics* (BLS). O modelo de tarefa está preocupado com as tarefas dentro de uma atividade, como, por exemplo, trabalhar em um escritório ou fazer reuniões com outras pessoas. A base desse modelo foi feita de levantamentos que incluem reuniões com pessoas e atividades realizadas pelos trabalhadores durante o seu trabalho.

Finalmente, o modelo de agente define como o nodo se move para o destino desejado. Esse modelo é baseado em pesquisas de planejamento urbano que incluem resultados da mobilidade dos pedestres. Nesse modelo, destinos aleatórios são escolhidos e nodos escolhem o caminho mais curto durante seu movimento. A velocidade é definida a partir de um intervalo, e os nodos podem ter velocidades diferentes. Além disso, como esse modelo de mobilidade usa modelos gráficos da área urbana, é necessária a descrição de localizações de prédios, o tamanho dos prédios (incluindo número de andares), a orientação dos prédios, ruas e calçadas. A descrição detalhada desse modelo está apresentada em (BOHACEK et al., 2004).

Segundo (SRIDHARA; KIM; BOHACEK, 2005), esse modelo foi utilizado para avaliar o desempenho de redes *mesh* em ambientes urbanos. Nessa avaliação as seguintes métricas foram consideradas: cobertura, taxa de bit máxima e desempenho de aplicações, tal como transferência de arquivos, voz sobre IP e *streaming* de música. Esse estudo utilizou o modelo de mobilidade para pedestres durante um dia de trabalho.

---

[4] Disponível em: http://www.gis.com

5 TRABALHOS ESTUDADOS

Neste trabalho serão utilizados modelos de mobilidade realísticos. Artigos sobre este tema têm sido escrito com diversas propostas para evoluir estes modelos sob vários aspectos.

Entre os artigos estudados podemos citar:

Modelo de Mobilidade Realístico OM:

- "*Towards Realistic Mobility Models For Mobile Ad hoc Networks*", desenvolvido por Amit Jardosh, Elizabeth M. Belding-Royer, Kevin C. Almeroth e Subhash Suri, do departamento de ciência da Computação da Universidade da Califórnia em Santa Bárbara;

Onde é descrito o estudo do padrão de mobilidade realístico OM (JARDOSH et al., 2006) utilizando o protocolo de roteamento AODV e após é realizado uma comparação com o Modelo de Mobilidade sintético de Ponto de Mudança de Rota (*Random Waypoint Mobility Model*) (THOMA; NUNES, 2006).

Modelo de Mobilidade Realístico UDEL:

- "*Models and Methodologies for Simulating Mobile Ad-hoc Networks*", "*Realistic Simulation of Urban Mesh Networks – Part I: Urban Mobility*" e "*Realistic Propagation Simulation of Urban Mesh Networks*", escrito por Vinay Sridhara, Jonghyun Kim e Stephan Bohacek, do Departamento de Engenharia Electrotécnica e de Computadores da Universidade de Delaware.

Onde são descritas técnicas utilizadas para simulação de redes *ad hoc*, incluindo métodos para simular a realidade, propagação e mobilidade, e também descrevem a implementação do simulador UDEL, que é utilizado para experimentação do modelo. (KIM; SRIDHARA; BOHACEK, 2006), (KIM; BOHACEK, 2005), (SRIDHARA; BOHACEK, 2007), (SRIDHARA; KIM; BOHACEK, 2005).

Modelo de Mobilidade Realístico UPF:

- "*Getting Urban Pedestrian Flow from Simple Observation: Realistic Mobility Generation in Wireless Network Simulation*", desenvolvido por Kumiko Maeda, Kazuki Sato, Kazuki Konishi, Akiko Yamasaki, Akira Uchiyama, Hirozumi Yamaguchi, Keiichi Yasumoto[†] e Teruo Higashino, graduados na Escola de Informação Ciência e Tecnologia, em Osaka no Japão.

Onde são descritos teste realizados com o simulador MobiREAL (MAEDA et al., 2005), e comparações realizadas entre o modelo realístico e o modelo de ponto de mudança de rota (*Random Waypoint Mobility Model*) (THOMA; NUNES, 2006).

Modelo de Mobilidade Sintéticos:

- "Avaliação de Desempenho de Redes Ad Hoc Utilizando Modelos de Mobilidade", desenvolvido por Virgínia Mattos Thoma e Cristiana Moreira Nunes, no Centro Universitário La Salle – Unilasalle, em Canoas no Brasil.

Onde são descritos testes realizados com o simulador NS-2, efetuando comparações entre os protocolos de roteamento AODV, DSR e DSDV, usando os Modelos de Mobilidade de Percurso Aleatório (Random Walk Mobility Model), Modelo de Mobilidade de Ponto de Mudança de Rota (Random Waypoint Mobility Model), Modelo de Mobilidade de Direção Aleatória (Random Direction Mobility Model) e a versão Probabilística do Modelo de Mobilidade de Percurso Aleatório (A Probabilistic Version of the Random Walk Mobility Model) (THOMA; NUNES, 2006).

Nos estudos realizados em todos os modelos citados (OM, UDEL e UPF) foi verificado que nenhum comparativo entre os mesmos foi realizado, ou seja, todos os modelos foram validados utilizando simuladores específicos (GlomoSim, UDEL ou QualNet e MobiREAL) e utilizando métricas específicas, fazendo com que esta seja a principal motivação deste trabalho, onde serão abordados os modelos utilizando o mesmo simulador NS-2, as mesmas métricas e o mesmo protocolo de roteamento, para que a comparação seja possível e válida.

Sendo assim não foi possível traçar uma relação entre os trabalhos estudados com este trabalho realizado, sendo que os mesmos somente servirão como fonte de estudo e conhecimento dos modelos.

6 FERRAMENTAS GERADORAS DE CENÁRIOS

Para realizar os experimentos será utilizada, para cada modelo de mobilidade realístico, uma ferramenta geradora de cenário, que fornece através de um arquivo trace a mobilidade dos nodos dentro do cenário estipulado. Abaixo segue uma descrição de cada ferramenta a ser utilizada

- *Plugin* do Obstacle-Model: é um *plugin* desenvolvido pela universidade da Califórnia para geração dos *traces* de mobilidade do modelo OM
- UdelMobilitySim: é o simulador que será utilizado para gerar os *traces* de mobilidade do modelo UDEL, neste simulador é necessário identificar a área urbana, sendo que a descrição é simplificada, sendo necessário somente informar a localização, os tamanhos e a orientação dos edifícios, estradas e calçadas. Possui uma ferramenta integrada chamada UdelMapBuilder que pode ser utilizada para geração do mapa da área urbana.
    - Algumas observações importantes sobre a ferramenta:
        - Para que a geração do movimento seja possível é necessário indicar de onde os nodos devem surgir, sendo que as alternativas são: Prédios residenciais, ou Estação de Trem (Metro), ou Automóveis, podendo ser dos três ao mesmo tempo.
        - Sempre deve ter na área simulada um prédio do tipo restaurante (*Office*/*Stores*)
        - Não podem existir no cenário curvas, somente retas.
        - Neste modelo de mobilidade a ferramenta geradora de mobilidade entende que nem todos os nodos estarão em movimento no cenário, pois os mesmos podem estar, por

exemplo, em um prédio comercial, onde as usuário podem estar sentados em suas mesas, e não em movimento, por tanto, alguns nodos permanecem sem movimento durante a geração do *trace* de mobilidade para o NS-2

  - Para utilizar o *trace* de mobilidade no NS-2 é necessário instalar um *patch* que contém dois arquivos: o Makefile que deve ser gravado no diretório “/ns-allinone-2.33/ns-2.33” substituindo o existente, e o Mobilenode.cc que deve ser incluído no diretório “/ns-allinone-2.33/ns-2.33/common” também substituindo o existente, e após a atualização desses arquivos o NS-2 deve ser re-compilado.

- MobiREAL: é o simulador que será utilizado para gerar os *traces* de mobilidade do modelo UPF, sendo que vinculamos um cenário criado na ferramenta *FlowMaker* e *Animator* no simulador para que a mobilidade do nodo seja gerada a partir do cenário criado.

7 SIMULAÇÃO

A simulação é a técnica de estudar o comportamento e reações de um determinado sistema através de modelos, que imitam na totalidade ou em parte as propriedades e comportamentos deste sistema em uma escala menor, permitindo sua manipulação e estudo detalhado (ERLANG, 2005).

A simulação permite analisar um sistema sem necessidade de interferir no mesmo. Qualquer mudança realizada em um cenário ocorre somente no meio computacional, não refletindo no mundo real. Através da simulação é possível utilizar vários cenários e configurações de redes, sendo que todas essas características são extremamente difíceis de realizar no mundo real através de experimentos. Por isso a simulação tornou-se uma ferramenta popular, para o desenvolvimento de protocolos e estudo das redes, como *ad hoc*.

Neste trabalho, foi utilizado o *Network Simulator* – NS-2 (NS, 2002), por ser bastante utilizado na área acadêmica, além de ser uma ferramenta voltada para a pesquisa na área de redes.

Após o termino das simulações, o NS-2 gera uma arquivo de saída (*trace*), que contém todas as informações referente à simulação realizada, e interpretando esses arquivos é possível analisar os resultados através das métricas estipuladas e gerar os gráficos para visualização e comparação dos resultados. Na próxima seção será descrito mais detalhadamente o NS-2.

7.1 Network Simulator

O simulador de redes NS-2 (*Network Simulator*) (COUTINHO, 2003) baseia-se em eventos discretos e é orientado a objetos. O objetivo do NS-2 é proporcionar um ambiente para o desenvolvimento de pesquisas em torno dos protocolos que

constituem a Internet (KAMIENSKI et al., 2002). Ao realizar uma simulação, o NS-2 produz dados detalhados, que podem ser direcionados para um arquivo. Esses dados podem ser utilizados para realizar diversas análises.

O NS-2 possui diversas plataformas computacionais, como: FreeBsd, Linux, SunOS, Solaris e Windows. A biblioteca de protocolos e mecanismos é bastante extensa, abrange implementação dos protocolos TCP (*Transmission Control Protocol*), UDP (*User Datagram Protocol*), IP (*Internet Protocol*), além de categorias de serviços WFQ (*Weight Fair Queueing*), protocolos de redes móveis, suporte a tecnologias em redes com e sem fio. Além disso, conforme (KAMIENSKI et al., 2002), oferece bibliotecas de funções para a geração de alguns tipos de tráfego como CBR (*Constant Bit Rate*), utilizado para voz, e VBR (*Variable Bit Rate*), utilizado para taxas variáveis.

Este simulador originou-se em 1989, a partir de uma variação do REAL *Network Simulator*, desenvolvido na universidade de Cornell com o objetivo de ser uma ferramenta para analise do controle de fluxo e congestionamento. Atualmente, o NS-2 é mantido pela DARPA (*Defense Advanced Research Projects Agency*) e pela NSF (*National Science Foundation*).

As principais vantagens de se utilizar o NS-2 são: é gratuito e seu código fonte é aberto, possibilitando ajustes para cada caso, possui uma grande quantidade de protocolos e tecnologias implementadas como *multicast*, redes *ad hoc*, MPLS (*Multiprotocol Label Switching*), redes de satélite, disponibilidade de ser um simulador padrão para a comunidade acadêmica e científica, e a possibilidade de utilizar ambientes controlados (KAMIENSKI et al., 2002).

O NS-2 utiliza duas linguagens de programação: C++ e OTCL (*Object-oriented Toll Command Language*). O C++ é utilizado para definição da estrutura básica como protocolos e agentes, e também por ser uma linguagem onde é possível compilar algoritmos com um grande número de dados. O OTCL é utilizado como *frontend*, onde são escritas as simulações, pois é fácil a manipulação e a troca de parâmetros com número de nodos no cenário, enlace, entre outros.

Todos os experimentos realizados utilizando o NS-2 é possível verificar os resultados através dos arquivos de saída chamados de *traces* (logs). Este arquivo é no formato texto e possui todos os dados referente à simulação realizada.

Entre várias ferramentas existentes para capturar os dados relevantes dos arquivos traces, existe uma chamada AWK, uma linguagem que utiliza padrões para leitura dos dados existentes no arquivo de saída.

O NS-2 também gera um arquivo trace para o programa animador de redes NAM (*Network Animator*), onde é possível verificar se forma gráfica o resultado da simulação.

7.2 Network Animator

O NAM (*Network Animator*) é uma ferramenta que mostra de forma animada à simulação realizada, com as informações de topologia, fluxo dos pacotes, e conteúdo. Para que a simulação possa ser visualizada no NAM é necessário que o NS-2 durante a simulação gere o arquivo de saída específico para a visualização da animação. A geração desse tipo de arquivo de saída é opcional, tendo que ser adicionado os comandos para geração no script Otcl (NAM, 2002).

A figura 6 apresenta a tela principal do NAM com 25 nós móveis se comunicando entre si.

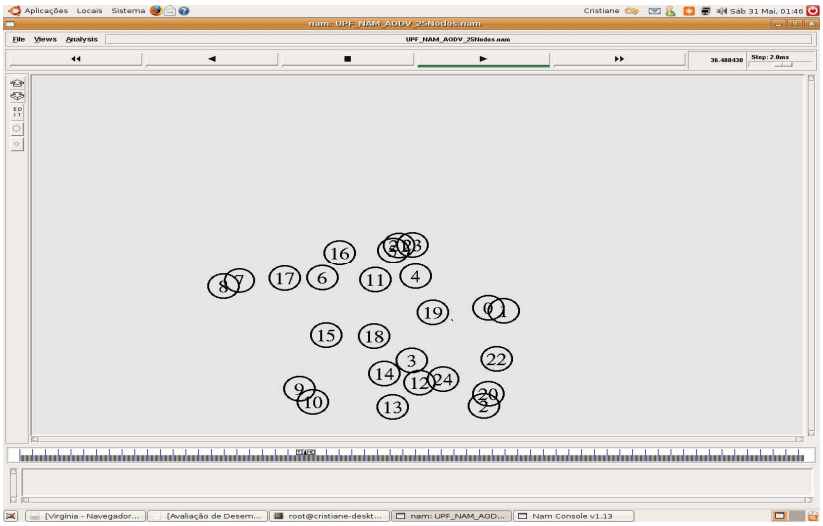

Figura 6 – Visualização da animação no NAM
Fonte: Autoria própria, 2008.

Na tela do NAM é possível verificar no canto superior direito um campo é registrado o tempo da simulação e ao lado um campo *step*, onde é possível avançar os quadros de animação. Possui botões como *play*, *fast foward*, *rewind*, *stop*, para manipulação da animação.

7.3 Arquivo Tace

Os arquivos *traces*, como citado anteriormente são os arquivos de saída do NS-2 que contem todos os dados referente à simulação realizada. É uma arquivo texto com um formato específico, onde pode ser verificado o transporte dos pacotes de um nodo origem para um nodo destino, mostrando toda a trajetória do pacote, sendo que se houverem nodos intermediários, e houver roteamento dos pacotes através desses, esta informação também estaram presentes nesse arquivo.

Os dados exibidos nesse arquivo dependem das informações que serão necessárias, a seguir uma descrição de alguns desses campos:

Exemplo de Arquivo Trace com campos limitados

1º	2º		3º		4º		5º		6º		7º		8º		9º		10º		11º		12º	
s	-t	13.069917510	-Hs	4	-Hd	-2	-Nx	161.19	-Ny	170.52	-N1	AGT	-Is	4.0	-Id	5.0	-It	tcp	-Il	1040	-Ii	1244
r	-t	13.069917510	-Hs	4	-Hd	-2	-Nx	161.19	-Ny	170.52	-N1	RTR	-Is	4.0	-Id	5.0	-It	tcp	-Il	1040	-Ii	1244
s	-t	13.069917510	-Hs	4	-Hd	5	-Nx	161.19	-Ny	170.52	-N1	RTR	-Is	4.0	-Id	5.0	-It	tcp	-Il	1040	-Ii	1244

Figura 7 – Campos arquivo *Trace*
Fonte: Autoria própria, 2008.

1º Campo: pode possuir os valores s, r, d, f, sendo que o valor “s” indica que o pacote foi enviado, o “r” indica que o mesmo foi recebido, o “d” indica que foi descartado e o “f” o pacote foi encaminhado;

2º Campo: valor “-t” (tempo) que indica o instante em segundos em que ocorreu o evento;

3º Campo: valor “-Hs” (Hop origem) que indica o número do nó que esta enviando o pacote;

4º Campo: valor “-Hd” (Hop destino) que indica o número do nó para onde esta sendo enviado o pacote. Podendo ser encaminhamento, sendo assim esse valor por não ser o nó destino, mas sim a rota;

5º Campo: valor “-Nx” (coordenada x) que indica a posição do nó em um instante de tempo;

6º Campo: valor “-Ny” (coordenada y) que indica a posição do nó em um instante de tempo;

7º Campo: valor “-N1” (nível de *trace*) que indica a camada onde se encontra o pacote podendo ser AGT, RTR e MAC;

8º Campo: valor “-Is” (endereço origem) que indica o endereço origem a qual enviou o pacote;

9º Campo: valor “-Id” (endereço destino) que indica o destino que recebera o pacote;

10º Campo: valor “-It” (tipo de pacote) que indica o tipo de pacote que esta sendo enviado ao nó;

11º Campo: valor “-I1” (tamanho do pacote) que indica o tamanho do pacote enviado;

12º Campo: valor “-Ii” (identificador do pacote) indica o número de identificação do pacote.

Na figura 7 é exibido um exemplo de um tráfego de pacotes, sendo que o pacote **“1244”**, identificado pelo campo **“-Ii”**, que é do tipo **TCP**, conforme o campo **“It”**. E o mesmo foi enviado do nó **“4”** para o nó destino **“5”**, como mostra nos campos **“Is”** e **“Id”** respectivamente. O campo **“I1”** mostra o tamanho do pacote enviado, que nesse exemplo é de **“1040”** *bytes*.

Devido a esses arquivos serem geralmente extensos, para realizar a análise dos mesmos e obter os dado necessários para gerar os gráficos de comparação entre as métrica, é necessário utilizar uma ferramenta para interpretar esse arquivos. Uma das formas utilizadas para interpretação desses arquivos são scripts desenvolvidos na linguagem de programação AWK (GAWK, 2004).

8 METODOLOGIA

Como já mencionado anteriormente, este trabalho foi desenvolvido para realizar uma análise comparativa entre alguns padrões de mobilidade realísticos para redes *ad hoc* estudados.

Para alcançar o objetivo desse trabalho foi utilizado o sistema operacional Linux e o simulador de redes NS-2 (*Network Simulator*). Utilizando este ambiente computacional, foram simulados os modelos de mobilidade com o protocolo de roteamento, ambos já explicados nos capítulos 3 e 4, respectivamente. O protocolo AODV foi escolhido por já estar implementado no NS-2 e em algumas bibliografias é citado que o mesmo possui um alto suporte a mobilidade. [Monte e Bernardo 2005]

No caso dos modelos de mobilidade realísticos, foi preciso utilizar as ferramentas “geradoras de cenário” (movimento), definidas no capítulo 6. Estas ferramentas geram arquivos traces contendo a posição inicial do nodo e toda a movimentação dos mesmos dentro do padrão de mobilidade realístico selecionado.

Um dos fatores mais importantes que devem se considerados quando se está definindo os cenários, diz respeito à quantidade de nodos que irão compô-lo. Isso por que em redes *ad hoc*, a quantidade de usuários reflete diretamente em seus parâmetros de desempenho.

Foram gerados cinco cenários para cada modelo de mobilidade, contendo 25, 160, 182, 425 e 564 nodos em movimento baseado no ambiente modelado sobre a figura 8, sendo que a partir do mesmo obteve-se os arquivos traces de mobilidade dos modelos UPF e UDEL.

A quantidade de nodos foi definida baseado na ferramenta MobiREAL, utilizada para gerar os cenários do modelo UPF, que conforme sua topologia não permite informar o número de nodos no cenário, pois a ferramenta inclui os nodos de

acordo com a densidade da área onde os nodos deveram se movimentar, conforme descrito em (Konishi, et. al., 2005).

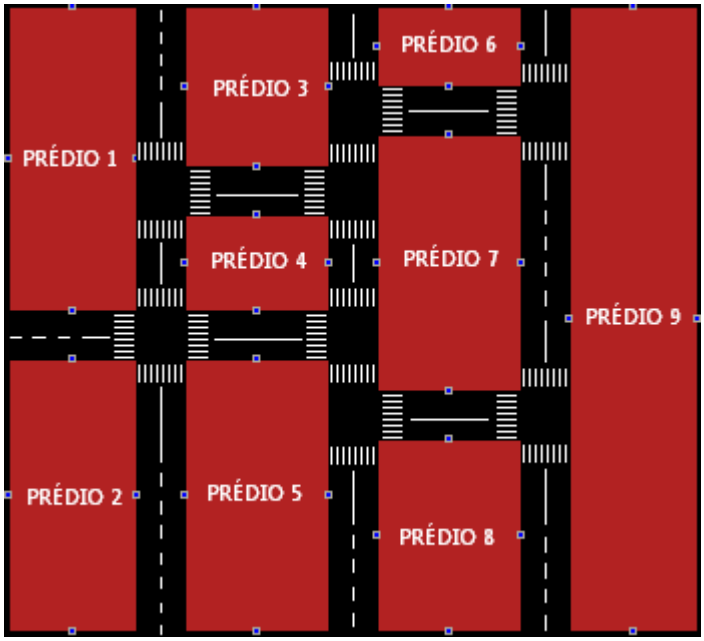

Figura 8 – Cenário Modelado
Fonte: Autoria própria, 2008.

A partir desses arquivos de movimento foram gerados os *scripts* TCL utilizando os parâmetros exibidos na tabela 1 [NS,2006], para executar a simulação no NS-2.

Tabela 1: Parâmetros de Configuração dos *scripts* TCL

Parâmetro	**Definição**	**Valor Atribuído**
Channel	Tipo de meio Físico	*Channel/WirelessChannel*
Propagation	Modelo de propagação	*Propagation/TwoRayGround*
phyType	Tipo de Interface	*Phy/WirelessPhy*
macType	Tipo de camada MAC	*Mac/802_11*
ifqType	Tipo de gerenciamento de fila empregado na interface	*Queue/DropTail/PriQueue*
llType	Abstração da camada *link*	LL
antType	Define o tipo de antena	*Antenna/OmniAntenna*
ifqLen	Tamanho máximo de pacotes na fila da interface	50
adhocRouting	Protocolo de roteamento empregado	AODV

Fonte: Autoria própria, 2008.

A quantidade de tráfego nas simulações foi determinada sem considerar estudos específicos sobre o tema. Foi definido que o número máximo de conexões na simulação seriam de acordo com a tabela 2.

Para gerar as conexões foi utilizado o script cbrgen.tcl, distribuído juntamente com o NS. Ele possui a capacidade de gerar tráfego aleatório de conexões TCP e CBR (*Constant Bitrate*) entre os nodos. Uma série de parâmetros podem ser ajustados para a geração do tráfego, como o tipo de conexão, o número de nodos, o número de conexões desejadas e a taxa de transferência, no caso do tráfego CBR. Um exemplo de arquivo de tráfego pode ser encontrado no Apêndice B.

Comando utilizado para geração do tráfego:

Sintaxe:

```
./ns /ns-allinone-2.33/ns-2.33/indep-utils/cmu-scen-gen/cbrgen.tcl –type <tipo da conexão> -nn <nº de nodos> -seed 0 –mc <nº de conexões> > <Nome do arquivo a ser salvo>
```

Comando executados para 25 nodos:

```
./ns /ns-allinone-2.33/ns-2.33/indep-utils/cmu-scen-gen/cbrgen.tcl –type tcp -nn 25 -seed 0 –mc 12 > traffic-tcp-25Nodos.txt
```

Foram atribuídos percentuais diferente dependendo do número de nodos, devido à quantidade de trafego gerado durante a simulação e na presunção de que nem todos os membros de uma rede permanecem utilizando-a durante 100% do tempo.

Devido aos *tracefiles* do NS serem extremamente grandes, o tempo máximo de simulação ficou ajustado para 120 segundos. Nos cenários com mais de 200 nodos, este tempo precisou ser reduzido para 100 segundos, uma vez que estava extrapolando o limite de 2GB por arquivo. A amostragem foi de 15 execuções devido ao tempo que leva cada simulação, chegando próximo a 2 horas de execução nos cenários com 425 e 565 nodos. A tabela 2 apresenta os dez cenários simulados.

Todos os cenários das simulações foram realizadas em uma área de 500m x 500m, com os nodos móveis inicialmente em posições aleatórias, e todos os nodos se movimentam simultaneamente conforme o padrão de mobilidade definido.

Tabela 2: Simulações no NS

Cenário	Mobilidade	Nodos	Protocolo	Conexões % por Nº de nodos	Tempo	Aplicação
1	UPF	25	AODV	50%	120s	FTP
2	UDEL	25	AODV	50%	120s	FTP
3	UPF	160	AODV	50%	120s	FTP
4	UDEL	160	AODV	50%	120s	FTP
5	UPF	182	AODV	50%	120s	FTP
6	UDEL	182	AODV	50%	120s	FTP
7	UPF	425	AODV	25%	100s	FTP
8	UDEL	425	AODV	25%	100s	FTP
8	UPF	564	AODV	25%	100s	FTP
9	UDEL	564	AODV	25%	100s	FTP

Fonte: Autoria própria, 2008.

8.1 Ambiente Utilizado nas Simulações

Em todas as simulações foi utilizado a mesma máquina com a seguinte configuração de hardware e software:

Hardware:

- Pentium 4
- CPU 3.00Ghz
- 512 Memória

Software:

- Para a criação dos cenários :
  - Sistema Operacional *Windows* XP
  - UDelMapBuilder[5] (cenários do UDEL)
  - *Animator*[6] e *FlowMaker*[7] (cenários do UPF)
- Para geração dos *traces* de mobilidade :
  - UDelProcessMap[8] (processamento do mapa do cenários UDEL)
  - UDelMobilitySim[9] (geração do *trace* de mobilidade do UDEL para NS)

[5] http://udelmodels.eecis.udel.edu/mapbuilder1.php
[6] http://www.mobireal.net/
[7] http://www.mobireal.net/
[8] http://udelmodels.eecis.udel.edu/download.php
[9] http://udelmodels.eecis.udel.edu/mobility1.php

  - Behavior[10] e Mobinet[11] (geração do *trace* de mobilidade do UPF para NS).
- Para realizar as simulações:
  - Sistema Operacional Linux (Distribuição Ubuntu 7.10)
  - Ns-allinone-2.33 (realizar as simulações)
  - Nam-1.13 (visualizar a simulação)
- Para Gráficos de comparação:
  - Microsoft Excel

8.2 Metodologia de Desenvolvimento

Após a geração dos *scripts* no NS, com a ferramenta NAM foi visualizado a simulação, observando a topologia, movimentação e comunicação entre os nodos. No fim da simulação o NS gerou um arquivo *trace* de saída, que foi analisado utilizando os scripts codificados na linguagem AWK. Através desses *scripts* foram capturadas as informações relevantes para análise das métricas.

A figura 9 mostra os passos realizados durante as simulações e obtenção dos resultados.

[10] http://www.mobireal.net/behavior.html
[11] http://www.mobireal.net/model.html

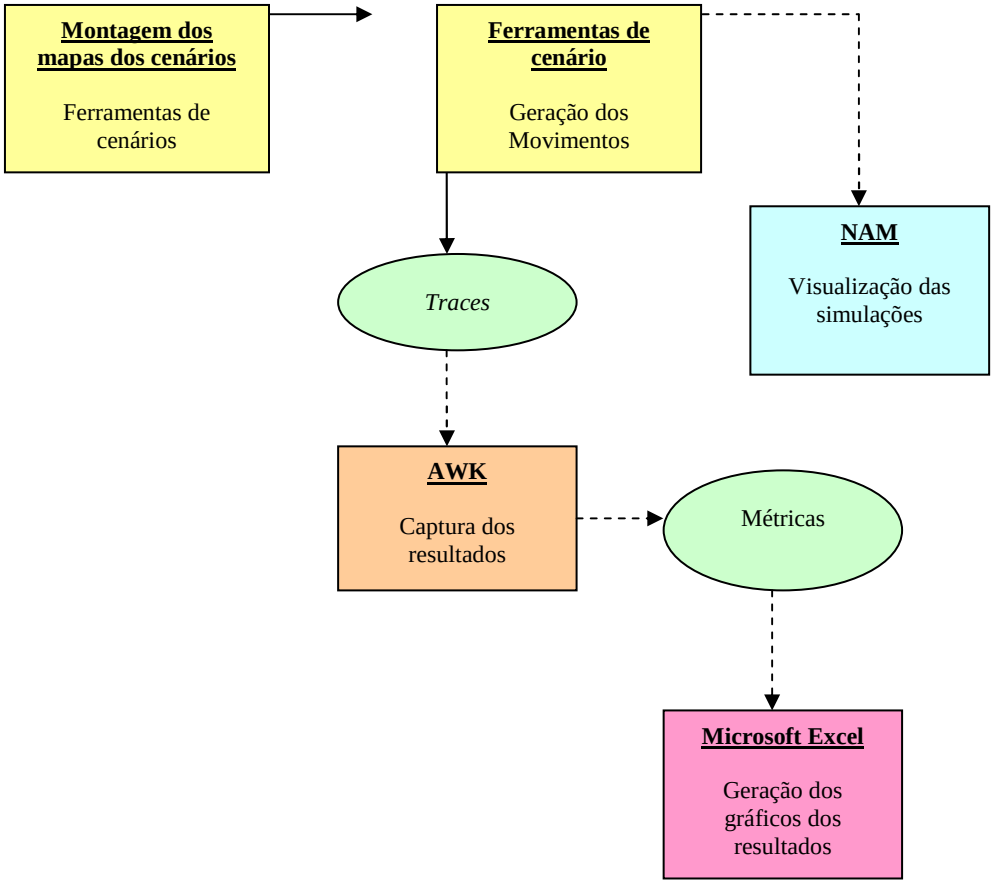

Figura 9 – Metodologia de Desenvolvimento
Fonte: Autoria própria, 2008.

Após a captura dos resultados com o AWK, foram gerados gráficos utilizando a ferramenta Microsoft Excel, para permitir a visualização e análise do comportamento dos modelos de mobilidade propostos.

Para realizar esta análise, foram utilizadas várias métricas de desempenho, para validação dos resultados, sendo que as métricas utilizadas estão descritas a seguir.

8.3 Métricas de Validação

As métricas são parâmetros muito importantes para que se tenha um controle do estado da rede durante uma determinada tarefa, para avaliar os modelo de mobilidade realísticos foram estabelecidas as seguintes métricas (JARDOSH; et al. 2005), (JARDOSH; et al. 2006), (KIM; SRIDHARA; BOHACEK, 2006), (KIM; BOHACEK, 2005), (MAEDA et al. 2005), (SRIDHARA; BOHACEK, 2007), (SRIDHARA; KIM; BOHACEK, 2005), (SRIDHARA; KIM; BOHACEK, 2005):

a) Número de pacotes enviados: Esta métrica avalia o desempenho da rede em relação à quantidade de pacotes enviados.

b) Número de pacotes perdidos: Quantidade de pacotes não entregues ao nodo de origem devido a erros durante a transmissão.

c) Número de pacotes recebidos: Esta métrica é utilizada para calculo da métrica vazão, é a quantidade de pacotes recebidos no nodo destino selecionados.

d) *Overhead* de roteamento: Pacotes de controle de roteamento, pacotes de descoberta e manutenção de rotas enviados pelos nodos origem ou pelos nodos intermediários. Estes pacotes incluem pacotes de *reply* e *request*.

e) Taxa de entrega: Razão entre pacotes recebidos nos nodos destinos e enviados nos nodos origem.

f) Taxa de perda: A taxas de perda de pacotes é calculada como a razão entre a quantidade de pacotes descartados pela quantidade de pacotes transmitidos. Quanto menor a perda de pacotes, melhor a qualidade da rede

g) Vazão: É utilizada para analisar a transmissão de dados na rede e determinar a quantidade de dados movida entre cada um dos nodos durante a simulação. Permite calcular a efetividade da rede, pois através dela é possível verificar a quantidade de bits recebidos por segundo com sucesso.

h) *Delay*: Tempo médio que os pacotes levam para ir de um ponto ao outro na rede. É calculada considerando o tempo em que o pacote chega no canal de comunicação através do nodo de origem e o recebimento do último bit do pacote no nodo destino. Quanto menor for a latência, menor será o tempo de resposta da rede.

i) *Jitter*: É a variação média no atraso de pacotes de dados causada por fatores como congestionamento da rede, pequenos deslocamentos no tempo previsto de recepção ou mudanças de rota.

9 LIMITAÇÕES

Duas principais limitações foram encontradas para execução desse trabalho, a primeira foi o *plugin* para o modelo de mobilidade OM, pois o mesmo é bastante antigo (de 2006) e foi desenvolvido para a versão NS-2.26, não sendo possível à utilização em versões mais recentes devido à passagem de parâmetros das funções, por exemplo, no *plugin* uma determinada função tem como parâmetro 8 variáveis e nas versões mais recentes do NS a mesma função utilizas como parâmetro 10 variáveis.

Durante a instalação da versão NS-2.26 ocorreram vários problemas, sendo que estes problemas foram solucionados conforme descrito no Apêndice A, porém ao tentar compilar o NS-2.26 após a atualização do *plugin*, ocorreram novos problemas. Sendo assim este modelo não foi analisado conforme a proposta inicial do trabalho, devido à instabilidade no NS e no *plugin*.

A segunda limitação foi à ferramenta geradora de cenário para o modelo BM, pois a ferramenta necessita de autorização para realizar o *download*, e esta autorização não foi concedida pela universidade mantenedora *da mesma*. Devido a essa limitação o modelo não foi utilizado e citado no trabalho.

10 SIMULAÇÕES E RESULTADOS

A partir da análise realizada nos traces gerados pelo NS, obteve-se os resultados os quais são apresentados neste capítulo. Abaixo, estão descritas as análises realizadas em cada cenário a partir das métricas descritas e de acordo com os modelos de mobilidade estudados.

Os resultados obtidos estão representados através de gráficos, sendo que os mesmo estão apresentados nas próximas seções deste capítulo, facilitando a visualização do comportamento dos modelos de mobilidade dentro dos cenários construídos.

Na próxima seção é descrito através das métricas o comportamento da rede utilizando os Modelos de Mobilidade aplicados nos cenários desenvolvidos, bem como o comportamento do protocolo de roteamento AODV utilizado durante os experimentos.

10.1 Pacotes Enviados

Foram realizados os experimentos utilizando 5 cenários para cada modelo de mobilidade. Resultando nas figuras 10, 11, 12, 13 e 14, onde é possível verificar a comparação realizada entre os modelos UPF e UDEL, para os cenários com 25, 160, 182, 425 e 565 nodos em movimento.

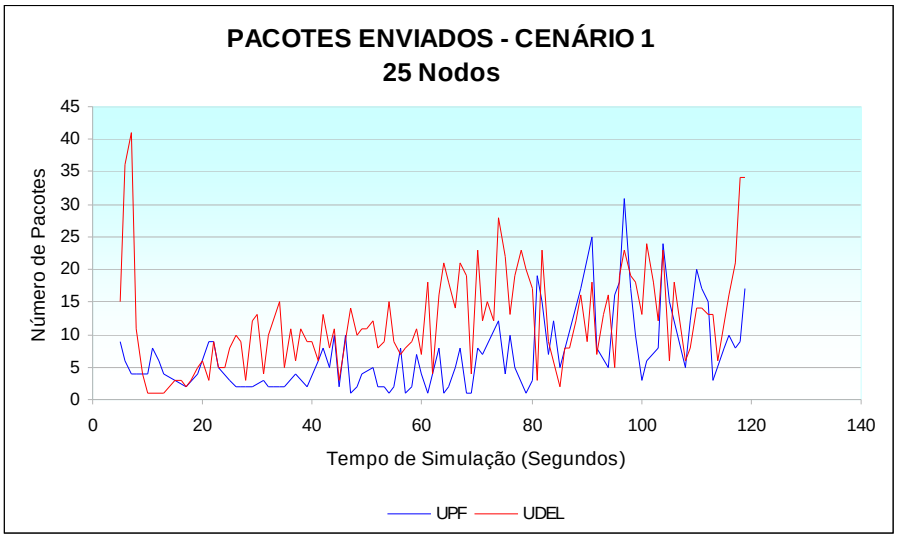

Figura 10 – Cenário 1 - 25 Nodos - Pacotes Enviados nos modelos UPF e UDEL
Fonte: Autoria própria, 2008.

Tabela 3: Comparativo Pacotes Enviados - Cenário 1 - 25 Nodos

Cenário 1 - 25 Nodos		
Segundos	Pacotes Enviados (UPF)	**Pacotes Enviados (UDEL)**
0s à 20s	55	**131**
21s à 40s	51	**157**
41s à 60s	82	**182**
61s à 80s	77	**339**
81s à 100s	208	**241**
101s à 120s	177	**280**
Total	**650**	**1330**

Fonte: Autoria própria, 2008.

No modelo de mobilidade UDEL pode-se perceber que há uma diferença no desempenho do protocolo AODV em comparação com o modelo de mobilidade UPF. Consegue-se visualizar através da figura 10, que o comportamento do modelo de mobilidade UDEL varia bastante, tendo picos de envio de pacotes. Nos dois modelos o envio de pacotes inicia em aproximadamente 5 segundos de simulação.

Conforme a tabela 3, podemos verificar que o modelo UPF mantém um equilíbrio no número de pacotes enviados até o intervalo de 61 a 80 segundos de simulação, tendo um aumento no número de pacotes enviados a partir do intervalo 81 a 100 segundos, se mantendo no intervalo de 101 a 120 segundos.

Porém o modelo de mobilidade UDEL possui o melhor desempenho no cenário com 25 nodos quando utilizado com o protocolo de roteamento AODV, pois o mesmo consegue enviar uma quantidade maior de pacotes que o modelo UPF.

O mesmo ocorre para os cenários de 182 e 425 nodos, conforme mostrado nas figuras 11 e 12 e nas tabelas 4 e 5, onde ocorrem variações no número de pacotes enviados, porém o modelo UDEL se mantém com o melhor índice de pacotes enviados.

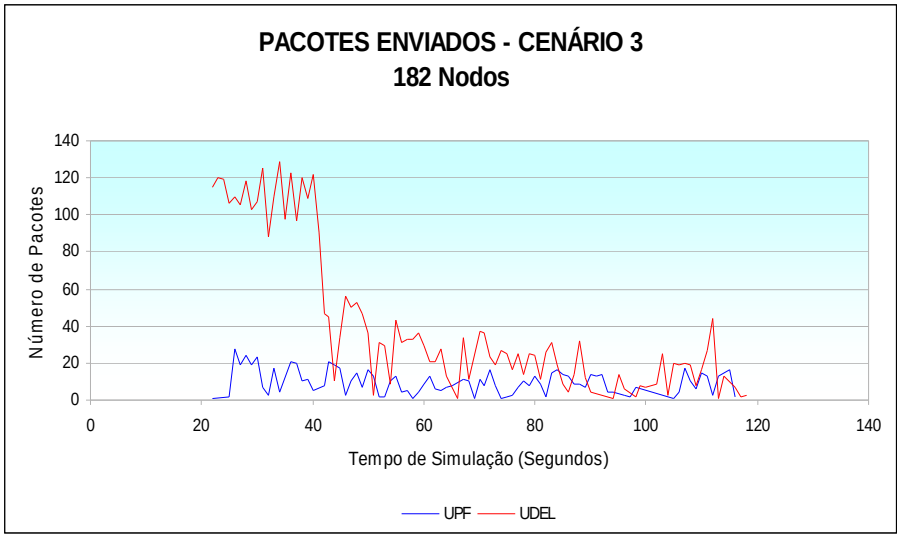

Figura 11 - Cenário 3 - 182 Nodos - Pacotes Enviados nos modelos UPF e UDEL
Fonte: Autoria própria, 2008.

Tabela 4: Comparativo Pacotes Enviados - Cenário 3 - 182 Nodos

Cenário 3 - 182 Nodos		
Segundos	Pacotes Enviados (UPF)	**Pacotes Enviados (UDEL)**
0s à 20s	-	**-**
21s à 40s	214	**2124**
41s à 60s	179	**746**
61s à 80s	148	**400**
81s à 100s	155	**200**
101s à 120s	100	**236**
Total	**796**	**3706**

Fonte: Autoria própria, 2008.

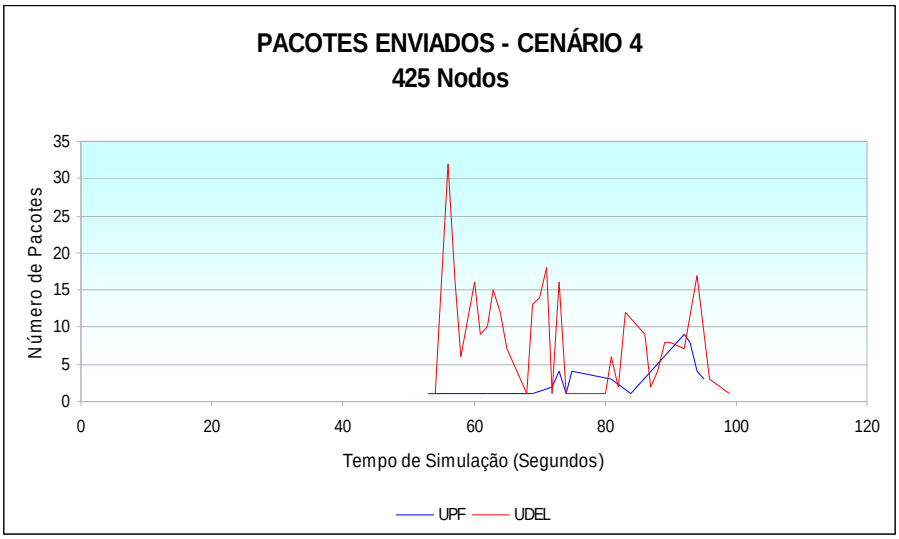

Figura 12 - Cenário 4 - 425 Nodos - Pacotes Enviados nos modelos UPF e UDEL
Fonte: Autoria própria, 2008.

Tabela 5: Comparativo Pacotes Enviados - Cenário 4 - 425 Nodos

Cenário 4 - 425 Nodos		
Segundos	Pacotes Enviados (UPF)	**Pacotes Enviados (UDEL)**
0s à 20s	-	**-**
21s à 40s	-	**-**
41s à 60s	4	**72**
61s à 80s	14	**119**
81s à 100s	28	**79**
Total	**46**	**270**

Fonte: Autoria própria, 2008.

No cenário 3 com 182 nodos no modelo UDEL ocorre um pico no envio de pacotes no intervalo de 21 a 40 segundos de simulação, e conforme o tempo avança, o número de pacotes enviados vai decrescendo. Enquanto que no modelo UPF a quantidade de pacotes enviados se mantém na mesma faixa de 1 a 40 pacotes enviados. Conforme podemos verificar na figura 11 e tabela 4.

Na figura 12 e tabela 5 podemos verificar que no cenário 4 com 425 nodos se movimentando, os pacotes começam a ser enviados no intervalo de 41 a 60, mas precisamente as 53 segundos de simulação, e o fluxo de envio ocorre da mesma forma que no cenário 3. Mantendo o modelo UDEL como o melhor desempenho na rede.

Nos cenário 2 e 5, com 160 e 565 nodos respectivamente, ocorre uma situação contraria do que foi visto nos cenário anteriores, pois o modelo UPF se mostra com melhor desempenho no envio de pacotes, esta informação pode ser vista nas figuras 13 e 14 e nas tabelas 6 e 7.

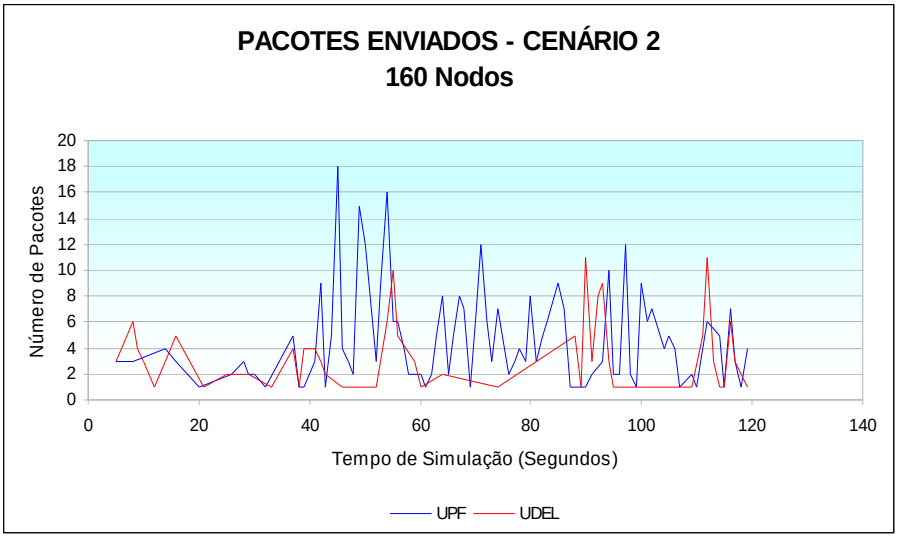

Figura 13 - Cenário 2 - 160 Nodos - Pacotes Enviados nos modelos UPF e UDEL
Fonte: Autoria própria, 2008.

Tabela 6: Comparativo Pacotes Enviados - Cenário 2 - 160 Nodos

Cenário 2 - 160 Nodos		
Segundos	**Pacotes Enviados (UPF)**	Pacotes Enviados (UDEL)
0s à 20s	14	**25**
21s à 40s	17	**23**
41s à 60s	**120**	37
61s à 80s	**94**	3
81s à 100s	**76**	44
101s à 120s	**61**	35
Total	**382**	**167**

Fonte: Autoria própria, 2008.

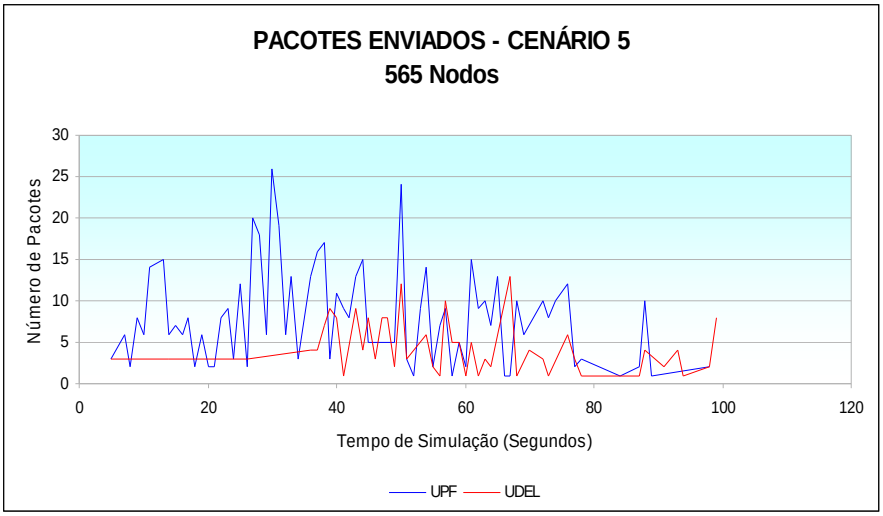

Figura 14 - Cenário 5 - 565 Nodos - Pacotes Enviados nos modelos UPF e UDEL
Fonte: Autoria própria, 2008.

Tabela 7: Comparativo Pacotes Enviados - Cenário 5 - 525 Nodos

Cenário 5 - 565 Nodos		
Segundos	**Pacotes Enviados (UPF)**	Pacotes Enviados (UDEL)
0s à 20s	**91**	9
21s à 40s	**207**	35
41s à 60s	**132**	88
61s à 80s	**117**	43
81s à 100s	16	**22**
Total	**563**	**197**

Fonte: Autoria própria, 2008.

10.2 Vazão

Assim como a métrica anterior analisada, pode-se visualizar uma diferença no desempenho do modelo de mobilidade UDEL em relação ao modelo de mobilidade UPF. Percebe-se que existe uma relação entre pacotes enviados e pacotes recebidos, fazendo com que os gráficos sejam parecidos.

Ao visualizar a figura 11 referente ao cenário com 25 nodos, nota-se que o comportamento do modelo de mobilidade UDEL é bem variável, assim como no envio de pacotes.

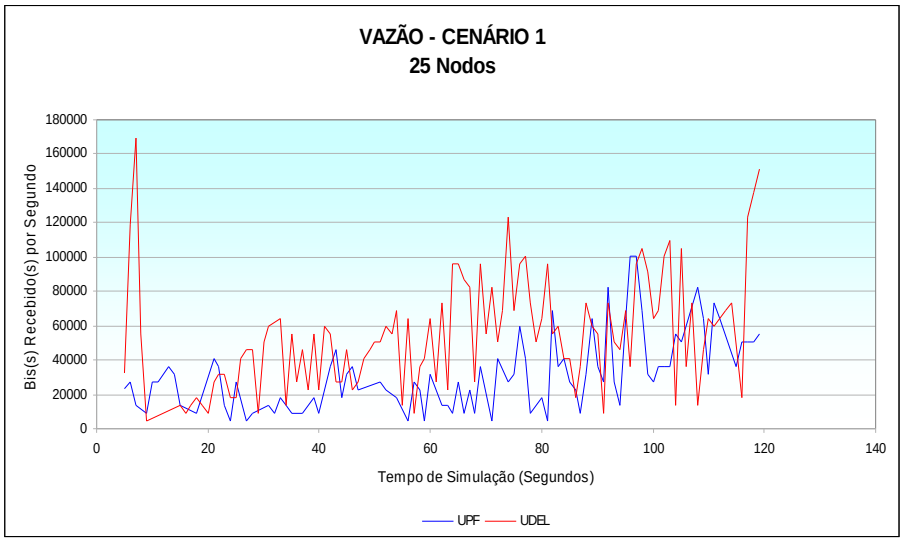

Figura 15 – Cenário 1 - 25 Nodos - Bits Recebidos nos modelos UPF e UDEL
Fonte: Autoria própria, 2008.

O modelo de mobilidade UPF se comporta praticamente da mesma maneira, porém analisando conclui-se que da mesma forma que a métrica anterior, o modelo UDEL é o que possui melhor desempenho. Isto pode ser visualizado na tabela 8, apresentada abaixo.

Tabela 8: Comparativo Bits Recebidos - Cenário 1 - 25 Nodos

Cenário 1 - 25 Nodos		
Segundos	Bits Recebidos (UPF)	**Bits Recebidos (UDEL)**
0s à 20s	220.180	**458.080**
21s à 40s	237.952	**686.400**
41s à 60s	352.352	**864.864**
61s à 80s	375.232	**1.441.440**
81s à 100s	887.744	**1.176.032**
101s à 120s	773.344	**1.262.976**
Total	**2.846.804**	**5.889.792**

Fonte: Autoria própria, 2008

O mesmo ocorre para os cenários de 182 e 425 nodos, conforme mostrado nas figuras 16 e 17 e nas tabelas 9 e 10, onde ocorrem variações no número bits recebidos por segundo, porém o modelo UDEL se mantém com o melhor índice de vazão.

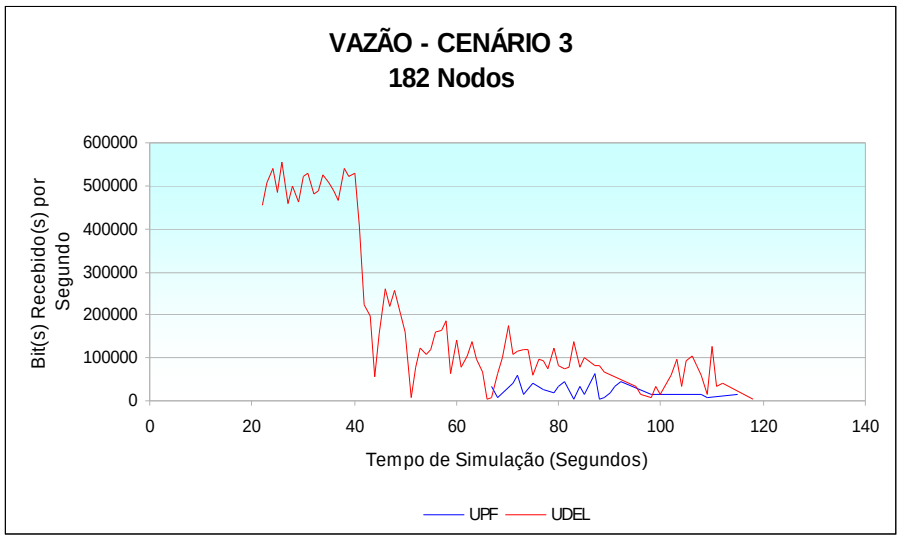

Figura 16 – Cenário 3 - 182 Nodos - Bits Recebidos nos modelos UPF e UDEL
Fonte: Autoria própria, 2008.

Tabela 9: Comparativo Bits Recebidos - Cenário 3 - 182 Nodos

Cenário 3 - 182 Nodos		
Segundos	Bits Recebidos (UPF)	**Bits Recebidos (UDEL)**
0s à 20s	-	**-**
21s à 40s	-	**9.564.320**
41s à 60s	-	**3.294.720**
61s à 80s	275.040	**1.825.824**
81s à 100s	283.712	**800.800**
101s à 120s	36.608	**745.888**
Total	**595.360**	**16.231.552**

Fonte: Autoria própria, 2008

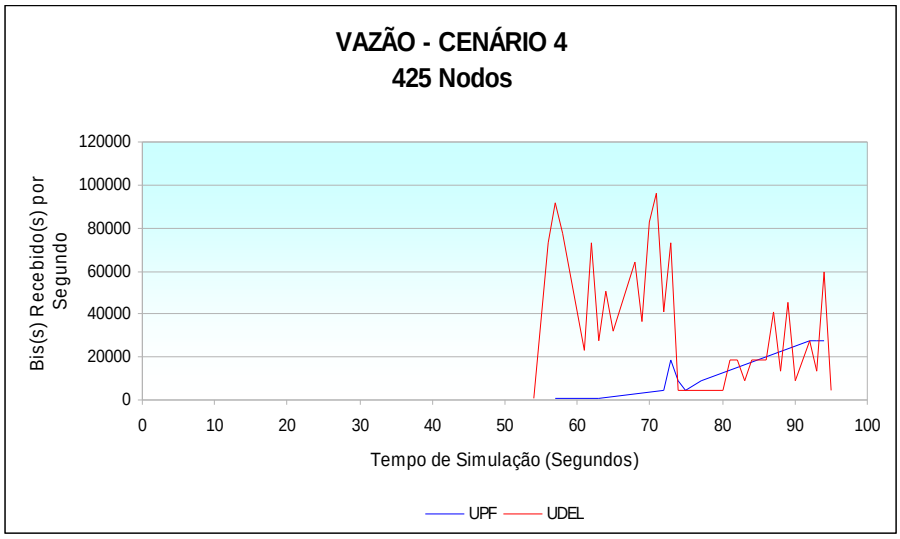

Figura 17 – Cenário 4 - 425 Nodos - Bits Recebidos nos modelos UPF e UDEL
Fonte: Autoria própria, 2008.

Tabela 10: Comparativo Bits Recebidos - Cenário 4 - 425 Nodos

Cenário 4 - 425 Nodos		
Segundos	Bits Recebidos (UPF)	**Bits Recebidos (UDEL)**
0s à 20s	-	**-**
21s à 40s	-	**-**
41s à 60s	480	**243.008**
61s à 80s	46.240	**608.608**
81s à 100s	82.368	**297.440**
Total	**129.088**	**1.149.056**

Fonte: Autoria própria, 2008

No cenário 3 com 182 nodos no modelo UDEL possui um alto índice de pacotes recebidos no intervalo de 20 a 41 segundos de simulação, tendo uma queda no intervalo de 40 a 60 segundos, mantendo-se com a mesma média durante o restante da simulação. Enquanto que no modelo UPF a quantidade de pacotes recebidos se mantém na mesma faixa de 0 a 100000 pacotes recebidos. Conforme podemos verificar na figura 16 e tabela 9.

Na figura 17 e tabela 9 podemos verificar que no cenário 4 com 425 nodos se movimentando, os pacotes começam a ser recebidos no intervalo de 41 a 60, e o fluxo de envio ocorre da mesma forma que no cenário 3. Mantendo o modelo UDEL como o melhor desempenho na rede.

Nos cenário 2 e 5, com 160 e 565 nodos respectivamente, ocorre uma situação contraria do que foi visto nos cenário anteriores, pois o modelo UPF se mostra com melhor desempenho no recebimento de pacotes, esta informação pode ser vista nas figuras 18 e 19 e nas tabelas 11 e 12, mantendo a relação entre

pacotes enviados e pacotes recebidos conforme a métrica anterior. Essa relação faz com que os gráficos sejam parecidos.

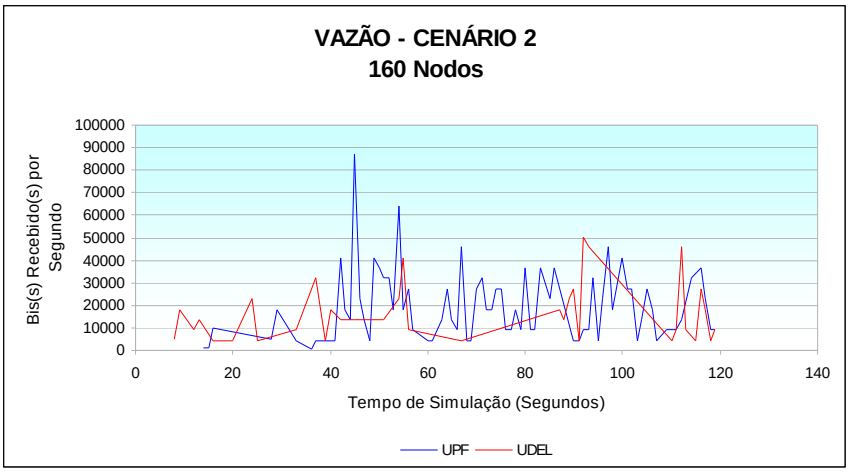

Figura 18 – Cenário 2 - 160 Nodos - Bits Recebidos nos modelos UPF e UDEL
Fonte: Autoria própria, 2008.

Tabela 11: Comparativo Bits Recebidos - Cenário 2 - 160 Nodos

Cenário 2 - 160 Nodos		
Segundos	Bits Recebidos (UPF)	**Bits Recebidos (UDEL)**
0s à 20s	12.032	**55.392**
21s à 40s	32.992	**92.000**
41s à 60s	**489.632**	114.400
61s à 80s	**359.928**	4.576
81s à 100s	**306.592**	183.040
101s à 120s	**265.408**	132.704
Total	**1.466.584**	**582.112**

Fonte: Autoria própria, 2008

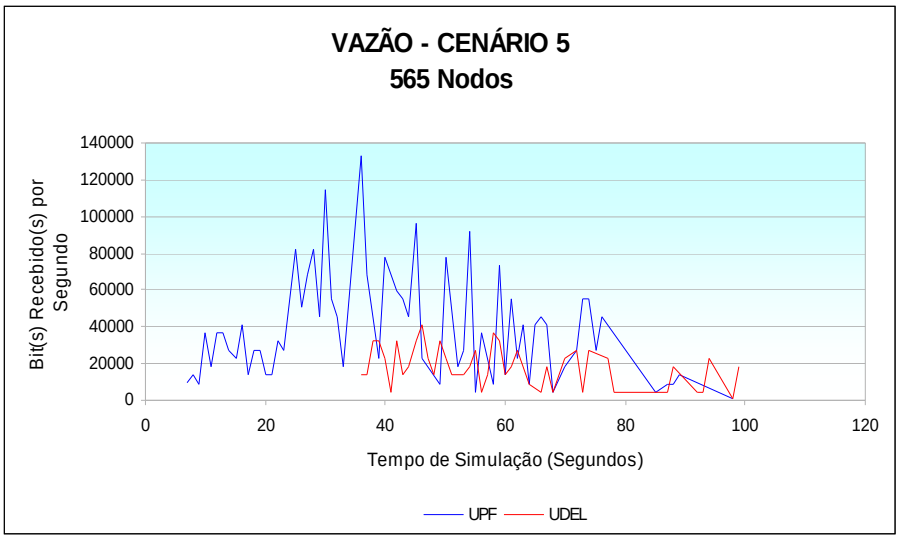

Figura 19 – Cenário 5 - 565 Nodos - Bits Recebidos nos modelos UPF e UDEL
Fonte: Autoria própria, 2008.

Tabela 12: Comparativo Bits Recebidos - Cenário 5 - 565 Nodos

Cenário 5 - 565 Nodos		
Segundos	Bits Recebidos (UPF)	**Bits Recebidos (UDEL)**
0s à 20s	**334.528**	-
21s à 40s	**938.080**	114.880
41s à 60s	**640.640**	407.264
61s à 80s	**489.632**	192.192
81s à 100s	37.088	**74.176**
Total	**2.439.968**	**788.512**

Fonte: Autoria própria, 2008

10.3 Pacotes Perdidos

Para todos os cenários simulados, podemos observar que o modelo de mobilidade UPF obteve níveis mais elevados de perda de pacotes em comparação com o modelo UDEL. A variação nas perdas ocorre somente em tempo de simulação, pois em alguns intervalos de tempo, o modelo UDEL perde mais pacotes que o UPF, porém na quantidade total de perda, o UDEL é o que possui o melhor desempenho nesta métrica, pois a quantidade de pacotes perdidos é menor em relação ao modelo UPF. Isto pode ser observado nas Figuras 20, 21, 22, 23 e 24 e comprovado nas Tabelas 13, 14, 15, 16 e 17 onde é mostrado o número total de pacotes perdidos em todos os intervalos de tempo.

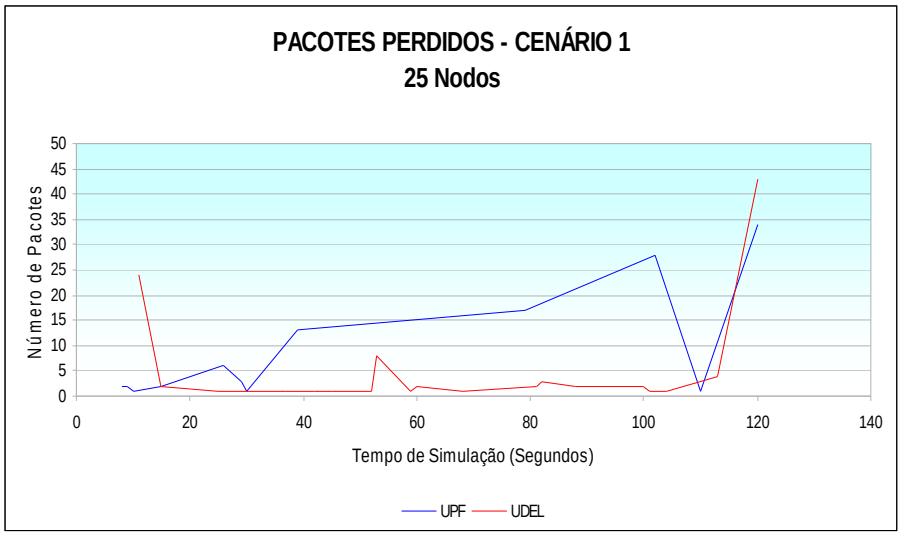

Figura 20 – Cenário 1 - 25 Nodos - Pacotes Perdidos nos modelos UPF e UDEL
Fonte: Autoria própria, 2008.

Tabela 13: Comparativo Pacotes Perdidos - Cenário 1 - 25 Nodos

Cenário 1 - 25 Nodos		
Segundos	Pacotes Perdidos (UPF)	**Pacotes Perdidos (UDEL)**
0s à 20s	**7**	26
21s à 40s	23	**2**
41s à 60s	**-**	14
61s à 80s	17	**1**
81s à 100s	**-**	9
101s à 120s	63	**49**
Total	**110**	**101**

Fonte: Autoria própria, 2008

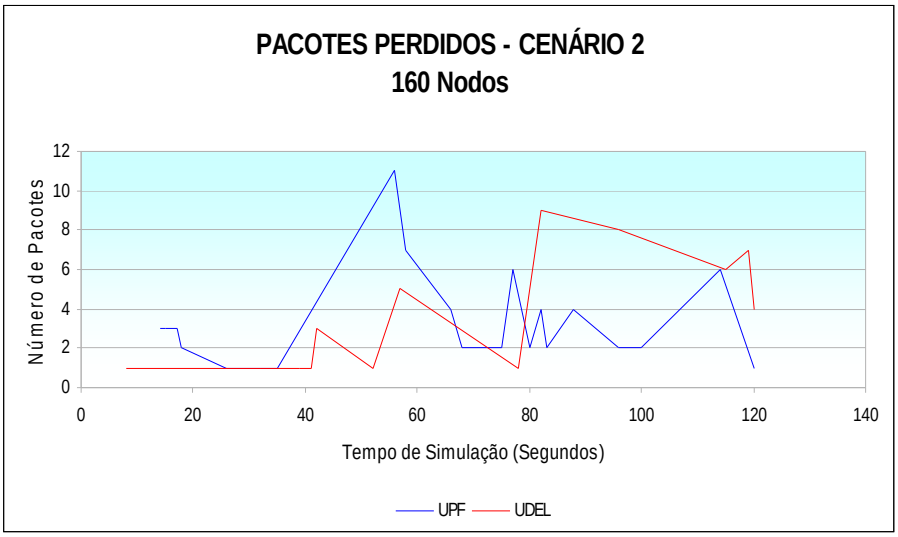

Figura 21 – Cenário 2 - 160 Nodos - Pacotes Perdidos nos modelos UPF e UDEL
Fonte: Autoria própria, 2008.

Tabela 14: Comparativo Pacotes Perdidos - Cenário 2 - 160 Nodos

Cenário 2 - 160 Nodos		
Segundos	Pacotes Perdidos (UPF)	**Pacotes Perdidos (UDEL)**
0s à 20s	11	**1**
21s à 40s	3	**1**
41s à 60s	18	**10**
61s à 80s	16	**1**
81s à 100s	**14**	17
101s à 120s	**7**	17
Total	**69**	**47**

Fonte: Autoria própria, 2008

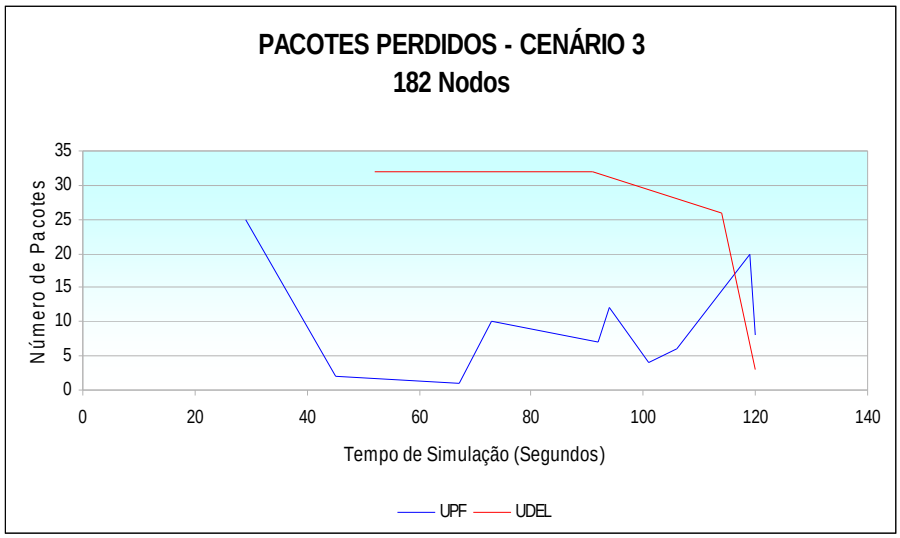

Figura 22 – Cenário 3 - 182 Nodos - Pacotes Perdidos nos modelos UPF e UDEL
Fonte: Autoria própria, 2008.

Tabela 15: Comparativo Pacotes Perdidos - Cenário 3 - 182 Nodos

Cenário 3 - 182 Nodos		
Segundos	Pacotes Perdidos (UPF)	**Pacotes Perdidos (UDEL)**
0s à 20s	-	**-**
21s à 40s	25	**-**
41s à 60s	**2**	32
61s à 80s	11	**-**
81s à 100s	**19**	32
101s à 120s	38	**29**
Total	**95**	**93**

Fonte: Autoria própria, 2008

Neste cenário, houve uma aproximação no número de pacotes perdidos, porém o UDEL permanece com menos pacotes perdidos que o modelo UPF.

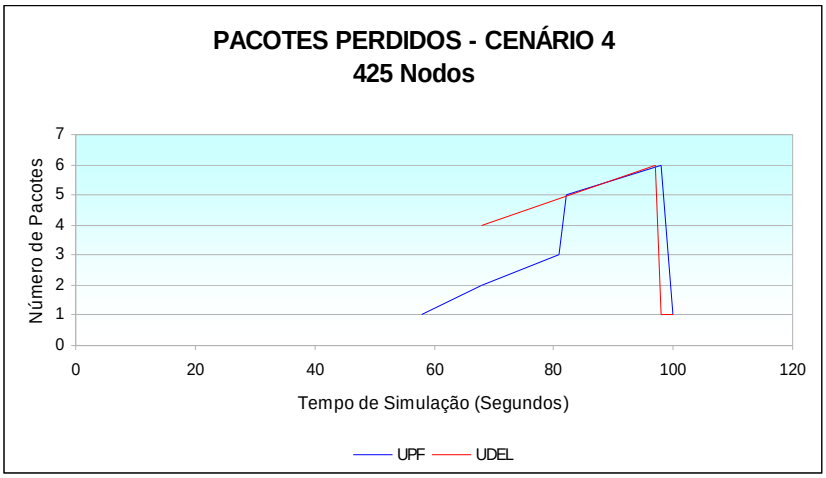

Figura 23 – Cenário 4 - 425 Nodos - Pacotes Perdidos nos modelos UPF e UDEL
Fonte: Autoria própria, 2008.

Tabela 16: Comparativo Pacotes Perdidos - Cenário 4 - 425 Nodos

Cenário 4 - 425 Nodos		
Segundos	Pacotes Perdidos (UPF)	**Pacotes Perdidos (UDEL)**
0s à 20s	-	**-**
21s à 40s	-	**-**
41s à 60s	1	**-**
61s à 80s	**2**	4
81s à 100s	15	**8**
Total	**18**	**12**

Fonte: Autoria própria, 2008

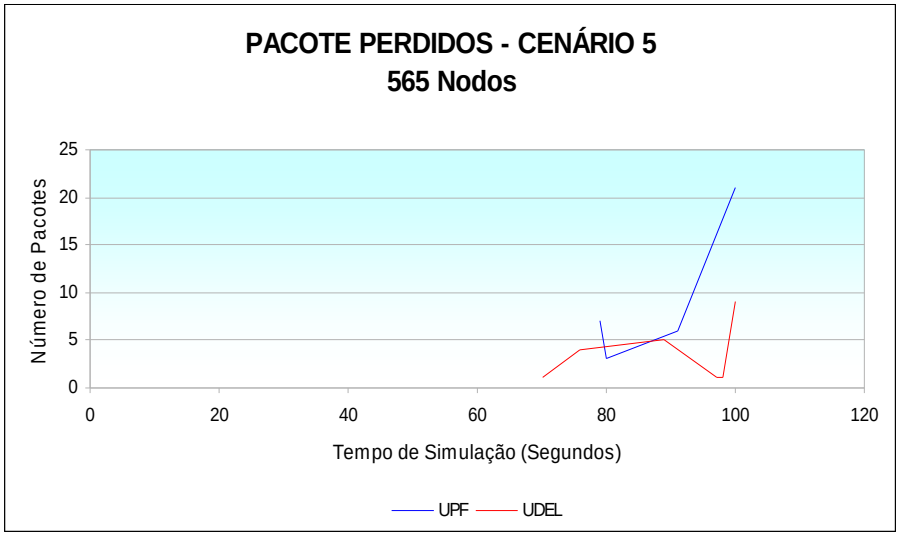

Figura 24 – Cenário 5 - 565 Nodos - Pacotes Perdidos nos modelos UPF e UDEL
Fonte: Autoria própria, 2008.

Tabela 17: Comparativo Pacotes Perdidos - Cenário 5 - 565 Nodos

Cenário 5 - 565 Nodos		
Segundos	Pacotes Perdidos (UPF)	**Pacotes Perdidos (UDEL)**
0s à 20s	-	**-**
21s à 40s	-	**-**
41s à 60s	-	**-**
61s à 80s	10	**5**
81s à 100s	27	**16**
Total	**37**	**21**

Fonte: Autoria própria, 2008

10.4 Taxa de Entrega

Na figura 25, percebe-se que o modelo de mobilidade UDEL apresentou um melhor desempenho na maioria dos cenários. O UPF manteve um bom desempenho em envio e recebimento de pacotes no cenário 2 com 160 nodos, como pode ser visualizado abaixo.

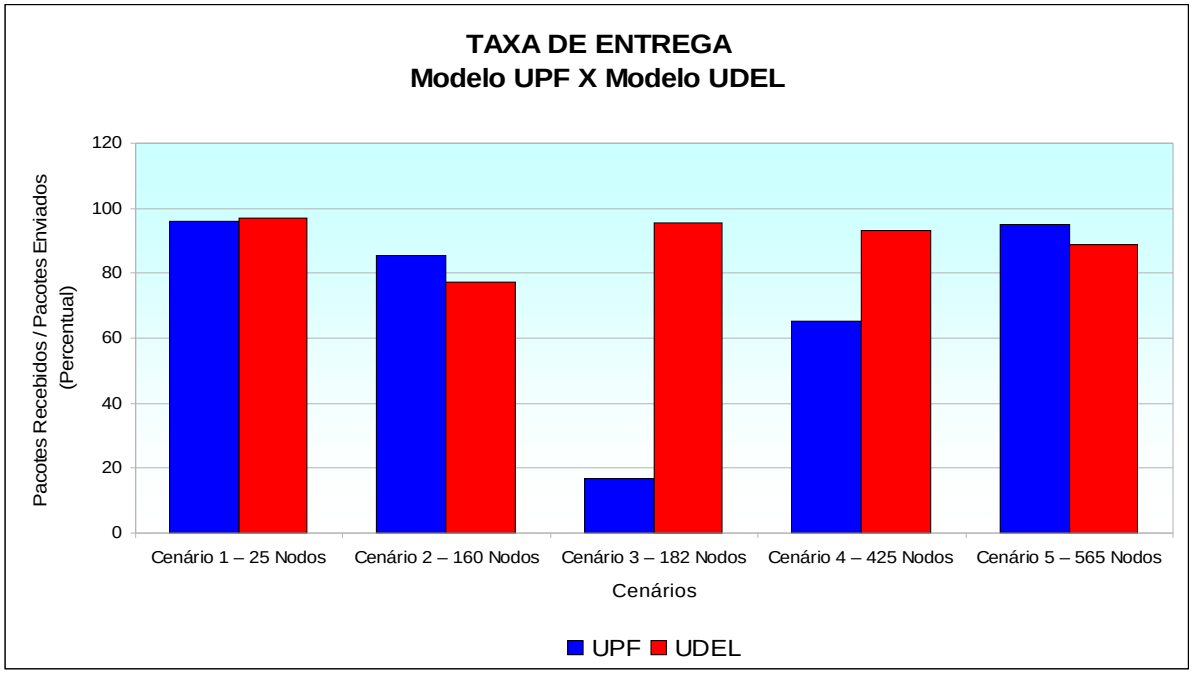

Figura 25 – Taxa de Entrega – Modelo UPF X Modelo UDEL
Fonte: Autoria própria, 2008.

O modelo UPF, nos cenário 3 e 4, obtiveram uma taxa de entrega baixa, pois nesses cenário foi necessário utilizar o processo de descoberta de rotas do protocolo de roteamento AODV. Como mostra a Figura 27.

10.5 Taxa de Perda

A Figura 26 representa o percentual de pacotes perdidos durante as simulações, separadas por cenários. Nota-se nesta figura que o modelo UDEL teve uma quantidade maior de pacotes perdidos nos cenários 2 e 5, enquanto que nos demais cenários o modelo UPF perdeu um número maior de pacotes.

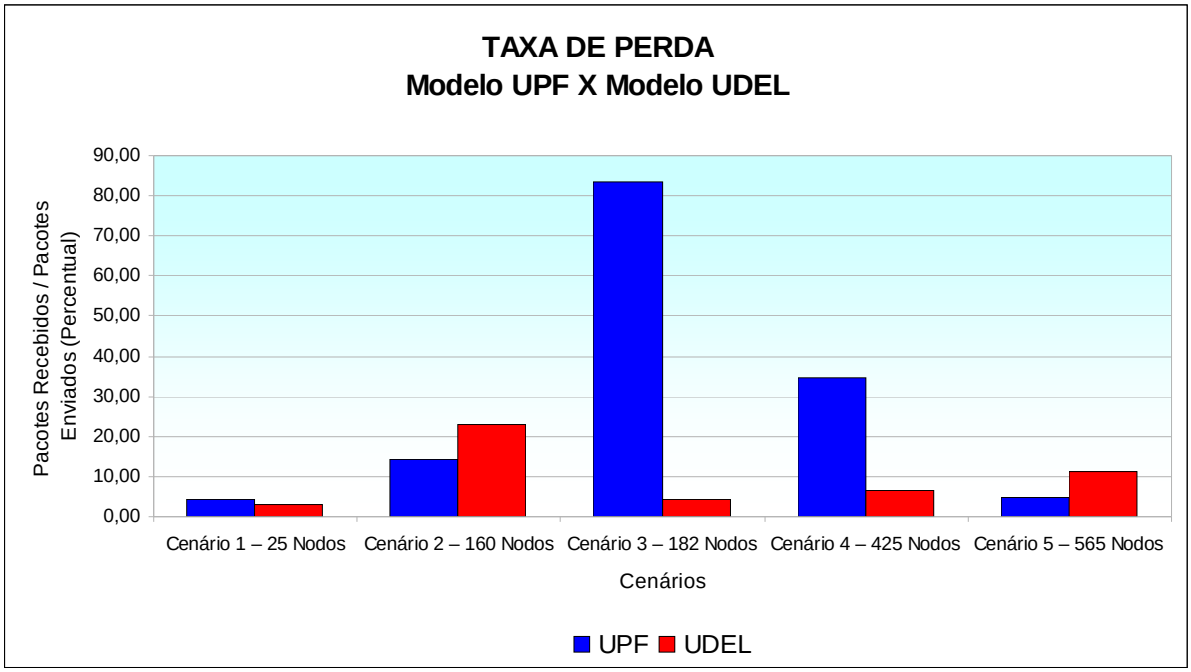

Figura 26 – Taxa de Perda – Modelo UPF X Modelo UDEL
Fonte: Autoria própria, 2008.

Realizando uma comparação entre os dois modelos, percebemos que o modelo UDEL obteve um melhor desempenho, pois na maioria dos cenários obteve uma taxa de perda menor.

10.6 Overhead de Roteamento

Na figura 27 representa o percentual de *overhead* gerado durante as simulações. Em destaque, percebe-se nesta figura que, o modelo UPF utilizando o protocolo de roteamento AODV matem um nível mais elevado de pacotes de controle na rede (*overhead*), pois o AODV não consegue manter uma comunicação entre os nodos, o que se faz necessário a inicialização do processo de descoberta de rotas. Foi por esse motivo que analisando as figuras acima se percebe a razão da menor taxa de entrega e da alta taxa de perda comparada com o modelo UDEL.

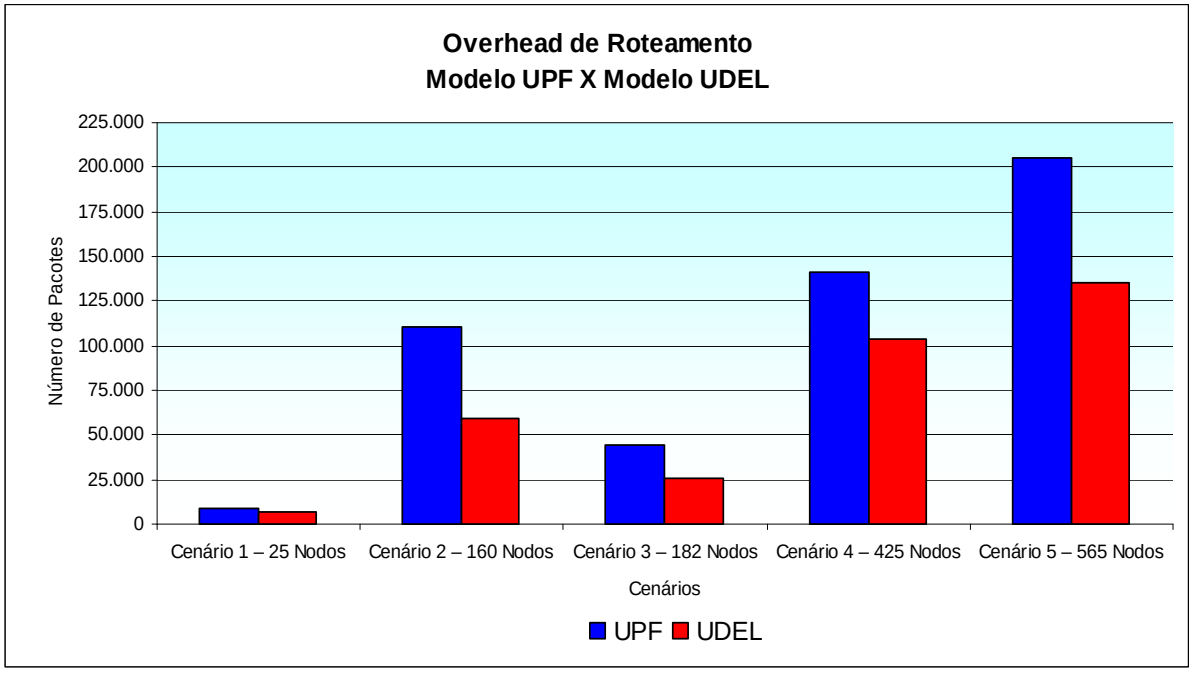

Figura 27 – Overhead de Roteamento – Modelo UPF X Modelo UDEL
Fonte: Autoria própria, 2008.

Outra análise que pode ser realizada é que quanto maior o número de nodos no cenário, maior é o overhead gerado pela rede. Desta forma ocorre um congestionamento na rede ocasionando a perda de pacotes, conforme mostrado na figura 26 anteriormente explicada.

Conclui-se então que, pela média de todos os cenários simulados, o modelo UDEL é o que tem o melhor desempenho comparado ao modelo UPF, nesta métrica.

10.7 Delay

O atraso na entrega dos dados tem impacto sobre a qualidade da rede, sendo desejado que esta métrica tenha sempre o menor valor possível. Podemos verificar na figura 28 que o modelo UDEL possui os menores valores de d*elay* em relação ao modelo UPF

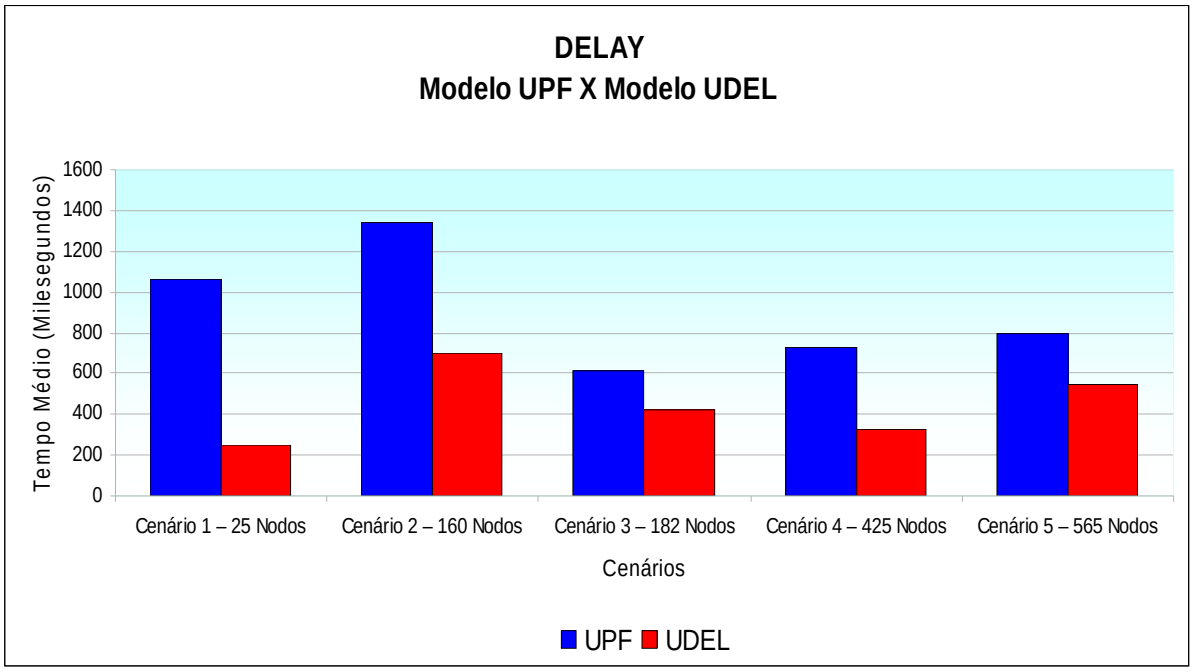

Figura 28 – Delay – Modelo UPF X Modelo UDEL
Fonte: Autoria própria, 2008.

O modelo UPF mostram valores altos de delay, enquanto que o modelo UDEL mantém uma média de atraso entre todos os cenários simulados. Isso demonstra que o modelo UPF tem um retardo maior na entrega dos pacotes.

Sendo assim, para essa métrica o modelo UDEL obteve o melhor desempenho, pois manteve na maioria dos cenários a o menor atraso na entrega dos dados.

10.8 Jitter

Podemos observar na figura 29, que o modelo UDEL os valores de *jitter* são menores em comparação com o modelo UPF. Quanto menores forem os valores do *jitter* melhor será a qualidade das transmissões.

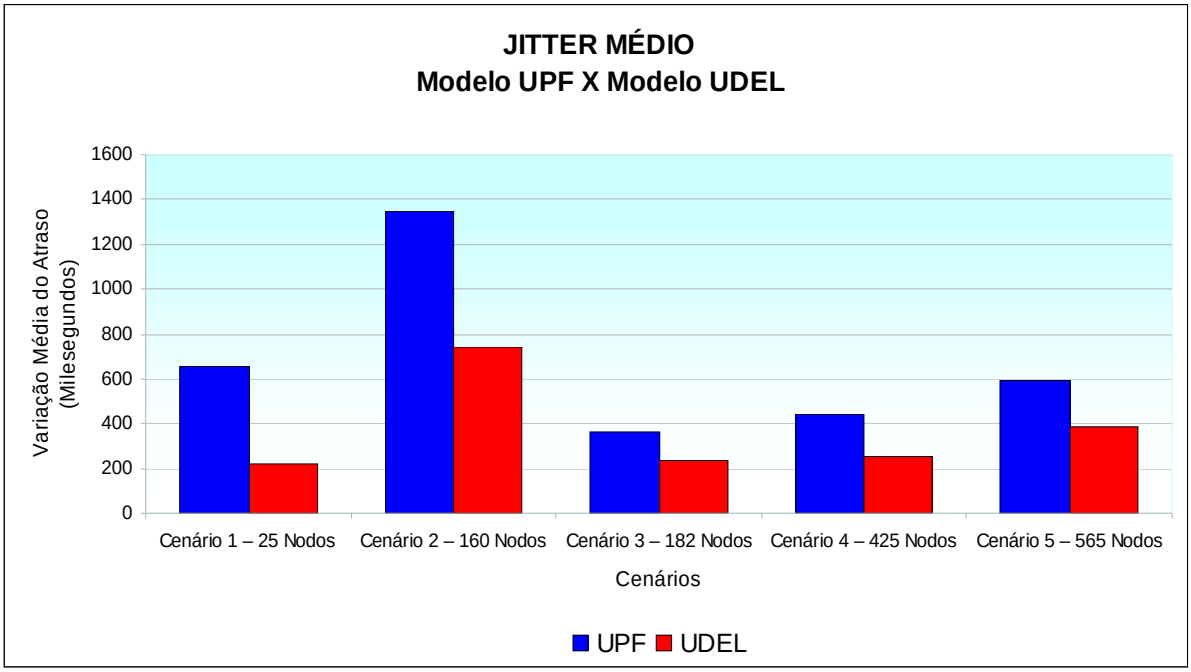

Figura 29 – Jitter – Modelo UPF X Modelo UDEL
Fonte: Autoria própria, 2008.

Sendo assim, para essa métrica o modelo UDEL obteve o melhor desempenho, pois manteve na maioria dos cenários a mesma faixa de valores no *Jitter,* ou seja, uma menor variação do atraso (variação do *delay*).

11 CONCLUSÃO

As redes sem fio estão sendo bastante utilizadas em ambientes empresariais e acadêmicos e devido a esse crescimento na utilização desse tipo de rede, é válido aprofundar o conhecimento dessas redes.

Devido a fácil configuração e a flexibilidade existem vários produtos e tecnologias com suporte a esse tipo de rede, gerando um aumento no consumo.

Este trabalho apresentou um estudo de 2 modelos de mobilidade realísticos e um protocolo de roteamento desenvolvido para redes *ad hoc*. Os modelos apresentados representam o comportamento dos nós móveis em uma rede real.

Baseado neste estudo pode-se apresentar uma avaliação comparativa entre os modelos de mobilidade realísticos UPF e UDEL, utilizando o protocolo de roteamento AODV.

Utilizando as ferramentas geradoras de cenários e o simulador de redes NS, foram criados os cenários onde caracterizam os modelos de mobilidade, aplicando o protocolo sobra cada um dos modelos conseguiu-se capturar as informações relevantes para análise das métricas de desempenho, sendo que as mesmas foram apresentadas na forma de gráficos e tabelas.

Com a análise realizada das métricas, pode-se perceber que o desempenho de cada modelo de mobilidade se altera para o protocolo de roteamento utilizado. Por essa razão, é importante que os modelos possam representar da melhor forma possível o movimento dos nodos dentro dos cenários que se pretende simular. Desta forma é possível utilizar o modelo mais apropriado em cada cenário.

Neste trabalho, podê-se concluir que o modelo de mobilidade UDEL teve um melhor desempenho em cenários com menor número de nodos em movimento e que o modelo UPF obteve o melhor desempenho no cenário com maior número de nodos na rede, isso demonstra que de acordo com o grau de conectividade existente

entre os nodos na rede o desempenho é alterado, porém para validar essas informações seria necessário realiza uma análise de conectividade.

Neste trabalho, podê-se concluir que o melhor modelo de mobilidade realístico a ser utilizado é o UDEL, em todas as métricas avaliadas foi o modelo que apresentou o melhor desempenho quando utilizado o protocolo de roteamento AODV.

Através destes resultados, este trabalho pode justificar a importância da análise realizada, pois se consegue concluir que um dos maiores problemas das redes ad hoc é justamente a grande mobilidade dos nós em ambientes reais. Fazendo com que a topologia seja alterada causando impacto no desempenho do protocolo.

Uma das limitações encontradas neste trabalho foi referente à impossibilidade de conseguir simular os 3 modelos de mobilidade realísticos propostos inicialmente. Esta limitação se deu devido a problemas encontrados durante a instalação e configuração das ferramentas geradoras de cenário, conforme pode ser visto no Apêndice A.

11.1 Trabalhos Futuros

Para trabalhos futuros, indica-se três sugestões que possam dar continuidade na análise do desempenho dos modelos de mobilidade.

A primeira é realizar uma comparação entre os modelos de mobilidade realísticos e os modelos sintéticos já apresentados em (THOMA; NUNES, 2006).

A segunda é realizar um estudo e comparação dos protocolos de roteamento DSDV, DSR, DYMO, etc. utilizando os mesmos modelos de mobilidade realísticos apresentados neste trabalho.

E a última sugestão trata-se da tentativa de alterar o *plugin* do modelo OM, atualizando o mesmo para uma versão mais recente do NS e realizar as simulações. Com estes dados, o mesmo poderia ser comparado com os resultados deste trabalho em específico.

REFERÊNCIAS

BOHACEK, Stephan; SRIDHARA, Vinay; SINGH, Gaurav; ILIC, Andjela. (2004). **"The Udel Models – MANET Mobility and Path Loss in an Urban/Suburban Environment".** Technical Report.

CAMP, Tracy; BOLENG, Jeff; DAVIES, Vanessa. (2002). **"A survey of mobility models for ad hoc network research."** Wireless Communication & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, 2(5):483–502.

CAMPOS, Carlos A. V. (2003). **"Uma modelagem da mobilidade individual para redes móveis ad hoc"**. Dissertação de Mestrado. Universidade Federal do Rio de Janeiro, Rio de Janeiro.

CAPKA, Joe; BOUTABA, Raouf. (2005). **"A Mobility Management Tool – The Realistic Mobility Model"**. In: School of Computer Science at the University of Waterloo, Canada.

COUNTINHO, M.M. (2003) **"Network Simulator – Guia Básico para Iniciantes"**. Disponível em: <http://www.cci.unama.br/margalho/ networksimulator>

ERLANG. (2005) **"O que é Simulação"**. Disponível em: <http://www.erlang.com.br/brsimulad.asp>

GAWK. (2004). **"Gawk: Effective AWK Programming"**. Disponível em: <http://www.gnu.org/software/gawk/manual/>.

HELBING, Dirk; FARKAS, Illes; and VICSEK, Tamas. (2000). **"Simulating dynamical features of escape panic"**. Nature, 407:487-490.

JARDOSH, Amit; M. BELDING-ROYER, Elizabeth; C. ALMEROTH, Kevin; SURI, Subhash. (2005). **"Real-world environment models for mobile ad hoc networks"**. IEEE Journal on Special Areas in Communications - Special Issue on Wireless Ad hoc Networks.

JARDOSH, Amit; M. BELDING-ROYER, Elizabeth; C. ALMEROTH, Kevin; SURI, Subhash. (2006). **"Towards Realistic Mobility Models For Mobile Ad hoc Networks"**. The Pennsylvania State University CiteSeer Archives.

JUNGKEUN,Yoon; BRIAN, D. Noble; MINGYAN, Liu; and MINKYONG, Kim. (2006). **"Building realistic mobility models from coarse-grained traces"**. In MobiSys '06: Proceedings of the 4th international conference on Mobile systems, applications and services, pages 177-190.

KAMIENSKI, A.C., SADOK, D., CAVALCANTI, T.A.D., SOUSA, T.M.D. e DIAS, L.K. (2002) **"Simulando a Internet: Aplicações na Pesquisa e no Ensino"**, Disponível em: <http://www.cin.ufpe.br/~cak/publications/jai2002-capitulo2.pdf>.

KIM, Jonghyun; SRIDHARA, Vinay; BOHACEK, Stephan. (2006). **“Realistic Simulation of Urban Mesh Networks – Part I: Urban Mobility”**. U. Delaware Technical Report.

KIM, Jonghyun; BOHACEK, Stephan. (2005). **“A Survey-Based Mobility Model of People for Simulation os Urban Mesh Networks”**. University of Delaware.

KONISHI, Kazuki; MAEDA, Kumiko; SATO, Kazuki; YAMASAKI, Akiko; YAMAGUCHI, Hirozumi; YASUMOTO, Keiichi; HIGASHINO, Teruo. (2005) **“MobiREAL Simulator -- Evaluating MANET Applications in Real Environments”**. in Proceedings of 13th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems.

KYRIAKAKOS, M., FRANGIADAKIS, N., HADJIEFTHYMIADES, S., MERAKOS, L. (2002). **“Realistic Mobility Pattern Generator: Design and Application in Path Prediction Algoritm Evaluation”**.IEEE.

LASSILA, Pasi; HYYTI¨A, Esa; KOSKINEN, Henri. (2005**). “Connectivity properties of random waypoint mobility model for ad hoc networks”**. Proceedings of the Fourth Annual Mediterranean Workshop on Ad Hoc Networks.

LEGENDRE, Franck; BORREL, Vincent; AMORIM, Marcelo D.; and FDIDA, Serge. (2006) **“Reconsidering microscopic mobility modeling for self-organizing networks”**. IEEE Network, pages 4-12.

MAEDA, Kumiku; SATO, Kazuki; KONISHI, Kazuki; YAMASAKI, Akiko; UCHIYAMA, Akira; YAMAGUCHI, Hirozumi; YASUMOTO, Keiichi; HIGASHINO, Teruo. (2005). **“Getting Urban Pedertrian Flow from Simple Observation: Realistic Mobility Generation in Wireless Network Simulation”**. International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems, Pages: 151 - 158, Year of Publication: 2005, ISBN:1-59593-188-0

M. Overmars M. de Berg, M. van Kreveld and O. Schwarzkopf. (2000). **“Computational geometry: Algorithms and applications”**. Springer Verlag.

MONTE, Luis R; BERNARDO, Rodrigo. et alli. (2005).**“Protocolos de Roteamento e Protocolos TCP/UDP em redes ad hoc”**, Disponível em: <http://www.teleco.com.br/tutoriais/tutorialprotocolo/default.asp>.

NAM. (2002). **“NAM - Network Simulator”**. Disponível em: <http://www.isi.edu/nsnam/nam/>.

NS. (2002). **“The Network Simulator Manual”**. Disponível em: <http://www.isi.edu/nsnam/ns/ns-documentation.html>.

PEREIRA, Ivana Cardial de Miranda. (2004). **“Análise do roteamento em redes móveis ad hoc em cenários de operações militares”**. Rio de Janeiro. Disponível em: < www.gta.ufrj.br/ftp/gta/TechReports/Ivana04/ivana04.pdf.gz>

PERKINS, C. E.; BELDING-ROYER, Elizabeth M. et alli. (2005). **“Ad hoc On-Demand Distance Vector (AODV) Routing”**. Disponível em: <http://www.ietf.org/rfc/rfc3561.txt?number=3561>.

PINHEIRO, José Mauricio Santos. (2005). **“Redes Móveis Ad Hoc”**. Disponível em: < http://www.projetoderedes.com.br/artigos/artigo_redes_moveis_ad_hoc.php>.

SILVA, Alan P. T. da; OLIVEIRA, Gustavo A. (2005). **“Comparação de protocolos de roteamento sob diferentes modelos de mobilidade em redes Ad Hoc”**. 2005. Trabalho de Conclusão de Curso de Engenharia. Universidade de Brasília, Brasília.

SRIDHARA, Vinay; BOHACEK, Stephan. (2007). **“Realistic Propagation Simulation of Urban Mesh Networks”**. Computer Networks: The International Journal of Computer and Telecommunications Networking Volume 51, Issue 12, Pages: 3392-3412, Year of Publication: 2007, ISSN:1389-1286

SRIDHARA, Vinay; KIM, Jonghyun; BOHACEK, Stephan. (2005). **“Models and Methodologies for Simulating Mobile Ad Hoc Networks”**. The 3rd IEEE International Workshop on Mobility Management and Wireless Access.

SRIDHARA, Vinay; KIM, Jonghyun; BOHACEK, Stephan.(2005). **“Performance of Urban Mesh Networks”**. In MSWiM ’05: Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems, Pages: 269–277.

STALLINGS, William. (2005). **“Wireless Communications & Networks”**. Ed. Prentice-Hall. New Jersey.

TANEMBAUM, Andrew. (1997). **“S. Redes de Computadores”**. 3ª ed. Rio de Janeiro: Campus, [s.n.].

THOMA, Virgínia Mattos; NUNES, C. M. (2006). **“Avaliação de Desempenho de Redes Ad Hoc Utilizando Modelos de Mobilidade”**. In: IV Escola Regional de Redes de Computadores, 2006, Passo Fundo.

THOMA, Virgínia Mattos. (2006). **“Avaliação de Desempenho de Redes Ad Hoc Utilizando Modelos de Mobilidade”**. Trabalho de Conclusão de Curso em Ciências da Computação. Centro Universitário La Salle, Canoas.

TUGCU, Tuna; ERSOY, Cem. (2001). **“How a New Realistic Mobility Model Can Effect the Relative Performance of a Mobile Networking”**. In: IEEE VTC, Rhodes, Spring.

U. C. Bureau. (2006) **“Topologically Integrated Geographic Encoding and Referecing (TIGER)”**. Disponível em: <http://www.census.gov/geo/www/tiger/>

APÊNCIDE A – Problemas Encontrados na Instalação do NS-2.26 e do Plugin para o Modem de Mobilidade Realístico OM

Ao tentar instalar o NS-2.26 ocorreram os seguintes problemas com as correções realizadas:

1º erro:

```
./mobile/god.h:88: error: extra qualification 'vector::' on member
'operator='
./mobile/god.h:93: error: extra qualification 'vector::' on member
'operator+='
./mobile/god.h:98: error: extra qualification 'vector::' on member
'operator=='
./mobile/god.h:101: error: extra qualification 'vector::' on member
'operator!='
make: *** [src_rtg/sragent.o] Error 1
Ns make failed!
See http://www.isi.edu/nsnam/ns/ns-problems.html for problems
```

Solução para o problema:
Utilizar o Patch econtrado no caminho http://www.isi.edu/nsnam/dist/ns-2.1b7-god.patch
Alterar Todos os "vector" por "Vector" (primeira letra em maiusculo)

2º erro:

```
setdest.h:26: error: extra qualification 'vector::' on member 'operator='
setdest.h:31: error: extra qualification 'vector::' on member 'operator+='
setdest.h:36: error: extra qualification 'vector::' on member 'operator=='
setdest.h:39: error: extra qualification 'vector::' on member 'operator!='
make[1]: *** [setdest.o] Error 1
```

Solução:
Utilizar o Patch disponível no caminho http://mailman.isi.edu/pipermail/ns-users/2007-January/058412.html
Retirar o "Vector" da frente dos "Operador"
Exemplo:
Alterar inline void vector::operator=(const vector a) {
Por inline void operator=(const vector a) {

3º erro:

```
mobile/god.cc: In member function 'bool God::IsNeighbor(int, int)':
mobile/god.cc:476: error: 'vector' was not declared in this scope
mobile/god.cc:476: error: expected `;' before â€˜aâ€™
mobile/god.cc:477: error: expected `;' before â€˜bâ€™
mobile/god.cc:478: error: expected `;' before â€˜dâ€™
mobile/god.cc:480: error: â€˜dâ€™ was not declared in this scope
make: ** [mobile/god.o] Erro 1
```

Solução:
Alterar todas os "vector" por "Vector"

4º erro:
Foi necessário realizar várias mudanças nos arquivos configure para compensar a análise mais rigorosa. Todas as linhas dos arquivos:

```
./tcl8.3.2/unix/configure:4701
./tcl8.3.2/unix/configure:5705
./tk8.3.2/unix/configure:1520
./otcl-1.0a8/configure:4167
```

```
Solução:
Linhas: system=MP-RAS-`awk '{print }' /etc/.relid'`
Alterar para: system=MP-RAS-`awk '{print }' /etc/.relid`
Deve ser removido o caracter "'" do fim.

5º erro:
queue/cbq.cc:112: error: ISO C++ forbids declaration of â€˜CBQueueâ€™ with
no type
queue/cbq.cc:112: error: expected â€˜;â€™ before â€˜*â€™ token
queue/cbq.cc: In member function â€˜virtual int
CBQueue::insert_class(CBQClass*)â€™:
queue/cbq.cc:488: error: â€˜class CBQClassâ€™ has no member named
â€˜cbq_â€™
queue/cbq.cc: In constructor â€˜CBQClass::CBQClass()â€™:
queue/cbq.cc:805: error: class â€˜CBQClassâ€™ does not have any field named
â€˜cbq_â€™
queue/cbq.cc: In member function â€˜virtual void CBQClass::recv(Packet*,
Handler*)â€™:
queue/cbq.cc:850: error: â€˜cbq_â€™ was not declared in this scope
queue/cbq.cc:856: error: â€˜cbq_â€™ was not declared in this scope
queue/cbq.cc: In member function â€˜void CBQClass::update(Packet*,
double)â€™:
queue/cbq.cc:873: error: â€˜cbq_â€™ was not declared in this scope
queue/cbq.cc: In member function â€˜int CBQClass::desc_with_demand()â€™:
queue/cbq.cc:928: error: â€˜cbq_â€™ was not declared in this scope
queue/cbq.cc: In member function â€˜void CBQClass::newallot(double)â€™:
queue/cbq.cc:975: error: â€˜cbq_â€™ was not declared in this scope
queue/cbq.cc: In member function â€˜virtual int CBQClass::command(int,
const char* const*)â€™:
queue/cbq.cc:1002: error: â€˜cbq_â€™ was not declared in this scope
make: ** [queue/cbq.o] Erro 1

Solução:
No arquivo /ns-allinone-2.33/ns-2.33/queue/cpq.cc esta faltando a linha :
class CBQueue; antes da class CBQClass : public Connector {

6º erro:
webcache/webtraf.cc: In member function â€˜virtual int
WebTrafPool::command(int, const char* const*)â€™:
webcache/webtraf.cc:416: error: ISO C++ forbids initialization in array new
make: ** [webcache/webtraf.o] Erro 1

Solução:
Incluir a linha "#include <vector>" abaixo da linha "#include <iostream>"
Alterar a linha "server_ = new WebServer[nServer_](this);" por
"std::vector<WebServer> server_(nServer_, this);"

7º erro:
./tora/tora_neighbor.h:72: error: ISO C++ forbids declaration of
â€˜toraAgentâ€™ with no type
./tora/tora_neighbor.h:72: error: expected â€˜;â€™ before â€˜*â€™ token
make: ** [tora/tora.o] Erro 1

Solução:
Colocar a linha "class toraAgent;" que estava faltando e
Alterar "LINK_UN = 0x0004,  // undirected" por "LINK_UN = 0x0004  //
undirected"

8º erro:
```

```
dsr/dsragent.cc: In member function â€˜void
DSRAgent::handleFlowForwarding(SRPacket&, int)â€™:
dsr/dsragent.cc:787: error: â€˜XmitFlowFailureCallbackâ€™ was not declared
in this scope
dsr/dsragent.cc: In member function â€˜void
DSRAgent::sendOutPacketWithRoute(SRPacket&, bool, Time)â€™:
dsr/dsragent.cc:1344: error: â€˜XmitFailureCallbackâ€™ was not declared in
this scope
dsr/dsragent.cc:1345: error: â€˜XmitFlowFailureCallbackâ€™ was not declared
in this scope
dsr/dsragent.cc:1362: error: â€˜XmitFailureCallbackâ€™ was not declared in
this scope
make: ** [dsr/dsragent.o] Erro 1
```

Solução:
Falta a linha "Time send_timeout = SEND_TIMEOUT; // (sec) how long a packet can live in sendbuf" depois da linha "Time rt_rq_max_period = 10.0;   // (sec) maximum time between rt reqs"

Passar as linhas

```
void
XmitFailureCallback(Packet *pkt, void *data)
{
  DSRAgent *agent = (DSRAgent *)data; // cast of trust
  agent->xmitFailed(pkt);
}

void
XmitFlowFailureCallback(Packet *pkt, void *data)
{
  DSRAgent *agent = (DSRAgent *)data;
  agent->xmitFlowFailed(pkt);
}
```

Para antes do

```
void
SendBufferTimer::expire(Event *)
{
  a_->sendBufferCheck();
  resched(BUFFER_CHECK + BUFFER_CHECK * Random::uniform(1.0));
}
```

Alterar a linha "if (Scheduler::instance().clock() - send_buf[c].t > SEND_TIMEOUT) {" por "if (Scheduler::instance().clock() - send_buf[c].t > send_timeout) {"

9º erro:

```
diffusion/diffusion.cc: In member function â€˜void
DiffusionAgent::MACprepare(Packet*, nsaddr_t, int, bool)â€™:
diffusion/diffusion.cc:404: error: â€˜XmitFailedCallbackâ€™ was not
declared in this scope
make: ** [diffusion/diffusion.o] Erro 1
```

Solução:
Recortar a função:

```
void XmitFailedCallback(Packet *pkt, void *data)
{
  DiffusionAgent *agent = (DiffusionAgent *)data;  // cast of trust
  agent->xmitFailed(pkt);
}
```

```
Que esta abaixo da função "void DiffusionAgent::MACsend(Packet *pkt, Time
delay)"
e colar em cima de "char *MsgStr[]= {"", "INTEREST", "DATA", "DATA_READY",
"DATA_REQUEST","

10º erro:
diffusion/omni_mcast.cc: In member function â€˜void
OmniMcastAgent::MACprepare(Packet*, nsaddr_t, unsigned int, bool)â€™:
diffusion/omni_mcast.cc:367: error: â€˜OmniMcastXmitFailedCallbackâ€™ was
not declared in this scope
make: ** [diffusion/omni_mcast.o] Erro 1

Solução:
Recortar a função:
void OmniMcastXmitFailedCallback(Packet *pkt, void *data)
{
  OmniMcastAgent *agent = (OmniMcastAgent *)data;  // cast of trust
  agent->xmitFailed(pkt);
}
Que esta abaixo da função "void OmniMcastAgent::MACsend(Packet *pkt, Time
delay)"
e colar em cima de "static class OmniMcastClass : public TclClass {"

11º erro:
tcp/tcp-sack-rh.cc:68: error: extra qualification â€˜SackRHTcpAgent::â€™ on
member â€˜newackâ€™
make: ** [tcp/tcp-sack-rh.o] Erro 1

Solução:
Alterar a linha "virtual void SackRHTcpAgent::newack(Packet* pkt);" para
"virtual void newack(Packet* pkt);"

12º erro:
linkstate/ls.h: In member function â€˜void LsList<_Tp>::eraseAll()â€™:
linkstate/ls.h:89: error: there are no arguments to â€˜beginâ€™ that depend
on a template parameter, so a declaration of â€˜beginâ€™ must be available
linkstate/ls.h:89: error: (if you use â€˜-fpermissiveâ€™, G++ will accept
your code, but allowing the use of an undeclared name is deprecated)
linkstate/ls.h:89: error: there are no arguments to â€˜endâ€™ that depend
on a template parameter, so a declaration of â€˜endâ€™ must be available
linkstate/ls.h: In member function â€˜void LsMap<Key, T>::eraseAll()â€™:
linkstate/ls.h:112: error: there are no arguments to â€˜beginâ€™ that
depend on a template parameter, so a declaration of â€˜beginâ€™ must be
available
linkstate/ls.h:112: error: there are no arguments to â€˜endâ€™ that depend
on a template parameter, so a declaration of â€˜endâ€™ must be available
make: ** [linkstate/ls.o] Erro 1

Solução:
Alterar a linha "LS_MSG_LSM = 5," para "LS_MSG_LSM = 5"
Alterar a linha "baseList::erase(begin(), end());" para
"baseList::erase(baseList::begin(), baseList::end());"
Alterar a linha "void eraseAll() { erase(begin(), end()); }" para "void
eraseAll() { erase(baseMap::begin(), baseMap::end()); }"

13º erro:
pgm/pgm-agent.cc:278: error: extra qualification â€˜PgmAgent::â€™ on member
â€˜trace_eventâ€™
make: ** [pgm/pgm-agent.o] Erro 1

Solução:
```

Alterar a linha "void PgmAgent::trace_event(char *evType, double evTime);" para "void trace_event(char *evType, double evTime);"

14º erro:
pgm/pgm-sender.cc:160: error: extra qualification â€˜PgmSender::â€™ on member â€˜trace_eventâ€™
make: ** [pgm/pgm-sender.o] Erro 1

Solução:
Alterar a linha "void PgmSender::trace_event(char *evType, nsaddr_t daddr, double evTime);" para "void trace_event(char *evType, nsaddr_t daddr, double evTime);"

15º erro:
pgm/pgm-receiver.cc:157: error: extra qualification â€˜PgmReceiver::â€™ on member â€˜trace_eventâ€™
make: ** [pgm/pgm-receiver.o] Erro 1

Solução:
Alterar a linha "void PgmReceiver::trace_event(char *evType, double evTime);" para "void trace_event(char *evType, double evTime);"

16º erro:
diffusion3/lib/nr/nr.hh:207: error: an explicit specialization must be preceded by â€˜template <>â€™
diffusion3/lib/nr/nr.hh:222: error: an explicit specialization must be preceded by â€˜template <>â€™
make: ** [diffusion3/lib/nr/nr.o] Erro 1

Solução:
Incluir a linha "virtual ~NR () {}" entre "typedef long handle;" e "class Callback {"
Incluir a linha "virtual ~Callback () {}" entre "class Callback {
public:" e "virtual void recv(NRAttrVec *data, handle h) = 0;"
Incluir a linha "template <>" entre "// string specialization" e "class NRSimpleAttribute<char *>: public NRAttribute {"
Incluir a linha "template <>" entre "// blob specialization" e "class NRSimpleAttribute<void *>: public NRAttribute {"

17º erro:
./diffusion3/diffusion/diffusion.hh:103: error: expected `)' before â€˜*â€™ token
make: ** [diffusion3/ns/diffagent.o] Erro 1

Solução:
Incluir a linha "class DiffRoutingAgent;" depois de "class NeighborEntry;"

18º erro:
/usr/include/c++/4.1.3/bits/vector.tcc:452: error: expected unqualified-id before â€˜(â€™ token
make: ** [diffusion3/ns/difftimer.o] Erro 1

Solução:
Alterar a informaÃ§Ã£o "std::max(__old_size, __n);" por "max(__old_size, __n);

19º erro:
/usr/include/c++/4.1.3/bits/stl_bvector.h:897: error: expected unqualified-id before â€˜(â€™ token
make: ** [diffusion3/ns/difftimer.o] Erro 1

Solução:
Alterar a informaÃ§Ã£o "std::max(__old_size, __n);" por "max(__old_size, __n);

Após a instalação no NS-2.26, foram realizadas as alterações conforme documentação do Plugin para o OM, porém ao tentar complilar o NS com o plugin ocorreram os seguinte problemas e soluções

1º erro:
./mobile/god.h:88: error: extra qualification â€˜vector::â€™ on member â€˜operator=â€™
./mobile/god.h:93: error: extra qualification â€˜vector::â€™ on member â€˜operator+=â€™
./mobile/god.h:98: error: extra qualification â€˜vector::â€™ on member â€˜operator==â€™
./mobile/god.h:101: error: extra qualification â€˜vector::â€™ on member â€˜operator!=â€™make: *** [src_rtg/sragent.o] Error 1
Ns make failed!
See http://www.isi.edu/nsnam/ns/ns-problems.html for problems

Solução:
Utilizar o Patch que esta disponivel no caminho http://www.isi.edu/nsnam/dist/ns-2.1b7-god.patch
Alterar todas os "vector" por "Vector"

2º erro:
./mobile/god.h:88: error: extra qualification â€˜vector::â€™ on member â€˜operator=â€™
./mobile/god.h:93: error: extra qualification â€˜vector::â€™ on member â€˜operator+=â€™
./mobile/god.h:98: error: extra qualification â€˜vector::â€™ on member â€˜operator==â€™
./mobile/god.h:101: error: extra qualification â€˜vector::â€™ on member â€˜operator!=â€™make: *** [src_rtg/sragent.o] Error 1
Ns make failed!
See http://www.isi.edu/nsnam/ns/ns-problems.html for problems

Solução:
Utilizar o patch disponível no caminho http://mailman.isi.edu/pipermail/ns-users/2007-January/058412.html
Retirar o "Vector" da frente dos "Operador"
Exemplo:
alterar inline void vector::operator=(const vector a) {
por inline void operator=(const vector a) {

3º erro:
mobile/god.cc: In member function â€˜bool God::IsNeighbor(int, int)â€™:
mobile/god.cc:476: error: â€˜vectorâ€™ was not declared in this scope
mobile/god.cc:476: error: expected `;' before â€˜aâ€™
mobile/god.cc:477: error: expected `;' before â€˜bâ€™
mobile/god.cc:478: error: expected `;' before â€˜dâ€™
mobile/god.cc:480: error: â€˜dâ€™ was not declared in this scope
make: ** [mobile/god.o] Erro 1

Solução:
Alterar todas os "vector" por "Vector"

4º erro:
./mobile/obstacle.h:14: error: extra qualification â€˜Obstacle::â€™ on member â€˜getPointâ€™

```
mobile/tworayground.cc: In member function â€˜virtual double
TwoRayGround::Pr(PacketStamp*, PacketStamp*, WirelessPhy*)â€™:
mobile/tworayground.cc:62: error: too few arguments to function â€˜int
intersect(double, double, double, double, double, double, double, double,
double*, double*)â€™
mobile/tworayground.cc:160: error: at this point in file
mobile/tworayground.cc:180: error: â€˜uniformâ€™ was not declared in this
scope
mobile/tworayground.cc:185: error: â€˜uniformâ€™ was not declared in this
scope
mobile/tworayground.cc:186: error: â€˜uniformâ€™ was not declared in this
scope
make: ** [mobile/tworayground.o] Erro 1
```

Solução:
Alterar a linha "int Obstacle::getPoint(int i, double *ax, double *ay);"
para "int getPoint(int i, double *ax, double *ay);"
Deletar a linha "int intersect(double p1x, double p1y, double p2x, double
p2y, double nodeX, double nodeY, double destX, double destY,double *a,
double *b);" no inicio do arquivo
Copiar a funÃ§Ã£o:
//Function to find the intersection between two lines

```
int
intersect(double p1x, double p1y, double p2x, double p2y, double nodeX,
double nodeY, double destX, double destY)
{
        double den, ua, ub, intx, inty;

        den = ((p2y - p1y)*(nodeX - destX) - (p2x - p1x)*(nodeY - destY));
        if (den != 0)
        {
                ua = (((p2x - p1x)*(destY - p1y) - (p2y - p1y)*(destX -
p1x)) / (den));
                ub = (((nodeX - destX)*(destY - p1y) - (nodeY -
destY)*(destX - p1x)) / (den));
                if((ua != 0) && (ub != 0))
                {
                        if((ua > 0) && (ua < 1) && (ub > 0) && (ub < 1))
                        {
                                //Find the intersecting points
                                intx = destX + (ua * (nodeX - destX));
                                inty = destY + (ua * (nodeY - destY));
#if DEBUG > 3
                                fprintf(stdout,"Found at %lf,%lf From :
%lf,%lf To : %lf,%lf\n",
                                        intx, inty, nodeX, nodeY, destX,
destY);
#endif
                                return 1;
                        }
                }

        }

        return 0;
}
```

Que esta acima da linha "//random number generator from 0 to 1"
Para Antes do comando "double TwoRayGround::TwoRay(double Pt, double Gt,
double Gr, double ht, double hr, double L, double d)"

```
Recortar a função:
//random number generator from 0 to 1
double
uniform()
{
  double a;
  a = (double)(rand()%1000000)/1000000;
  return a;
}
```

e colar depois do fim da função "intersect(double p1x, double p1y, double p2x, double p2y, double nodeX, double nodeY, double destX, double destY)"

```
5º erro:
/home/cristiane/ns-allinone-2.26/lib/libtcl8.3.a(tclUnixPipe.o): In
function `TclpCreateTempFile':
tclUnixPipe.c:(.text+0x7ab): warning: the use of `tmpnam' is dangerous,
better use `mkstemp'
mobile/tworayground.o: In function `TwoRayGround::Pr(PacketStamp*,
PacketStamp*, WirelessPhy*)':
tworayground.cc:(.text+0x491): undefined reference to
`Obstacle::getPoint(int, double*, double*)'
tworayground.cc:(.text+0x4d9): undefined reference to
`Obstacle::getPoint(int, double*, double*)'
mobile/tworayground.o: In function `TwoRayGround::TwoRayGround()':
tworayground.cc:(.text+0xa67): undefined reference to
`Obstacle::Obstacle()'
tworayground.cc:(.text+0xba0): undefined reference to
`Obstacle::init(double, double, double, double, double, double, double,
double)'
mobile/tworayground.o: In function `TwoRayGround::TwoRayGround()':
tworayground.cc:(.text+0xcc9): undefined reference to
`Obstacle::Obstacle()'
tworayground.cc:(.text+0xe02): undefined reference to
`Obstacle::init(double, double, double, double, double, double, double,
double)'
collect2: ld returned 1 exit status
make: ** [ns] Erro 1
```

PROBLEMA SEM SOLUÇÃO

APÊNDICE B – Arquivo de Tráfego Gerado com cbrgen.tcl

```
#
# nodes: 25, max conn: 12, send rate: 0.0, seed: 0
#
#
# 3 connecting to 4 at time 5.0
#
set tcp_(0) [$ns create-connection  TCP $node_(3) TCPSink $node_(4) 0]
$tcp_(0) set window_ 32
$tcp_(0) set packetSize_ 512
set ftp_(0) [$tcp_(0) attach-source FTP]
$ns at 5.0 "$ftp_(0) start"
#
# 4 connecting to 5 at time 5.0
#
set tcp_(1) [$ns create-connection  TCP $node_(4) TCPSink $node_(5) 0]
$tcp_(1) set window_ 32
$tcp_(1) set packetSize_ 512
set ftp_(1) [$tcp_(1) attach-source FTP]
$ns at 5.0 "$ftp_(1) start"
#
# 7 connecting to 8 at time 5.0
#
set tcp_(2) [$ns create-connection  TCP $node_(7) TCPSink $node_(8) 0]
$tcp_(2) set window_ 32
$tcp_(2) set packetSize_ 512
set ftp_(2) [$tcp_(2) attach-source FTP]
$ns at 5.0 "$ftp_(2) start"
#
# 8 connecting to 9 at time 5.0
#
set tcp_(3) [$ns create-connection  TCP $node_(8) TCPSink $node_(9) 0]
$tcp_(3) set window_ 32
$tcp_(3) set packetSize_ 512
set ftp_(3) [$tcp_(3) attach-source FTP]
$ns at 5.0 "$ftp_(3) start"
#
# 10 connecting to 11 at time 5.0
#
set tcp_(4) [$ns create-connection  TCP $node_(10) TCPSink $node_(11) 0]
$tcp_(4) set window_ 32
$tcp_(4) set packetSize_ 512
set ftp_(4) [$tcp_(4) attach-source FTP]
$ns at 5.0 "$ftp_(4) start"
#
# 10 connecting to 12 at time 5.0
#
set tcp_(5) [$ns create-connection  TCP $node_(10) TCPSink $node_(12) 0]
$tcp_(5) set window_ 32
$tcp_(5) set packetSize_ 512
set ftp_(5) [$tcp_(5) attach-source FTP]
$ns at 5.0 "$ftp_(5) start"
#
# 11 connecting to 12 at time 5.0
#
set tcp_(6) [$ns create-connection  TCP $node_(11) TCPSink $node_(12) 0]
$tcp_(6) set window_ 32
$tcp_(6) set packetSize_ 512
set ftp_(6) [$tcp_(6) attach-source FTP]
```

```
$ns at 5.0 "$ftp_(6) start"
#
# 12 connecting to 13 at time 5.0
#
set tcp_(7) [$ns create-connection  TCP $node_(12) TCPSink $node_(13) 0]
$tcp_(7) set window_ 32
$tcp_(7) set packetSize_ 512
set ftp_(7) [$tcp_(7) attach-source FTP]
$ns at 5.0 "$ftp_(7) start"
#
# 17 connecting to 18 at time 5.0
#
set tcp_(8) [$ns create-connection  TCP $node_(17) TCPSink $node_(18) 0]
$tcp_(8) set window_ 32
$tcp_(8) set packetSize_ 512
set ftp_(8) [$tcp_(8) attach-source FTP]
$ns at 5.0 "$ftp_(8) start"
#
# 17 connecting to 19 at time 5.0
#
set tcp_(9) [$ns create-connection  TCP $node_(17) TCPSink $node_(19) 0]
$tcp_(9) set window_ 32
$tcp_(9) set packetSize_ 512
set ftp_(9) [$tcp_(9) attach-source FTP]
$ns at 5.0 "$ftp_(9) start"
#
# 21 connecting to 22 at time 5.0
#
set tcp_(10) [$ns create-connection  TCP $node_(21) TCPSink $node_(22) 0]
$tcp_(10) set window_ 32
$tcp_(10) set packetSize_ 512
set ftp_(10) [$tcp_(10) attach-source FTP]
$ns at 5.0 "$ftp_(10) start"
#
# 21 connecting to 23 at time 5.0
#
set tcp_(11) [$ns create-connection  TCP $node_(21) TCPSink $node_(23) 0]
$tcp_(11) set window_ 32
$tcp_(11) set packetSize_ 512
set ftp_(11) [$tcp_(11) attach-source FTP]
$ns at 5.0 "$ftp_(11) start"
#
#Total sources/connections: 9/12
#
```

APÊNDICE C – Exemplo de Arquivo de Mobilidade do UPF

```
$node_(0) set X_ 265.441
$node_(0) set Y_ 254.067
$node_(0) set Z_ 0
$ns at 0 "$node_(0) setdest 267.091 0.0001 1.53962"
$node_(1) set X_ 276.795
$node_(1) set Y_ 134.606
$node_(1) set Z_ 0
$ns at 0 "$node_(1) setdest 276.795 0.0001 1.35494"
$node_(2) set X_ 267.582
$node_(2) set Y_ 19.2311
$node_(2) set Z_ 0
$ns at 0 "$node_(2) setdest 318 228.248 1.63976"
$node_(3) set X_ 172.596
$node_(3) set Y_ 114.056
$node_(3) set Z_ 0
$ns at 0 "$node_(3) setdest 318 172.596 1.40676"
$node_(4) set X_ 61.6027
$node_(4) set Y_ 164.462
$node_(4) set Z_ 0
$ns at 0 "$node_(4) setdest 0.0001 163.569 1.53152"
$node_(5) set X_ 170.11
$node_(5) set Y_ 263.141
$node_(5) set Z_ 0
$ns at 0 "$node_(5) setdest 318 170.653 1.51433"
$node_(6) set X_ 218.183
$node_(6) set Y_ 209.587
$node_(6) set Z_ 0
$ns at 0 "$node_(6) setdest 0.0001 140.968 1.51512"
$node_(7) set X_ 80.5885
$node_(7) set Y_ 163.633
$node_(7) set Z_ 0
$ns at 0 "$node_(7) setdest 0.0001 162.465 1.62386"
$node_(8) set X_ 262.99
$node_(8) set Y_ 283.923
$node_(8) set Z_ 0
$ns at 0 "$node_(8) setdest 264.317 0.0001 1.64941"
$node_(9) set X_ 76.8824
$node_(9) set Y_ 31.838
$node_(9) set Z_ 0
$ns at 0 "$node_(9) setdest 77.2615 0.0001 1.47906"
$node_(10) set X_ 81.8896
$node_(10) set Y_ 134.625
$node_(10) set Z_ 0
$ns at 0 "$node_(10) setdest 318 83.2089 1.58857"
$node_(11) set X_ 111.422
$node_(11) set Y_ 167.486
$node_(11) set Z_ 0
$ns at 0 "$node_(11) setdest 350 172.456 1.26626"
$node_(12) set X_ 71.5394
$node_(12) set Y_ 56.6802
$node_(12) set Z_ 0
$ns at 0 "$node_(12) setdest 72.2142 0.0001 1.42538"
$node_(13) set X_ 167.546
$node_(13) set Y_ 58.5997
$node_(13) set Z_ 0
$ns at 0 "$node_(13) setdest 167.546 0.0001 1.52772"
$node_(14) set X_ 160.615
$node_(14) set Y_ 99.3437
```

```
$node_(14) set Z_ 0
$ns at 0 "$node_(14) setdest 0.0001 101.789 1.48181"
$node_(15) set X_ 81.2493
$node_(15) set Y_ 160.033
$node_(15) set Z_ 0
$ns at 0 "$node_(15) setdest 84.8821 0.0001 1.53698"
$node_(16) set X_ 76.9971
$node_(16) set Y_ 209.637
$node_(16) set Z_ 0
$ns at 0 "$node_(16) setdest 37.5552 0.0001 1.59303"
$node_(17) set X_ 47.1664
$node_(17) set Y_ 157.445
$node_(17) set Z_ 0
$ns at 0 "$node_(17) setdest 318 343.756 1.49786"
$node_(18) set X_ 76.5738
$node_(18) set Y_ 199.806
$node_(18) set Z_ 0
$ns at 0 "$node_(18) setdest 76.5738 0.0001 1.76566"
$node_(19) set X_ 175.328
$node_(19) set Y_ 154.736
$node_(19) set Z_ 0
$ns at 0 "$node_(19) setdest 175.328 0.0001 1.56295"
$node_(20) set X_ 264.473
$node_(20) set Y_ 52.9224
$node_(20) set Z_ 0
$ns at 0 "$node_(20) setdest 264.473 0.0001 1.56061"
$node_(21) set X_ 188.473
$node_(21) set Y_ 199.175
$node_(21) set Z_ 0
$ns at 0 "$node_(21) setdest 318 96.8692 1.42142"
$node_(22) set X_ 222.115
$node_(22) set Y_ 209.391
$node_(22) set Z_ 0
$ns at 0 "$node_(22) setdest 350 172.285 1.34095"
$node_(23) set X_ 167.328
$node_(23) set Y_ 253.456
$node_(23) set Z_ 0
$ns at 0 "$node_(23) setdest 318 167.967 1.41405"
$node_(24) set X_ 186.466
$node_(24) set Y_ 46.2683
$node_(24) set Z_ 0
$ns at 0 "$node_(24) setdest 75.7116 0.0001 1.31529"
$ns at 4.2 "$node_(24) setdest 0.0001 67.3269 1.31529"
$ns at 4.4 "$node_(24) setdest 0.0001 50.5671 1.31529"
$ns at 4.6 "$node_(24) setdest 0.0001 30.8927 1.31529"
$ns at 4.8 "$node_(24) setdest 0.0001 6.40838 1.31529"
$ns at 5 "$node_(24) setdest 179.493 0.0001 1.31529"
$ns at 6.6 "$node_(10) setdest 0.0001 193.541 1.58857"
$ns at 6.8 "$node_(10) setdest 0.0001 289.466 1.58857"
$ns at 8 "$node_(21) setdest 0.0001 283.048 1.42142"
$ns at 8.2 "$node_(10) setdest 318 81.8217 1.58857"
$ns at 8.2 "$node_(21) setdest 0.0001 295.257 1.42142"
$ns at 8.4 "$node_(21) setdest 0.0001 309.533 1.42142"
$ns at 8.6 "$node_(21) setdest 318 13.3249 1.42142"
$ns at 8.8 "$node_(21) setdest 318 39.0446 1.42142"
$ns at 9 "$node_(21) setdest 318 62.8409 1.42142"
$ns at 9.2 "$node_(21) setdest 318 85.5221 1.42142"
```

APÊNDICE D – Exemplo de Arquivo de Mobilidade do UDEL

```
set god_ [God instance]
$node_(0) set Z_ 3.50
$node_(0) set Y_ 282.67
$node_(0) set X_ 132.00
$node_(1) set Z_ 1500067.50
$node_(1) set Y_ 1500000.00
$node_(1) set X_ 1500000.00
$node_(2) set Z_ 10.50
$node_(2) set Y_ 80.56
$node_(2) set X_ 45.16
$node_(3) set Z_ 1500067.50
$node_(3) set Y_ 1500000.00
$node_(3) set X_ 1500000.00
$node_(4) set Z_ 0.00
$node_(4) set Y_ 114.33
$node_(4) set X_ 63.14
$node_(5) set Z_ 17.50
$node_(5) set Y_ 270.40
$node_(5) set X_ 268.00
$node_(6) set Z_ 0.00
$node_(6) set Y_ 86.09
$node_(6) set X_ 169.60
$node_(7) set Z_ 0.00
$node_(7) set Y_ 75.43
$node_(7) set X_ 317.72
$node_(8) set Z_ 10.50
$node_(8) set Y_ 80.56
$node_(8) set X_ 45.76
$node_(9) set Z_ 0.00
$node_(9) set Y_ 235.33
$node_(9) set X_ 44.86
$node_(10) set Z_ 1500067.50
$node_(10) set Y_ 1500000.00
$node_(10) set X_ 1500000.00
$node_(11) set Z_ 1500067.50
$node_(11) set Y_ 1500000.00
$node_(11) set X_ 1500000.00
$node_(12) set Z_ 0.00
$node_(12) set Y_ 134.86
$node_(12) set X_ 336.00
$node_(13) set Z_ 1500067.50
$node_(13) set Y_ 1500000.00
$node_(13) set X_ 1500000.00
$node_(14) set Z_ 0.00
$node_(14) set Y_ 35.18
$node_(14) set X_ 155.20
$node_(15) set Z_ 0.00
$node_(15) set Y_ 92.13
$node_(15) set X_ 92.19
$node_(16) set Z_ 0.00
$node_(16) set Y_ 63.67
$node_(16) set X_ 210.78
$node_(17) set Z_ 1500067.50
$node_(17) set Y_ 1500000.00
$node_(17) set X_ 1500000.00
$node_(18) set Z_ 0.00
$node_(18) set Y_ 84.11
$node_(18) set X_ 92.55
```

```
$node_(19) set Z_ 1500067.50
$node_(19) set Y_ 1500000.00
$node_(19) set X_ 1500000.00
$node_(20) set Z_ 1500067.50
$node_(20) set Y_ 1500000.00
$node_(20) set X_ 1500000.00
$node_(21) set Z_ 0.00
$node_(21) set Y_ 92.27
$node_(21) set X_ 95.18
$node_(22) set Z_ 0.00
$node_(22) set Y_ 84.24
$node_(22) set X_ 95.54
$node_(23) set Z_ 1500067.50
$node_(23) set Y_ 1500000.00
$node_(23) set X_ 1500000.00
$node_(24) set Z_ 0.00
$node_(24) set Y_ 223.00
$node_(24) set X_ 275.25
$ns at 0.00 "$node_(2) setdest-udelmobility 45.16 78.97 10.50 1.587444"
$ns at 0.00 "$node_(8) setdest-udelmobility 45.76 79.69 10.50 0.866217"
$ns at 0.00 "$node_(17) setdest-udelmobility 199.13 304.40 0.00
2597824.485074"
$ns at 0.00 "$node_(20) setdest-udelmobility 196.06 306.88 0.00
2597824.824665"
$ns at 1.00 "$node_(2) setdest-udelmobility 45.16 77.38 10.50 1.587444"
$ns at 1.00 "$node_(8) setdest-udelmobility 45.76 78.82 10.50 0.866217"
$ns at 1.00 "$node_(17) setdest-udelmobility 199.44 293.19 0.00 11.206705"
$ns at 1.00 "$node_(19) setdest-udelmobility 198.89 313.43 0.00
2597819.413842"
$ns at 1.00 "$node_(20) setdest-udelmobility 196.29 298.65 0.00 8.227338"
$ns at 2.00 "$node_(2) setdest-udelmobility 45.16 75.79 10.50 1.587444"
$ns at 2.00 "$node_(8) setdest-udelmobility 45.76 77.96 10.50 0.866217"
$ns at 2.00 "$node_(17) setdest-udelmobility 199.74 281.99 0.00 11.206705"
$ns at 2.00 "$node_(19) setdest-udelmobility 199.13 304.47 0.00 8.963891"
$ns at 2.00 "$node_(20) setdest-udelmobility 196.51 290.43 0.00 8.227338"
$ns at 3.00 "$node_(2) setdest-udelmobility 45.16 74.21 10.50 1.587444"
$ns at 3.00 "$node_(8) setdest-udelmobility 45.76 77.09 10.50 0.866217"
```