UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Rafael Ogayar Gomes

Análise de Desempenho de Algoritmos de Roteamento de Nodos em Uma Rede de Sensores Sem Fio Aplicada a Cidades Inteligentes

Santa Maria, RS, Brasil

2018

Rafael Ogayar Gomes

Análise de Desempenho de Algoritmos de Roteamento de Nodos em Uma Rede de Sensores Sem Fio Aplicada a Cidades Inteligentes

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Mestre em Ciência da Computação**

Orientador: Prof. Dr. João Baptista dos Santos Martins

Santa Maria, RS, Brasil

2018

OGAYAR GOMES, RAFAEL
Análise de Desempenho de Algoritmos de Roteamento de Nodos em Uma Rede de Sensores Sem Fio Aplicada a Cidades Inteligentes / RAFAEL OGAYAR GOMES.- 2018.
81 p.; 30 cm

Orientador: João Baptista dos Santos Martins
Dissertação (mestrado) - Universidade Federal de Santa Maria, Centro de Tecnologia, Programa de Pós-Graduação em Informática, RS, 2018

1. REDES DE SENSORES SEM FIO 2. INTERNET DAS COISAS I. dos Santos Martins, João Baptista II. Título.

Sistema de geração automática de ficha catalográfica da UFSM. Dados fornecidos pelo autor(a). Sob supervisão da Direção da Divisão de Processos Técnicos da Biblioteca Central. Bibliotecária responsável Paula Schoenfeldt Patta CRB 10/1728.

Rafael Ogayar Gomes

Análise de Desempenho de Algoritmos de Roteamento de Nodos em Uma Rede de Sensores Sem Fio Aplicada a Cidades Inteligentes

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Mestre em Ciência da Computação.**

Aprovado em 31 de agosto de 2018.

João Baptista dos Santos Martins, Dr. (UFSM)
(Presidente/Orientador)

Mateus Beck Rutzig, Dr. (UFSM)

Sergio Jose Melo De Almeida, Dr. (UCPEL)

Santa Maria, RS

2018

AGRADECIMENTO

Início meus agradecimentos aos meus pais, especialmente a minha mãe, meu infinito agradecimento que sempre acreditou em minha capacidade e não mediu esforços em minha trajetória acadêmica, também a minha namorada pelo incentivo, ajuda e apoio em todo esse tempo. Isso só me fortaleceu e me fez tentar, não ser o melhor, mas a fazer o melhor de mim.

A presente dissertação de mestrado não poderia chegar a bom porto sem o precioso apoio de várias pessoas. Em primeiro lugar, não posso deixar de agradecer ao meu orientador, Professor Doutor João Baptista dos Santos Martins, por toda a paciência, empenho e sentido prático com que sempre me orientou neste trabalho e em todos aqueles que realizei. Muito obrigado por me ter corrigido quando necessário sem nunca me desmotivar. Agradeço também ao Professor Mestre Alessandro André Mainardi de Oliveira, que desde o início me apoiou em todas as etapas até chegar aqui.

Desejo igualmente agradecer a todos os meus colegas do Mestrado em Computação. Por último, quero agradecer à minha família e amigos pelo apoio incondicional que me deram ao longo da elaboração deste trabalho.

RESUMO

Análise de Desempenho de Algoritmos de Roteamento de Nodos em Uma Rede de Sensores Sem Fio Aplicada a Cidades Inteligentes

AUTOR: Rafael Ogayar Gomes

ORIENTADOR: João Baptista dos Santos Martins

Com o avanço de pesquisas envolvendo Cidades Inteligentes, grande parte dessas estão focadas na iteração entre sensores e atuadores, no entanto, pode se citar o problema com roteamento de pacotes que vem sendo estudado, pelo fato da necessidade de se obter uma melhor comunicação entre todos os nodos e também sem a perda de pacotes, aspectos como energia, alcance e dispersão do sinal são pontos cruciais neste assunto. Neste trabalho foram feitos estudos e análises de alguns algoritmos de roteamento (OADV, OSDV, DSR) os quais são os mais utilizados para aplicações, pelo fato de trabalharem de forma satisfatória com a mobilidade dos nodos. Foram feitas análises e comprovado a prospecção de cada algoritmo em diferentes cenários, esses serão mostrados durante o desenvolvimento deste trabalho, chegando assim a conclusão de qual mais adequado a ser utilizado em diferente situação, local e especificação determinada, como por exemplo: campo aberto, cidades, casas, edifícios.

Palavras-Chave: Cidades Inteligentes, Algoritmos de Roteamento, AODV, OSDV, DSV, Consumo de Energia, Dispersão de Sinal.

ABSTRACT

Performance Analysis of Node Routing Algorithms in Wireless Sensors Network Applied to Smart Cities

AUTHOR: Rafael Ogayar Gomes

ADVISORS: João Baptista dos Santos Martins

With the advancement of research involving Smart Cities, most of these are focused on the iteration between sensors and actuators, however, we can cite the problem with packet routing that has been studied, due to the need to obtain better communication between all the nodes and also without the loss of packets, aspects such as energy, reach and signal dispersion are crucial points in this subject. In this work, we performed studies and analysis of some routing algorithms (OADV, OSDV, DSR) which are the most used for applications, because they work in a satisfactory way with node mobility. The analysis of each algorithm was carried out in the different scenarios, and these will be shown during the development of this work, arriving at the conclusion of which more suitable to be used in different situation, location and specific specification, such as open field, cities, houses, buildings.

Keywords: Smart Cities, Routing Algorithms, AODV, OSDV, DSV, Power Consumption, Signal Dispersion.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

LEACH	Low Energy Adaptive Clustering Hierarchy
TEEN	Threshold-sensitive Energy Efficient Protocol
BCDCP	Station Controlled Dynamic Protocol
MECN	Minimum Energy Communication Network
NS-2	Simulador de Códigos de Roteamento
DF	Direct Diffusion
AODV	Ad Hoc On-Demand Distance Vector
DSDV	Destination-Sequenced Distance Vector
DSR	Dynamic Source Routing

Anexo

Sumário

1. INTRODUÇÃO

Cidades Inteligentes são definidas pelo fato de ambientes serem integrados de forma interativa utilizando a comunicação de pontos, formados por sensores ou atuadores, esses pontos exigem um alcance de sinal e consumo de energia eficiente para melhor dispersão de dados (STEVENTON e WRIGHT, 2006). Partindo destas afirmações, este trabalho se propõe a trazer contribuições para a melhor escolha de roteamento para serem utilizados em Cidades Inteligentes.

Existem diversos algoritmos de roteamento, uns privilegiam melhor gerenciamento de pacotes e outros um consumo de energia reduzido, dependendo da aplicabilidade. No nosso caso serão analisados os seguintes algoritmos: AODV, OSDV e DSR, os quais satisfazem as condições de estudo de gerenciamento de pacotes e consumo de energia.

Existem diversos Softwares que realizam a análise e testes do roteamento de pacotes, dentre os mais citados na metodologia temos: NT++, NS-2, NS3, CUPCarbon. Para este trabalho foi escolhido o software NS-2 por este disponibilizar uma plataforma flexível a edição em código fonte, o que auxilia na implementação de casos de testes.

O presente estudo é apresentado da seguinte forma: a seguir são discutidos os aspectos conceituais relacionados a Cidades Inteligentes e Algoritmos de Roteamento. Logo após são apresentadas as interfaces para testes como técnica de visualização e mapeamento de nodos sensores; na sequência são descritos os procedimentos metodológicos aplicados a este estudo, os resultados obtidos, a conclusão e os trabalhos futuros.

1.1. OBJETIVOS

O presente trabalho tem como objetivo analisar e implementar testes de comparação entre algoritmos de roteamento de nodos sensores sem fio, ao qual

agrega a sua escolha adequada a determinado cenário e especificação para futuras aplicações em cidades inteligentes.

1.2. OBJETIVOS ESPECÍFICOS

- Análise sobre dispersão de sinal, consumo de energia e envio de dados sem perda;
- Análise de plataformas de Simulação;
- Análise dos Algoritmo AODV, OSDV, DSR na interface de simulação NS-2;
- Comparação de algoritmos em diferentes casos de uso como parâmetros e obstáculos e consumo de energia.

1.3. JUSTIFICATIVA

Com a difusão em massa da internet das coisas a realidade das cidades inteligentes está cada vez mais presente na vida das pessoas, com isso o tamanho e a complexidade das redes aumentam conjuntamente com o número de dispositivos que começam a fazer parte desse novo modelo de computação (HRIBERNIK et al. 2011).

No estudo de Algoritmos de Roteamentos em Cidades Inteligentes, se observou que o alcance de sinal em determinada região, estava sendo perdido pelo fato de prédios, rodovias e arvores obstruírem o alcance de sinal, fazendo assim com que grande parte dos dados que estavam sendo enviados fossem perdidos chegando assim a conclusão que os dados não são íntegros.

Outro ponto crucial e determinante para a comparação e implementação deste trabalho é o consumo de energia, uma variável considerável para a estabilidade do sistema. No entanto, essas duas variáveis são aspectos que dificilmente são explorados pelo fato de que quanto maior o alcance do sinal maior o consumo de energia e vice versa, neste contexto o trabalho pretende avaliar essas variáveis que

são de suma importância para que um rede de sensores em Cidade Inteligente seja executada de forma satisfatória.

1.4. ORGANIZAÇÃO

A dissertação encontra-se organizada conforme a descrição a seguir.

O capitulo 1 descreve os objetivos e justificativa da dissertação.

O capitulo 2 mostra a revisão bibliográfica sobre cidades inteligentes bem como faz um estudo sobre alguns algoritmos existentes como LEACH, TEEN, APTEEN, BCDCP, MECN, TTDD, MECN, GAF, GEAR, SPIN, DIRECT DIFFUSION, PEDAP, COUGAR, ACQUIRE. Neste Capítulo também é descrito os simuladores de redes de sensores mais comuns, dentre eles estão NS-2, CUP CARBON.

O capitulo 3 mostra e descreve detalhadamente os Protocolos utilizados neste trabalho.

No capítulo 4 são apresentadas as contribuições científicas juntamente com a metodologia utilizada e seus casos de testes.

O capitulo 5 trata dos resultados obtidos com as implementações propostas e análise de dados.

O capitulo 6 chega na conclusão do trabalho proposto juntamente com os trabalhos futuros.

E por fim, no capítulo 7 são apresentadas as Referências Bibliográficas utilizadas na dissertação.

2. REVISÃO BIBLIOGRÁFICA

Nessa sessão serão apresentados de forma geral, aspectos relacionados ao trabalho, sendo eles conceitos sobre redes de sensores, conceitos relacionados a algoritmos de roteamentos, Cidades Inteligentes, Simuladores, entre os específicos utilizados no projeto.

2.1. CIDADES INTELIGENTES

Smarty City é definida como a comunicação que integra (TICs), fazendo com que se gerem ambientes iterativos trazendo comunicação para todo o meio em que se encontra. Uma cidade é considerada Smarty City quando ambientes onde possuem tecnologia da informação ou comunicação passam a ser invisíveis pois são embutidos em artefatos e ambientes do dia a dia (STEVENTON e WRIGHT, 2006).

Muitas pesquisas nos dias de hoje estão focadas em cidades inteligentes pelo fato de ser o futuro, essas pesquisas mostram que em poucos anos inúmeras cidades já vão estar com sensoriamento e atuadores remotos para melhorar a vida dos moradores desses lugares. A Figura 2-1 ilustra a imagem de uma Cidade Inteligente onde podemos observar diversas conexões em todos os lados mostrando que a conectividade entre os pontos e periféricos é possível.

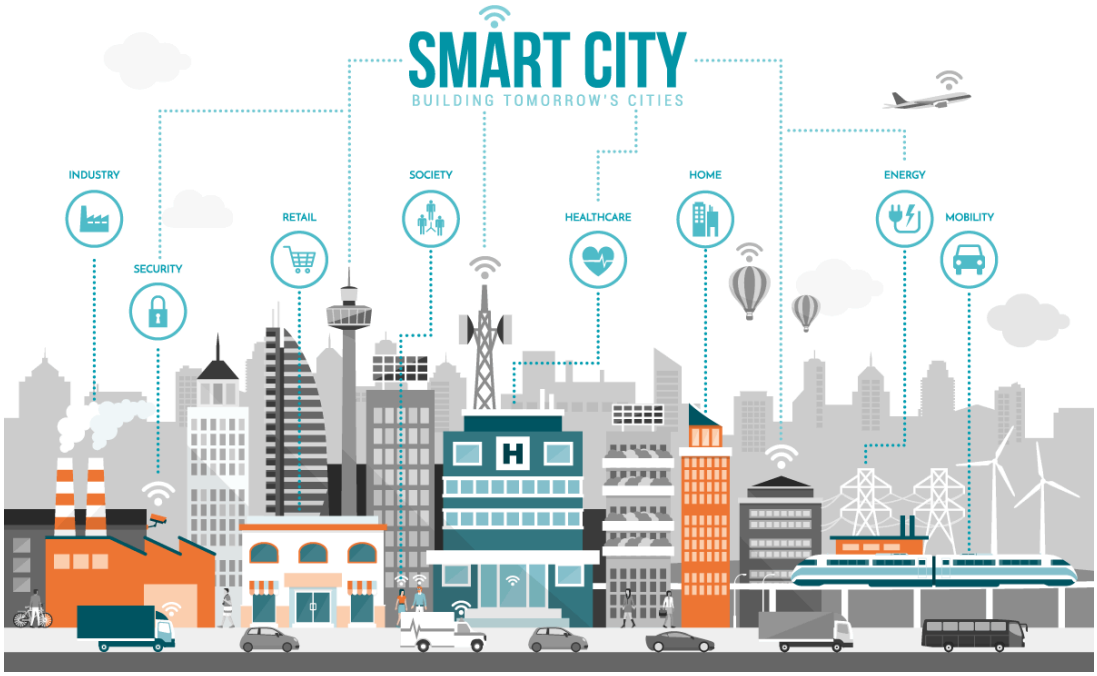
2.2. REDES DE SENSORES SEM FIO

Uma RSSF é definida como dezenas, centenas e até milhares de nodos sensores espalhados em um determinado local (MALLADI; AGRAWAL, 2002), esses nodos tem a capacidade de processar e comunicar, verificando e enviando determinado dado de sua coleta seja ele temperatura, umidade, luminosidade, entre outros.

A pesquisa sobre novas topologias e novos algoritmos vem contribuindo com o avanço dos sensores e os deixando cada vez menores. Na maioria das vezes é

efetuada a comunicação sem fio facilitando assim a sua utilização. Tais sensores também são utilizados em áreas como da agricultura, hospitalar, saúde, controle de fenômenos ambientais, dentre outros. (AKKAYA; YOUNIS, 2005). Na Figura 2-1 é ilustrado o exemplo de comunicação de uma cidade inteligente, conectando seus sensores e atuadores.

Figura 2-1 Exemplo de Cidade Inteligente.



Fonte: (ARC ADVISORY, 2017).

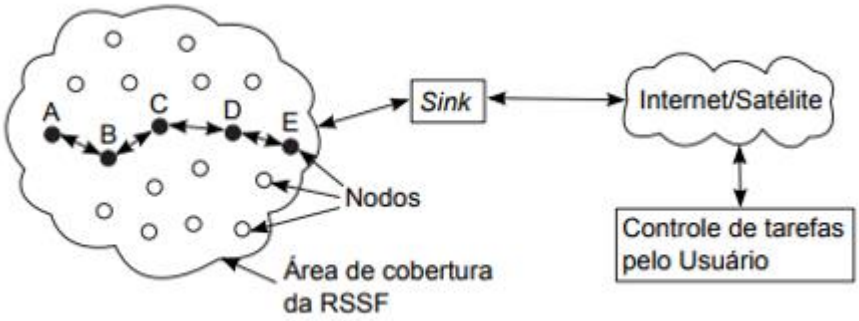
2.2.1. Arquitetura de uma Rede de Sensores sem Fio

A organização clássica de uma rede de sensores pode ser vista na Figura 2-2. Os dados são coletados pelos nodos até a chegada ao ***sink***. Esse dado então é transportado para a internet, que por sua vez, faz com que o usuário possa ter controle dos dados recebidos, executando assim uma determinada função imposta pela central de controle.

A influência de vários fatores nas RSSF gera diversas áreas de pesquisa buscando tentar solucionar diversos fatores tais como: consumo de energia,

verificação de envio e recebimento de dados sem perda, tolerância a falhas nos nodos entre outros que são citados em (AKYILDIZ et al., 2002).

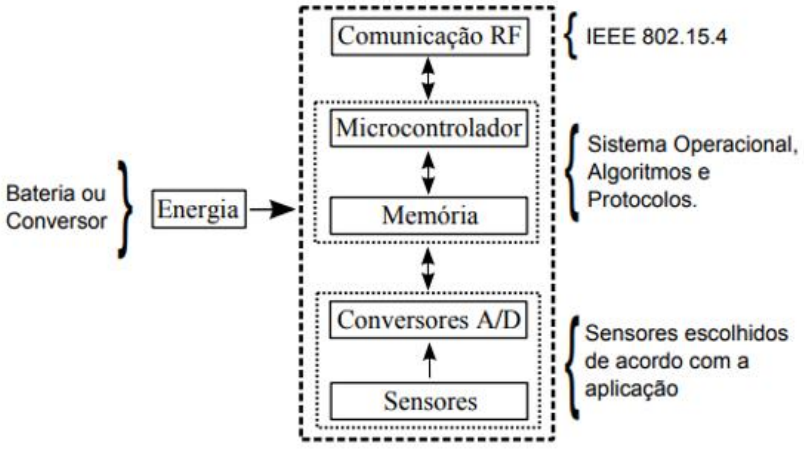
Figura 2-2. Nodos espalhados em uma area determinada



Fonte: adaptado de (AKYILDIZ et al, 2002)

A arquitetura de um nodo sensor é ilustrado na Figura 2-3. Essa arquitetura é definida em 4 partes: sensoriamento, processamento, comunicação e fornecimento de energia, demanda que esses passos sejam executados, com ênfase em pesquisas nas RSSF, que é controle de consumo de cada nodo sensor (PANTAZIS; VERGADOS, 2007).

Figura 2-3. Componentes de um nodo RSSF.



Fonte: (PANTAZIS; VERGADOS, 2007)

2.3. PROTOCOLOS DE ROTEAMENTO EM RSSF

Protocolos de roteamento em RSSF, segundo (WEILIAN et al., 2005), influenciam na capacidade em que um nodo pode fornecer certa informação para o usuário, além de auto gerenciar os nodos sensores em diversas camadas, de modo a completar as tarefas desejadas pelo mesmo.

Promove melhoras de rotas em relação ao trafego dos dados, reduzindo falhas, otimizando a vida útil da rede que irá acarretar em um melhor consumo de energia.

Existem diversas classificações para esses protocolos, sendo as principais descritas a seguir: estruturada em dados, hierárquicos e geográfico.

2.3.1. Protocolo de Roteamento Hierárquico

Para resolver o gargalo existente nos nodos da rede, são estudados protocolos capazes de mantes a eficiência da rede, quando o número de nodos é grande causando latência nas comunicações. O protocolo hierárquico tem como objetivos principais a eficiência do consumo de energia, através da comunicação *multi-hop (multi saltos)*, realizando a fusão de dados para diminuir o número de mensagens enviadas pelo coletor. A seguir apresenta-se a descrição dos principais roteamentos hierárquicos.

2.3.1.1. LEACH

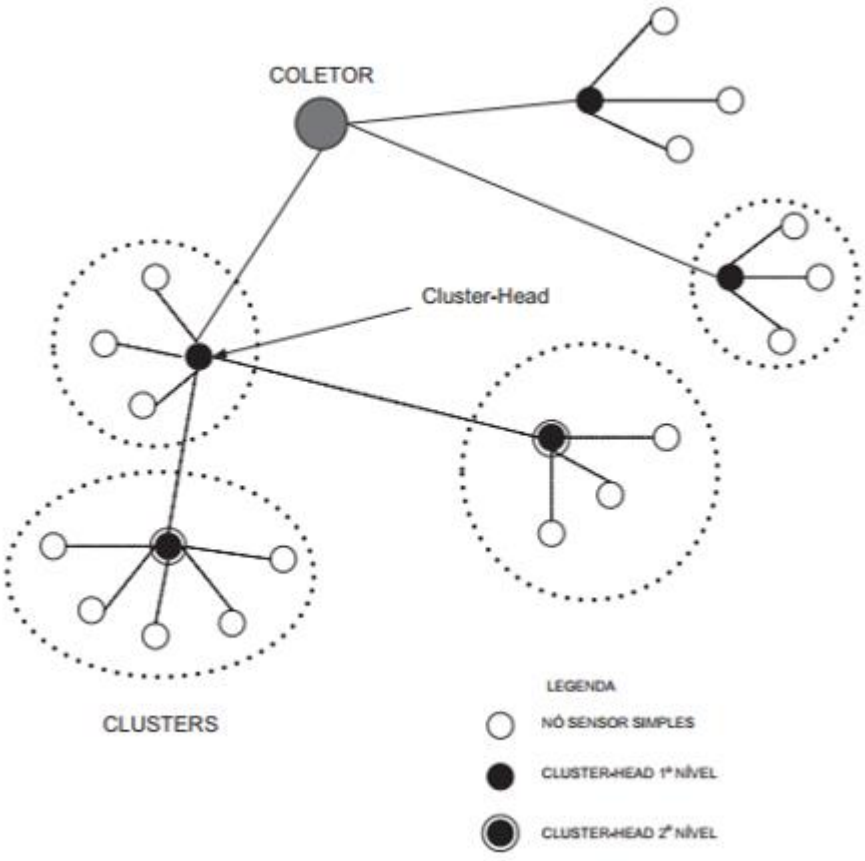
O protocolo LEACH (Low Energy Adaptive Clustering Hierarchy) (HEINZELMAN 2000), é o primeiro e um dos principais trabalhos na área de RSSF. Ele funciona da seguinte forma: é feita a escolha de alguns nodos da rede de forma autônoma para servirem de *cluster-head*, que serve para distribuir tarefas entre eles. Também é feita a eleição de cada nodo que fará parte do cluster em cada execução,

lembrando que um nodo só será novamente eleito após todos os outros já terem passado por este processo.

2.3.1.2. TEEN

O protocolo TEEN (Threshold-sensitive Energy Efficient Protocol) mostrado na Figura 2-4 é desenvolvido para executar em mudanças que são repentinas em um monitoramento, como por exemplo, em um monitoramento de temperatura, ou até mesmo, de umidade. A arquitetura é feita de forma que nodos próximos, formam clusters, repetindo assim até a informação chegar ao nodo base. Esse processo é bastante utilizado em aplicações onde necessitam de tolerância a falhas na recepção dos dados. No entanto, a coleta de dados é feita apenas quando algum dado é modificado (MANJESHWAR 2001).

Figura 2-4. Arquitetura TEEN e APTEEN



Fonte: adaptado de (AKYILDIZ et al. 2002).

2.3.1.3. APTEEN

Para suprir a deficiência em que há no protocolo TEEN, foi desenvolvido o APTEEN (*Adaptive Periodic Threshold-sensitive Energy Efficient sensor Network protocol*). Aparentemente com a arquitetura igual da TEEN, na parte de consultas existem 3 três novas implementadas:

- One Time – Analisa a visão geral da rede;
- Histórica – Analisa dados já capturados;
- Persistente – monitora por um período de tempo determinado.

2.3.1.4. BCDCP

O objetivo deste algoritmo é distribuir a energia igualmente entre todos os nodos fazendo assim a melhor eficiência energética. Ele funciona da seguinte forma, em cada envio de dados a base faz a verificação de cada nodo, o qual é comparado com um limiar já definido. Dessa forma, os nodos que estiverem com a energia maior se tornam candidatos a participar do cluster (MURUGUNATHAN 2005).

2.3.1.5. MECN

O protocolo MECN (*Minimum Energy Communication Network*) criado por (RODOPLU 1999), cria uma sub-rede na qual a transmissão é feita entre todos os nodos até chegar a base. Na sub-rede criada, para cada par (i, j) de nodos conectados na rede original, existe um caminho entre i e j na sub-rede que garante o uso mínimo de energia para transmissão de mensagens entre os dois nodos (LAZZAROTTO, 2008).

2.3.1.6. TTDD

O algoritmo TTDD (FAN 2002), é feita uma hierarquia de 2 níveis para determinar o envio dos dados, pontos de interseção que identificam informação de encaminhamento de dado são iniciados, cada nodo cria uma grade e faz o encaminhamento da informação no sensor mais próximo a grade.

Uma das desvantagens é o gasto com recálculo das grades, no caso de redes que fazem o movimento com frequência não é aconselhável o uso do protocolo TTDD (*Two-Tier Data Dissemination*).

2.4. PROTOCOLOS DE ROTEAMENTO BASEADO EM LOCALIZAÇÃO

Protocolos baseados em localização utilizam em sua arquitetura o componente chamado GPS, que permite melhorar o cálculo de distância entre os nodos, e o consumo de energia, para se chegar até um determinado ponto. Um dos aspectos principais desse protocolo é o não envio desnecessário para nodos que não necessitam da informação que está sendo enviada, como já vimos em protocolos anteriores. Alguns dos principais protocolos serão descritos abaixo.

2.4.1. MECN

Proposto por (XU; HEIDEMANN; ESTRIN, 2001), esse protocolo cria uma rede de baixo consumo utilizando a localização de GPS. O protocolo escolhe um nodo mestre que envia a informação até o nodo destino, onde o mestre é escolhido de acordo com o nodo com maior eficiência energética. O principal objetivo deste protocolo é minimizar o máximo as sub redes, com o menor consumo possível para chegar a localização.

Uma das vantagens da utilização deste protocolo é sua autonomia na reconfiguração de nodos, tanto em novos nodos como a mobilidade de cada.

2.4.2. GAF

Utilizado em redes ad hoc, a técnica consiste em desligar o nodo enquanto o mesmo não é utilizado, sem afetar a rede em que se encontra. Também é criada

uma grade de nodos onde cada um utilizando a sua localização por GPS para associar à grade. Caso tenha mais de um ponto em um mesmo raio de posição, considerados equivalentes um deles é desnecessário para a sua utilização, então o mesmo entra em modo de dormência. Existem 3 tipos de estados para o nodo no protocolo GAF:

- Descoberta
- Ativo
- Dormência

Em testes feitos por (XU; HEIDEMANN; ESTRIN, 2001), este protocolo apresenta um ótimo desempenho em redes com alta latência e perda de pacotes.

2.4.3. GEAR

Utilizando informações geográficas foi proposto o protocolo GEAR por (XU; HEIDEMANN; ESTRIN, 2001), onde utiliza informações geográfico heurísticas de nodos para o encaminhamento de informações entre eles. Cada nodo nesse protocolo possui o custo estimado para enviar dados e chegar até seu nodo vizinho. O algoritmo possui duas fases:

1. Encaminhamento de pacotes para a região-alvo: o nodo inicial verifica se há algum nodo entre seus vizinhos que esteja mais próximo ao destino. Caso exista mais que um, é feita a seleção entre os dois, caso não exista vizinhos, o algoritmo escolherá um nodo baseado em custo aprendido.

2. Encaminhamento dos pacotes dentro da região: se o nodo destino está na mesma grade do nodo origem, o mesmo é enviado por difusão, inundação restrita ou encaminhamento geográfico.

2.5. PROTOCOLOS DE ROTEAMENTO CENTRADOS EM DADOS

Com a grande densidade de sensores, a localização de sensores se torna um tanto difícil pelo fato do número de nodos ser muito alto se tornando inviável a

identificação individual de uma grande rede, fazendo com que esses protocolos sejam utilizados em redes de grande número de sensores, funcionando geralmente dessa maneira enviando os dados para quem necessitar na região em que se encontra, fazendo assim um broadcast na rede.

2.5.1. SPIN

Desenvolvido por (KULIK 2002), esse protocolo tem o objetivo de enviar informações para todos os nodos, tendo em vista que todos os nodos tenhamos a capacidade de enviar as informações até o coletor base.

Este protocolo parte do pressuposto que os nodos obtém a mesma coleta de informação. Nesse caso, é levado em consideração que apenas será necessário enviar dados para nodos distantes que não possuem a informação do grupo anterior.

Na redundância de dados, o SPIN estabelece que cada nodo deve nomear e negociar a transição da informação antes do envio. Caso importante também no SPIN é que os nodos também permitem basear a decisão de envio de informação de acordo com o nível de bateria que está restando nos outros nodos para então enviar a informação.

2.5.2. Direct Diffusion

Desenvolvido por (INTANAGONWIWAT 2000), um protocolo baseado em índice de cada dado que é enviado pelo sensor, o objetivo é unir dados de diferentes partes diminuindo envio desnecessários e redundância de dados, isto é efetuado a partir de múltiplas rotas com o mesmo destino, propiciando a agregação de dados em seu envio.

O objetivo é criar uma árvore que agrega dados dos nodos base e enviam juntamente para o nodo principal. Os nodos criam um quadro de dados das suas vizinhanças. Quando o nodo principal faz a requisição de dados através da rede, essa requisição é enviada a toda rede. Quando um nodo sensor verifica que a

requisição é para sua vizinhança, o mesmo inicia a retransmissão para o nodo principal.

2.5.3. PEGASISVER

O algoritmo PEGASISVER (LINDSEY 2002) é baseado em que cada nodo pode se comunicar com seu vizinho, o envio de dados é feito na topologia onde um vizinho envia os dados para o outro até chegar a um nodo que enviará a informação até o coletor base. Entretanto estes nodos modificam-se a cada envio de dado requisitado. O objetivo deste algoritmo é diminuir a carga na rede, visto que o nodo apenas necessita enviar uma vez sua informação.

O objetivo deste procedimento é reduzir ao máximo o gasto total de energia da rede, visto que cada nodo só precisará fazer uma transmissão, e a uma curta distância (pois transmitirá para o vizinho mais próximo). O protocolo trabalha com a premissa de que os dados podem sempre ser agrupados (cada nodo, apesar de ter um pacote de dados próprio mais o pacote de dados do nodo vizinho, sempre transmitirá somente um pacote para o próximo nodo da corrente). Com isso, limita-se a aplicação deste protocolo a redes com grande redundância de dados (se não há redundância, o acúmulo de dados sobrecarrega os nodos sensores localizados ao longo da corrente).

2.5.4. PEDAP

Por [TAN 2003], é utilizado o princípio do PEGASIS, onde apenas um nodo é selecionado para transmitir a base, a diferença é que todos os nodos são organizados em forma de árvore, onde apenas o nodo pode receber dados de dois vizinhos. Este algoritmo é utilizado em redes com pouca redundância de dados.

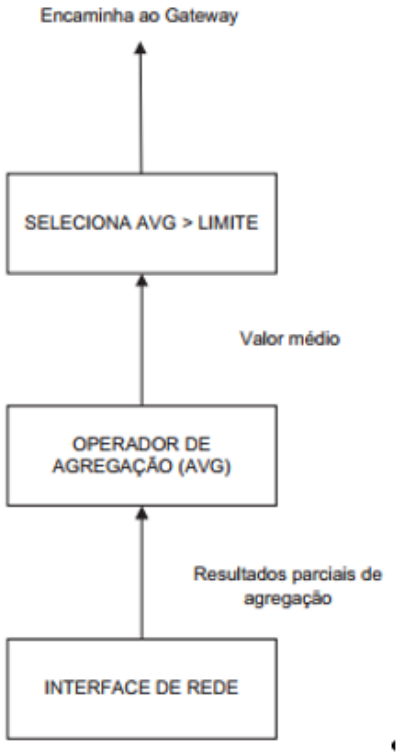
2.5.5. COUGAR

Utilizando a banco hierárquico em sua arquitetura o algoritmo COUGAR, utiliza o banco de dados para cadastrar a localização de nodos, o objetivo principal é utilizar

consulta de baixa densidade ou também de alta densidade, para saber as agregações dos nodos, tendo exatamente cada informação do mesmo.

A Figura 2-5, apresenta a arquitetura de um nodo que é selecionado para ser o responsável pela consulta que obtem informações relevantes dos nodos, efetuando o cálculo do valor de distância entre eles. O protocolo oferece processamento para todos os nodos, garantindo a economia de energia principalmente em grandes nodos, pois segundo (AKYILDIZ et al., 2002), o desgaste de procurar os nodos mais próximo será feito antes da transmissão de informações.

Figura 2-5. Plano de Consulta de um nodo sensor.



Fonte: adaptado de (AKYILDIZ et al. 2002)

2.5.6. ACQUIRE

Utilizando a abordagem de funcionamento do COUGAR, o algoritmo ACQUIRE segundo (SADAGOPAN; KRISHNAMACHARI; HELMY, 2003), utiliza a consulta que é distribuída a toda a cadeia de sensores em sua rede, o nodo possui um cache onde são utilizadas para verificar se a consulta pode ser respondida no instante, e

os nodos também verificam com seus vizinhos essa informação, até o ponto em que a informação possa chegar até o coletor, que foi descoberta com a troca de informações entre os sensores da rede.

O algoritmo utiliza número de saltos que o nodo pode encaminhar a consulta a outro nodo sensor, no entanto o problema com a seleção dos nodos próximos já foi estudado e a seleção é feita escolhendo aleatoriamente baseado na satisfação máxima da consulta.

2.5.7. Energy-Aware Routing

O algoritmo Energy-Aware Routing (SHAH 2002), é focado em economia de energia, seu funcionamento é de acordo com a probabilidade do melhor caminho que o nodo pode tomar, essa probabilidade é feita de acordo com o consumo do nodo. É assumido que cada nodo tem a sua localização e seu tipo esta operação é dividido em 3 tipos:

- Configuração: ocorre a busca para localizar a rota entre todos os nodos origem e destino, e calculando o custo de cada caminho, criando tabelas de rota;
- Comunicação: o nodo envia um pacote aleatoriamente em sua tabela, com a probabilidade inversa proporcional ao custo do nodo;
- Manutenção: realizando uma busca para descobrir os caminhos quebrados e reparar as tabelas.

Em comparação com o algoritmo *Direct Diffusion*, esse protocolo obteve melhores resultados, porém com descobertas mais complexas de rotas devido ao fato de obter a localização e a configuração do endereçamento de nodos.

2.6. PROTOCOLOS DE ROTEAMENTO PROATIVOS

Esses protocolos utilizam mais de uma tabela em cada nodo de sua rede, a atualização dessas tabelas é feita gradativamente, uma de suas vantagens é ter o

roteamento atualizado de seus sensores, no entanto sua utilização não é aconselhável a redes com muita mobilidade entre seus nodos, pelo motivo em que suas tabelas de roteamentos ficariam cada vez maior.

2.6.1. CGSR

Utilizando o conceito de cluster, o algoritmo CGSR (JAYAKUMAR; GOPINATH, 2007), todos os dados são encaminhados ao seu cluster mestre que logo encaminham ao coletor da rede, cada nodo tem duas tabelas uma delas contém o endereço do nodo cluster e a outra contém os passos para encontrar o coletor da rede, a mobilidade dos nodos é dificilmente utilizada pois a cada redescoberta de rede é enviada duas tabelas atualizadas, ocasionando um atraso no envio dos dados.

2.7. SIMULADORES

A seguir alguns dos simuladores mais utilizados na bibliografia.

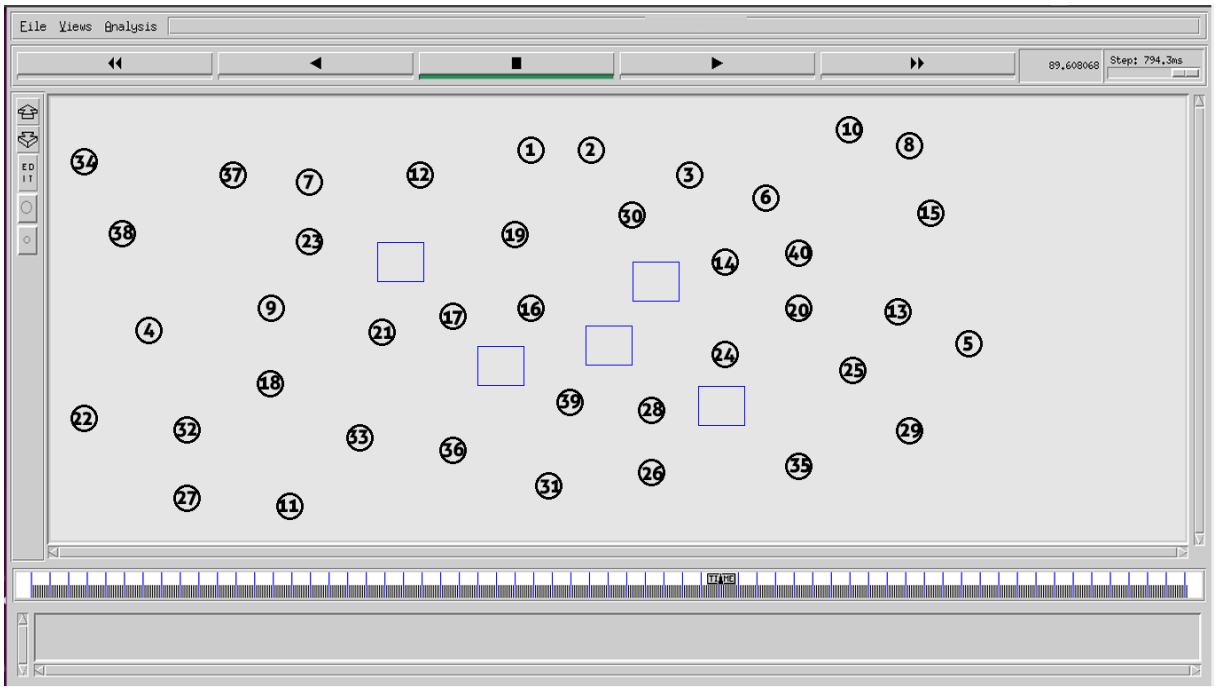
2.7.1. Interface NS-2

O NS-2 é um simulador de eventos focado na pesquisa para desenvolvimento e análise de uma rede de computadores, o qual prevê variações de protocolos, multicast tendo a facilidade de tracing (coleta e registro de dados obtida a cada evento que é executado pela rede). Dentre as várias aplicações deste software destacam-se simulações de redes sem fio e roteamento de satélites. Este software foi desenvolvido para programação em linguagem C++, no entanto a linguagem interpretada é o OTcl [OTCL. 2002].

O NS-2 utiliza modelos específicos de redes de computadores que podem ser, pacotes perdidos, vazão e consumo de energia ligado a transmissão de dados.

O simulador utilizado neste trabalho também é utilizado com grande frequência em pesquisas na área de rede de sensores que é chamado de Network Simulator 2 (NS-2), cuja interface é mostrada na Figura 2-6, foi desenvolvido em 1989 como projeto da Comell University (JAIN, R. 2002).

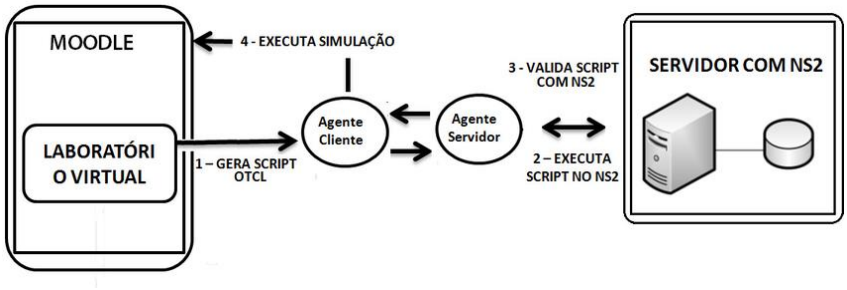
Figura 2-6. Interface principal do NS-2.



Fonte: (Autor, 2018)

A interface entre o software e o usuário final como já especificado a cima se dá através de scripts OTcl. Essa linguagem, juntamente com o C++, permite velocidade e flexibilidade para controle do software da implementação. As Figuras 2-7 e 2-8 mostram a arquitetura do software onde o conjunto constituído entre C++ e o interpretador OTcl e o esquemático do NS-2.

Figura 2-7. Arquitetura do NS-2

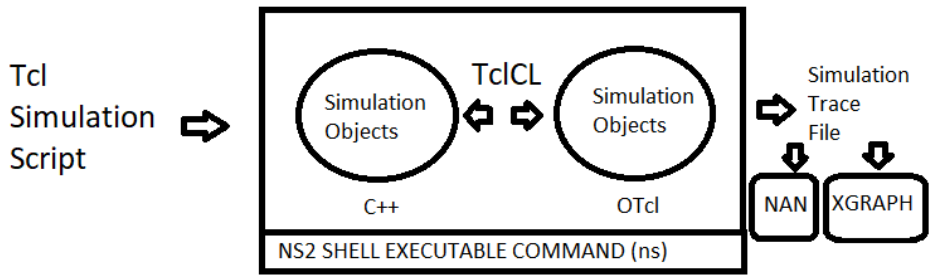


Fonte: (Laboratório Virtual para o Ensino de Redes de Computadores no Moodle, 2018).

Para iniciar uma aplicação, primeiramente é necessário escrever uma aplicação na linguagem OTcl, chamada de script, contendo uma estrutura básica, como segue:

- Criação do objeto Simulador
- Abertura de arquivos para tracing e análise posterior
- Criação da topologia de rede
- Criação de nodos ou nodos
- Conexão dos nodos entre si (links)
- Criação das filas de saída
- Criação dos agentes de 4ª. camada e conexão com nodos
- Criação dos geradores de tráfego (nível de Aplicação) e conexão com agentes de 4ª. camada (nível de Transporte)
- Programação dos escalonadores e timers
- Fechamento da simulação, animação e geração de estatísticas
- O processo de simulação pode ser assim resumido:
- Confecção do script (arquivo texto comum)
- Execução do script com o comando ns nome do script.tcl
- Os arquivos de tracing serão gerados com registro de cada evento simulado
- Após conclusão da simulação:
- Imprimir estatísticas calculadas no script
- Visualizar os eventos com o nam
- Analisar resultados através dos arquivos de tracing com apoio de ferramentas apropriadas.

Figura 2-8. Esquemático NS-2.



Fonte: (Garg, A., Narasimha Reddy A.L, 2002. pp: 45 – 53)

As Redes de Computadores possuem oportunidades de pesquisa seja ela em dispersão do sinal, pacotes enviados e recebidos, teste de velocidade na rede.

A utilização destes simuladores para os modelos propostos é inteiramente necessária para validação dos testes.

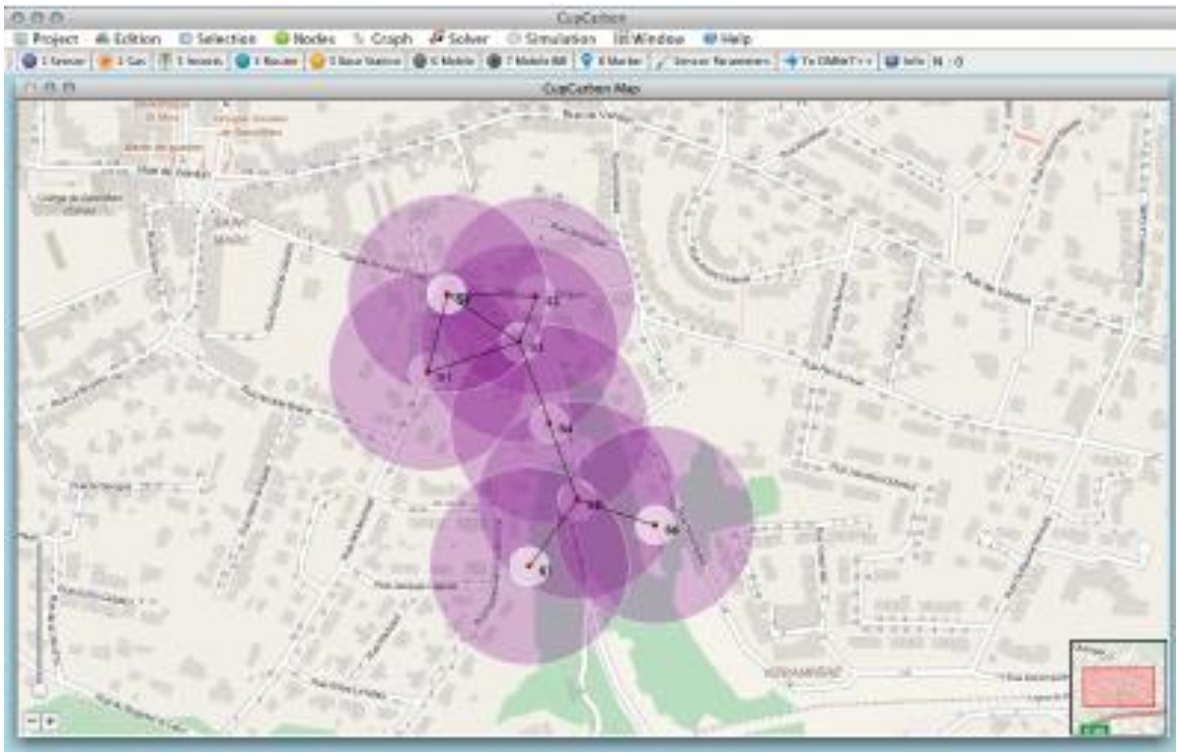
2.7.2. Interface CUPCARBON

Um simulador utilizado para rede de sensores inteligente, tem como principal objetivo prover o planejamento, visualização, depuração e validação os nodos dos sensores, também tem como opções criar cenários como: incêndio, gás, meteoros, entre outros. Planejado também para cientistas testarem suas topologias e protocolos de comunicação em suas redes de sensores [CUPCARBON, 2015].

Simulação baseada em aplicações dos nodos, faz com que a ação de cada sensor seja dada de forma real, também em cada sensor pode ser simulado o algoritmo que está nele [CUPCARBON, 2015].

A Figura 2-9 mostra a interface do CupCarbon nela à uma simulação de uma rede de sensor sem o mapa atrás, mostrando assim que pode ser utilizar a qualquer circunstância.

Figura 2-9. Interface CupCarbon.



Fonte: (CUPCARBON 2016)

3. PROTOCOLOS DE ROTEAMENTOS REATIVOS

A rota é feita por método de inundação, com tempo de processamento melhor do que os protocolos proativos. No entanto o tempo gasto para descobrir as rotas é muito maior (PERKINS; ROYER, 1999). Exemplos desses pacotes são os protocolos AODV e DSR, descritos a seguir.

Este capítulo apresenta os protocolos utilizados na implementação deste trabalho, foram escolhidos pelo fato de criarem tabelas de rotas, onde uma específica de forma diferente a sua tabela e modifica a escolha de variáveis para localização de nodos sensores.

3.1. AODV

Sendo um protocolo Ad Hoc, feito para ser adaptativo a diversos cenários e com mobilidade alta, com a intensão de diminuir a largura da banda e processamento de seus nodos, que atuam juntamente como roteadores de toda a rede. Utilizando tabelas para seu roteamento e armazenando apenas o seu próximo salto que será o seu destino. Sendo um protocolo que realiza atualização constantemente o consumo de energia é alto.

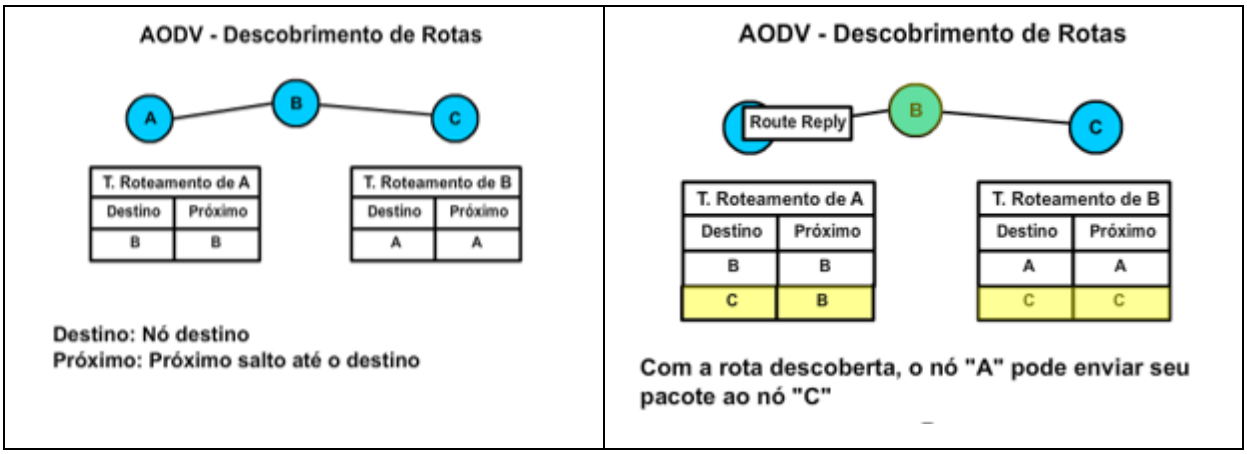
Protocolo foi programado para ser adaptativo a cenários com sua mobilidade alta e com uma largura de banda baixa.

3.1.1. Funcionamento do Protocolo AODV

Por ser um protocolo adaptativo quando não encontra seu próximo destino é iniciado o processo de descoberta de novas rotas entre todos os seus nodos, a Figura 3-1 mostra que a tabela do nodo A tem seu destino e também qual será seu próximo nodo destino até chegar ao final da rota. O mesmo acontece com o nodo B que tem sua rota para o nodo A. Caso um nodo não saiba o destino, o pacote é enviado para o próximo nodo, que é chamado de nodo intermediário, que envia os

dados até que algum nodo conheça a rota desejada. Caso isso não aconteça, é feita novamente a redescoberta de rede.

Figura 3-1. Funcionamento do Protocolo AODV.



Fonte: (DALLORA MORAES, FERNANDES XAUD, 2007)

Neste protocolo também é de suma importância a atualização de rotas para avaliar a existência das rotas já definidas em sua tabela de roteamento, pois seus nodos são móveis e isso pode acarretar uma ruptura de ligação entre os nodos já existentes, por exemplo, quando um nodo não consegue enviar o pacote para o seu nodo destino, o mesmo começa a efetuar a detecção do erro da transmissão, e então envia imediatamente um pacote sinalizando um erro para o nodo de origem, fazendo assim com que todos os outros saibam de sua ruptura de transmissão.

A maneira de manutenção de rotas do protocolo AODV é diferente dos demais que recebem broadcast de todos os outros nodos sensores a cada espaço de tempo. Nesse caso, a atualização da tabela só se dá ao sinal de um erro em sua rede.

3.2. DSDV

Descrito por (PERKINS; BHAGWAT, 1994), sendo um protocolo pró ativo, onde cada nodo sensor envia a sua tabela para o vizinho, baseado no DV (Distance – Vector) com a implementação de números sequenciais permitindo assim com que se saiba quando a tabela foi alterada. A explicação do funcionamento do protocolo que será implementado será descrito a seguir.

3.2.1. Funcionamento do Protocolo DSDV

Tabela de roteamento possuem os seguintes campos Destino, Próximo, Métrica, Número de Sequência, Tempo de Registro, Estabilidade dos Dados.

- Endereço de Destino: Endereço do nodo de origem;
- Número de Sequência: Identificador para cada nova rota.;
- Métrica: Numero de saltos até chegar ao destino;
- Endereço do Próximo nodo: Endereço para o próximo encaminhamento da informação;
- Tempo de Registro: é uma marcação temporal que através dela será decidido deletar ou não uma nova topologia recebida;
- Estabilidade de Dados: Serve como um ponteiro informando sobre a estabilidade das rotas.

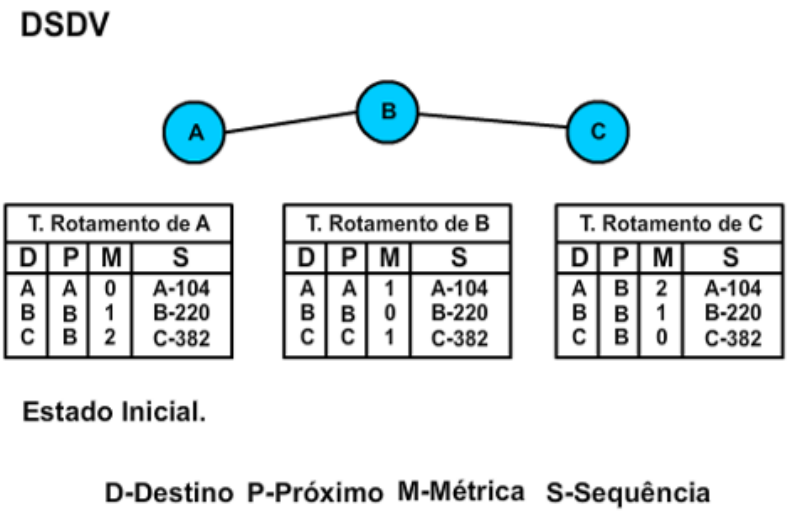
Atualizações é quando o nodo recebe informações de nodos vizinhos, e compara com informações já consistente em sua tabela, para modificar ou não.

- Se a informação da mais recente, a atualização é feita imediatamente;
- Se a intuição for igual, só será feita a atualização caso a métrica seja diferente;

Entretanto, para não existir congestionamento entre as rotas, foi implementado incremental update, onde o nodo só enviara a informação não repetida para seu vizinho. Também existe o Full Update, que é quando o nodo envia toda as informações de sua tabela para seu vizinho.

A Figura 3-2, apresenta o funcionamento do algoritmo DSDV, onde uma tabela é atualizada constantemente ou quando é verificada alguma falha em algum sensor da rede que envia dados para os nodos, atualizando sua tabela a cada iteração com o nodo, sabendo assim o seu destino de origem e chegada.

Figura 3-2. Funcionamento do Protocolo DSDV



Fonte: (DALLORA MORAES, FERNANDES XAUD, 2007)

3.3. DSR

Obter melhor desempenho em redes que tenham uma mobilidade baixa é um dos principais objetivos do protocolo citado, entretanto em casos contrários não há nenhum outro tipo de solução para o problema da mobilidade.

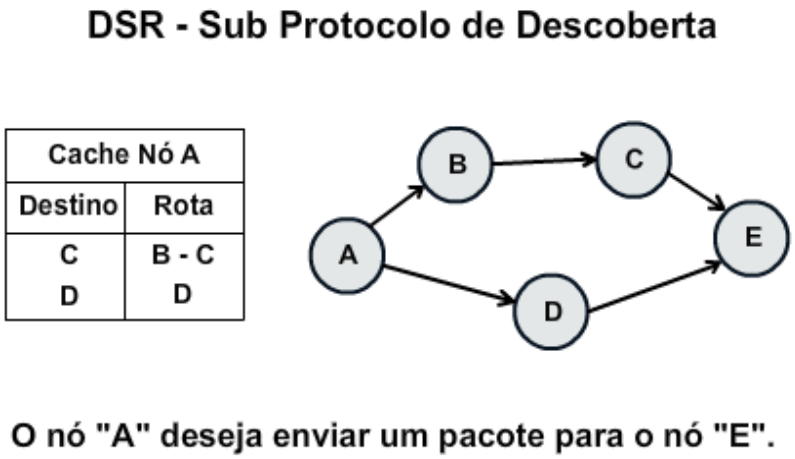
3.3.1. Funcionamento do Protocolo DSR

Cada nodo, Figura 3-3 tem em seu cache o armazenamento da rota que é conhecida por ele. No caso, tenha um número alto de informações trafegando pela rede, os caminhos nessas redes se tornam inválidos rapidamente pelo fato de que

as rotas entre os nodos se modificaram antes de se ter uma nova tabela de roteamento.

- O protocolo é possível ser dividido em sub-protocolos.
- Descoberta e Manutenção de Rotas: funciona de forma semelhante ao AODV, onde as rotas são enviadas periodicamente para seus nodos vizinhos.

Figura 3-3. Funcionamento do Protocolo DSR.



Fonte: (DALLORA MORAES, FERNANDES XAUD, 2007)

Uma das grandes diferenças entre o DSR e o AODV é que um DSR é utilizado mais em mobilidade baixa enquanto o AODV é utilizado em velocidade baixa, respectivamente.

4. Proposta de Analise de Protocolos de Roteamento

Este capítulo apresenta a proposta de análise dos protocolos: *AODV, DSR, DSDV*. É apresentada a metodologia proposta, a validação e a interfase NS-2 para o desenvolvimento dos roteamentos.

4.1.A proposta

A grande densidade de dispositivos presentes em cidades inteligentes utilizando sensores, atuadores e comunicadores, em áreas com pouca cobertura de sinal e obstáculos como prédios, veículos, áreas verdes entre outros (ZHANG, 2007), torna por muitas vezes a comunicação não confiável, e um consumo exagerado de energia entre os nodos para o envio de dados.

Com a utilização de diversos sensores em cidades inteligentes o protocolo de roteamento adaptável é imprescindível para se obter um envio de dados, consumo de energia e dispersão do sinal, satisfatório.

Considerando os itens citados a cima, a proposta consiste na elaboração de cenários de testes com obstáculos ou não que simulam os casos já descritos em uma cidade inteligente, que serão implementados juntamente com os protocolos de roteamento AODV, DSR, DSDV, em uma interface de simulação.

Na implementação serão avaliados: consumo de energia, confiabilidade de dados e tempo de envio.

Esses tópicos serão especificados e detalhados na metodologia e validação nos tópicos a seguir, trazendo consigo resultados para que futuros desenvolvedores possam efetuar uma escolha sensata do protocolo de roteamento para determinado desenvolvimento de aplicações em cidades inteligentes, visando as especificações necessária solicitadas.

4.1.1. Interface de Teste

A interface escolhida para a implementação do trabalho foi a NS-2, escolha feita por ser uma plataforma onde o código de implementação fosse editável para melhores detalhamentos durante os testes e cenário.

Outras interfaces têm seus casos de testes engessados em mapas, ou outros cenários já definidos. No entanto, no NS-2 Figura 4-1, nenhum mapa ou cenário é pré-definido, fazendo com que o desenvolvedor faça todo o código para seus testes. Por esses motivos a escolha por este simulador a Tabela 4.1 mostra a comparação entre os dois simuladores, já descritos nesta dissertação.

A limitação do Simulador NS-2 sobre a quantidade de memória utilizada nas gravações dos dados dificultou alguns testes, diminuindo o número de dados definidos inicialmente no trabalho.

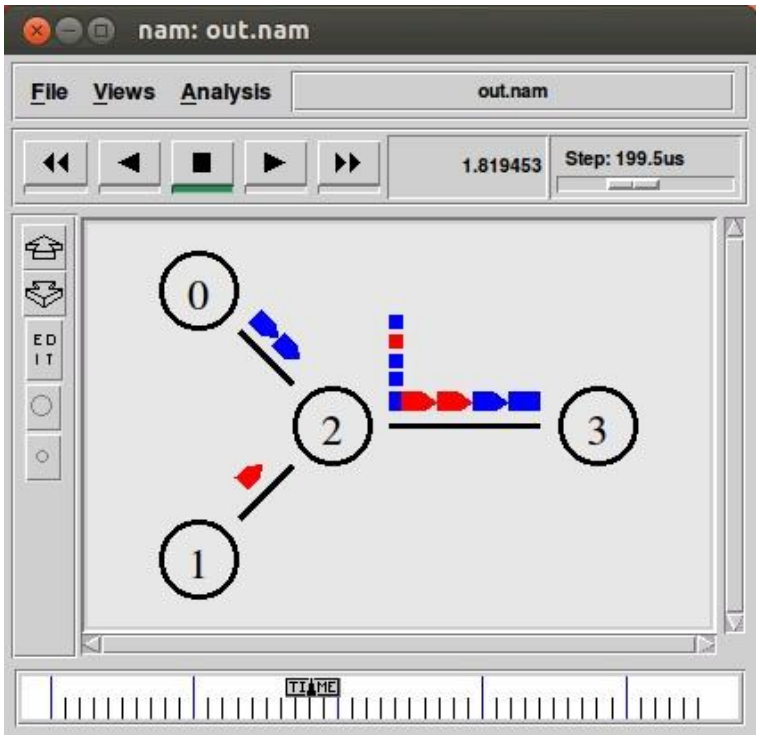
Por fim, os cenários propostos foram executados com essa limitação, satisfazendo os testes.

Tabela 4-1 Comparação entre interfaces de simulação NS-2 e CUPCARBON.

Observação	NS-2	CUPCARBON
Código Fonte Open Source da Interface	X	X
Bibliotecas para implementação	X	-
Comunidade para duvidas	X	-
Cenário Definidos	-	X
Implementação de Protocolos não Definidos	-	X
Limitação de Memória de Gravação de dados	X	-

Fonte: (Autor 2018).

Figura 4-1. NS-2 interseção de nodos exemplo em Uma Rede



Fonte: (DALLORA MORAES, FERNANDES XAUD, 2007)

4.1.2. Protocolos

Entre os diversos Protocolos de Roteamentos existentes, os escolhidos para serem implementados foram: o AODV, DSDV e DSR, pelo fato de na literatura serem os mais estudados, e os mais utilizados.

As implementações foram feitas de acordo com diretrizes estudadas, com tabelas guardando roteamento, entre outras especificações destes algoritmos que podem ser conferidos no código fonte.

Foram implementados e testados os itens citados a cima na proposta e a seguir serão detalhados na metodologia e validação.

4.2. Metodologia

O Diagrama de Fluxo na Figura 4-2 mostra a arquitetura de desenvolvimento da implementação feita neste trabalho, avaliando os requisitos propostos e justificando-os em cada etapa da implementação (BARNAGHI, 2014). Esses quesitos são:

- Número de Nos;
- Comunicação entre os nodos;
- Tempo de Comunicação entre os nodos;
- Consumo de Energia;
- Perda de Pacotes na rede.

A interface NS-2 foi utilizado nas etapas da implementação e testes da dissertação que por ser software livre foi utilizado para estudo de rede de computadores que neste trabalho teve a finalidade de criar diretrizes para o desenvolvimento de cenários e protocolos de roteamento.

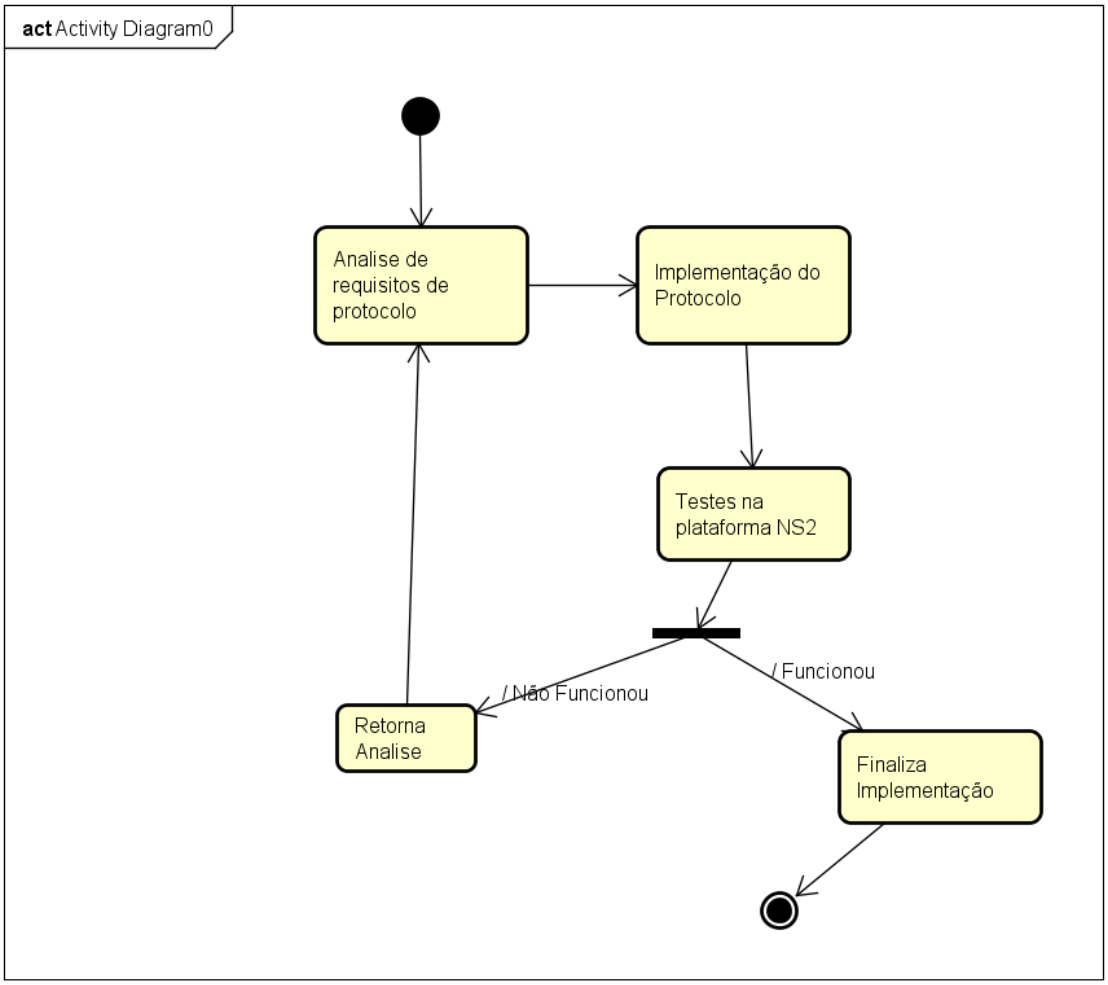
Foram implementados 3(três) protocolos de roteamento, sendo eles AODV, DSDV, DSR, e cenários que são eles:

- Cenário 1: Sem obstáculos testando assim um campo aberto em uma cidade inteligente;
- Cenário 2: Com obstáculos em forma de quadrados, simulando casas e prédios;
- Cenário 3: Com alguns retângulos simulando uma área verde e alguns prédios.

Todos os protocolos foram testados nos cenários descritos.

Ao final da implementação que segue o fluxo na Figura 4-2 foram guardados todos os resultados dos testes da simulação em um arquivo no formato txt. A seguir são analisados os pontos de número de nodos, dados da comunicação entre os nodos, tempo de comunicação entre os nodos, consumo de energia e a perda de pacotes na rede, para então se chegar na conclusão que permitirá o resultado desse trabalho.

Figura 4-2 Fluxograma da metodologia utilizada na implementação dos protocolos de roteamento



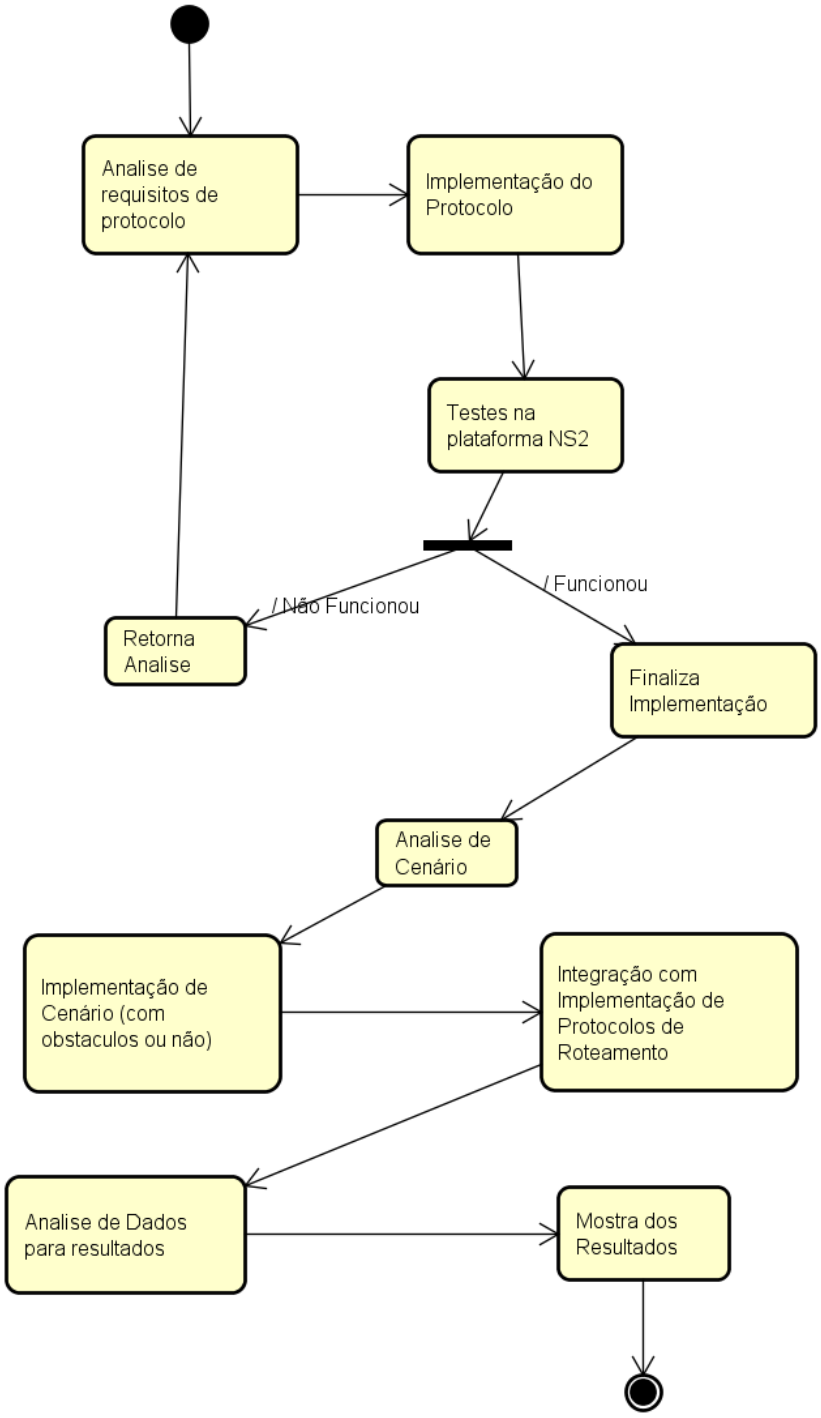
Fonte: (Autor, 2018).

4.2.1. Validação

Os Algoritmos foram implementados com 40 nodos, todos em cenários iguais e com as mesmas diretrizes, sendo os 40 (quarenta) nodos a limitação do simulador em questão. Na Figura 4-3 é ilustrado o simulador com o protocolo do AODV.

A Figura 4-3 mostra ainda, o diagrama de atividades da implementação geral da proposta juntamente com os protocolos.

Figura 4-3 Diagrama de Atividade de Implementação e Testes.



Fonte: (Autor, 2018).

4.2.2. Cenários de Simulação

Foram implementados os algoritmos de roteamentos escolhidos em diferentes cenários, após esse desenvolvimento foram observados aspectos de número de nodos; Comunicação entre os Nodos; Tempo de Comunicação entre os Nodos; Consumo de Energia; Perda de Pacotes na rede. A Figura 4-4 mostra um exemplo de implementação.

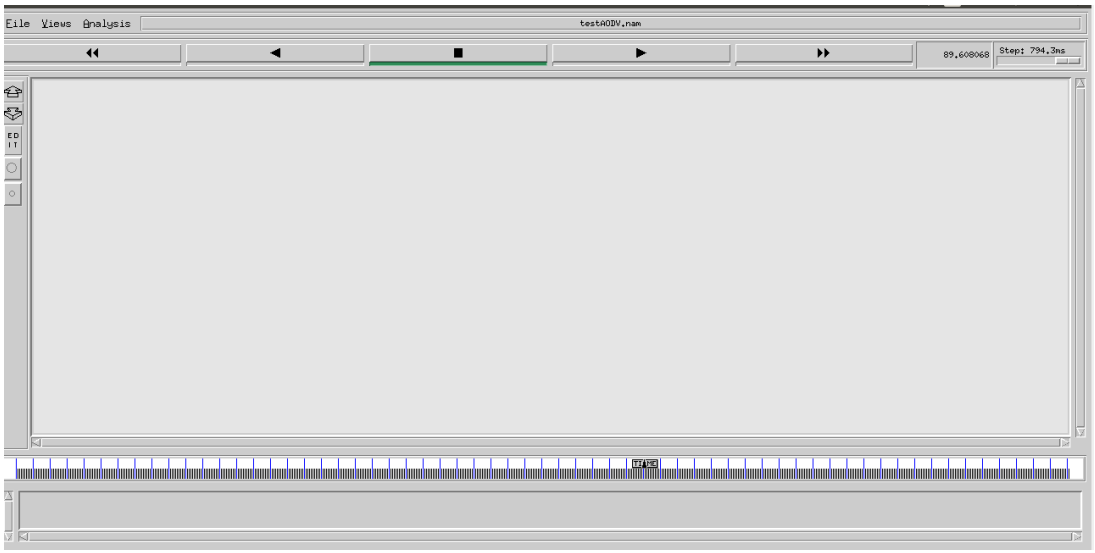
Figura 4-4 Exemplo de Implementação.

Fonte: (Autor, 2018).

4.2.2.1. Cenário 1

A Figura 4-5 mostra o cenário 1 que consiste em uma simulação de campo aberto, que caracteriza um local em uma cidade que não contém nenhum obstáculo. Nele é possível observar a mobilidade dos nodos, o envio dos pacotes e o consumo médio de energia de todos os nodos da rede.

Figura 4-5 Cenário 1, Campo aberto sem obstáculos.

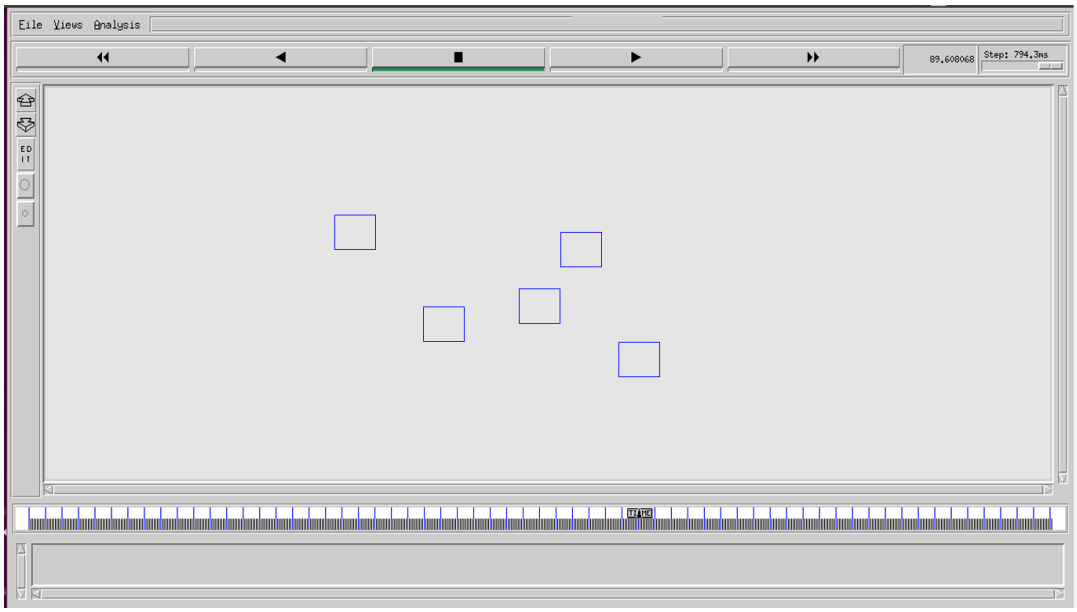


Fonte: (Autor, 2018).

4.2.2.2. Cenário 2

O cenário 2 é constituído de 5 (cinco) obstáculos que obstruem a passagem do sinal. O software não possibilita a implementação de altura desses obstáculos, apenas identifica que existe algo impedindo a passagem de sinal, conforme mostrado na Figura 4-6.

Figura 4-6 - Cenário 2 , alguns obstáculos, simulação de prédios

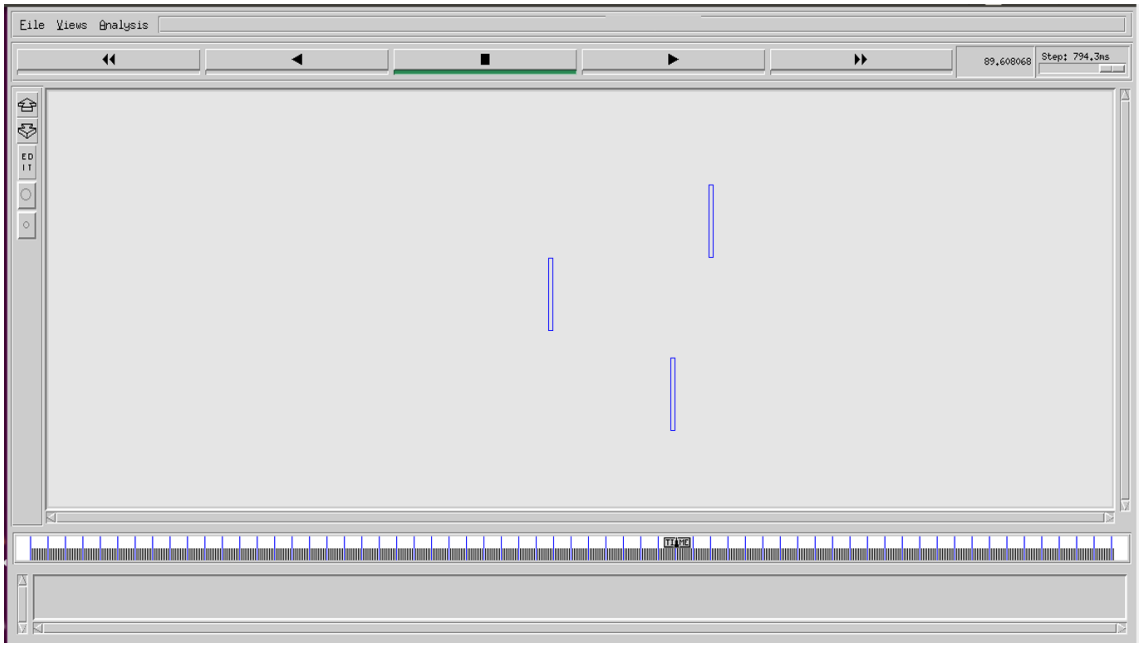


Fonte: (Autor, 2018).

4.2.2.3. Cenário 3

O cenário 3 mostra alguns retângulos, os quais simulam situação de alguns obstáculos como arvores ou casas em menor tamanho. O modelo consiste também em mobilidade dos nodos dinâmicos e estáticos.

Figura 4-7 Cenário 3, alguns obstáculos retangulares, simulando arvores.



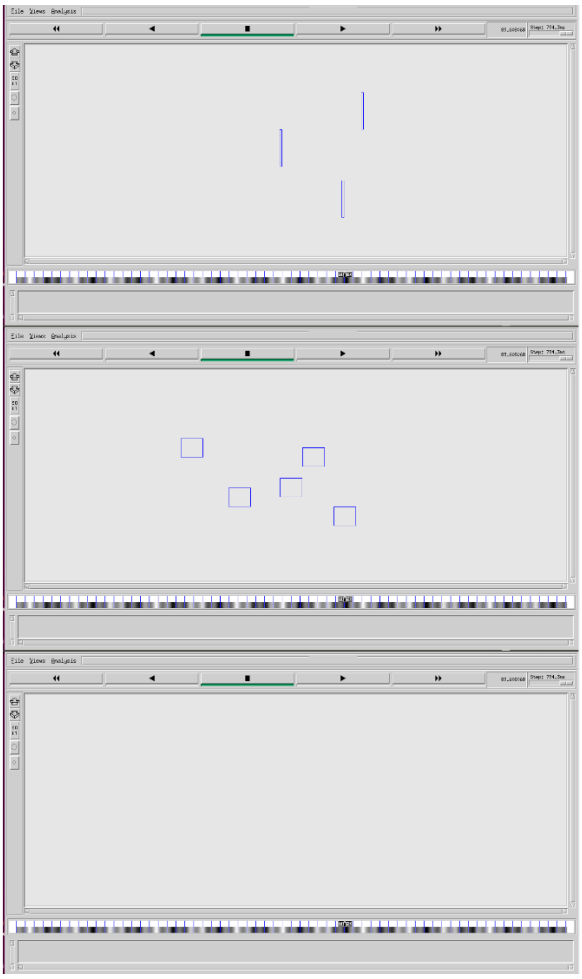
Fonte: (Autor, 2018).

4.2.3. Cenários de Testes

Existem 3 tipos de cenários analisados neste trabalho que compõem os principais problemas em uma rede de sensores, que são obstáculos que atrasam a informação e consomem mais energia.

Os testes foram efetuados nesses ambientes para obter resultados de comportamento de cada protocolo de roteamento quando colocado em alguma aplicação deste modo. A Figura 4-8 mostra todos os cenários que foram desenvolvidos.

Figura 4-8 Cenários Testados.



Fonte: (Autor, 2018).

A Tabela 4-2 mostra a comparação de todos os cenários tendo em vista os obstáculos e mobilidade dos nodos.

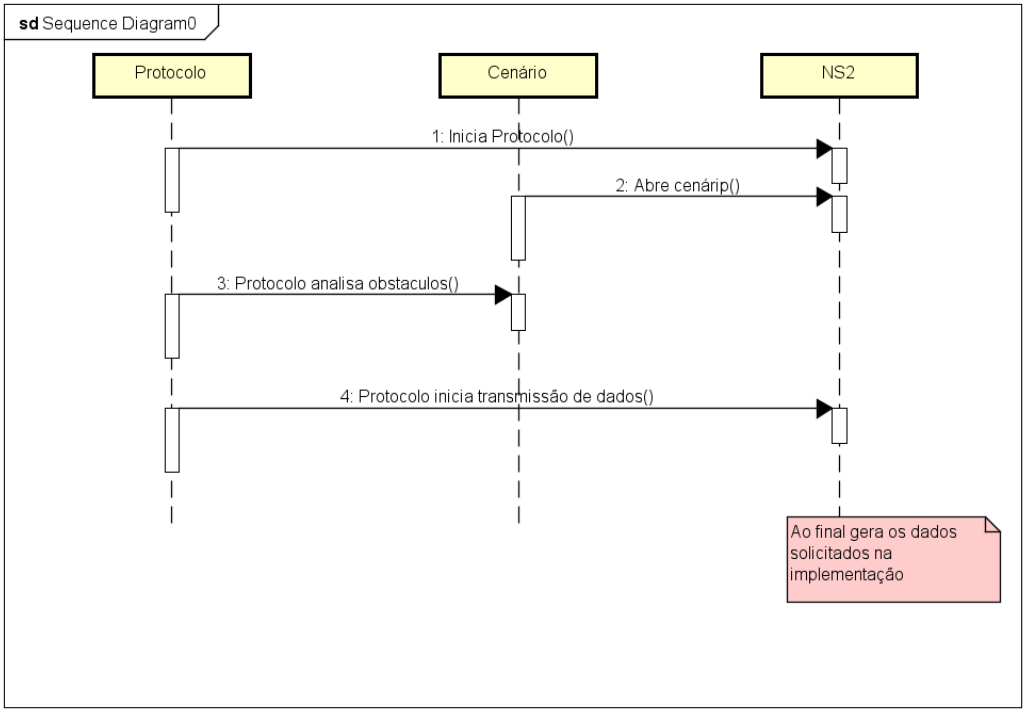
Tabela 4-2 Comparação entre os cenários.

	Cenário 1	Cenário 2	Cenário 3
Obstáculos	NÃO	SIM	SIM
Nodos estáticos	NÃO	NÃO	NÃO

Fonte: (Autor, 2018).

Por fim, a Figura 4-9 mostra o desenvolvimento do código de implementação dos protocolos escolhidos. O protocolo é integrado ao cenário que é executado pelo simulador NS-2, o qual salva estes dados programados para análise a posterior.

Figura 4-9 Diagrama de sequência dos cenários.



Fonte: (Autor, 2018).

5. RESULTADOS

Este capítulo apresenta os resultados obtidos nas simulações. São comparados os três protocolos, em diferentes cenários de implementação.

5.1. RESULTADOS POR CENÁRIO

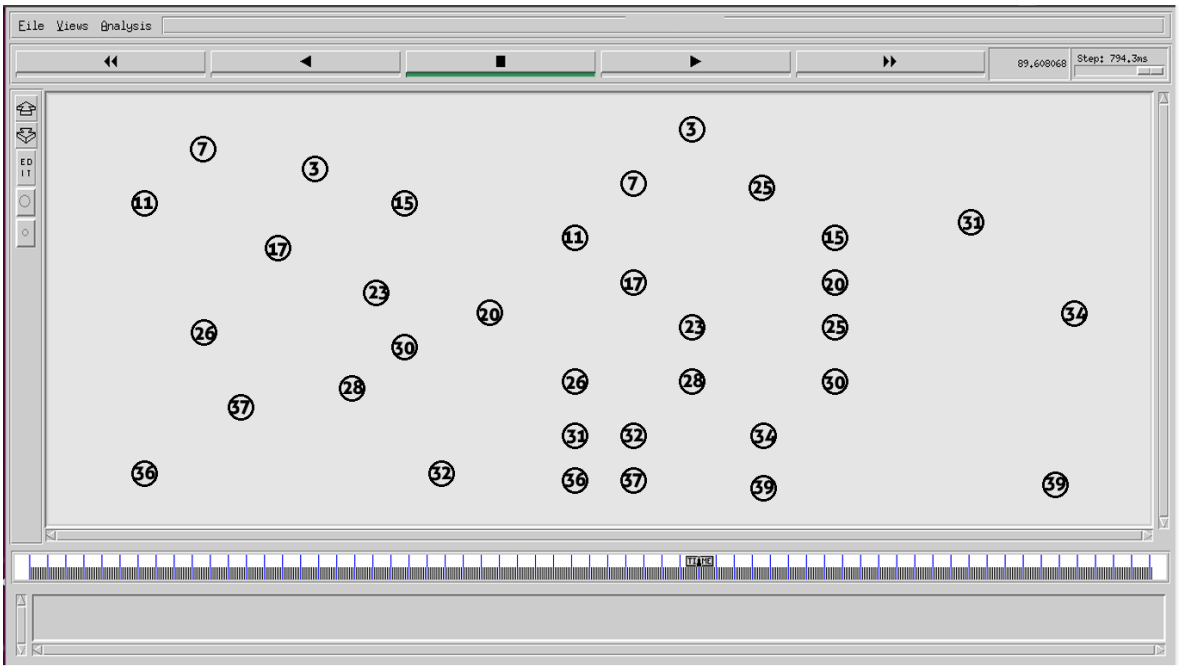
A seguir é mostrado os resultados individuais dos cenários que foram implementados.

5.1.1.1. Cenário 1

O cenário 1 mostrado na Figura 5-1 é constituído de 40 sensores, que enviam mensagens entre si sem a presença de obstáculo, simulando um campo aberto. Os sensores foram iniciados com carga de 0V que chegaria até o seu final de envio a 5V, simulando assim uma bateria.

Foram enviados 345 dados em intervalos aleatórios, que não ultrapassasse 25 minutos de envio. A Tabela 5-1 mostra a comparação dos dados enviados, recebidos junto com sua carga final de envio e também a taxa em porcentagem de não perda de pacotes de cada algoritmo que foi 92% para DSDV, 98% para DSR e 77% para AODV. A seguir será mostrado o comparativo de cada um dos algoritmos no cenário 1 de testes.

Figura 5-1 Protocolo com o primeiro cenário de testes



Fonte: (Autor, 2018).

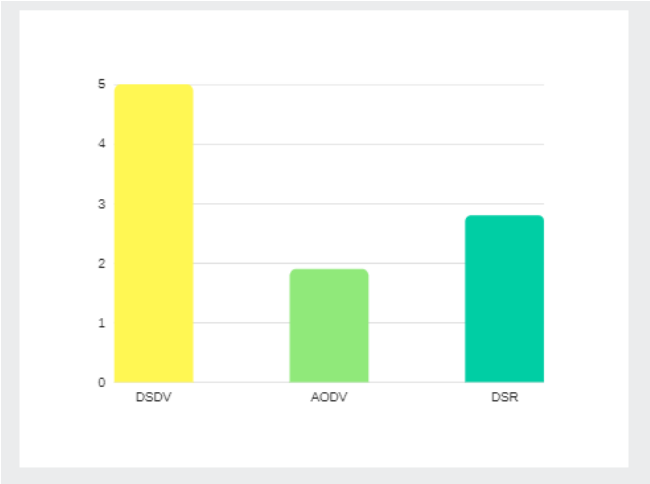
Tabela 5-1 Tabela de Resultados da implementação do cenário 1.

DSDV			DSR			AODV	
Envios	345		Envios	345		Envios	345
Recebimentos	319		Recebimentos	331		Recebimentos	265
Perda	26		Perda	14		Perda	80
Tempo	17min		Tempo	16min		Tempo	22min
Consumo	5V		Consumo	2,8V		Consumo	1,9V

Fonte: (Autor, 2018).

A Figura 5-2 mostra o desempenho em consumo de energia de cada um dos algoritmos, sendo observado que o algoritmo DSDV utilizou plena carga (tensão máxima), o AODV utilizou menos da metade de sua carga e o DSR ficando na média entre os dois resultados.

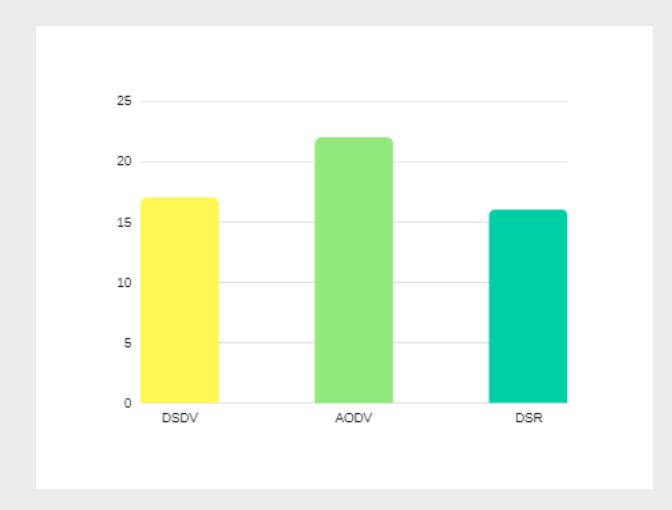
Figura 5-2 Consumo de energia dos protocolos.



Fonte: (Autor, 2018).

A Figura 5-3 ilustra o tempo que foi utilizado para o recebimento e envio de dados, entre todos os nodos da rede, onde o AODV demorou mais tempo no envio de dados que os demais algoritmos em estudo, porém o DSR teve melhor utilização do seu tempo para essa função.

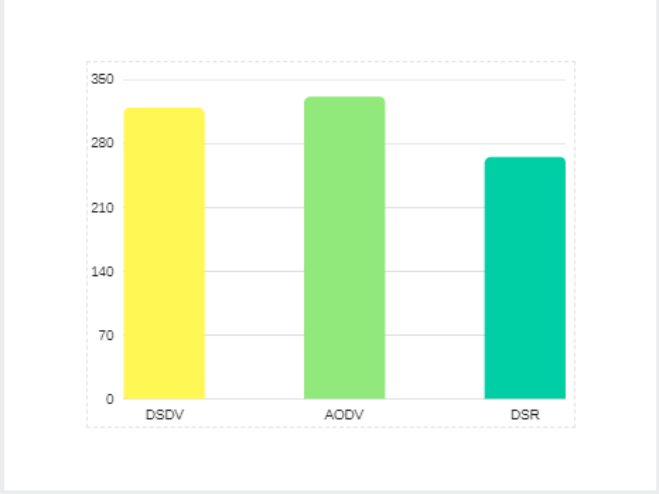
Figura 5-3 Tempo de Recebimento de Dados



Fonte: (Autor, 2018).

A Figura 5-4 mostra a taxa de recebimento dos dados, o AODV teve resultados satisfatórios durante todo o teste sem perda de dados.

Figura 5-4 Taxa de recebimento.



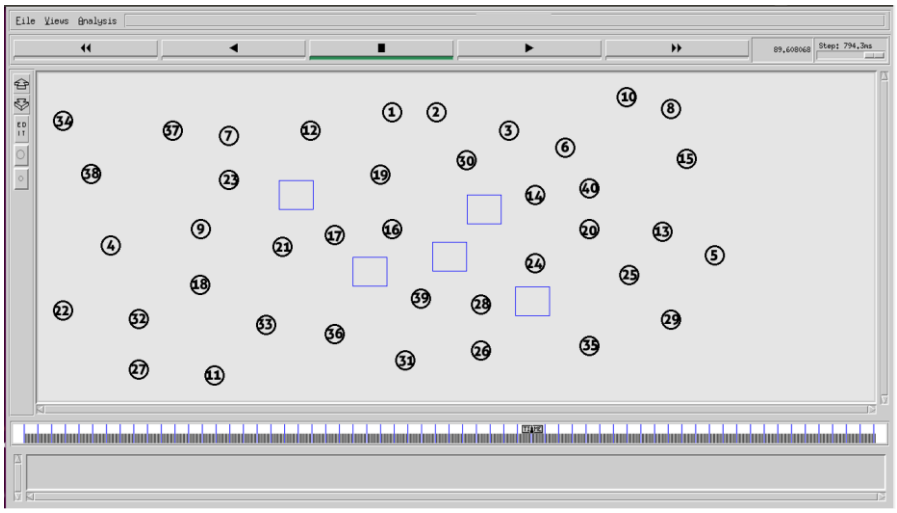
Fonte: (Autor, 2018).

5.1.1.2. Cenário 2

O cenário 2 mostrado na Figura 5-5 é constituído de 40 nodos sensores em movimento, com alguns obstáculos simulando um terreno com relevo acidentado, com alguns obstáculos, caracterizando assim uma cidade ou até mesmo uma indústria, onde a mobilidade não seria tão utilizada, porém os obstáculos ocasionariam em mais perdas de dados.

A Tabela 5-2 mostra os resultados obtidos neste teste, onde o AODV consome toda a energia de seu dispositivo, pelo fato de sua tabela na qual os dados são armazenados ser maior do que os outros, causando um consumo excessivo nesse cenário.

Figura 5-5 Imagem Cenário 2 com protocolo



Fonte: (Autor, 2018).

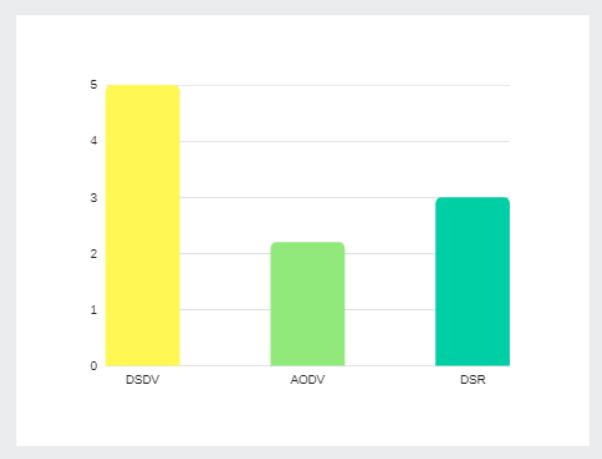
Tabela 5-2 Tabela de Resultados da implementação do cenário 2

DSDV			DSR			AODV	
Envios	345		Envios	345		Envios	345
Recebimentos	339		Recebimentos	321		Recebimentos	255
Perda	6		Perda	24		Perda	90
Tempo	15min		Tempo	17min		Tempo	20min
Consumo	5V		Consumo	3,0V		Consumo	2,2V

Fonte: (Autor, 2018).

A Figura 5-6, mostra o consumo dos algoritmos para um cenário 2 onde o DSDV utilizou toda sua carga para enviar os dados. O AODV utilizou menos energia, pois sua tabela de rotas é mais objetiva utilizando apenas 2 campos, e como citado a cima o DSDV continua com o consumo excessivo.

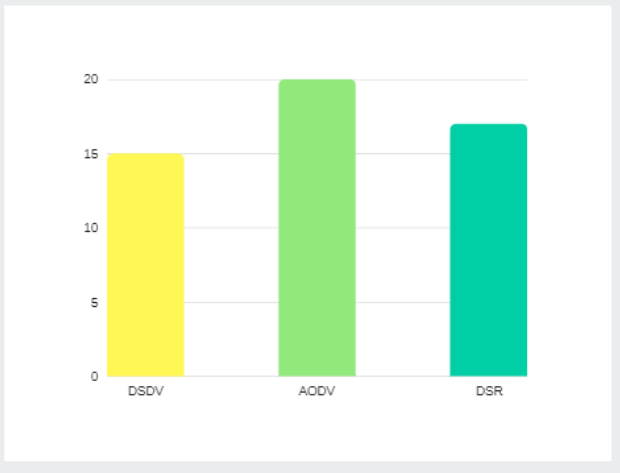
Figura 5-6 Consumo de Energia Cenário 2



Fonte: (Autor, 2018).

Na Figura 5-7 mostra o tempo de envio em que nenhum dos algoritmos excedeu essa variável, mostrando assim que todos são ágeis o suficiente.

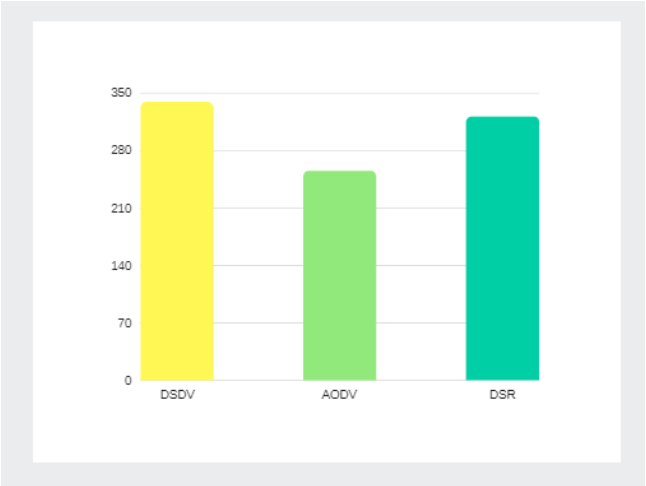
Figura 5-7 Tempo de Envio dos protocolos no cenário 2.



Fonte: (Autor, 2018).

A taxa de recebimento é mostrada na Figura 5-8, onde as medidas se coincidem, porém o algoritmo DSDV mais uma vez neste cenário não teve muitas perdas de dados.

Figura 5-8 Taxa de Envio, cenário 2.

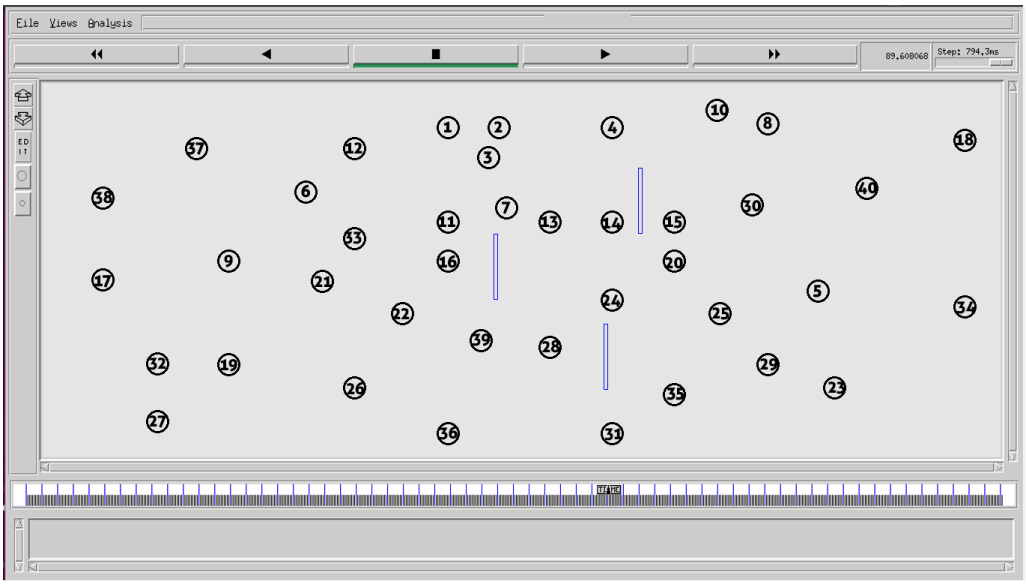


Fonte: (Autor, 2018).

5.1.1.3. Cenário 3

No cenário 3 como mostrado na Figura 5-9, são inseridos alguns obstáculos em sua implementação. Esse cenário simula um ambiente com árvores e outros obstáculos em comum, como uma casa, com diversos obstáculos para que as informações cheguem até seu destino. Os dados da implementação podem ser acompanhados na Tabela 5-3.

Figura 5-9 Cenário 3 com seus protocolos.



Fonte: (Autor, 2018).

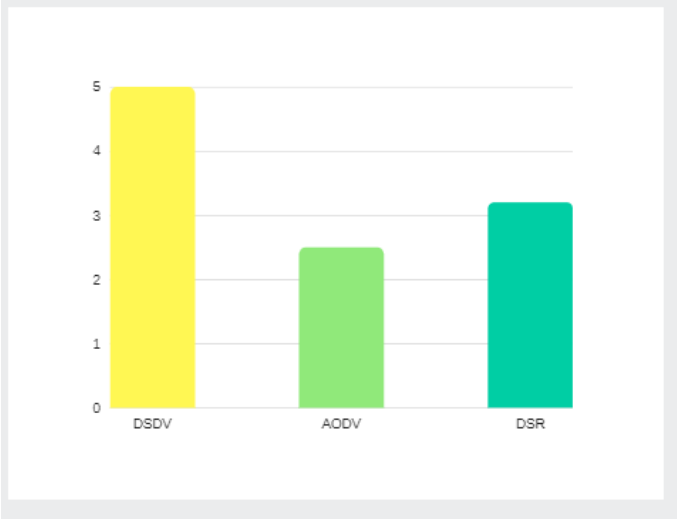
Tabela 5-3 Resultados cenário 3.

DSDV			DSR			AODV	
Envios	345		Envios	345		Envios	345
Recebimentos	323		Recebimentos	340		Recebimentos	280
Perda	22		Perda	5		Perda	65
Tempo	16min		Tempo	15min		Tempo	21min
Consumo	5v		Consumo	3,2v		Consumo	2.5

Fonte: (Autor, 2018).

A seguir a Figura 5-10, ilustra o consumo de cada algoritmo durante os testes, em que vemos que o AODV teve a melhor performance entre os outros que também foi ocasionado por sua tabela de rotas.

Figura 5-10 Consumo de Energia Cenário 3:



Fonte: (Autor, 2018).

Os tempos utilizados no Cenário 3, estão na Figura 5-11, onde é mostrado que o AODV teve o maior tempo de testes entre os outros.

Figura 5-11 Tempo de Envio de Dados.

Fonte: (Autor, 2018).

A Figura 5-12 mostra a taxa de recebimento que cada algoritmo teve no cenário 3, onde o DSR teve menos perda entre os outros algoritmos.

Figura 5-12 Taxa de Recebimento Cenário 3.

Fonte: (Autor, 2018).

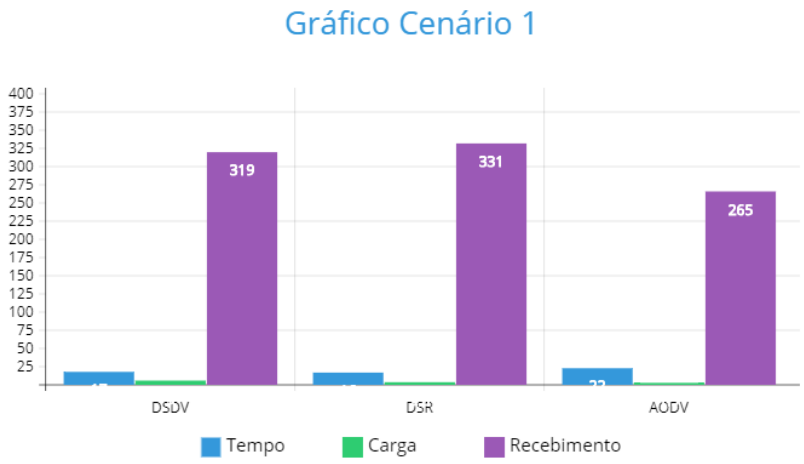
5.2. RESULTADOS GERAIS

A seguir é mostrado os resultados gerais dos cenários que foram implementados e protocolos utilizados.

5.2.1.1. Cenário 1

Podemos observar já no primeiro cenário (Figura 5-13) que cada algoritmo tem característica distinta um do outro, por exemplo, o algoritmo DSR teve o melhor desempenho em taxa de não perda dos dados e sua carga final foi de 2,8V, em um tempo de 16 min, podendo ser assim concluído que no primeiro cenário em campo aberto o melhor a ser usado seria o DSR não desconsiderando o AODV que teve uma taxa de 77% maior que a do DSR, porém a sua carga final foi de 1,9V mostrando que se o objetivo do projeto fosse a economia de energia o AODV seria o melhor a ser utilizado.

Figura 5-13. Gráfico resultado do Cenário 1.



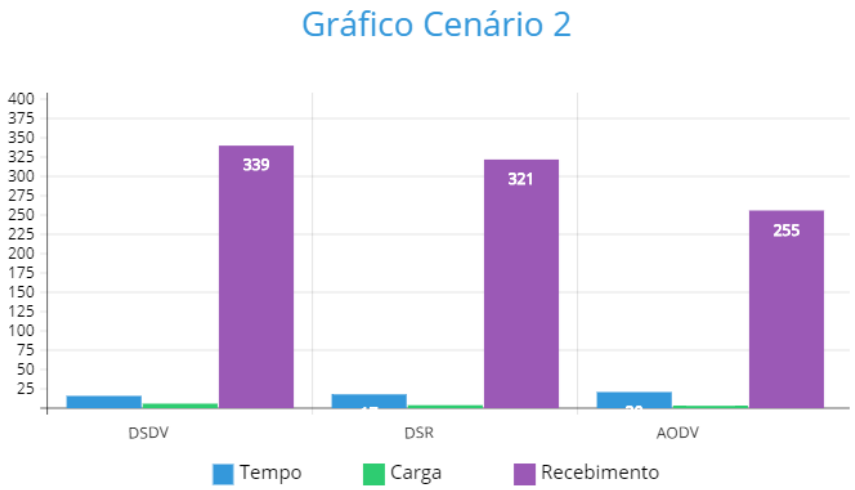
Fonte: (Autor, 2018).

5.2.1.2. Cenário 2

Na Figura 5-14 é mostrado o cenário 2, aonde o algoritmo DSDV teve um melhor envio de dados e um tempo menor de envio, porém utilizando novamente toda sua carga repetindo o cenário 1, o algoritmo AODV teve a maior taxa de perda de dados, porém sua carga chegou a até 2,2V com um tempo de simulação de 22min. Sendo assim também o DSDV em situação de tolerância a falhas poderia ser

melhor utilizado que os outros. Caso o cenário necessitasse um melhor desempenho de carga o AODV seria a melhor escolha para esse tipo de cenário.

Figura 5-14. Gráfico resultado do Cenário 2

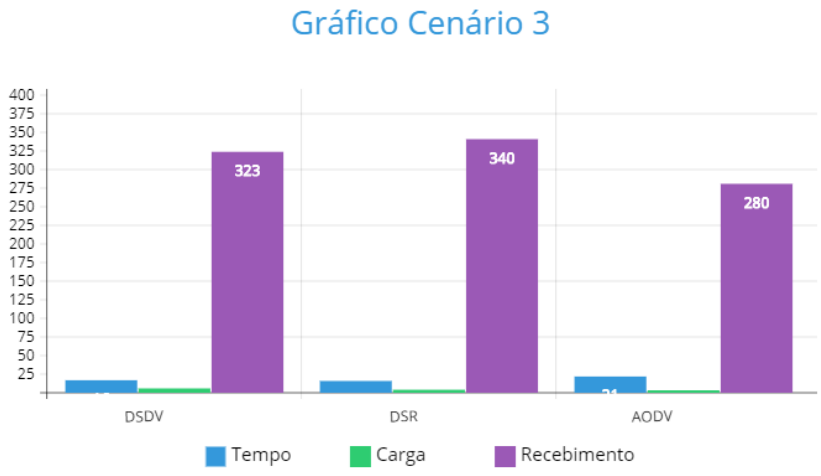


Fonte: (Autor, 2018).

5.2.1.3. Cenário 3

Na Figura 5-15 podemos observar que o algoritmo DSR se mostrou mais eficiente em todos os aspectos analisados durante a simulação tendo também o menor tempo entre todos os outros algoritmos, e seu consumo não chegou a 100%..

Figura 5-15. Gráfico resultado do Cenário 3.



Fonte: (Autor, 2018).

Durante as implementações dos protocolos neste trabalho foram encontrados alguns problemas no seu desenvolvimento, como por exemplo, o software NS-2 não possui uma base de dados para armazenamento de testes, por esse motivo os testes foram apenas com as variáveis citadas. Em segundo lugar, a implementação de uma nova linguagem desconhecida também ocasionou demora na implementação do trabalho.

6. CONCLUSÃO E TRABALHOS FUTUROS

Após inúmeros testes efetuados com os algoritmos AODS, DSR e AODV, conclui-se que cada um pode ser utilizado em situações diferentes onde há alguma necessidade específica de implementação.

Este trabalho mostrou a eficiência do algoritmo DSDV em ambientes com poucos obstáculos, como por exemplo em uma cidade inteligente, onde seus dados transmitidos, tempo de envio e carga utilizada tem uma estabilidade para este tipo de ambiente.

O algoritmo DSR se mostrou eficiente em casos em que não há obstáculos e em casos com muitos obstáculos. Neste caso ele pode ser usado de forma eficiente em ambientes rurais ou residências, dependendo do projeto em questão.

O algoritmo AODV não obteve resultados satisfatórios nos testes em que foram implementados, porém em alguns dos cenários como no 1, a sua carga foi a menor que utilizou, podendo assim satisfazer alguma condição de projeto nesta variável.

Algoritmos de roteamento são estudados e simulados em diversos trabalhos, porém sua eficiência depende de cada ambiente. Esse tema traz consigo uma discussão sobre o que utilizar em determinado local ou projeto. O IoT (*Internet of Things*) faz com que esse tipo de transmissão seja estudada para melhor experiência com o usuário.

6.1. Trabalhos Futuros

Esta dissertação cria oportunidades de pesquisas futuras, que são elas:

- Implementação de novos algoritmos de roteamentos vistos neste trabalho;
- Testes em novos cenários, em que se possa deixar os nodos escravos estáticos, sem locais com obstrução e sem obstrução de sinal durante;
- cenários criados com maior quantidade de nodos para verificar a perda de pacote e consumo dos sensores;
- Mudança de parâmetros como carga e tempo para verificação de outros modelos de sensores e comunicação;
- Implementação de testes reais com equipamentos e dispositivos, em determinado cenário estudado a cima.

- Estudo de algoritmos em diferentes situações (cada nodo tem um tipo de algoritmo em sua base de dados);
- Estudo em Big Data, dos dados armazenados durante os testes.
- Criação de um novo algoritmo de roteamento utilizando diretrizes dos algoritmos estudados nesse trabalho.

7. REFERÊNCIAS BIBLIOGRÁFICAS

AKKAYA, K.; YOUNIS, M. A survey on routing protocols for wireless sensor networks. Ad hoc networks, Elsevier, v. 3, n. 3, p. 325–349, 2005.

Ana Luiza Dallora Moraes • Arthur Fernandes dos Santos Xaud • Marco Fernandes dos Santos Xaud, 2007. Disponivel em: https://www.gta.ufrj.br/grad/09_1/versao-final/adhoc/intro.html

ANTONIO de Oliveira Domingues, Marco; Fawzi Hadj Sadok, Djamel. Uma abordagem para validação de protocolos de comunicação em ambientes de simulação. 2004. Dissertação (Mestrado). Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Pernambuco, Recife, 2004.

ARC ADVISORY, 2017. Disponibilizado em: https://www.arcweb.com/industries/smart-cities.

CORREIA, L. et al. Uma taxonomia para protocolos de controle de acesso ao meio em redes de sensores sem fio. Annales des Telecommunications.(Paris: 2005), 2005.

CUPCARBON. An cupcarbon.com. Disponível em: <http://www.cupcarbon.com>. Acesso em: 05 ago.2017.

FAN, Y., ZHANG, L. A Two-tier data dissemination model for large-scale wireless sensor networks. Proceedings of ACM/IEEE MOBICOM, 2002.

Garg, A., Narasimha Reddy A.L. “Mitigation of DoS attacks through QoS regulation” in Proc. Quality of Service,2002. pp: 45 – 53

HEINZELMAN, W., CHANDRAKASAN, A., BALAKRISHNAN, H. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In Proceedings of the 33rd International Conference on System sciences (HICSS ’00). Maui, Hawaii, Jan 2000.

HRIBERNIK, Karl A.; GHRAIRI, Zied; HANS, Carl; THOBEN, K. First experiences in the participatory design of intelligent products with arduino. Proceedings of the 2011 17th International Conference on Concurrent Enterprising (ICE 2011).

Issues, v. 8, pp. 89-93, 2011

JAIN, R. The art of computer systems perfomance analysis. John Wiley & Sons, 1991 apud BRITO, Sergio de Figueiredo et al. Simulation tool and usage strategy – a practical and pragmatic approach. In Anais do V INDUSCON 2002 (IEEE – Industry Applications Society), Salvador, BA, 03 jul. 2002

KARIMI, R., ITHNIN, N., RAZAK, S., NAJAFZADEH, S., “DTN Routing Protocols for VANETs: Issues and Approaches”, *IJCSI International Journal of Computer Science*
KESHAV, S. REAL 5.0 overview [online]. Disponível na Internet via URL: http://www.cs.cornell.edu/skeshav/real/overview.html. Arquivo capturado em 24.07.2002.

Laboratório Virtual para o Ensino de Redes de Computadores no Moodle - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/Figura-1-Arquitetura-basica_fig1_259146341 [accessed 16 Aug, 2018]

Lindgren, A.; Doria, A.; Schelén, O. “Probabilistic Routing in Intermittently Connected Networks”. *SIGMOBILE Mobile Computing and Communications Review*, vol. 7-3, Julho 2003, pp. 19–20.

MALLADI, R.; AGRAWAL, D. Current and future applications of mobile and wireless networks. Communications of the ACM, ACM, v. 45, n. 10, p. 144–146, 2002.

MANJESHWAR, A., AGARWAL, D. P. APTEEN: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks. Parallel and Distributed Processing Symposium., Proceedings International. IPDPS 2002, pp. 195-202.

MURUGUNATHAN, D., et al. A Centralized Energy-Efficient Routing Protocol for Wireless Sensor Networks. IEEE Radio Communications. March 2005 pp. S8-S13.

OTCL (MIT Object Tool Command Language) [online]. Disponível na Internet via URL: http://otcl-tclcl.sourceforge.net/otcl/. Arquivo capturado em 24.07.2002.

PANTAZIS, N.; VERGADOS, D. A survey on power control issues in wireless sensor networks. Communications Surveys Tutorials, IEEE, [S.1], v9, n.4, p.86-107, 2007

RODOPLU, V., MENG, T. H. Minimum Energy Mobile Wireless Networks. IEEE Journal Selected Areas in Communications. Vol. 17, no. 8, Aug. 1999, pp. 133344.

SPYROPOULOS, T., PSOUNIS, K., RAGHAVENDRA, C., “Spray and wait: an efficient routing scheme for intermittently connected mobile networks”. In: *WDTN '05 Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pp. 252-259, New York, 2005.

STEVENTON, A.and WRIGHT, S, (2006) “Intelligent spaces: The application of pervasive ICT”. London, Springer

WEILIAN S; ERDAL, C. O. B. Handbook of sensor networks: Compact wireless and wired sensing systems. In: Proceedings of 3rd International Symposium on Modeling and Op-timization in Mobile, Ad Hoc and Wireless Networks. S.l.: IEEE, Sessão IV - Capítulo 16, p. 16–1 à 16–14, 2005.

XMPP community. Disponível em: https://xmpp.org/about/technology-overview.html. Julho 2017.

Zhang, X.; Kurose, J.; Levine, B. N.; Towsley, D.; Zhang, H. “Study of a Bus-based Disruption-Tolerant Network: Mobility Modeling and Impact on Routing”. In: Tha Annual International Conference on Mobile Computing and Networking, 2007, pp. 195–206.

ANEXO A - Códigos utilizados na implementação do roteamento de sensores no software NS-2

```
set a 1
while {$a == 1 } {
puts &quot;Enter the Routing Agents in mobile networking&quot;
puts &quot;1. AODV&quot;
puts &quot;2. DSDV&quot;
puts &quot;3. DSR&quot;

set top [gets stdin]
if {$top == 1} {
set opt(chan)          Channel/WirelessChannel    ;# channel type
set opt(prop)          Propagation/TwoRayGround   ;# radio-propagation model
set opt(netif)         Phy/WirelessPhy            ;# network interface type
set opt(mac)           Mac/802_11                 ;# MAC type
set opt(ifq)           Queue/DropTail/PriQueue    ;# interface queue type
set opt(ll)            LL                         ;# link layer type
set opt(ant)           Antenna/OmniAntenna        ;# antenna model
set opt(ifqlen)        50                         ;# max packet in ifq
set opt(nn)            22                         ;# number of mobilenodes
set opt(rp)            AODV                       ;# routing protocol
set opt(x)             1800                  ;# X dimension of topography
set opt(y)             840                   ;# Y dimension of topography
### Setting The Simulator Objects

     set ns_ [new Simulator]
#create the nam and trace file:
     set tracefd [open  aodv.tr  w]
     $ns_ trace-all $tracefd
     set namtrace [open aodv.nam w]
     $ns_ namtrace-all-wireless $namtrace  $opt(x) $opt(y)
     set topo [new Topography]
     $topo load_flatgrid $opt(x) $opt(y)
     create-god $opt(nn)
     set chan_1_ [new $opt(chan)]

####  Setting The Distance Variables

     # For model &#39;TwoRayGround&#39;
     set dist(5m)  7.69113e-06
     set dist(9m)  2.37381e-06
     set dist(10m) 1.92278e-06
     set dist(11m) 1.58908e-06
     set dist(12m) 1.33527e-06
     set dist(13m) 1.13774e-06
     set dist(14m) 9.81011e-07
```

```
set dist(15m) 8.54570e-07
set dist(16m) 7.51087e-07
set dist(20m) 4.80696e-07
set dist(25m) 3.07645e-07
set dist(30m) 2.13643e-07
set dist(35m) 1.56962e-07
set dist(40m) 1.56962e-10
set dist(45m) 1.56962e-11
set dist(50m) 1.20174e-13
Phy/WirelessPhy set CSThresh_ $dist(50m)
Phy/WirelessPhy set RXThresh_ $dist(50m)
```

31
Anexo A
A_____

```
#  Defining Node Configuration

                $ns_ node-config -adhocRouting $opt(rp) \
                 -llType $opt(ll) \
                 -macType $opt(mac) \
                 -ifqType $opt(ifq) \
                 -ifqLen $opt(ifqlen) \
                 -antType $opt(ant) \
                 -propType $opt(prop) \
                 -phyType $opt(netif) \
                 -topoInstance $topo \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace ON \
                 -movementTrace ON \
                 -channel $chan_1_
###  Creating The WIRELESS NODES

set Server1 [$ns_ node]
set Server2 [$ns_ node]
set n2 [$ns_ node]
set n3 [$ns_ node]
set n4 [$ns_ node]
set n5 [$ns_ node]
set n6 [$ns_ node]
set n7 [$ns_ node]
set n8 [$ns_ node]
set n9 [$ns_ node]
set n10 [$ns_ node]
set n11 [$ns_ node]
set n12 [$ns_ node]
set n13 [$ns_ node]
```

```
set n14 [$ns_ node]
set n15 [$ns_ node]
set n16 [$ns_ node]
set n17 [$ns_ node]
set n18 [$ns_ node]
set n19 [$ns_ node]
set n20 [$ns_ node]
set n21 [$ns_ node]
set n22 [$ns_ node]

set opt(seed) 0.1
set a [ns-random $opt(seed)]
set i 0
while {$i &lt; 5} {
incr i
}

###  Setting The Initial Positions of Nodes
$Server1 set X_ 513.0
$Server1 set Y_ 517.0
$Server1 set Z_ 0.0

$Server2 set X_ 1445.0
$Server2 set Y_ 474.0
$Server2 set Z_ 0.0

$n2 set X_ 36.0
$n2 set Y_ 529.0
$n2 set Z_ 0.0
```



```
$n3 set X_ 143.0
$n3 set Y_ 666.0
$n3 set Z_ 0.0
$n4 set X_ 201.0
$n4 set Y_ 552.0
$n4 set Z_ 0.0

$n5 set X_ 147.0
$n5 set Y_ 403.0
$n5 set Z_ 0.0

$n6 set X_ 230.0
$n6 set Y_ 291.0
```

```
$n6 set Z_ 0.0
$n7 set X_ 295.0
$n7 set Y_ 419.0
$n7 set Z_ 0.0
$n8 set X_ 363.0
$n8 set Y_ 335.0
$n8 set Z_ 0.0
$n9 set X_ 334.0
$n9 set Y_ 647.0
$n9 set Z_ 0.0
$n10 set X_ 304.0
$n10 set Y_ 777.0
$n10 set Z_ 0.0

$n11 set X_ 412.0
$n11 set Y_ 194.0
$n11 set Z_ 0.0

$n12 set X_ 519.0
$n12 set Y_ 361.0
$n12 set Z_ 0.0
$n13 set X_ 569.0
$n13 set Y_ 167.0
$n13 set Z_ 0.0
$n14 set X_ 349.0
$n14 set Y_ 546.0
$n14 set Z_ 0.0
$n15 set X_ 466.0
$n15 set Y_ 668.0
$n15 set Z_ 0.0
$n16 set X_ 489.0
$n16 set Y_ 794.0
$n16 set Z_ 0.0
$n17 set X_ 606.0
$n17 set Y_ 711.0
$n17 set Z_ 0.0
$n18 set X_ 630.0
$n18 set Y_ 626.0
$n18 set Z_ 0.0
```



```
$n19 set X_ 666.0
$n19 set Y_ 347.0
$n19 set Z_ 0.0
```

```
$n20 set X_ 741.0
$n20 set Y_ 152.0
$n20 set Z_ 0.0
$n21 set X_ 882.0
$n21 set Y_ 264.0
$n21 set Z_ 0.0

$n22 set X_ 761.0
$n22 set Y_ 441.0
$n22 set Z_ 0.0

## Giving Mobility to Nodes

$ns_ at 0.75 &quot;$n2 setdest 379.0 349.0 20.0&quot;
$ns_ at 0.75 &quot;$n3 setdest 556.0 302.0 20.0&quot;
$ns_ at 0.20 &quot;$n4 setdest 309.0 211.0 20.0&quot;
$ns_ at 1.25 &quot;$n5 setdest 179.0 333.0 20.0&quot;
$ns_ at 0.75 &quot;$n6 setdest 139.0 63.0 20.0&quot;
$ns_ at 0.75 &quot;$n7 setdest 320.0 27.0 20.0&quot;
$ns_ at 1.50 &quot;$n8 setdest 505.0 124.0 20.0&quot;
$ns_ at 1.25 &quot;$n9 setdest 274.0 487.0 20.0&quot;
$ns_ at 1.25 &quot;$n10 setdest 494.0 475.0 20.0&quot;
$ns_ at 1.25 &quot;$n11 setdest 899.0 757.0 25.0&quot;
$ns_ at 0.50 &quot;$n12 setdest 598.0 728.0 25.0&quot;
$ns_ at 0.25 &quot;$n13 setdest 551.0 624.0 25.0&quot;
$ns_ at 1.25 &quot;$n14 setdest 397.0 647.0 25.0&quot;
$ns_ at 1.25 &quot;$n15 setdest 748.0 688.0 25.0&quot;
$ns_ at 1.25 &quot;$n16 setdest 842.0 623.0 25.0&quot;
$ns_ at 1.25 &quot;$n17 setdest 678.0 548.0 25.0&quot;
$ns_ at 0.75 &quot;$n18 setdest 741.0 809.0 20.0&quot;
$ns_ at 0.75 &quot;$n19 setdest 437.0 799.0 20.0&quot;
$ns_ at 0.20 &quot;$n20 setdest 159.0 722.0 20.0&quot;
$ns_ at 1.25 &quot;$n21 setdest 700.0 350.0 20.0&quot;
$ns_ at 0.75 &quot;$n22 setdest 839.0 444.0 20.0&quot;

## Setting The Node Size

$ns_ initial_node_pos $Server1 125
$ns_ initial_node_pos $Server2 125
$ns_ initial_node_pos $n2 70
$ns_ initial_node_pos $n3 70
$ns_ initial_node_pos $n4 40
$ns_ initial_node_pos $n5 70
$ns_ initial_node_pos $n6 70
$ns_ initial_node_pos $n7 70
$ns_ initial_node_pos $n8 70
$ns_ initial_node_pos $n9 70
```

```
$ns_ initial_node_pos $n10 70
$ns_ initial_node_pos $n11 70
$ns_ initial_node_pos $n12 70
$ns_ initial_node_pos $n13 70
$ns_ initial_node_pos $n14 70
$ns_ initial_node_pos $n15 70
$ns_ initial_node_pos $n16 70
$ns_ initial_node_pos $n17 70
$ns_ initial_node_pos $n18 70
$ns_ initial_node_pos $n19 70
$ns_ initial_node_pos $n20 70
```



```
$ns_ initial_node_pos $n21 70
$ns_ initial_node_pos $n22 70

#### Setting The Labels For Nodes

$ns_ at 0.0 &quot;$Server1 label Server1&quot;
$ns_ at 0.0 &quot;$Server2 label Server2&quot;

$ns_ at 0.0 &quot;$n2 label node2&quot;
$ns_ at 0.0 &quot;$n3 label node3&quot;
$ns_ at 0.0 &quot;$n4 label node4&quot;
$ns_ at 0.0 &quot;$n5 label node5&quot;
$ns_ at 0.0 &quot;$n6 label node6&quot;
$ns_ at 0.0 &quot;$n7 label node7&quot;
$ns_ at 0.0 &quot;$n8 label node8&quot;
$ns_ at 0.0 &quot;$n9 label node9&quot;
$ns_ at 0.0 &quot;$n10 label node10&quot;
$ns_ at 0.0 &quot;$n11 label node11&quot;
$ns_ at 0.0 &quot;$n12 label node12&quot;
$ns_ at 0.0 &quot;$n13 label node13&quot;
$ns_ at 0.0 &quot;$n14 label node14&quot;
$ns_ at 0.0 &quot;$n15 label node15&quot;
$ns_ at 0.0 &quot;$n16 label node16&quot;
$ns_ at 0.0 &quot;$n17 label node17&quot;
$ns_ at 0.0 &quot;$n18 label node18&quot;
$ns_ at 0.0 &quot;$n19 label node19&quot;
$ns_ at 0.0 &quot;$n20 label node20&quot;
$ns_ at 0.0 &quot;$n20 label node21&quot;
$ns_ at 0.0 &quot;$n22 label node22&quot;
$n2 color green
$ns_ at 0.0 &quot;$n2 color green&quot;
```

```
$n3 color green
$ns_ at 0.0 &quot;$n3 color green&quot;

$n4 color green
$ns_ at 0.0 &quot;$n4 color green&quot;

$n5 color green
$ns_ at 0.0 &quot;$n5 color green&quot;

$n6 color green
$ns_ at 0.0 &quot;$n6 color green&quot;

$n7 color green
$ns_ at 0.0 &quot;$n7 color green&quot;

$n8 color green
$ns_ at 0.0 &quot;$n8 color green&quot;
$n9 color yellow
$ns_ at 0.0 &quot;$n9 color yellow&quot;

$n10 color yellow
$ns_ at 0.0 &quot;$n10 color yellow&quot;
$n11 color yellow
$ns_ at 0.0 &quot;$n11 color yellow&quot;

$n12 color pink
$ns_ at 0.0 &quot;$n12 color pink&quot;
```

35
Anexo A
A_____

```
$n13 color pink
$ns_ at 0.0 &quot;$n13 color pink&quot;
$n14 color pink
$ns_ at 0.0 &quot;$n14 color pink&quot;
$n15 color pink
$ns_ at 0.0 &quot;$n15 color pink&quot;
$n16 color pink
$ns_ at 0.0 &quot;$n16 color pink&quot;
$n17 color orange
$ns_ at 0.0 &quot;$n17 color orange&quot;
$n18 color orange
$ns_ at 0.0 &quot;$n18 color orange&quot;
$n19 color orange
$ns_ at 0.0 &quot;$n19 color orange&quot;
$n20 color orange
```

```
    $ns_ at 0.0 &quot;$n20 color orange&quot;
    $n21 color orange
    $ns_ at 0.0 &quot;$n21 color orange&quot;
    $n22 color orange
    $ns_ at 0.0 &quot;$n22 color orange&quot;
    $Server1 color maroon
    $ns_ at 0.0 &quot;$Server1 color maroon&quot;

    $Server2 color maroon
    $ns_ at 0.0 &quot;$Server2 color maroon&quot;
    ## SETTING ANIMATION RATE
$ns_ at 0.0 &quot;$ns_ set-animation-rate 12.5ms&quot;
  #  COLORING THE NODES
$n9 color blue
$ns_ at 4.71 &quot;$n9 color blue&quot;
$n5 color blue
$ns_ at 7.0 &quot;$n5 color blue&quot;
$n2 color blue
$ns_ at 7.29 &quot;$n2 color blue&quot;
$n16 color blue
$ns_ at 7.59 &quot;$n16 color blue&quot;
$n9 color maroon
$ns_ at 7.44 &quot;$n9 color maroon&quot;
$ns_ at 7.43 &quot;$n9 label TTLover&quot;
$ns_ at 7.55 &quot;$n9 label \&quot;\&quot;&quot;
$n12 color blue
$ns_ at 7.85 &quot;$n12 color blue&quot;

####  Establishing Communication
    set udp0 [$ns_ create-connection UDP $Server1 LossMonitor $n18 0]
    $udp0 set fid_ 1
    set cbr0 [$udp0 attach-app Traffic/CBR]
```



```
    $cbr0 set packetSize_ 1000
    $cbr0 set interopt_ .07
    $ns_ at 0.0 &quot;$cbr0 start&quot;
    $ns_ at 4.0 &quot;$cbr0 stop&quot;

    set udp1 [$ns_ create-connection UDP $Server1 LossMonitor $n22 0]
    $udp1 set fid_ 1
    set cbr1 [$udp1 attach-app Traffic/CBR]
    $cbr1 set packetSize_ 1000
    $cbr1 set interopt_ .07
    $ns_ at 0.1 &quot;$cbr1 start&quot;
```

```
$ns_ at 4.1 &quot;$cbr1 stop&quot;


set udp2 [$ns_ create-connection UDP $n21 LossMonitor $n20 0]
$udp2 set fid_ 1
set cbr2 [$udp2 attach-app Traffic/CBR]
$cbr2 set packetSize_ 1000
$cbr2 set interopt_ .07
$ns_ at 2.4 &quot;$cbr2 start&quot;
$ns_ at 4.1 &quot;$cbr2 stop&quot;

set udp3 [$ns_ create-connection UDP $Server1 LossMonitor $n15 0]
$udp3 set fid_ 1
set cbr3 [$udp3 attach-app Traffic/CBR]
$cbr3 set packetSize_ 1000
$cbr3 set interopt_ 5
$ns_ at 4.0 &quot;$cbr3 start&quot;
$ns_ at 4.1 &quot;$cbr3 stop&quot;

set udp4 [$ns_ create-connection UDP $Server1 LossMonitor $n14 0]
$udp4 set fid_ 1
set cbr4 [$udp4 attach-app Traffic/CBR]
$cbr4 set packetSize_ 1000
$cbr4 set interopt_ 5
$ns_ at 4.0 &quot;$cbr4 start&quot;
$ns_ at 4.1 &quot;$cbr4 stop&quot;

set udp5 [$ns_ create-connection UDP $n15 LossMonitor $n16 0]
$udp5 set fid_ 1
set cbr5 [$udp5 attach-app Traffic/CBR]
$cbr5 set packetSize_ 1000
$cbr5 set interopt_ 5
$ns_ at 4.0 &quot;$cbr5 start&quot;
$ns_ at 4.1 &quot;$cbr5 stop&quot;

set udp6 [$ns_ create-connection UDP $n15 LossMonitor $n17 0]
$udp6 set fid_ 1
set cbr6 [$udp6 attach-app Traffic/CBR]
$cbr6 set packetSize_ 1000
$cbr6 set interopt_ 5
$ns_ at 4.0 &quot;$cbr6 start&quot;
$ns_ at 4.1 &quot;$cbr6 stop&quot;

set udp7 [$ns_ create-connection UDP $n14 LossMonitor $n4 0]
$udp7 set fid_ 1
set cbr7 [$udp7 attach-app Traffic/CBR]
$cbr7 set packetSize_ 1000
```

```
$cbr7 set interopt_ 5
$ns_ at 4.0 &quot;$cbr7 start&quot;
$ns_ at 4.1 &quot;$cbr7 stop&quot;

set udp8 [$ns_ create-connection UDP $n14 LossMonitor $n9 0]
$udp8 set fid_ 1
```



```
set cbr8 [$udp8 attach-app Traffic/CBR]
$cbr8 set packetSize_ 1000
$cbr8 set interopt_ 5
$ns_ at 4.0 &quot;$cbr8 start&quot;
$ns_ at 4.1 &quot;$cbr8 stop&quot;

set udp9 [$ns_ create-connection UDP $n4 LossMonitor $n3 0]
$udp9 set fid_ 1
set cbr9 [$udp9 attach-app Traffic/CBR]
$cbr9 set packetSize_ 1000
$cbr9 set interopt_ 5
$ns_ at 4.0 &quot;$cbr9 start&quot;
$ns_ at 4.1 &quot;$cbr9 stop&quot;

set udp10 [$ns_ create-connection UDP $n4 LossMonitor $n2 0]
$udp10 set fid_ 1
set cbr10 [$udp10 attach-app Traffic/CBR]
$cbr10 set packetSize_ 1000
$cbr10 set interopt_ 5
$ns_ at 4.0 &quot;$cbr10 start&quot;
$ns_ at 4.1 &quot;$cbr10 stop&quot;

set udp11 [$ns_ create-connection UDP $n9 LossMonitor $n16 0]
$udp11 set fid_ 1
set cbr11 [$udp11 attach-app Traffic/CBR]
$cbr11 set packetSize_ 1000
$cbr11 set interopt_ 5
$ns_ at 4.0 &quot;$cbr11 start&quot;
$ns_ at 4.1 &quot;$cbr11 stop&quot;

set udp12 [$ns_ create-connection UDP $n9 LossMonitor $n10 0]
$udp12 set fid_ 1
set cbr12 [$udp12 attach-app Traffic/CBR]
$cbr12 set packetSize_ 1000
$cbr12 set interopt_ 5
$ns_ at 4.0 &quot;$cbr12 start&quot;
$ns_ at 4.1 &quot;$cbr12 stop&quot;
```

```
    #ANNOTATIONS DETAILS
      $ns_ at 0.0 &amp;quot;$ns_ trace-annotate \&amp;quot;MOBILE NODE MOVEMENTS\&amp;quot;&amp;quot;
    $ns_ at 4.1 &amp;quot;$ns_ trace-annotate \&amp;quot;NODE27 CACHE THE DATA FRO SERVER\&amp;quot;&amp;quot;
      #$ns_ at 4.59 &amp;quot;$ns_ trace-annotate \&amp;quot;PACKET LOSS AT NODE27\&amp;quot;&amp;quot;
      $ns_ at 4.71 &amp;quot;$ns_ trace-annotate \&amp;quot;NODE10 CACHE THE DATA\&amp;quot;&amp;quot;

    ### PROCEDURE TO STOP
    proc stop {} {

                global ns_ tracefd
                $ns_ flush-trace
                close $tracefd
                exec nam aodv.nam &amp;amp;
                exit 0
            }
    puts &amp;quot;Starting Simulation.....&amp;quot;
    $ns_ at 25.0 &amp;quot;stop&amp;quot;
    $ns_ run


} elseif {$top == 2} {
```



```
# Define options
set val(chan)          Channel/WirelessChannel    ;# channel type
set val(prop)          Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)         Phy/WirelessPhy            ;# network interface type
set val(mac)           Mac/802_11                 ;# MAC type
set val(ifq)           Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)            LL                         ;# link layer type
set val(ant)           Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)        50                         ;# max packet in ifq
set val(nn)            7                          ;# number of mobilenodes
set val(rp)            DSDV                       ;# routing protocol
set val(x)             500                ;# X dimension of topography
set val(y)             400                ;# Y dimension of topography
set val(stop)          100              ;# time of simulation end
set ns          [new Simulator]
set tracefd       [open  dsdv.tr  w]
set namtrace      [open dsdv.nam w]
$ns use-newtrace
```

```
$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)


Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 1.0
Antenna/OmniAntenna set Gr_ 1.0
# set up topography object
set topo       [new Topography]

$topo load_flatgrid $val(x) $val(y)

create-god $val(nn)

#
#  Create nn mobilenodes [$val(nn)] and attach them to the channel.
#

# configure the nodes
$ns node-config -adhocRouting $val(rp) \
   -llType $val(ll) \
   -macType $val(mac) \
   -ifqType $val(ifq) \
   -ifqLen $val(ifqlen) \
   -antType $val(ant) \
   -propType $val(prop) \
   -phyType $val(netif) \
   -channelType $val(chan) \
   -topoInstance $topo \
   -agentTrace ON \
   -routerTrace ON \
   -macTrace OFF \
   -movementTrace ON

   for {set i 0} {$i &lt; $val(nn) } { incr i } {
      set node_($i) [$ns node]
   }

# Provide initial location of mobilenodes
$node_(0) set X_ 5.0
$node_(0) set Y_ 5.0
```

```
$node_(0) set Z_ 0.0

$node_(1) set X_ 490.0
$node_(1) set Y_ 285.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 150.0
$node_(2) set Y_ 240.0
$node_(2) set Z_ 0.0

$node_(3) set X_ 250.0
$node_(3) set Y_ 240.0
$node_(3) set Z_ 0.0

$node_(4) set X_ 180.0
$node_(4) set Y_ 70.0
$node_(4) set Z_ 0.0
$node_(5) set X_ 100.0
$node_(5) set Y_ 70.0
$node_(5) set Z_ 0.0

$node_(6) set X_ 380.0
$node_(6) set Y_ 70.0
$node_(6) set Z_ 0.0

# Generation of movements
$ns at 10.0 &quot;$node_(0) setdest 250.0 250.0 10.0&quot;
$ns at 15.0 &quot;$node_(1) setdest 45.0 285.0 10.0&quot;
$ns at 29.0 &quot;$node_(2) setdest 480.0 300.0 10.0&quot;
$ns at 70.0 &quot;$node_(3) setdest 180.0 30.0 10.0&quot;
$ns at 80.0 &quot;$node_(4) setdest 80.0 30.0 10.0&quot;
$ns at 90.0 &quot;$node_(5) setdest 98.0 30.0 10.0&quot;
$ns at 80.0 &quot;$node_(6) setdest 50.0 30.0 10.0&quot;

# Set a TCP connection between node_(0) and node_(1)
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(1) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 10.0 &quot;$ftp start&quot;

# Set a TCP connection between node_(0) and node_(2)
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
```

```
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(2) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 10.0 &quot;$ftp start&quot;
# Set a TCP connection between node_(0) and node_(3)
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(3) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
```



```
$ns at 10.0 &quot;$ftp start&quot;
# Set a TCP connection between node_(0) and node_(4)
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(4) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 10.0 &quot;$ftp start&quot;
# Set a TCP connection between node_(0) and node_(5)
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(5) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 10.0 &quot;$ftp start&quot;
# Set a TCP connection between node_(0) and node_(6)
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(6) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 10.0 &quot;$ftp start&quot;
```

```
# Define node initial position in nam
for {set i 0} {$i &lt; $val(nn)} { incr i } {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}

# Telling nodes when the simulation ends
for {set i 0} {$i &lt; $val(nn) } { incr i } {
    $ns at $val(stop) &quot;$node_($i) reset&quot;;
}

# ending nam and the simulation
$ns at $val(stop) &quot;$ns nam-end-wireless $val(stop)&quot;
$ns at $val(stop) &quot;stop&quot;
$ns at 150.01 &quot;puts \&quot;end simulation\&quot; ; $ns halt&quot;
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
    #Execute nam on the trace file
    exec nam dsdv.nam &amp;
    exit 0
}

$ns run

} elseif {$top == 3} {
set opt(chan)           Channel/WirelessChannel    ;# channel type
set opt(prop)           Propagation/TwoRayGround   ;# radio-propagation model
set opt(netif)          Phy/WirelessPhy            ;# network interface type

41
Anexo                                                                    A
A_____________________________________________________________________
set opt(mac)            Mac/802_11                 ;# MAC type
set opt(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
set opt(ll)             LL                         ;# link layer type
set opt(ant)            Antenna/OmniAntenna        ;# antenna model
set opt(ifqlen)         50                         ;# max packet in ifq
set opt(nn)             3                          ;# number of mobilenodes
set opt(rp)             DSR                        ;# routing protocol
set opt(x)              500                ;# X dimension of topography
set opt(y)              400                ;# Y dimension of topography
set opt(stop)           150                        ;
set ns              [new Simulator]
#Creating trace file and nam file
```

```
set tracefd       [open Dsr.tr w]
set namtrace      [open dsr.nam w]
$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $opt(x) $opt(y)
# set up topography object
set topo       [new Topography]
$topo load_flatgrid $opt(x) $opt(y)
create-god $opt(nn)
# configure the nodes
        $ns node-config -adhocRouting $opt(rp) \
                -llType $opt(ll) \
                -macType $opt(mac) \
                -ifqType $opt(ifq) \
                -ifqLen $opt(ifqlen) \
                -antType $opt(ant) \
                -propType $opt(prop) \
                -phyType $opt(netif) \
                -channelType $opt(chan) \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace OFF \
                -movementTrace ON

     for {set i 0} {$i &lt; $opt(nn) } { incr i } {
          set node_($i) [$ns node]
     }
# Provide initial location of mobilenodes
$node_(0) set X_ 5.0
$node_(0) set Y_ 5.0
$node_(0) set Z_ 0.0
$node_(1) set X_ 490.0
$node_(1) set Y_ 285.0
$node_(1) set Z_ 0.0
$node_(2) set X_ 150.0
$node_(2) set Y_ 240.0
$node_(2) set Z_ 0.0
# Generation of movements
$ns at 10.0 &quot;$node_(0) setdest 250.0 250.0 3.0&quot;
$ns at 15.0 &quot;$node_(1) setdest 45.0 285.0 5.0&quot;
$ns at 110.0 &quot;$node_(0) setdest 480.0 300.0 5.0&quot;
```



```
# Set a TCP connection between node_(0) and node_(1)
set tcp [new Agent/TCP/Newreno]
```

```
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(1) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 10.0 &quot;$ftp start&quot;

# Define node initial position in nam
for {set i 0} {$i &lt; $opt(nn)} { incr i } {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}
# Telling nodes when the simulation ends
for {set i 0} {$i &lt; $opt(nn) } { incr i } {
    $ns at $opt(stop) &quot;$node_($i) reset&quot;;
}
# ending nam and the simulation
$ns at $opt(stop) &quot;$ns nam-end-wireless $opt(stop)&quot;
$ns at $opt(stop) &quot;stop&quot;
$ns at 150.01 &quot;puts \&quot;end simulation\&quot; ; $ns halt&quot;
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
exec nam dsr.nam &amp;
exit 0
}
$ns run

}
puts &quot;want to continue (0/1)&quot;
set a [gets stdin]
}
```