UNIVERSIDADE FEDERAL DO ABC

VINICIUS ZANIN

AVALIAÇÃO DE DESEMPENHOS DE PROTOCOLOS DE ROTEAMENTO EM REDES AD-HOC SEM FIO COGNITIVAS.

SANTO ANDRÉ, SP

2017

VINICIUS ZANIN

AVALIAÇÃO DE DESEMPENHOS DE PROTOCOLOS DE ROTEAMENTO EM REDES AD-HOC SEM FIO COGNITIVAS.

Trabalho de graduação do curso
de Engenharia de Informação da
Universidade Federal do ABC
sob orientação do
Prof. Dr. João Henrique Kleinschmidt

SANTO ANDRÉ, SP
2017

RESUMO

Este projeto visa realizar testes da real eficiência de uma rede cognitiva, buscando comparar o desempenho de protocolos Ad-hoc. A primeira etapa consiste na construção de uma topologia de rede, simulando ela no simulador de redes NS-2 de forma cognitiva. Assim foram obtidos dados para que seja possível fazer uma análise de desempenho para comparação destas redes. Os resultados demonstraram que o protocolo de roteamento DSR tem um desempenho superior ao AODV nos cenários simulados.

Palavras-chave: NS-2. Rede cognitiva. Self-aware Ad-Hoc. Wireless. CRCN

1. Introdução

Com a evolução da necessidade da sociedade em manter-se conectada a uma rede de computadores, seja através de um desktop ou um smartphone, surge um crescente problema nas redes sem fio: a limitação da banda espectral. Neste cenário, foi desenvolvido um novo conceito de rede: as Redes Cognitivas. Este modelo de rede foi inspirado nos sistemas biológicos, nos quais os indivíduos deste sistema possuem total autonomia e se auto-organizam mesmo sem regras explícitas. são capazes de se adaptarem e superar possíveis eventos catastróficos. Levando este conceito para uma aplicação prática, é possível criar uma rede com mais resiliência, eficiência e otimização. Essa rede seria capaz de coletar dados sobre o ambiente, analisá-los e adaptar-se a este ambiente e, principalmente, ela seria capaz de aprender com eventos e com equipamentos novos inseridos no ambiente da rede. Com base neste aprendizado, geralmente uma rede cognitiva tem a capacidade de planejar a utilização futura dos recursos disponíveis. [1]

Este modelo de rede não visa a substituição total da interação humana, mas sim um melhor aproveitamento desta interação, restringindo a intervenção humana apenas para problemas mais complexos. A rede geralmente demonstra capacidade de autoconhecimento, capacidade de conhecer a si mesma e o ambiente, e conforme os dados sejam coletados e os problemas corrigidos, ela adquire conhecimento, evitando assim problemas futuros, além de efetuar mudanças com base nos conhecimentos previamente aprendidos. Ou seja, ela tem a capacidade de tomar decisões. [1]

A eficiência da rede é aumentada através de algoritmos que selecionam a melhor rota de acordo com a política previamente definida pelo administrador da rede e evitam congestionamento de dados. Essas políticas podem variar entre a rota que gasta menos energia, a rota que oferece melhor qualidade de link com menor preço, entre outros. [1]

É possível inspirar-se nos comportamentos de certos sistemas biológicos para aplicá-los a redes cognitivas, porém, é necessário tomar certos cuidados, evitando simplesmente imitar estes sistemas. Ao utilizá-los temos que otimizar e adaptar seus conceitos. Por exemplo, na evolução (que leva até milhares de anos) os sistemas biológicos que não conseguiram se adaptar ao ambiente deixaram de existir. Não poderíamos permitir que acontecesse o mesmo com uma rede de computadores. São necessários estudos aprofundados antes de implementar um sistema desta maneira. [1]

1.1 Motivação

Nos últimos anos houve um constante crescimento das redes sem fio e, com isso, o problema de interferência entre estas é cada vez mais comum, devido a várias redes operarem na

mesma frequência. Este problema poderia ser melhorado com a implementação do conceito de redes cognitivas, onde a própria rede poderia verificar os espectros de transmissão ocupados e realizar uma mudança para que as várias redes existentes ocupem espectros diferentes. Para que isso fosse possível, seria necessário que os equipamentos que gerenciam as redes sem fio se comunicassem para formar uma rede cognitiva. Isto leva a alguns dilemas: a dificuldade de comunicação entre estes dispositivos por conta de terem fabricantes distintos, possíveis riscos de segurança caso exista um dispositivo malicioso e necessidade de compatibilidade com os dispositivos atuais.

É necessário, portanto, realizar estudos para mensurar o desempenho destas redes cognitivas em cenários reais. Deste modo, será possível avaliar as vantagens da adoção deste tipo de rede.

1.2 Objetivos

O objetivo deste trabalho é avaliar o desempenho de protocolos de roteamento redes cognitivas ad-hoc sem fio. Para isso, será construída uma topologia fixa para mensurar esse desempenho em diferentes situações. Para fazer a simulação de uma rede cognitiva será utilizado simulador NS-2 [2] com a extensão *Cognitive Radio Netowrk* (CRCN). [3]

1.3 Organização do texto

A organização deste trabalho foi feita da seguinte forma: Capitulo 2 é introduzido os principais conceitos de redes sem fio, redes cognitivas e protocolos ad-hoc; No Capitulo 3 é explicado sobre o simulador NS-2[2] e as modificações feita pela extensão CRCN[3]; O Capitulo 4 é dedicado a descrever as ferramentas e técnicas utilizadas para configuração do ambiente de simulação; As topologias definidas, parâmetros da simulação e métricas estão descritos no Capitulo 5; No Capitulo 6 estão os resultados da simulação e breve análise destes; Por fim, no Capitulo 7 contém a conclusão e considerações sobre trabalhos futuros.

2. Redes sem Fio

2.1 Redes Cognitivas

As pesquisas sobre as teorias estudadas neste trabalho são relativamente recentes, sendo que o modelo de redes cognitivas tem passado por aperfeiçoamento no decorrer dos últimos anos. Portanto, é importante definir o que é uma rede cognitiva,

> Redes cognitivas são redes que podem alterar dinamicamente suas funcionalidades e/ou sua topologia de acordo com a mudança de necessidades dos seus usuários, levando em conta as condições atuais do ambiente. Essa modificação dinâmica é feita de acordo com as regras comerciais e as políticas regulatórias aplicáveis. [4]

Essa definição é importante para diferenciar redes cognitivas de redes autônomas, sendo que a segunda, apesar de possuir inteligência artificial para tomar decisões sem interferência humana, está restrita a um laço fechado configurado previamente de acordo com políticas definidas.

Um dos conceitos mais importante para entender o funcionamento de Redes Cognitivas é o de "*Self-Organization*". Nos sistemas biológicos pode-se observar que, apesar da autonomia dos indivíduos, em muitas atividades coletivas eles se auto-organizam mesmo não havendo regras explícitas de como agir individualmente. As principais vantagens destes sistemas biológicos são a escalabilidade e a resiliência, porém, ela não possui o mesmo desempenho de uma rede otimizada, sendo que seu desempenho possui uma variação considerável. A principal vantagem desta auto-organização se dá em situações adversas, nas quais redes comuns provavelmente seriam interrompidas enquanto redes cognitivas se adaptariam a este cenário e manteriam suas funções.

Para exemplificar como isto funcionaria em uma rede de computadores, é possível adotar o cenário hipotético onde seriam disparados vários pacotes por "n" rotas. A rota de envio de pacotes que retornasse o primeiro pacote-resposta, ou comprovante de recebimento do pacote enviado, mais rapidamente seria considerada a mais otimizada, sendo que nem sempre ela seria a mais curta, porém, no momento seria a mais rápida. [1]

O primeiro requisito fundamental para uma rede cognitiva [wireless] é a capacidade de varrer toda sua banda de espectro detectando a presença ou ausência de usuários primários. Este processo é chamado detecção de espectro e é executado localmente por usuários secundários ou

coletivamente por um grupo de usuários secundários. Assim, são aferidos dados da rede, como relação sinal ruído (SNR), taxa de erro, etc. Uma vez que esses dados foram coletados, analisados e a melhor banda escolhida, os usuários (ou redes) secundários ao detectarem uma transmissão de um usuário primário adotam o processo chamado *Spectrum Handoff* que consiste em mudar a faixa de banda do espectro licenciado para outra banda. [5]

As redes cognitivas são divididas basicamente em dois grupos: centralizadas e distribuídas. Nas centralizadas, os usuários secundários são geridos por bases secundárias que estão ligadas via cabo ao um *backbone*. Já nas distribuídas, os usuários comunicam-se uns com os outros de uma maneira ad-hoc. O sensoriamento espectral é feito de forma colaborativa em redes distribuídas. Este tipo de arquitetura também engloba a coexistência de duas ou mais redes sem fio operando em bandas não licenciadas (Exemplo: IEEE 802.11 com IEEE 802.16). [1]

Para operar este tipo de rede são utilizadas algumas técnicas:

Sensoriamento espectral: Habilidade de varrer a banda de espectro e identificar os canais vagos para transmissão. [5]

Análise de decisão do espectro: Cada banda do espectro possui características únicas como o range de frequência e número de usuários (Primários e secundários). O sensoriamento espectral determina uma lista de bandas de espectro livres, então os usuários secundários decidem qual banda é mais apropriada desta lista. Essa decisão dependerá de vários fatores como perda de pacotes, delay, etc. [5]

Mobilidade espectral: Refere-se à agilidade com que as redes cognitivas mudam dinamicamente o acesso entre os espectros. Por exemplo, os usuários secundários não têm acesso contínuo garantido em bandas licenciadas e têm que mudar frequentemente para outras que estão vagas. Essa habilidade pode ser afetada por alguns fatores, como o atraso na troca entre espectros (o que afeta diversos protocolos), a demora para detectar uma transmissão primária e varrer os espectros vagos ("*Spectrum Handoff*"). [5]

2.1.2 Tipos de redes cognitivas

*Centralizadas***:** Em uma rede cognitiva centralizada os usuários secundários são orientados pela infraestrutura. Isto é, a rede é dividida em células e cada uma é administrada por uma estação secundária. Essas estações controlam o acesso e os usuários secundários. Estes usuários são sincronizados com essas estações e podem realizar sensoriamento espectral periodicamente. As estações podem ser interconectadas através de um backbone de rede. [5]

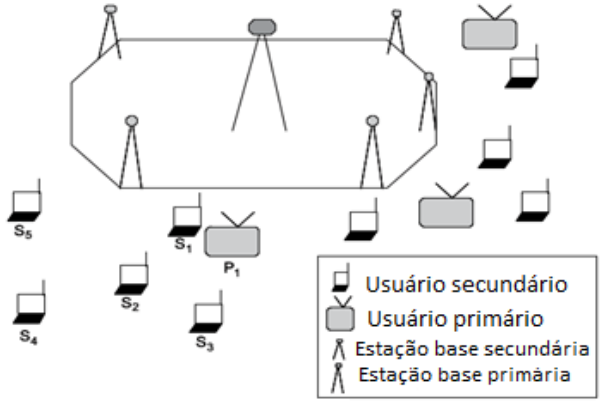

Figura 1- Rede Cognitiva centralizada (Imagem retirada do livro [5] página 274)

Descentralizadas: Os usuários secundários não são interconectados por uma infraestrutura, e sim, se comunicam de maneira ad-hoc. Se dois usuários secundários estão próximos eles podem trocar informações diretamente, enquanto os distantes trocam com múltiplos saltos. Neste tipo de rede, os usuários secundários tomam sua decisão sobre bandas de espectros, gasto de transmissão, etc. É baseado em observações locais ou cooperando em algumas funções para conseguir um melhor desempenho para todos os outros usuários. [5]

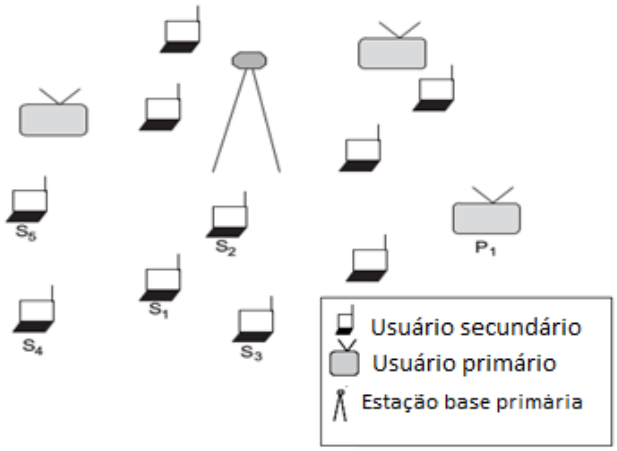

Figura 2- Rede Cognitiva descentralizada (Imagem retirada do livro [5] página 274)

2.2 Redes IEE 802.11

As redes sem fio possuem interferências que redes cabeadas não possuem. Elas podem causar perda de pacote, congestionamento e embaralhamento. As principais fontes de interferências são:

Variação na intensidade do sinal: As ondas magnéticas são afetas pelo meio em que são transmitidas, portanto, caso exista "n" materiais que impeçam ou dificulte a propagação das mesmas haverá atenuação do sinal. E também existe o fator da distância entre o receptor e emissor, sendo que quanto maior a distância menor a intensidade do sinal de comunicação. [6]

Interferência de outros sinais: Como o IEE 802.11xx opera em uma frequência não licenciada (2.4GHz ou 5 GHz) outros equipamentos podem utilizar esta mesma frequência para transmissão (Telefone sem fio por exemplo). [6]

Propagação multivias: Esta interferência ocorre quando há reflexão do sinal (em solos, objetos e etc.). Isto faz com que o sinal faça um caminho com distância diferente entre o emissor e receptor. [6]

Como esse tipo de rede sofre muito mais com erro de bits, devido aos problemas citados anteriormente, são empregados diversos códigos corretores de erro para contornar esta situação. Como por exemplo, o código de verificação de redundância cíclica (CRC).

As redes sem fio utilizam o protocolo *Code Division Mutiple Access* (CDMA) para que os sinais enviados por diferentes emissores não interfiram nos receptores. Esta técnica se baseia em multiplicar cada bit enviado por um sinal que muda com a taxa de "*chipping*", maior que a taxa da sequência original de bit de dados. [6]

O padrão IEE 802.11xx, é o utilizado na grande maioria das redes LANs sem fio atualmente. A Figura 3 exemplifica a arquitetura deste padrão de rede:

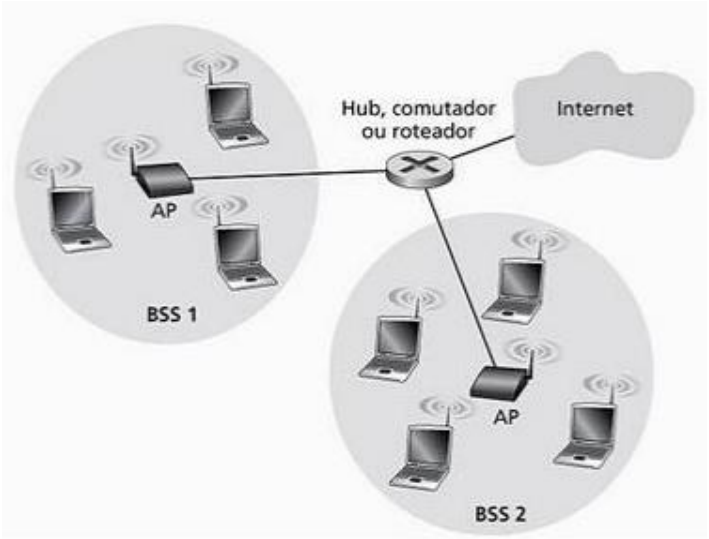

Figura 3 – Rede IEE 802.11 padrão (*Retirado do Livro [6] página 387)*

Cada BSS (conjunto básico de serviço) irá conter um ponto-de-acesso (AP) que é a estação base central e dispositivos, por exemplo, notebooks, celulares, etc. As BSS irão transmitir os dados para um comutador central (Roteador, hub, etc.) que terá a função de fazer a conexão com a internet.

As redes IEE 802.11xx também podem atuar de forma ad hoc, ou seja, os dispositivos se comunicam entrem si sem um AP. Na Figura 4 pode-se observar um exemplo:

Figura 4 – Rede IEE 802.11 Ad-hoc (Página 388 de [6]).

O 802.11 possui diversas versões, elas operam em faixas diferentes de frequência e possuem taxa de dados distintas, sendo que as versões mais recentes possuem taxas maiores como é mostrado na Tabela 1.

Tabela 1 – Padrões 802.11 [6]

Padrão	Faixa de frequência	Taxa de dados
802.11b	2.4-2.485 GHz	Até 11 Mbps
802.11ª	5.1 – 5.8 GHz	Até 54 Mbps
802.11g	2.4 – 2.485 GHz	Até 54 Mbps
802.11n	2.4 – 5 GHz	Até 200 Mbps

O padrão 802.11n também possui outras características como a utilização de múltiplas antenas para receber e transmitir o sinal (*Multiple-Input Multiple-Outpt* (MIMO)), portanto, conseguindo uma taxa de transferência de dados superior. Além disto, esses novos padrões utilizando codificação digital com esquema de multiplexação por divisão de frequência, sendo que a principal utilizada é a *Orthogonal frequency-division multiplexing* (OFDM).

Quando um dispositivo se associa a uma estação (que pode ser tanto um outro dispositivo como um AP) é necessária a utilização de um protocolo para gerenciar acessos múltiplos e coordenar as transmissões. Isto é feito na camada MAC e o protocolo adotado é o CSMA/CA que possui prevenção de colisão. [6]

Na camada de enlace é utilizado reconhecimento/retransmissão (ARQ) para que se minimize as taxas de erro de bit. Este processo é feito quando a estação destino receber um quadro verificado pelo CRC. Então, é esperado um curto período de tempo (Espaçamento Curto Interquadros (SIFS)) e então o quadro é devolvido. Caso não se obtenha resposta dentro de um período, a transmissora irá assumir que ocorreu um erro e assim retransmitir utilizando CSMA/CA *(Carrier sense multiple access with collision avoidance)* para acessar o canal. Se depois de um certo número de retransmissões a transmissora não receber um reconhecimento, ela irá desistir de transmitir o quadro e vai descartá-lo. [7]

O CSMA/CA está exemplificado na Figura 5 e atua da seguinte forma:

- Verifica se a estação está ociosa, se estiver transmite o quadro e aguarda SIFS.

- Caso a estação não estiver livre, ele irá escolher um valor aleatório de backoff (que é o tempo que uma estação vai aguardar para realizar uma retransmissão após ocorrer uma colisão) e fará uma contagem a partir deste valor se perceber que o canal estiver ocioso. Enquanto o canal não estiver ocioso, a contagem não começa.

- Quando a contagem zerar, será transmitido o quadro e então a fonte irá esperar um ACK.

- Se o ACK chegar, a estação irá saber que o quadro foi entregue corretamente. Voltando assim para a etapa de verificação de ociosidade caso existam outros quadros a serem

enviados. Caso contrário ela voltará novamente para a fase de backoff selecionando um valor maior de intervalo aleatoriamente. [4]

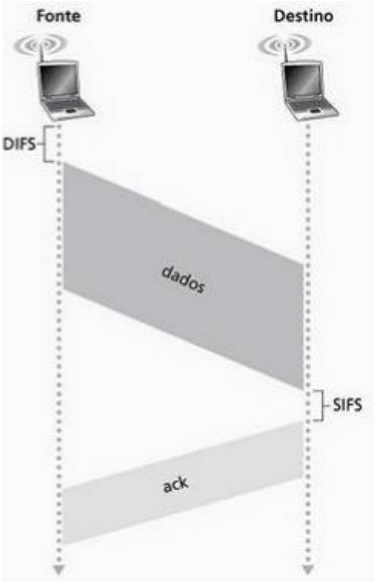

Figura 5- Transmissão utilizando CSMA/CA (Imagem retirada de [6] página 391).

2.3 Problemas em redes Ad-hoc sem fio.

Os dispositivos em um rede ad-hoc compartilham um canal de transmissão comum. Portanto, esse tipo de rede enfrenta problemas diferentes de redes cabeadas ou wireless IEE 802.11 padrão. Assim, o protocolo MAC precisa ser modificado para que esses problemas sejam minimizados ou resolvidos. Os principais problemas são: [7]

Eficiência de banda: Como o espectro é limitado, a banda disponível para comunicação é muito limitada. Portanto, o protocolo MAC tem que ser adaptador, para utilizar esta banda de maneira eficiente. Evitando que haja um overhead da banda disponível. [7]

Suporte a qualidade de serviço: Em uma rede ad-hoc os dispositivos estão em constante movimento, portanto, é muito difícil prover qualidade de serviço (QoS), pois uma vez que o dispositivo mude de região a reserva de banda se torna inválida. [7]

Sincronização: A sincronização é muito importante para que haja troca de pacotes entre os dispositivos. Porém, o controle desta sincronização não pode consumir muita banda. [7]

Terminais escondidos e expostos: O problema de terminal escondido se refere a colisão de pacotes em um dispositivo receptor. Isso se dá pois pode existir outros dispositivos transmitindo pacotes simultaneamente que não estão no alcance do transmissor, porém, estão no do receptor.

Um nó pode ter sua transmissão bloqueado pelos nós vizinhos, quando estes estão transmitindo pacotes: o problema chamado de terminal exposto.

Estes problemas reduzem significativamente a taxa de transferência da rede. [7]

Mobilidade: Como a rede não possui um ponto de acesso central, os dispositivos têm que trocar informações sobre posicionamento, fazendo assim que seja necessário que o protocolo MAC reduza o excesso de troca de informações para que não haja overhead. E também devido esta mobilidade a potência de sinal pode ser reduzida, assim afetando a taxa de transferência. [7]

2.4 Protocolos de roteamento para redes Ad-hoc.

Para superar esses desafios das redes Ad-hoc, os protocolos que são utilizados têm que possuir algumas características: [7]

1. É necessário que o sistema seja totalmente distribuído, pois com roteamento centralizado tem que ter um alto controle de *"overhead"*. Quando se tem um roteamento distribuído, existe uma maior tolerância a falha.
2. Tem que ser adaptativo, pois a topologia está em constantes mudanças devido a mobilidade dos nós.
3. A decisão e manutenção da rota deve envolver o mínimo número de nós possíveis e assim cada nó na rede terá um acesso rápido às rotas, ou seja, é desejado que exista um tempo mínimo de configuração na conexão.
4. Tem que ser localizado, pois a manutenção do estado geral da rede envolve um grande controle de propagação de "*overhead"*.
5. Não devem possuir laços e rotas obsoletas.
6. O número de colisões de pacote tem que ser mínimo, limitado ao número de transmissões feitas por cada nó. E cada transmissão tem que ser confiável, para reduzir a perda de pacotes e evitar que existam rotas obsoletas.
7. Tem que existir uma convergência paras as rotas mais otimizadas, para que essa rede seja estável. E esta convergência tem que acontecer de forma rápida.
8. Os recursos como largura de banda, capacidade computacional, memoria e etc. Tem que ser utilizados de forma otimizados por serem limitados.
9. Todos os nós devem tentar armazenar apenas informações referentes a topologia local estável. Desta forma as mudanças frequentes de topologia em partes da rede as quais o nó não está relacionado, não irá afetar este nó. Ou seja, mudanças em partes distantes na rede não vão causar atualizações desnecessárias nas informações mantidas pelos nós.
10. O protocolo deve ser capaz de fornecer um certo nível de QoS, que é exigido pelas aplicações e também deve ter suporte a tráfego sensível ao tempo.

2.4.2 Classificação dos protocolos de roteamento.

Os protocolos para redes ad-hoc sem fio podem ser classificados em vários tipos considerando diversos critérios. Muitas vezes essas classificações não são exclusivas, assim, existem protocolos que se encaixam em mais de uma classe. [7]

Portanto, este trabalho se focará nos protocolos que utilizam as informações do roteamento para fazer a atualização da rota. Estes podem ser classificados em três grandes categorias, que são:

Protocolos de roteamento proativos ou orientados por tabelas: Nesta categoria os protocolos mantêm as informações sobre a topologia da rede em forma de tabelas de roteamento, trocando informações periodicamente sobre o roteamento. As informações sobre roteamento são enviadas para todos na rede. E sempre que um nó for fazer uma transmissão, ele irá utilizar um algoritmo de busca de caminho nestas tabelas que ele mantém. Os protocolos "*Distance-vector routing protocol"* (DSDV)*, "wireless routing protocol"* (WRP) e *"source-tree adaptive routing protocol"* (START) são exemplos de protocolos de roteamento proativo. [7]

Protocolos de roteamento reativos ou sob demanda: Os protocolos que estão nesta categoria não mantêm informações sobre a topologia da rede. Eles obtêm os caminhos conforme é necessário, utilizando um processo de estabelecimento de conexão entre os nós. Assim os nós não trocam informações de roteamento periodicamente. Os principais protocolos de roteamento ativo são "*Dynamic Source Routing Protocol"* (DSR) e "*Ad hoc On-Demand Distance Vector Routing"* (AODV). [7]

Protocolos de roteamento híbridos: Esta categoria enquadra os protocolos que combinam as melhores características das duas categorias anteriores. Os nós próximos são classificados como uma zona de roteamento. Dentro desta zona é utilizada a abordagem de roteamento orientado por tabelas e para nós que estão fora desta zona é utilizada a abordagem sob demanda. Os protocolos *"Core extraction distrbuted ad hoc routing"* (CEDAR), *"Zone Routing Protocol"* (ZRP) e "*Zone-Based Hierachical Link State Routing Protocol"* (ZHLS) são exemplos roteamento híbrido. [7]

2.4.3 Protocolos utilizados.

Para realizar os testes de desempenho neste trabalho foram utilizados protocolos sob demanda, devido ao fato deles se adaptarem melhor a mudança de rotas e as topologias utilizadas neste trabalho não serem grandes, não sendo necessária a utilização de protocolos híbridos. Então,

foram selecionados os protocolos *Dynamic Source Routing Protocol* e *Ad hoc On-Demand Distance Vector Routing* por possuirem uma implementação menos complexa no simulador de rede NS-2.

2.4.3.1 *Dymanic Source Routing Protocol (DSR).*

O DSR consiste em um protocolo sob demanda que foi projetado para restringir a largura de banda consumida pelos pacotes de controle em uma rede sem fio Ad hoc. Isso é feito eliminando as mensagens periódicas de atualização de tabela. A maior diferença entre este protocolo e os outros sob demanda é que ele não exige transmissões periódicas de pacotes de sinalização (*hellopacket)*, que são utilizados para informar os demais nós da sua presença. Assim como todos os protocolos sob demanda, durante a seleção da rota são enviados diversos pacotes de requisição na rede, e quando o nó destino receber este pacote ele responderá enviando um pacote de resposta. O conceito básico por trás deste protocolo é que durante a fase de construção da rota a rede é inundada com pacotes de *RouteRequest.* Portanto, quando o nó de destino receber esta requisição ele irá devolver um pacote do tipo *RouteReply*, que vai conter as informações da rota percorrida pelo pacote *RouteRequest.* [7]

Para um melhor entendimento, considere um nó que vai realizar uma transmissão, porém, este não tem uma rota definida até o destino. Quando os pacotes de dados forem começar a ser transmitidos, é iniciado o processo de envio de pacotes de *RouteRequest* para a rede. Cada nó que estiver recebendo esse pacote vai retransmiti-lo para seus vizinhos, ao menos que esse pacote já tenha enviado por este nó, ele seja o nó de destino ou a contagem de tempo de vida (TTL) tenha sido excedida. Cada *RouteRequest* carrega um número gerado pelo nó de origem e pelo caminho que ele atravessou. Quando um nó receber o *RouteRequest* ele verifica esse número antes de retransmitir. Isso é feito para avaliar se não é um pacote duplicado, se for, ele não será transmitido. Essa técnica evita formações de laços e transmissões desnecessárias pelos nós intermediários. Portanto, todos os nós farão transmissão durante esta fase, com exceção do nó de destino. Quando o destino receber o *RouteRequest* ele responderá utilizando a rota que o pacote percorreu até chegar nele. [7]

Algumas técnicas de otimização foram implementadas no DSR básico para que seu desempenho seja maior. Uma dessas técnicas é a utilização de caches que armazenam todas as informações possíveis sobre a rota de origem que estão contidas nos pacotes de dados. Caso os nós atuem de forma promíscua (modo em que os nós podem receber pacotes que são enviados para os vizinhos e não são adereçados para eles) eles podem aprender sobre as rotas vizinhas que o pacote percorreu e assim na fase de manutenção de rota são evitadas rotas com falhas. Esse cache é utilizado durante a fase de construção da rota, se quando um nó intermediário receber um *RouteRequest* e já

possuir as informações da rota para o destino armazenada em cache, ele devolverá um *RouteReply* com todas essas informações para o nó de origem. Por fim, foi implementado um algoritmo exponencial de *backoff* que é utilizado para evitar um "spam" de *RouteRequest* na rede quando o nó de destino estiver em outro conjunto disjuntivo. E também é permitido que haja superposição de pacotes, para que seja possível a transmissão de *RouteRequests* e pacotes de dados simultaneamente na rede. [7]

A Figura 6 traz um exemplo da fase de construção de rota utilizando o DSR básico sem otimizações. O nó 1 pretende transmitir para o nó 15, assim é iniciado o processo de envio de *RouteRequest*, os nós intermediários irão se comunica entre si até o pacote chegar ao nó 15. Assim são devolvidos três caminhos possíveis para o nó 1, que irá selecionar o Caminho 2 por ser menor.

Já na Figura 7, está exemplificado uma situação que após o estabelecimento do Caminho 2, ocorre uma falha entre o nó 12 e 15. Quando isso acontece é gerado um pacote de *RouteError* para todos os nós adjacentes, que irão informar o nó de origem sobre essa interrupção. Assim será reiniciado o estabelecimento de rota, o Caminho 2 será retirado do cache dos nós intermediários e será utilizado um outro caminho.

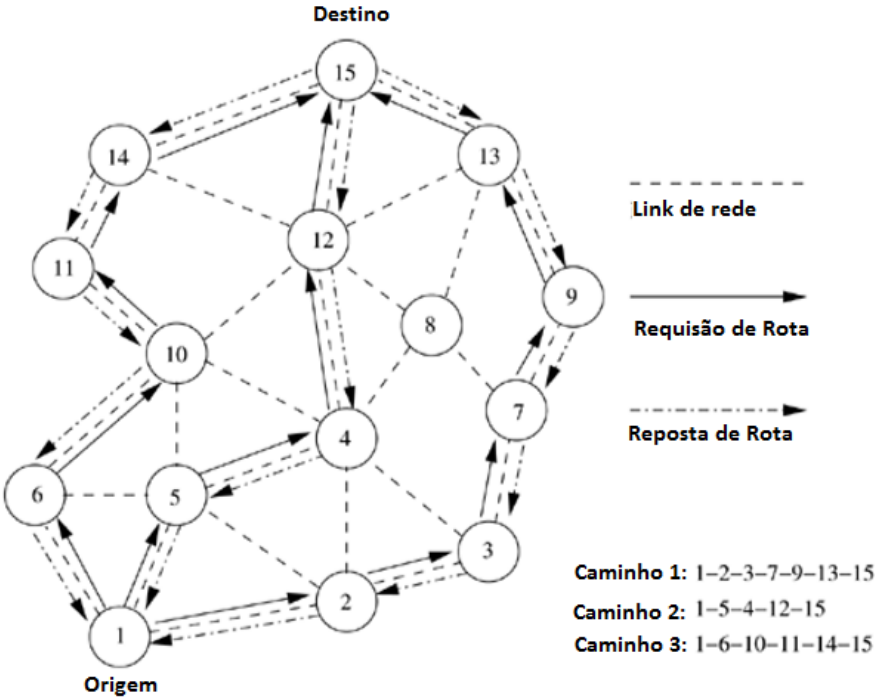

Figura 6 – Estabelecimento de rota com DSR. [7]

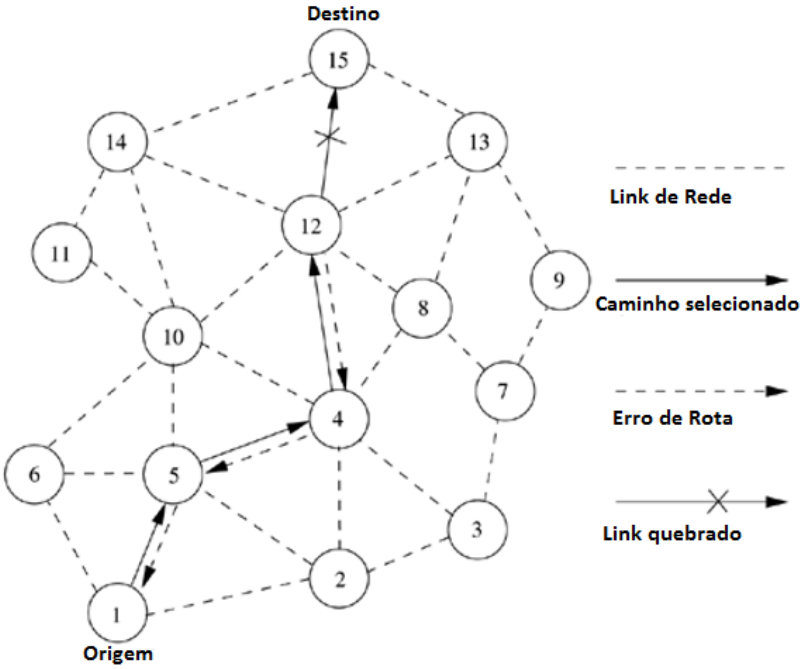

Figura 7 – Manutenção de rota com DSR. [7]

Esse protocolo possui algumas desvantagens, por exemplo, quando o mecanismo de manutenção de rota não corrigir localmente o caminho quando existe uma falha, o atraso para fazer a conexão é maior do que protocolos guiados por tabelas, apesar de funcionar bem em ambientes de baixa mobilidade os estáticos se degrada rapidamente com o aumento de mobilidade e pode causar uma sobrecarga (*overhead)* na rede devido ao mecanismo de *RouteRequest*, sendo que essa sobrecarga é diretamente proporcional ao comprimento do caminho. [7]

2.4.3.2 *Ad hoc On-Demand Distance Vector Routing* (AODV)

AODV é um protocolo que usa a abordagem sob demanda para encontrar rotas. Portanto, a rota só é estabelecida quando for requisitada pelo nó de origem para transmitir pacotes de dados. Ele utiliza o número de sequência do destino para identificar o caminho mais recente. [7]

A principal diferença entre o AODV e DSR é que o DSR utiliza roteamento de origem, pois um pacote de dados carrega o caminho completo percorrido. Contudo no AODV, o nó de origem e os nós intermediários armazenam a informação do próximo salto que corresponde ao fluxo de cada pacote transmitido. E a grande diferença entre o AODV e os outros protocolos sob demanda é que ele utiliza o número de sequência (DetSeqNum) para determinar uma atualização da rota para o destino. O nó só vai atualizar a rota caso o DetSeqNum seja maior que o previamente armazenado. [7]

Sendo um protocolo sob demanda, o AODV envia pacotes *RouteRequest* para a rede, afim de descobrir a melhor rota. O *RouteRequest* contém o identificador de origem (SrcID), de destino (DestID), o número de sequência da fonte (SrcSeqNum), o DetSeqNum, identificador de transmissão (BcastID) e o TTL. O DestSeqNum indica o quão nova a rota deve ser. Esse parâmetro foi definido pela origem. Quando um nó intermediário receber o pacote de *RouteRequest*, ele pode encaminhar o pacote ou então responder com um *RouteReply* se ele tiver uma rota válida para o destino. Essa validade é determinada comparando o DetSeqNumber com o número de sequência desse nó intermediário. Caso seja recebido um *RouteRequest* várias vezes, o que pode ser detectado utilizando o BcastID e SrcID, essas cópias são descartadas. Todos os nós intermediários que possuírem uma rota válida para o destino ou o próprio nó de destino podem enviar um pacote de *RouteReply* para a origem. Sempre quando um nó encaminhar um *RouteRequest*, ele vai entrar no endereço do nó anterior e no seu BCastID. Um timer é utilizado para deletar um pacote de *RouteReply* se ele não for recebido antes do tempo expirar. Isso ajuda o armazenamento de rotas ativas nos nós intermediários, pois o AODV não emprega o roteamento de origem nos pacotes de dados. Quando um nó recebe um *RouteReplay*, as informações sobre os nós anteriores que estão armazenadas nos pacotes também são armazenadas, para que na próxima transmissão o pacote seja encaminhado corretamente. [7]

Na Figura 8 é mostrado um exemplo de como ocorre a seleção da rota. O nó 1 vai transmitir para o nó 15, ele inicia o processo inundando a rede com pacotes de *RouteRequest*, levando em consideração que esses pacotes contém o DetSeqNum como 3 e o SrcSeqNum como 1. Quando os nos 2,5 e 6 receberem o pacote, eles vão checar suas rotas para o destino. Caso essa rota não esteja disponível, eles vão encaminhar o pacote para seus vizinhos. Assumindo, para simplificar, que os nós 3 e 10 já possuem rotas para o nó de destino, que são respectivamente, 3-7-9-13-15 e 10-14-15. Se o DestSeqNumber no nó 10 é de 4 e 1 no nó 3, então apenas o nó 10 poderá transmitir uma resposta com a rota para a origem. Isso porque a rota do nó 3 será mais velha se comparada com a do nó 10. Se o *RouteRequest* chegar ao nó de destino, utilizando o caminho 4-12-15 ou qualquer caminho alternativo, ele também poderá mandar um *RouteReply* para o nó de origem. No caso em ser transmitidos múltiplos pacotes de *RouteReply*, todos os nós intermediários que receberem esse pacote vão atualizar sua tabela de roteamento com o DetSeqNum mais recente. E também vão ser atualizadas as informações sobre as rotas que utilizam o menor caminho entre a origem e destino. [7]

O ADOV não repara localmente caminhos quebrados. Quando ocorre uma falha no caminho, ela é determinada observado as transmissões periódicas, quando é determinada uma falha os nós finais são notificados. Assim quando o nó de origem é notificado desta falha, ele mudará a rota. Se uma falha for detectada em um nó intermediário, o nó irá informar os nós finais enviando um *RouteReply* com a contagem de saltos definido como infinito.

Na Figura 9 temos uma situação em que acontece uma falha entre o nó 5 e 4. Assim que isso ocorre, os dois nós começam a transmissão de um pacote *RouteError* informando os nós de destino e origem. Assim que eles recebem essa informação, é removido da tabela de roteamento esse caminho. O nó de origem irá recomeçar o processo de construção de rota, com um novo BCastID e o mesmo DetSeqNum. [7]

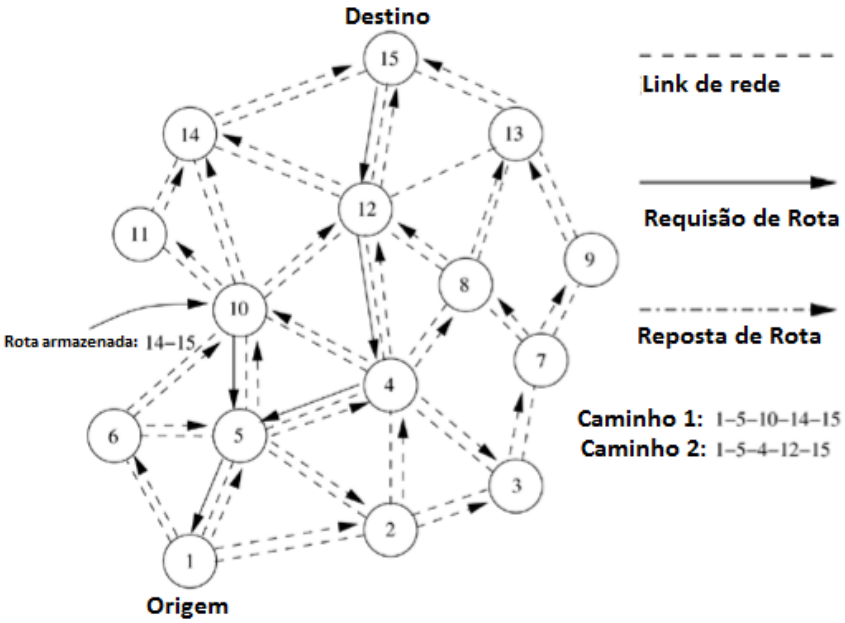

Figura 8 – Estabelecimento de rota AODV. [7]

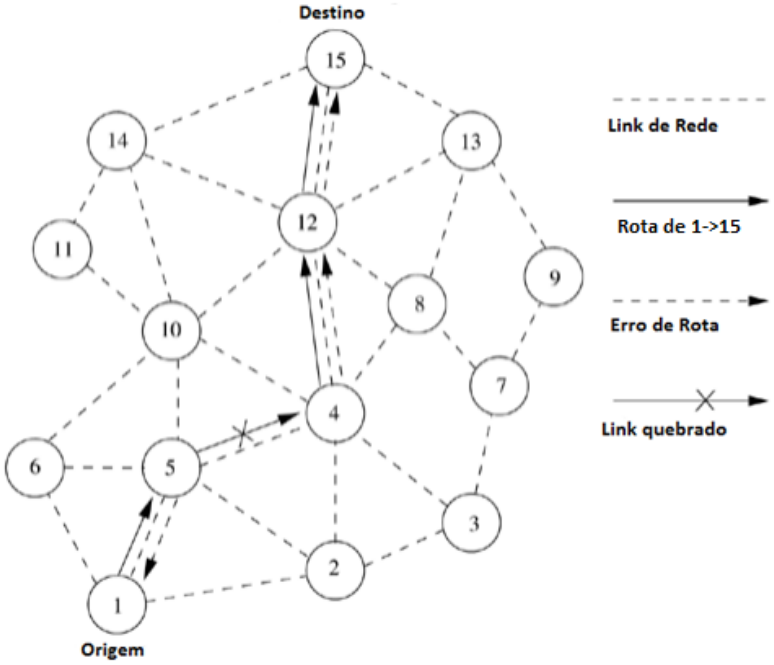

Figura 9 – Manutenção da rota com AODV. [7]

Uma desvantagem do protocolo AODV é que os nós intermediários podem levar a rotas inconsistentes. Isso ocorre se o SrcSeqNum for muito antigo e os nós intermediários possuírem

um DetSeqNum muito maior, porém, não mais recente. Outra desvantagem é que os múltiplos pacotes *RouteReply* que são utilizados para responder um único *RouteRequest* faz com que seja necessário um alto controle para não ter uma sobrecarga na rede. E por fim, o processo de transmissões periódicas para verificar os caminhos pode levar ao um consumo de banda desnecessário.

3. Simulador NS-2 e simulador de redes sem fio cognitivas.

3.1 Simulador NS-2.

O NS-2 é um projeto Open-Source e consiste em um simulador de eventos discretos direcionado à pesquisa de diferentes tipos de redes. Ele foi escolhido para este trabalho, por possuir a possibilidade de configuração de uma rede 802.11 comum e cognitiva, assim sendo possível a comparação entre as duas redes simuladas. A rede cognitiva será feita com o auxílio do patch *Cognitive Radio Cognitive Network Simulator (CRCN)* [3].

3.2 CRCN

O CRCN permite avaliar o desempenho para redes rádio cognitivas (Cognitive Radio (CR)), pois suporta algoritmos de controle de potências e protocolos alterados para funcionar de forma cognitiva, como por exemplo, o protocolo MAC. Ele simula um tráfego em topologias realistas e em cada nó é possível reconfigurar a camada física (PHY) para personalizar parâmetros de espectro (potência de transmissão, propagação, etc.). [3]

3.3 Camada MAC

O simulador adota duas técnicas distintas para implementar a camada MAC de forma cognitiva: a primeira é baseada em contenção e o segundo é o sem colisões.

3.3.1 MAC baseado em contenção

Na Figura 10 é descrito o modelo da camada MAC modificada. No primeiro instante serão coletadas, pela camada física, as informações sobre o ambiente da rede, como SNR, atraso e etc. Assim que a camada CR MAC tiver estas informações será tomada a decisão sobre qual canal

será utilizado e com qual potência o sinal vai ser transmitido. Assim, partindo da camada física (CR PHY), teremos os seguintes procedimentos:

Informação de tráfego, interferência sobre um canal especifico, comunicação informativa sobre o canal de controle (TICC): A camada física irá fornecer as informações de tráfego, interferência e comunicação para a camada MAC, assim estás informações contidas no "Bloco de Informação" vão ser fornecida para o CRCN. [3]

Potência de transmissão e seleção de canal (TC): Depois de o CR MAC receber o TICC ela fará a decisão sobre a potência de transmissão e vai selecionar o canal. O simulador vai mandar essas informações para o bloco "Suporte para multi-canal". Antes de passar por este bloco a camada MAC irá perceber apenas um único canal. [3]

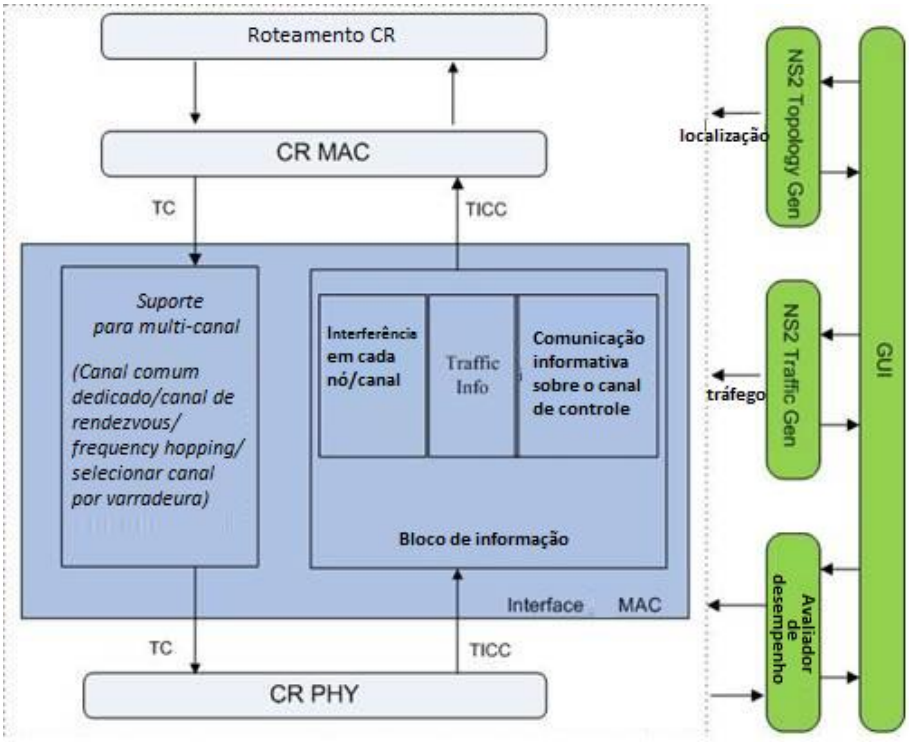

Figura 10 – Modificação camada MAC baseado em contenção. [3]

3.3.2 MAC sem colisões

O modelo sem colisões implementa o algoritmo DSA (*Dynamic spectrum acess*), que consiste em uma técnica para aumentar a eficiência na utilização do espectro, permitindo que um pedaço do espectro possa ser alocado para um ou mais usuários, por exemplo, é alocado para os usuários primários uma parte do espectro, porém, enquanto eles não utilizarem essa parte, ela poderá ser utilizada pelos usuários secundários. O DSA foi colocado entre a camada MAC e o roteamento

CR, isto faz com que a decisão seja tomada com base em condições pré-determinadas pelo usuário. [3] Na Figura 11, partindo da camada física, as informações sobre o ambiente da rede serão coletadas, enviadas para o bloco de "*Data MAC"* que possuem os dados das condições pré-determinada pelo usuário e então encaminhado para o bloco do DSA. O processo ocorre da seguinte forma:

Informação de tráfego, interferência sobre um canal de dados (TIC): A camada física vai prover as informações sobre o tráfego e interferência para o "bloco de informação" que irá auxiliar o CRCN a tomar as decisões. [3]

Potência de transmissão e seleção de canal (TC): Neste caso após o DSA decidir a potência de transmissão e o canal, o simulador irá mandar as informações para o bloco "Suporte para Multi-canal". As métricas escolhidas pelo DSA vão ser comparadas com as escolhidas pelo usuário através da interface (GUI), caso elas não satisfaçam estas condições elas serão devolvidas ao DSA informando que é necessária uma nova escolha. [3]

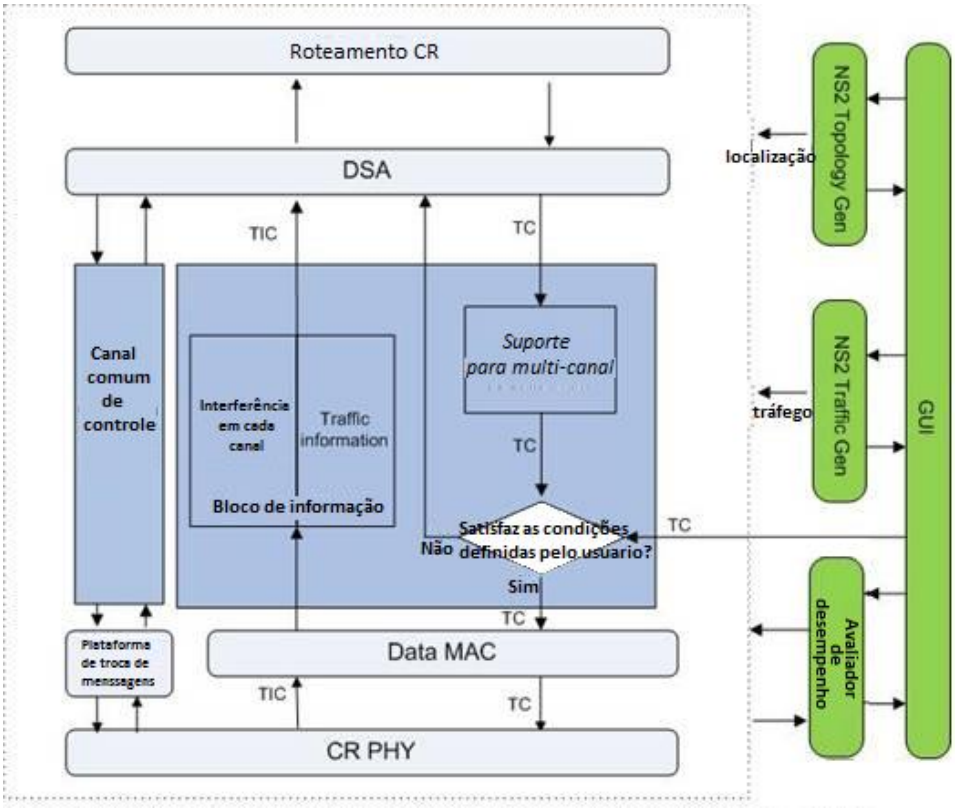

Figura 11 – Modificação na camada MAC sem colisões. [3]

3.4 Camada de Roteamento

A camada de roteamento irá receber informações da camada MAC e vai utiliza-las para selecionar o canal, rota e rádio. [3]. Na Figura 12 está o detalhamento deste processo, que vai seguir os seguintes procedimentos:

Informação sobre utilização de espectro, tráfego e interferência do canal e rádio selecionado (STICR): A camada inferior coleta as informações necessárias para o roteamento CR e manda estas informações para as camadas superiores. Esta informação é processada nos blocos de Estimativas (tráfego, utilização de espectro e interferência) e passada para a camada de roteamento CR. [3]

Decisão de Canal/ Seleção do Rádio/ Seleção da rota (CRR): A decisão do canal irá definir o número do canal e a potência de transmissão no canal selecionado. Na decisão da rota será definido qual caminho os pacotes irão seguir. [3]

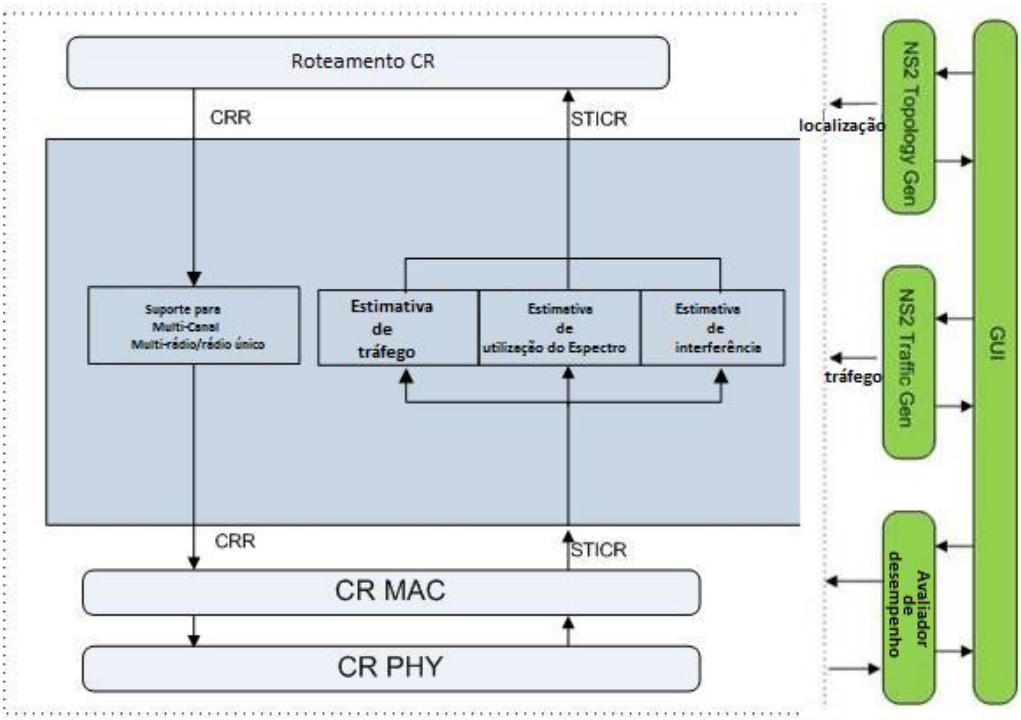

Figura 12 – Esquemático da camada de roteamento. [3]

3.5 Considerações gerais da modificação no NS-2.

A extensão CRCN modifica a biblioteca TCL do NS-2 que contém os algoritmos dos protocolos AODV e DSR. Assim, quando a fase de construção e manutenção de rota for finalizada, o nó de origem vai utilizar as informações coletadas além dos procedimentos descritos no capitulo 2

para decidir a melhor rota. E também para modificar essa rota caso haja necessidade, como por exemplo, falhas ou perda excessiva de pacotes. Ambos os protocolos funcionam de modo promíscuo nesta implementação.

A modificação feita na biblioteca TCL para o protocolo 802.11 traz a possibilidade de utilizar vários canais e os procedimentos descritos na Seção 3.3.

O simulador possui algumas limitações, como por exemplo, não foi implementada a política de nós primários e secundários, assim todos os nós da rede possuem a mesma prioridade. Assim, quando dois nós ocupam o mesmo canal, é tratado como colisão de pacotes e ambos desocupam o canal e esperam uma retransmissão.

No NS-2 sem modificações o nó receptor que decide qual canal utilizar e qual a potência de transmissão será usada [2]. No CRCN, essa seleção é feita pelo nó transmissor, que irá transmitir para toda a rede as informações sobre potência e canal.

4. Ambiente e ferramentas de simulação.

4.1 Configuração do ambiente de simulação.

O simulador foi instalado no sistema operacional Ubuntu 12.04 32 Bits, com 4 gigabytes de memória RAM e processador Phenom X4 3.2 GHz e fornece uma interface gráfica que pode auxiliar na construção de algumas topologias. Na seção 4.1.2 é explicado como está interface funciona.

4.1.2 Interface gráfica do simulador.

4.1.2.1 Aba “main”.

Como é possível observar na figura 13, na interface principal fornecida pelo simulador é possível definir os principais parâmetros da rede.

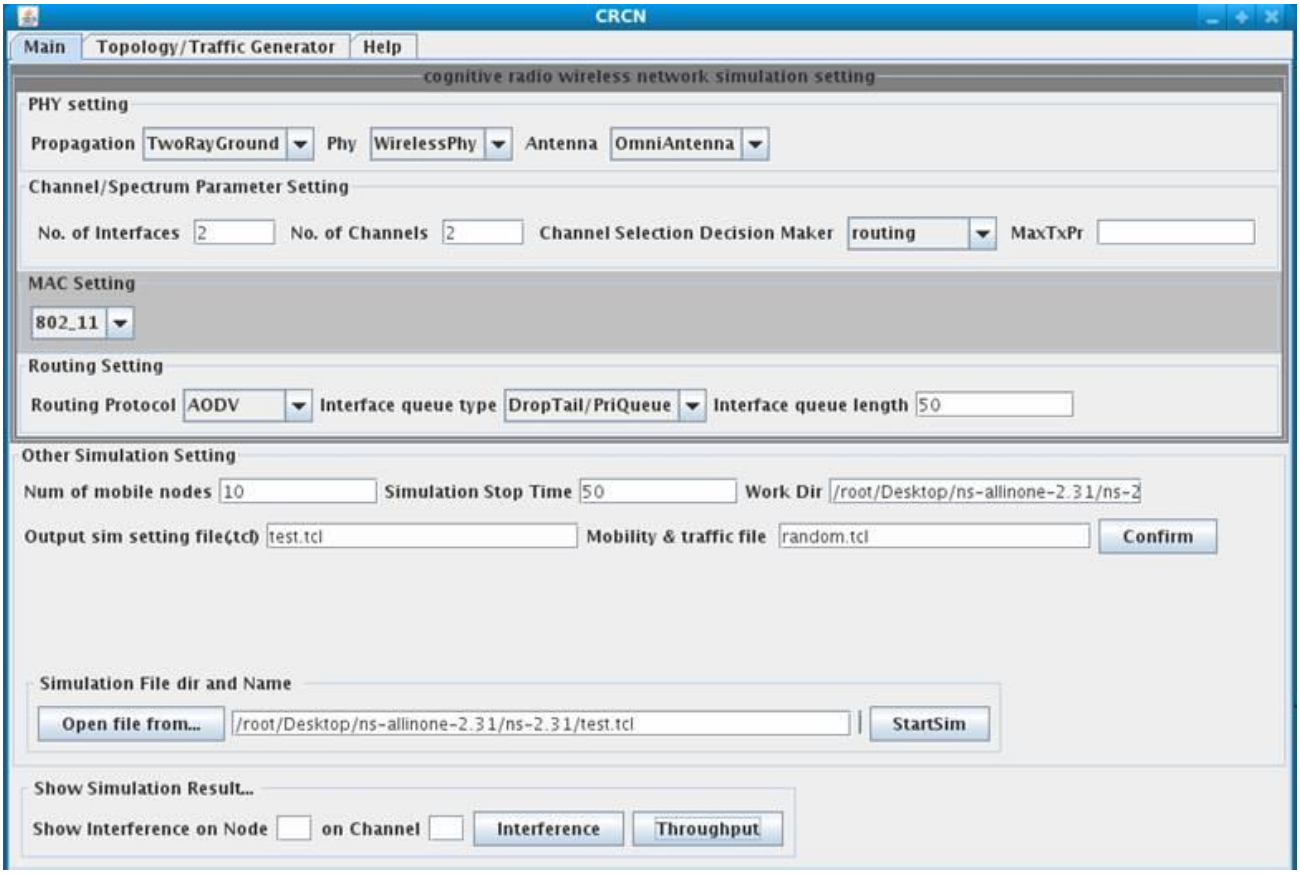

Figura 13 – Interface principal do Simulador

Na categoria "*PHY Settings*" é definido a camada física da rede. Os tipos de propagações suportadas são no espaço livre ou o modelo de dois raios. Nesta versão só é suportado antenas omnidirecionais.

Em "*Channel/Spectrum Parameter Setting*" pode ser definido o número de interfaces por nó e o número de canais da rede. Outro parâmetro importante é "Channel Selection Decision Maker", em que existem duas opções "routing" e "routing/DSA. A opção "routing" significa que os múltiplos canais serão visíveis apenas para a camada de roteamento. Já na opção "routing/DSA" os múltiplos canais serão visíveis para camada de roteamento e MAC.

É possível escolher o tipo de protocolo da camada MAC em "Mac Setting", sendo que o simulador fornece os protocolos 802_11, TDMA, Maccon e Macng.

Os protocolos de roteamento podem ser definidos em "Routing Setting", existindo três opções que são AODV, DSR, WCETT.

Na categoria "Other Simulation Setting" serão estabelecidos os números de nós moveis, o tempo de simulação, o diretório onde serão salvos os resultados, o nome do arquivo que será salvo com as configurações da simulação e por fim, o arquivo em que se encontra a topologia da rede.

O simulador fornece ainda uma ferramenta que pode gerar gráficos de interferência entre os nós e a vazão da rede.

4.1.2.2 Aba “Topology/Traffic Generator”.

Para gerar a topologia da rede e o tráfego é fornecida a ferramenta da Figura 14.

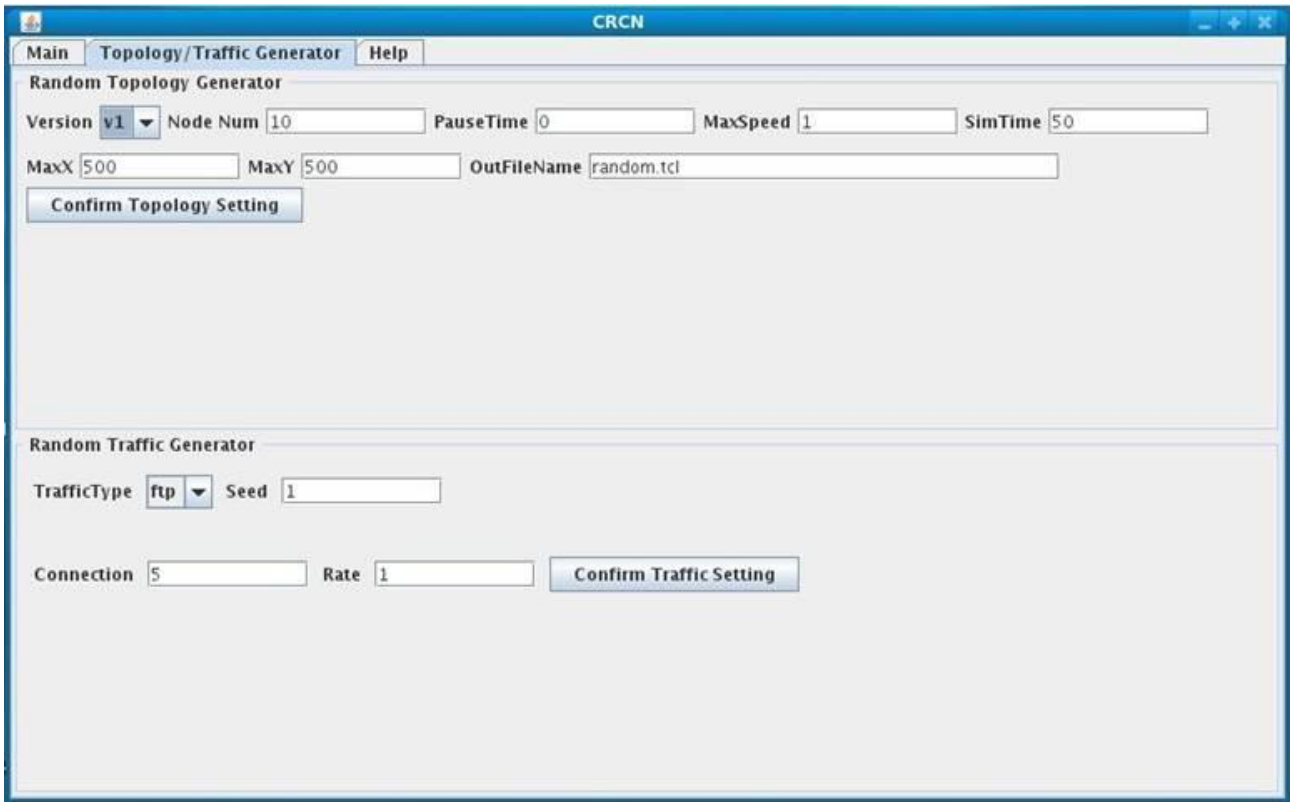

Figura 14 – Interface para gerar tráfego e topologia

Portanto, é definido o número de nós da rede em “Node num”, o tempo em que os nós ficam parados em “Pause Time”, a velocidade máxima com que cada nó se movimenta em “MaxSpeed”, o tempo de simulação em “SimTime”, o tamanho XY da grade em “MaxX” e “MaxY” e o nome do arquivo onde ficará armazenado esses dados “OutFileName”.

Na categoria “Random Traffic Generator”, pode-se escolher entre dois tipos de tráfego (TCP e CBR), a semente para a geração aleatória, o número máximo de conexões e a taxa de pacotes.

4.2 Ferramentas auxiliares.

4.2.1 Gerador de topologia e simulação.

Para um melhor controle da topologia da rede foi feito um programa em Python 3, que gera um script de simulação conforme os parâmetros definidos. O código pode ser visualizado no apêndice A. Assim que iniciado serão executados os seguintes passos:

- Solicitar para o usuário o nome do arquivo de simulação (Exemplo: Topologia1), que será salvo no formato “topologia1.tcl “(Formato utilizado pelo simulador NS-2 para realizar a simulação).
- Será solicitada a dimensão da grade da topologia e o tempo de simulação.
- Quantos nós estarão presentes nesta topologia.
- Se o usuário deseja que os nós sejam posicionados de forma aleatória. Caso seja selecionado de forma manual, o programa irá pedir para que seja definido as coordenadas x e y de cada nó da topologia.
- Será definido então pelo usuário quantos nós farão a transmissão, o tipo de tráfego (CBR ou TCP) e se ele será aleatório ou não.
- Caso o tráfego não seja aleatório, será solicitado o nó de origem e destino, o momento que a simulação vai começar e, para o caso do CBR, o intervalo de transmissão.
- Então o usuário terá que definir se haverá movimentação ou não na simulação. Se for uma topologia com movimentação, o usuário decide quantos nós irão se movimentar e a velocidade, para qual coordenada o nó vai e em qual instante. Porém, existe a opção de uma movimentação aleatória, que o usuário só precisa definir quantos nós irão se movimentar e com qual velocidade.

No apêndice B, temos um exemplo de saída do script .”tcl” para um cenário sem movimentação, utilizando o protocolo ad-hoc AODV e com tráfego UDP. E no apêndice C, um exemplo de uma topologia com movimentação, utilizando DSR e com tráfego TCP.

4.2.2 Códigos para obtenção de resultados.

Para se obter os resultados do arquivo de trace gerado pelo NS-2, foram utilizados scripts na linguagem de programação AWK [8] que foram retirados de [9] para serem adaptados. No anexo A existe o código utilizado para obter o atraso e taxa de entrega de pacotes (PDR), ele percorre o arquivo de saída da simulação, obtém o atraso de cada pacote e faz a média total do atraso da rede. Para calcular a PDR é percorrido o arquivo e verificado quantos pacotes foram recebidos e quantos foram enviados, assim obtendo a porcentagem de pacotes entregues.

No anexo B é calculada a vazão média do sistema, em que é percorrido o arquivo, armazenado o tempo total de simulação, o momento que a simulação acaba e a quantidade de dados recebidos na origem. Assim calculando a vazão do sistema em quilo bit por segundo.

Para automatizar a obtenção de resultados foi feito um código em shell que percorre os diretórios que foram salvos a simulação e juntam essas informações em arquivos do tipo texto. Esse código está localizado no apêndice D.

5. Metodologia e métricas.

Para fazer a análise das redes modeladas neste trabalho, será mensurado a vazão da rede e o atraso dos pacotes. Estes dois conceitos são de fundamental importância para mensurar o desempenho de uma rede.

5.1 Atraso de pacotes.

O atraso consiste no tempo que um pacote demora para atravessar a rede da origem até seu destino, passando pelos nós intermediários. Existem quatro principais tipos de atraso: atraso de processamento, fila, transmissão e propagação. [7]

Atraso de processamento: É o tempo que os nós demoram parar verificar o cabeçalho do pacote, sua integridade e definir o destino para o qual ele deve ser encaminhado. Usualmente esse atraso é da ordem de microssegundos. [7]

Atraso de Fila: Consiste no tempo que cada pacote tem que esperar nas filas dos nós intermediários até serem encaminhados para o próximo nó. O tempo deste atraso é variável pois depende do tráfego da rede, podendo ser da ordem de milissegundos. [7]

Atraso de transmissão: É o tempo necessário para que cada bit dos pacotes seja transmitido dentro da rede. [7]

Atraso de propagação: É o tempo de viagem até o próximo nó, sendo que ele vai depende da distância entre os nós e da velocidade da propagação no meio físico. [7]

Assim, o atraso total será a soma de todos atrasos em cada um dos nós no caminho. [7]

5.2 Vazão da rede.

A vazão é o número de bits transmitido na rede em um determinado espaço de tempo, ou seja, é medida em bits/segundo.

5.3 Taxa de entrega de pacotes(PDR).

PDR é a relação de pacotes que são entregues com sucesso para o destino com o número de pacotes que foram enviados.

5.3 Parâmetros da simulação.

Para realizar as simulações foi escolhido utilizar o padrão 802.11, fixado então um tamanho de grade de 500 metros por 500 metros variando o número de nós de 9 até 30, com uma topologia fixa. E foi criado outros cenários com movimentações destes nós.

O tráfego da rede foi simulado utilizando TCP e UDP, com pacotes de 512 quilo bytes e 1 nó transmitindo. O tipo de propagação foi definido um cenário com modelo de dois raios e a transmissão feita com uma antena omnidirecional. Por fim, definido 1 interface, utilizando os protocolos de roteamento AODV e DSR.

5.4 Topologias padrões.

Para as simulações foi criada uma topologia em que os nós foram posicionados na grade de 500 por 500, mantendo sempre a proporcionalidade de distância entre eles. Na Figura 15 é mostrado o arranjo inicial, que foi aumentado gradualmente em número de nós até a grade ficar completa, como na Figura 16.

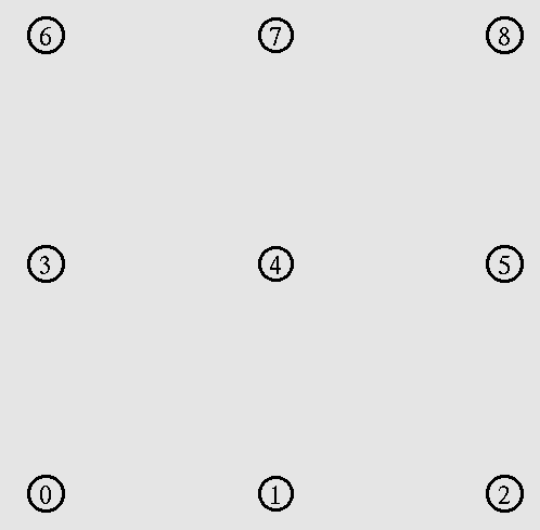

Figura 15 – Topologia básica com 9 nós

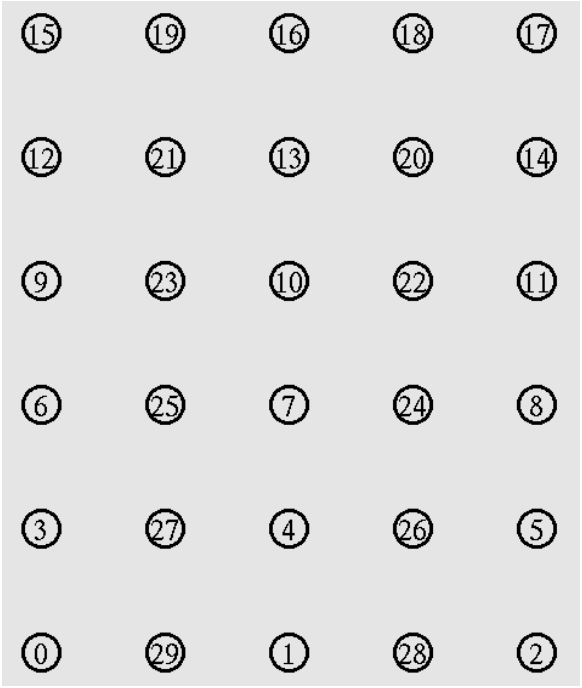

Figura 16 – Topologia básica com 30 nós

6. Resultados.

6.1 Resultados e análise para TCP.

Neste cenário o protocolo DSR teve uma taxa de vazão 61% superior como pode ser visto na figura 17 e um atraso médio 55,33% menor como pode-se observar na figura 18, porém, temos uma PDR praticamente igual como indicado na tabela 2. Assim, na situação de nós sem mobilidade e com uma transmissão TCP que prove garantia de entrega, o DSR tem um desempenho melhor, sendo que isso pode ser atribuído ao fato de a construção e manutenção de rotas se darem de maneira mais eficiente. Isso porque os pacotes no DSR carregam todo caminho percorrido e guardam em cache esse caminho.

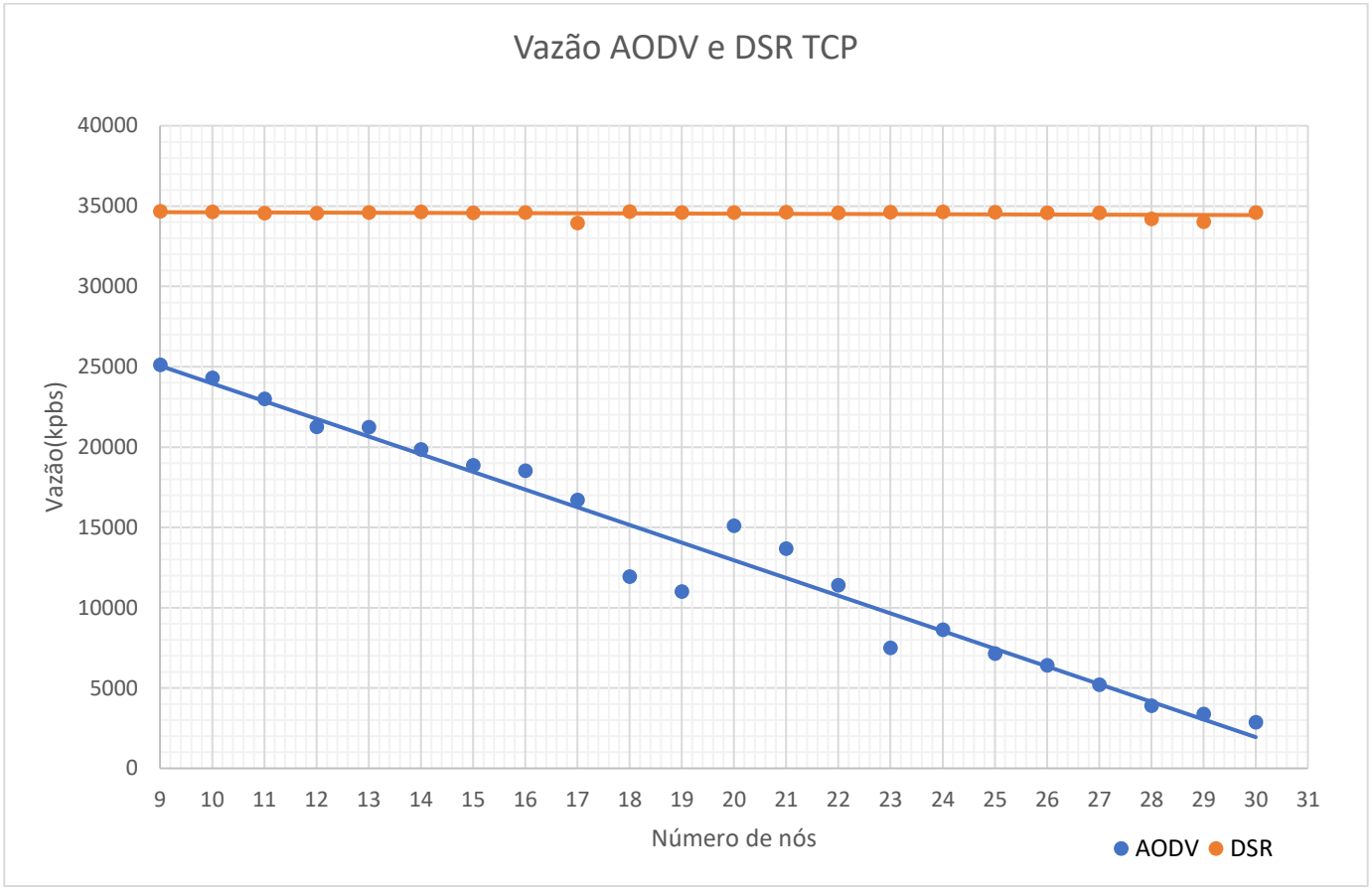

Figura 17 – Vazão TCP.

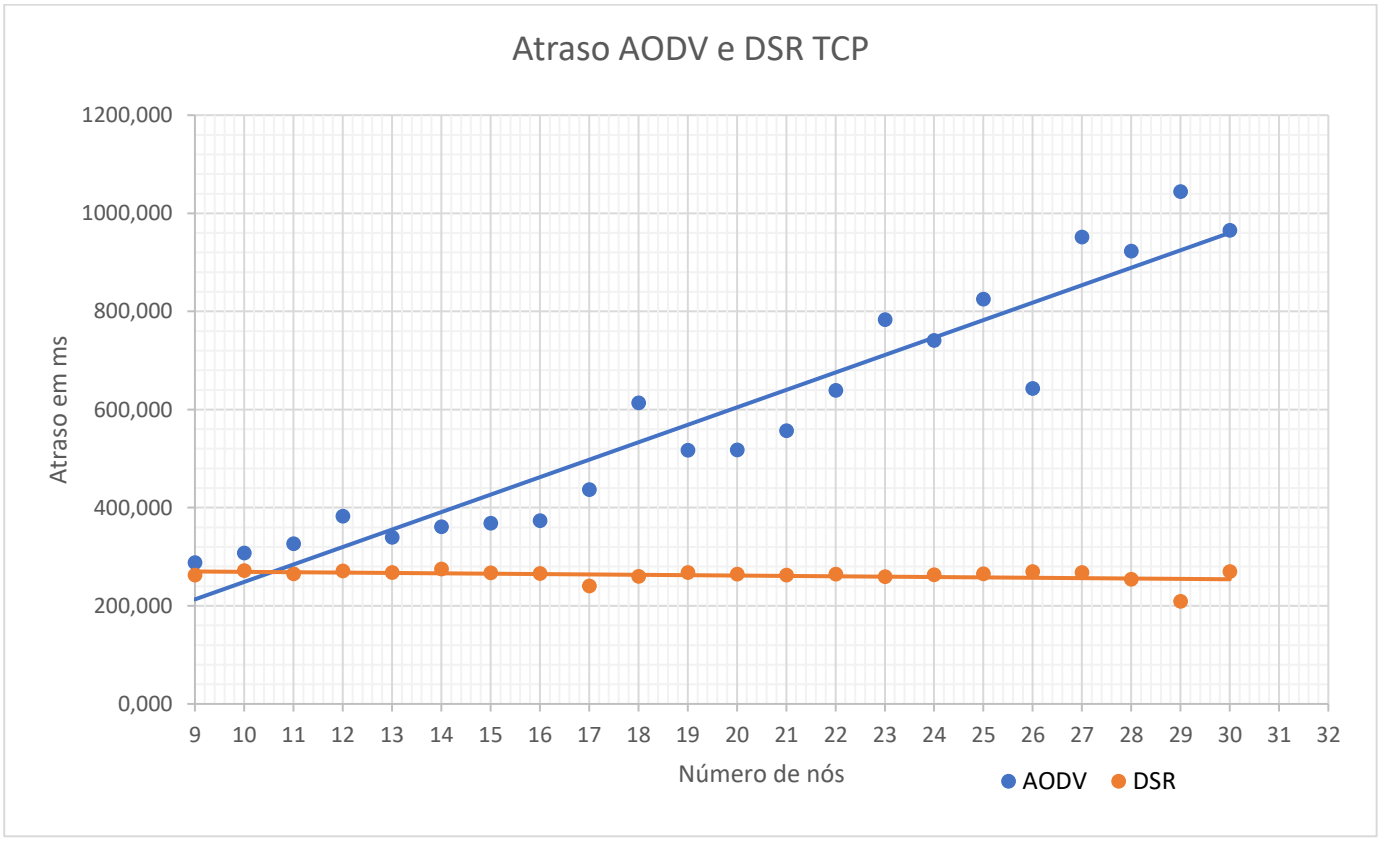

Figura 18 - Atraso TCP.

Tabela 2 – Resultados dos protocolos AODV e DSR utilizando protocolo TCP

	AODV	**DSR**
Vazão média(kpbs)	13500,983	34530,208
Atraso médio(ms)	586,638	262,023
PDR(%)	98,913	99,723

6.2 Resultados e análise para UDP.

No cenário UDP o DSR teve uma taxa de vazão média aproximadamente 20,44% maior como está demonstrado na Figura 19 e um atraso médio 73,6% menor que pode ser observado na Figura 20. Se comparado ao cenário com o TCP apesar de continuar um desempenho inferior o protocolo AODV se aproximou do desempenho do DSR, como pode ser observado se comparada as Tabela 2 e 3.

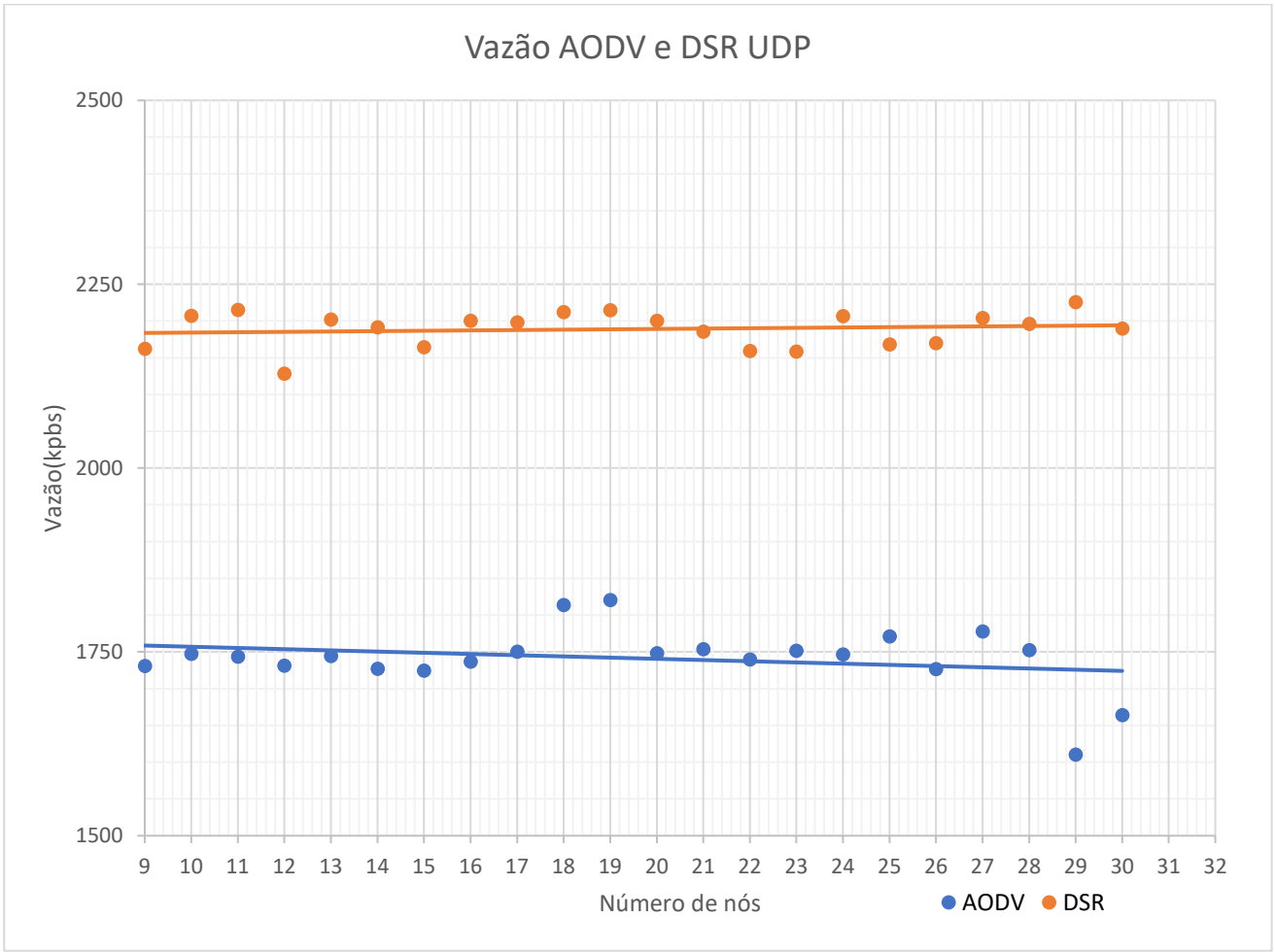

Figura 19 –Vazão UDP.

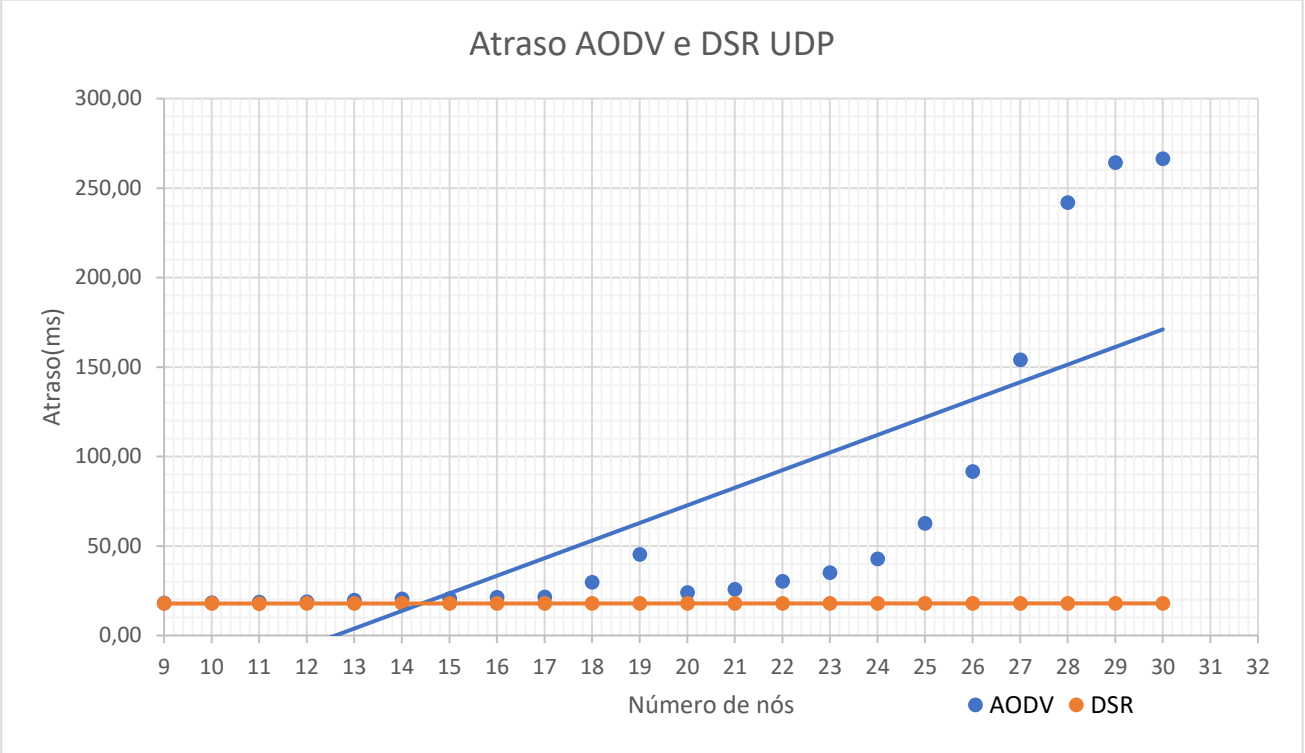

Figura 20 - Atraso UDP.

Tabela 3 – Resultados dos protocolos AODV e DSR utilizando protocolo UDP.

	AODV	**DSR**
Vazão média(kpbs)	1741,382	2188,915
Atraso médio(ms)	67,791	17,902
PDR(%)	98,766	99,564

6.3 Resultados cenários diversos.

6.3.1 Cenário com movimentação aleatória e 1 nó transmitindo.

Nesta simulação foi utilizado 30 nós conforme a Figura 16**,** sendo que o nó 0 transmitindo para o nó 16 utilizando o protocolo TCP. Então foi determinado que todos os eles se movimentassem de maneira aleatório com velocidade de 10 m/s e durante 150 segundos. Desta maneira, foram obtidos os seguintes resultados:

Tabela 4 – Resultados dos protocolos AODV e DSR com movimentação.

	AODV	**DSR**
Vazão (kpbs)	1823,79	26207,94
Atraso (ms)	528,087	373,605
PDR (%)	96,087	98,8438

Com uma movimentação aleatória e constante, temos uma queda de vazão de aproximadamente 24,25% para o protocolo DSR e 36,22% para o AODV se compararmos as Tabelas 4 e 2. Assim temos que a rede sofre degradação conforme se aumenta a mobilidade dos nós, conforme previsto na literatura [7].

6.3.2 Cenário com vários nós transmitindo.

Foi utilizado uma topologia conforme a figura 16, com todos os nós transmitindo durante 150 segundos utilizando o protocolo TCP. Deste modo, os resultados são:

Tabela 5 – Resultados dos protocolos AODV e DSR com muitas transmissões

	AODV	**DSR**
Vazão (kpbs)	6324,330	96107.66
Atraso (ms)	1501,200	2315.26
PDR (%)	92,941	45,948

Comparando está simulação com a 6.1, é possível observar que o atraso aumentou consideravelmente. Isto deve-se ao fato de que, com várias transmissões, ocorrer uma maior ocupação dos canais disponíveis para transmissão na rede. No caso do protocolo DSR ele utiliza um cache do caminho da fonte para o destino, até que a rota seja atualizada pacotes poderão ser perdidos e isto pode ser observado neste caso que o PDR ficou em aproximadamente em 45%.

6.3.3 Cenário com movimentação e várias transmissões.

Para este cenário foi utilizado a topologia base idêntica a figura 16, então os nós movimentaram durante 150 segundos, com uma velocidade de 10 m/s e todos transmitiram utilizando o protocolo TCP. Assim, foram obtidos os seguintes resultados:

Tabela 6 – Resultados dos protocolos AODV e DSR com muitas transmissões e movimentação constante.

	AODV	**DSR**
Vazão (kpbs)	3819,74	67284,70
Atraso (ms)	824,181	1969,32
PDR (%)	91,4388	79,8476

Com a junção de movimentação e transmissões, temos uma queda na vazão da rede em ambos os protocolos. Isso é devido ao aumento do atraso e diminuição da PDR. O atraso também teve um aumento grande comparado ao cenário com apenas movimentação, sendo que para o DSR esse atraso pode ter corrido devido as limitações do protocolo em adaptar rapidamente a rota. A taxa de entrega de pacotes do protocolo AODV foi aproximadamente 11,6% melhor. Isso mostra que em situações com muita mobilidade e transmissões o protocolo DSR vai inundar a rede com muito mais pacotes de controle para que seja possível manter a melhor rota gerando assim sobrecarga. Portanto, é possível perceber que a mobilidade aliada ao alto tráfego de pacotes impõe uma degradação severa do desempenho da rede.

6.3.4 Cenário com falhas.

Este cenário foi idealizado para testar o desempenho da rede em ambientes com falhas severas e observar como os protocolos se comportam e adaptam conforme ocorrem falhas na topologia. Foi utilizado a topologia da Figura 16, definido uma transmissão utilizando o protocolo TCP entre o nó 0 e 16. Inicialmente a topologia começou com os 30 nós, e então foi retirado gradualmente(falhas) os nós intermediários durante a simulação, por exemplo, aos 23 segundos da simulação o nó 6 sofre uma falha, depois aos 30 o nó 25 também falha, etc. Na Figura 21 é mostrado como a topologia ficou depois deste processo e na tabela a seguir os resultados obtidos.

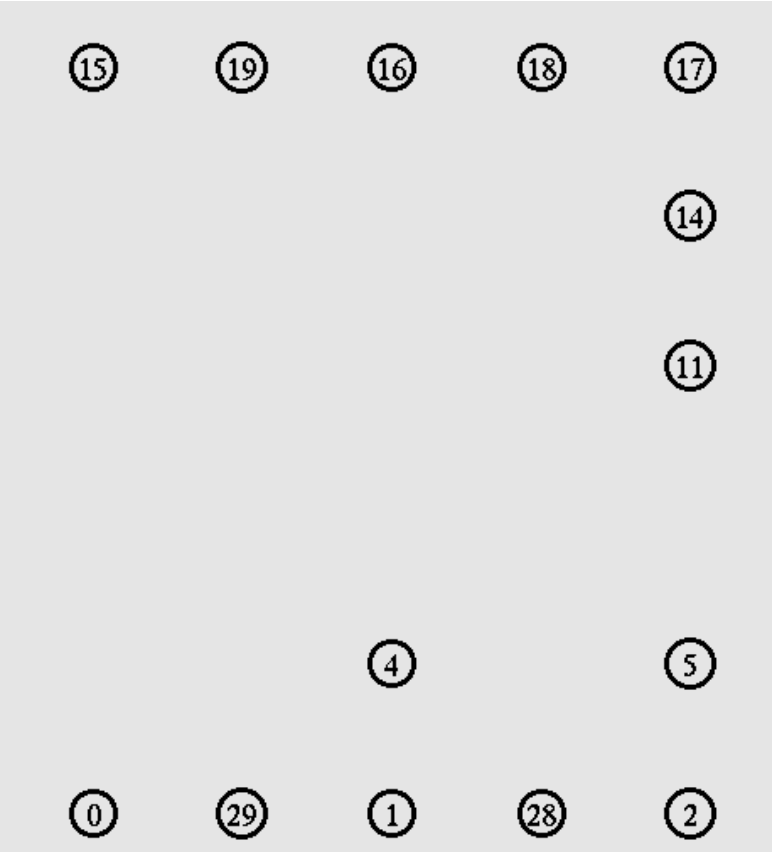

Figura 21 – Topologia com falhas.

Tabela 7 – Resultados dos protocolos AODV e DSR com falhas.

	AODV	**DSR**
Vazão (kpbs)	806,72	25469,19
Atraso (ms)	643,199	375,979
PDR (%)	93,307	70,576

Comparando esses resultados obtidos listados na Tabela 7 com os da Seção 6.1, temos uma queda de vazão de 71,78% utilizando o protocolo AODV e 26,37% no caso do DSR. Também existe um aumento no atraso sendo de 8,79% e 30,30% respectivamente. Assim, pode-se concluir que a rede mantém seu funcionamento mesmo em cenários extremos. Porém, principalmente para o protocolo AODV a vazão da mesma será seriamente comprometida, pois, é necessário que a rota para se chegar no nó de destino mude constantemente. Apesar de a vazão do DSR não ter tido uma queda muito alta, temos que a perda de pacote será maior e isso pode afetar a confiabilidade da rede. Um dos motivos que podem ter levado a essa perda é porque o protocolo DSR não corrige a rota localmente. Se houvesse mais falhas a rede não funcionaria, pois não haveria mais rotas disponíveis para se chegar nó de destino.

6.3.5 Simulação não cognitiva.

Neste cenário foi utilizado o simulador NS-2 sem a extensão CRCN. Como observado nas figuras 22 e 23 atuando de forma não cognitiva, o protocolo DSR se manteve superior ao AODV, porém, com um atraso de pacotes maior. Assim, é possível concluir que na implementação do cenário cognitivo esta relação não será modificada.

Comparando esse cenário não cognitivo com a Seção 6.1, temos que o desempenho da rede atuando de forma cognitiva é menor. Um dos possíveis motivos é que em um cenário onde não existe limitação da banda de espectro, movimentação e tem uma baixa densidade de nós os protocolos conseguem atuar em uma condição ideal, necessitando de menos recursos para funcionar. Assim, são necessários mais estudos para investigar essa relação e definir em quais cenários seria mais adequada a utilização de redes cognitivas.

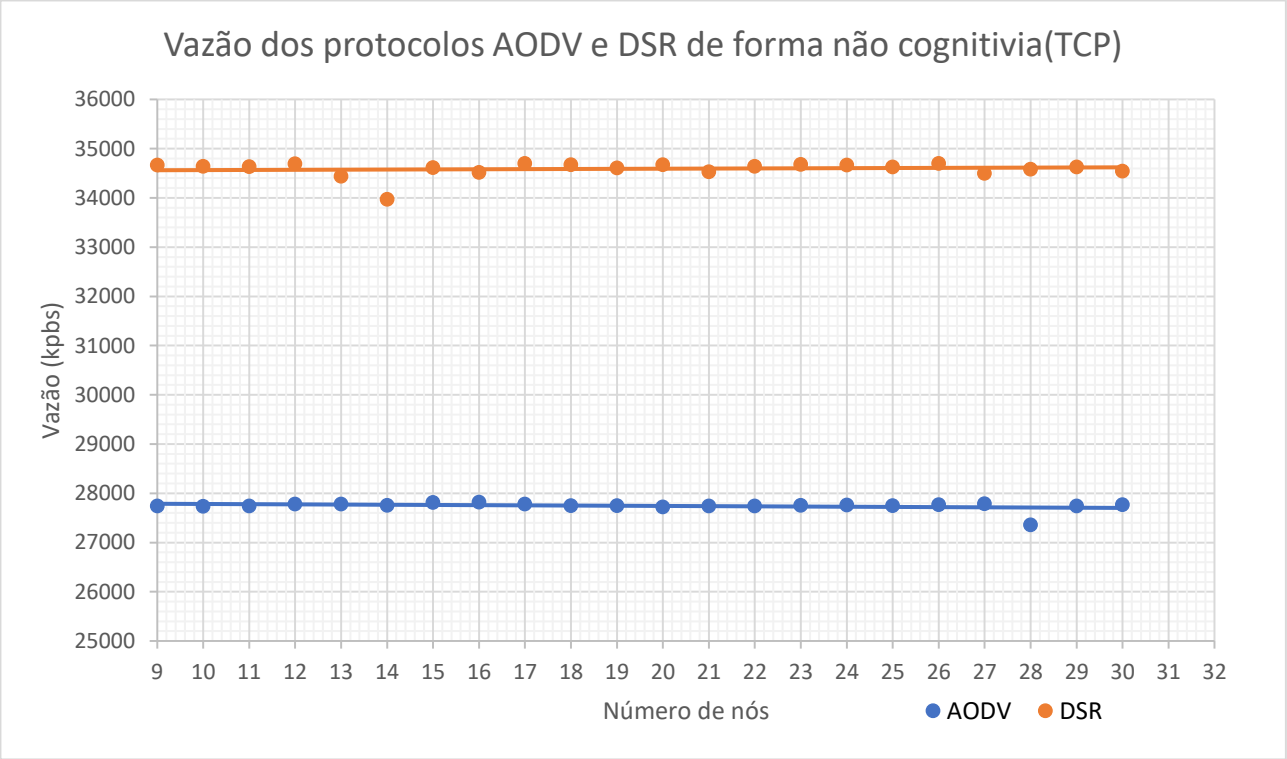

Figura 22 - Vazão dos protocolos agindo de forma não cognitiva.

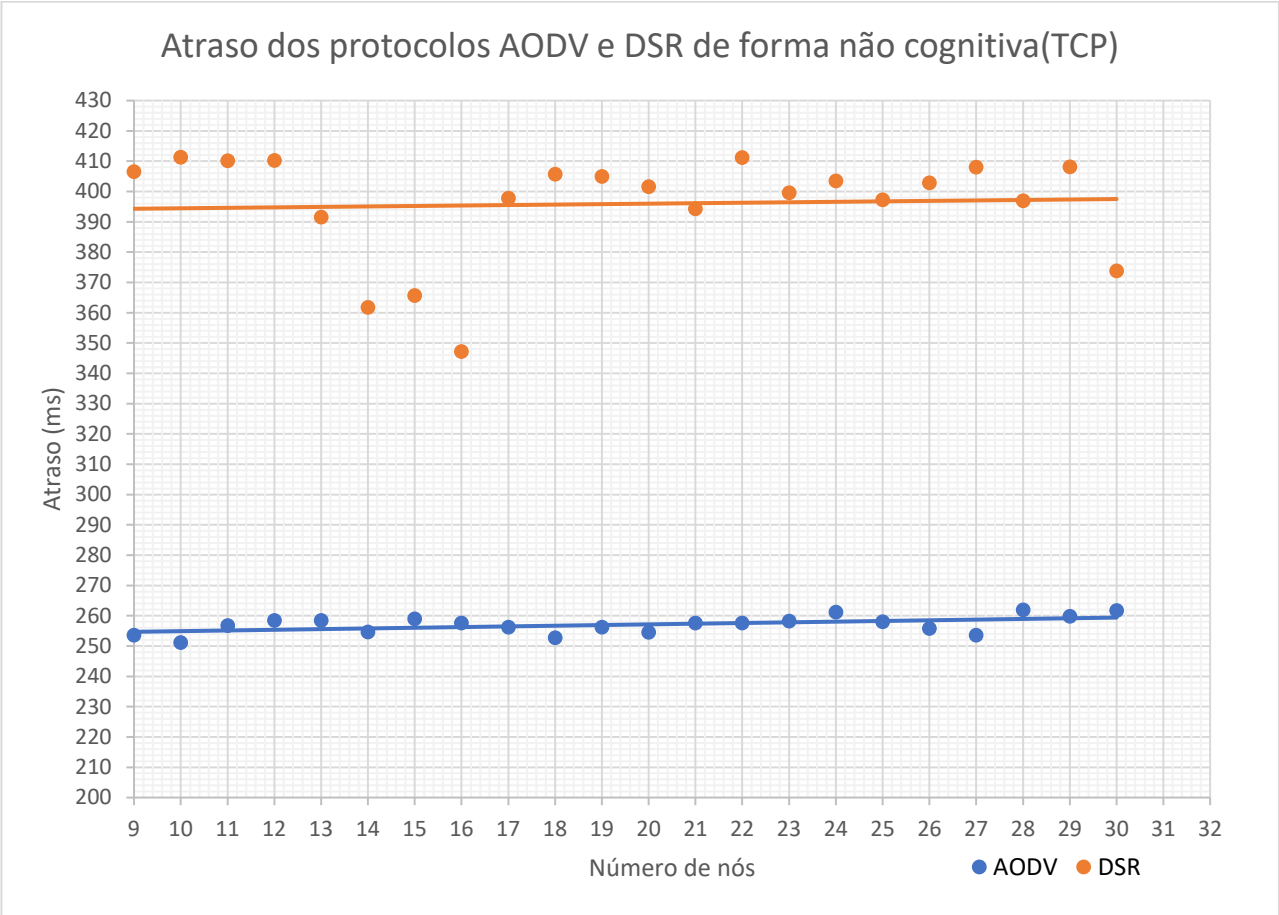

Figura 23 – Atraso dos protocolos agindo de forma não cognitiva.

Tabela 8 – Resultados dos protocolos AODV e DSR de forma não cognitiva.

	AODV	**DSR**
Vazão média(kpbs)	27744,840	34591,108
Atraso médio(ms)	257,043	395,911
PDR(%)	99,815	96,587

7. Conclusão

As redes cognitivas sugiram com a intenção de solucionar os problemas decorrido do crescimento de redes sem fio, que impactam o dia-a-dia dos usuários, como por exemplo, o aumento de transmissores em uma mesma frequência, o limite de canais disponíveis para transmissão, a mobilidade dos nós entre outros. Neste trabalho foi possível concluir que as redes cognitivas podem ser piores em alguns cenários, porém, certamente são uma excelente solução futura para uma utilização mais eficaz da alocação espectral. Antes de sua implantação, entretanto, é necessário enfrentar a resolução de alguns problemas, como desenvolvimento de hardware e software pelos fabricantes de equipamentos e preocupações sobre a segurança da transmissão da informação que são estudadas a fundo em outros trabalhos [10] e [11] .

No cenário simulado 6.1, em que os nós não possuem mobilidade e existe apenas um nó transmitindo utilizando protocolo TCP, foi possível concluir que o protocolo DSR, por ser elaborado para ser um protocolo que possui o foco de ter uma eficiência melhor na fase de construção da rota e manutenção, teve um desempenho superior ao protocolo AODV. Outro fator que pode influenciar diretamente o desempenho destes protocolos é a densidade de nós, que não foi explorada neste trabalho, sendo que isto é uma desvantagem do protocolo DSR, que pode acabar sobrecarregando a rede pelo excesso de *RouteRequest* enviados.

No Cenário 6.2 foi utilizado a mesma topologia do primeiro, modificando apenas a transmissão para UDP. Apesar do DSR possuir um desempenho também superior, esta diferença foi menor. Isso pode estar ligado ao fato de que a transmissão UDP gera um overhead menor na rede.

O Cenário 6.3.1 foi construído de forma que todos os nós se movessem aleatoriamente pela grade, com isso foi percebido a degradação da rede quando se tem muita movimentação. Isso vem do fato de que os protocolos, principalmente o DSR, não possuírem um mecanismo para correção de rota localmente, ou seja, a nova rota só será calculada após o nó de origem receber a informação que existe uma falha.

Com várias transmissões simultâneas pode levar a congestionamento em nós intermediários da rede, assim foi feita essa simulação no cenário 6.3.2. O atraso dos pacotes aumentou severamente, sendo que o aumento para o protocolo DSR foi de 89% se comparado ao Cenário 1 e a PDR foi diminuído em aproximadamente 54%. Assim, nesta situação pode-se concluir que o protocolo AODV é muito menos afetado por excesso de tráfego que o DSR. No Cenário 6.3.3 foi unido as condições dos dois cenários anteriores. Portanto, foi construída uma simulação com movimentação aleatória e várias transmissões. E o protocolo DSR continua a ser o mais afetado por este cenário.

No cenário 6.3.4 foi construído uma situação em que ocorrem diversas falhas na rede durante a transmissão, isto foi feito para avaliar o comportamento dos protocolos em situações que é necessário que a rota mude várias vezes. Assim, foi observado que o protocolo AODV possui uma baixa tolerância a falha, pois a vazão da rede caiu em 71,78%. Esse problema decorre da demora para atualização de rotas obsoletas nos nós intermediários.

Por fim, foram simulados ambos os protocolos de forma não cognitiva. Assim foi possível observar que a relação entre o desempenho do protocolo AODV e DSR foi mantida com a densidade de 30 nós em uma grade de 500 metros por 500 metros. Existem outros trabalhos que exploram melhor essa relação em cenários não cognitivos como em [12] e [13].

De um modo geral, temos que o protocolo DSR mesmo em cenários cognitivos, tem um desempenho superior ao AODV. Porém, os cenários em que existe muita mobilidade e transmissão o desempenho dele é degradado rapidamente.

As modificações cognitivas realizadas pelo simulador se mostraram úteis para reduzir a degradação do desempenho dos protocolos em alguns cenários, como por exemplo, no cenário com falhas, em que o DSR foi pouco afetado.

Em trabalhos futuros é possível modificar o simulador para que o protocolo 802.11 seja implementado com a possibilidade de se utilizar mais canais de transmissão e com a diferenciação entre nós primários e secundários. Outra sugestão é se focar na comparação entre os protocolos utilizados neste trabalho e o WCETT que é um protocolo desenvolvido para cenários cognitivos. Também é possível construir um cenário utilizando outros protocolos MAC (por exemplo: Maccon e Macng) que estão implementados neste simulador. Outro cenário interessante de estudo seria o aumento da densidade de nós, para avaliar o impacto nos dois protocolos, assim se aproximando mais de um cenário real, como por exemplo, o campus de uma faculdade

Referências Bibliográficas

1. LEIBNITZ, Kenji; WAKAMIYA, Naoki; MURATA, Masayuki. Biologically Inspired Networking. In: MAHMOUD, Qusay H. (Ed.). **COGNITIVE NETWORKS:** Towards Self-Aware Networks. Londres: Guelph: Wiley-interscience, 2007. p. 1-19.

2. INFORMATION SCIENCES INSTITUTE. University Of Southern California. **The Network Simulator - NS-2.** Los Angeles, 2012. Disponível em: <http://www.isi.edu/nsnam/ns/>. Acesso em: 27 out. 2015

3. CHIGAN, Chunxiao Tricia. **Cognitive Radio Cognitive Network Simulator (NS2 Based).** Lowell, 2013. Disponível em: <http://faculty.uml.edu/Tricia_Chigan/Research/CRCN_Simulator.htm>. Acesso em: 15 abril 2017.

4. STRASSNER, John. The Role of Autonomic Networking in Cognitive Networks. In: MAHMOUD, Qusay H. (Ed.). **COGNITIVE NETWORKS:** Towards Self-Aware Networks. Londres: Guelph: Wiley-interscience, 2007. p. 23-51.

5. ATHUR, Chetan N.; SUBBALAKSHMI, K. P.. Security Issues in Cognitive Radio Networks. In: MAHMOUD, Qusay H. (Ed.). **COGNITIVE NETWORKS:** Towards Self-Aware Networks. Londres: Guelph: Wiley-interscience, 2007. p. 271-275.

6. J. F. KUROSE, K. W. ROSS**, Redes de Computadores e a Internet,** Addison Wesley, 5a Ed., 2010.

7. MURTHY, C. Silva Ram; MANOJ, B. S.. **Ad Hoc Wireless Networks:** Architectures and Protocols. New Jersey: Prentice Hall Professional Technical Reference, 2004. 879 p.

8. Free Software Foundation, Inc. **, The GNU Awk User's Guide**. Disponível em : <https://www.gnu.org/software/gawk/manual/gawk.html>. Acesso em: 03 nov. 2017.

9. TAHILIANI, Mohit P.. **Few more AWK Scripts for NS2. 2010**. Disponível em: <http://mohittahiliani.blogspot.dk/2010/02/few-more-awk-scripts-for-ns2.htm>. Acesso em: 03 nov. 2017

10. J Li, Z Feng, Z Feng et al., "A survey of security issues in cognitive radio networks. [J]", China Communications, vol. 12, no. 3, pp. 132-149, 2015.

11. REDDY, Yenumula B.. Security Issues and Threats in Cognitive Radio Networks. In: THE NINTH ADVANCED INTERNATIONAL CONFERENCE ON TELECOMMUNICATIONS, 9., 2013, Roma. Proceedings... . Roma: Logothetis, M.d., 2013. v. 1, p. 85 - 91. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.670.9778&rep=rep1&type=pdf >. Acesso em: 04 nov. 2017.

12. BanojKumar Panda, Bibhudatta Dash, Rupanita Das, Ajit Sarangi, **Mobility and its impact on Performance of AODV and DSR in Mobile Ad hoc Network**, IEEE, 2012.

13. S. Mohapatra, P. Kanungo, **Performance analysis of AODV, DSR, OLSR and DSDV Routing Protocols using NS2 Simulator**, In Procedia Engineering, Volume 30, 2012, Pages 69-76, ISSN 1877-7058, https://doi.org/10.1016/j.proeng.2012.01.835. (http://www.sciencedirect.com/science/article/pii/S1877705812008454) Keywords: Mobile Adhoc Network; Routing protocols; NS2 (Simulator); Throughput; Delay; Packet Delivery Ratio; Control Overhead

APÊNDICE A – CÓDIGO FONTE PARA GERAR *SCRIPT* DE SIMULAÇÃO

```
import random

nomesim = input("Defina um nome para simulacao \n")
protocolo = input("Qual o protocolo? (AODV,DSR,WCETT)\n")
tx = int(input("Qual dimensao da topografia em x?\n"))
ty = int(input("Qual dimensao da topografia em y?\n"))
i = int(input("Quantos nos?\n"))
temposimu = int(input("Qual tempo da simualacao?\n"))
nnos = i
randnos = input("Deseja que os nos sejam aleatorios? 1 Para sim e 0 para nao\n")
movimenta = int(input("Deseja que os nos se movimentem? 1 Para e 0 Para nao \n"))
ntrans = int(input("Quantos nos irao transmitir?\n"))
tipotraf = int(input("Qual tipo de trafego? 1 para CBR ou 2 para TCP\n"))
randtraf = int(input("Deseja que o trafego seja aleatorio? 1 para sim ou 0 para nao\n"))
traffic = " "
tcpWindows = "\n\n #Exibe o tamanho da janela\n"+"proc plotWindow {tcpSource file}
{\n"+"global ns \n"+"set time 0.01 \n"+"set now [$ns now] \n"+"set cwnd [$tcpSource set cwnd_]
\n" +'puts $file "$now $cwnd"\n'+'$ns at [expr $now+$time] "plotWindow $tcpSource
$file" }\n'+'$ns at 10.1 "plotWindow $tcp $windowVsTime2"\n'
if tipotraf == 2:
        while ntrans > 0:
                if(randtraf == 0):
                        norigem = int(input("Qual no de origem?\n"))
                        nodestino = int(input("Qual no de destino?\n"))
                        strans = float(input("Quando a transmissao ira comecar?\n"))
                        traffic = traffic + "\n#Define a conexao TCP entre "+str(norigem)+" e
"+str(nodestino)+" Comecando em "+str(strans)+"\nset tcp [new Agent/TCP/Newreno]\n"+"$tcp set
class_ 2 \n"+"set sink [new Agent/TCPSink] \n"+"$ns attach-agent $node_("+str(norigem)+")
$tcp\n" + "$ns attach-agent $node_("+str(nodestino)+") $sink\n"+"$ns connect $tcp $sink\n"+"set
ftp [new Application/FTP]\n"+"$ftp attach-agent $tcp\n"+"$ns at " +str(strans)+ ' "$ftp start"'
                        ntrans -=1
                else:
                        norigem = int(random.randrange(0,nnos-1,1))
```

```
                    nodestino = int(random.randrange(0,nnos-1,1))
                    while(norigem == nodestino):
                        nodestino = int(random.randrange(0,nnos-1,1))
                    strans = float(random.randrange(0,temposimu,1))
                    traffic = traffic + "\n#Define a conexao TCP entre "+str(norigem)+" e
"+str(nodestino)+" Comecando em "+str(strans)+"\nset tcp [new Agent/TCP/Newreno]\n"+"$tcp set
class_ 2 \n"+"set sink [new Agent/TCPSink] \n"+"$ns attach-agent $node_("+str(norigem)+")
$tcp\n" + "$ns attach-agent $node_("+str(nodestino)+") $sink\n"+"$ns connect $tcp $sink\n"+"set
ftp [new Application/FTP]\n"+"$ftp attach-agent $tcp\n"+"$ns at " +str(strans)+ ' "$ftp start"'+"\n"
                    ntrans -=1

elif tipotraf == 1:
    while ntrans > 0:
        if(randtraf==0):
            norigem = int(input("Qual no de origem?\n"))
            nodestino = int(input("Qual no de destino?\n"))
            strans = float(input("Quando a transmissao ira comecar?\n"))
            intervalo = float(input("Qual intervalo de transmissao?\n"))
            cbr = "cbr_("+ str(ntrans)+")"
            traffic = traffic + "\n #Define a conexao UDP entre "+str(norigem)+" e"+str(nodestino)+"
Comecando em "+str(strans)+"\nset udp_("+str(ntrans)+") [new Agent/UDP]\n"+"$ns attach-agent
$node_("+str(norigem)+") $udp_("+str(ntrans)+")\n"+"set null_("+str(ntrans)+") [new
Agent/Null]\n"+"$ns attach-agent $node_("+str(nodestino)+") $null_("+str(ntrans)+")\n"+"set
"+cbr +" [new Application/Traffic/CBR]\n"+"$"+cbr+" set packetSize_ 512\n"+"$"+cbr+" set
interval_ "+str(intervalo)+"\n$"+cbr+" set random_ 1\n"+"$"+cbr+" set maxpkts_
10000\n"+"$"+cbr+" attach-agent $udp_("+str(ntrans)+")\n"+"$ns connect $udp_("+str(ntrans)+")
$null_("+str(ntrans)+")\n"+"$ns at "+str(strans)+' "$'+cbr+' start"'
            ntrans -=1

        else:
            norigem = int(random.randrange(0,nnos-1,1))
            nodestino = int(random.randrange(0,nnos-1,1))
            while(norigem == nodestino):
                nodestino = int(random.randrange(0,nnos-1,1))
            strans = float(random.randrange(0,temposimu,1))
```

```
        intervalo = 0.05
        cbr = "cbr_("+ str(ntrans)+")"
        traffic = traffic + "\n #Define a conexao UDP entre "+str(norigem)+" e"+str(nodestino)+"
Comecando em "+str(strans)+"\nset udp_("+str(ntrans)+") [new Agent/UDP]\n"+"$ns attach-agent
$node_("+str(norigem)+") $udp_("+str(ntrans)+")\n"+"set null_("+str(ntrans)+") [new
Agent/Null]\n"+"$ns attach-agent $node_("+str(nodestino)+") $null_("+str(ntrans)+")\n"+"set
"+cbr +" [new Application/Traffic/CBR]\n"+"$"+cbr+" set packetSize_ 512\n"+"$"+cbr+" set
interval_ "+str(intervalo)+"\n$"+cbr+" set random_ 1\n"+"$"+cbr+" set maxpkts_
10000\n"+"$"+cbr+" attach-agent $udp_("+str(ntrans)+")\n"+"$ns connect $udp_("+str(ntrans)+")
$null_("+str(ntrans)+")\n"+"$ns at "+str(strans)+' "$'+cbr+' start"'+"\n"
        ntrans -=1
else:
    print("Opcao invalida")
    exit()

if tipotraf == 2:
    traffic = traffic + tcpWindows

posicaono = " "
while i > 0:
    if(int(randnos) != 0):
        posicaono = posicaono + "\n$node_(" + str(i-1) + ") set X_
"+str(random.randrange(1,tx,1))+".0\n"
        posicaono = posicaono + "$node_(" + str(i-1) + ") set Y_
"+str(random.randrange(1,ty,1))+".0\n"
        posicaono = posicaono + "$node_(" + str(i-1) + ") set Z_ 0.0\n"
        i -=1
    else:
        print("Entre a posicao do no:"+str(i)+"\n")
        posicaono = posicaono + "\n$node_(" + str(i-1) + ") set X_ "+input("Coordenada X\n")+".0\n"
        posicaono = posicaono + "$node_(" + str(i-1) + ") set Y_ "+input("Coordenada Y\n")+".0\n"
        posicaono = posicaono + "$node_(" + str(i-1) + ") set Z_ 0.0\n"
        i -=1
```

```
move = " "
if movimenta == 1:
    nmovs = int(input("Quanto nos irao se movimentar?\n"))
    velocidade = float(input("Qual velocidade dos nos?\n"))
    while nmovs > 0:
        nomv = int(input("Qual no vai se movimentar?\n"))
        v = velocidade
        t = float(random.randrange(0,temposimu,1))
        move = move + "\n$ns at "+str(t)+' "$node_('+str(nomv)+') setdest ' +
str(random.randrange(1,tx,1))+".0 " + str(random.randrange(1,ty,1)) + ".0 "+str(v)+""
        nmovs -=1

comentscript = "#Simulacaoo gerada pelo script em python feito por Vinicius Zanin\n" +
"#Simualcao com "+ str(nnos) + " nos utilizando protocolo " + protocolo + " com a grade tendo " +
str(tx) + " x " + str(ty) + "\n"
options = "#Define as opcoes\n"+"set val(chan)          Channel/WirelessChannel    ;# channel type
\n"+"set val(prop)          Propagation/TwoRayGround   ;# radio-propagation \n"+"set val(netif)
Phy/WirelessPhy          ;# network interface type \n"+"\n"+"set val(mac)
Mac/802_11               ;# MAC type\n"+"set val(ifq)          Queue/DropTail/PriQueue    ;#
interface queue typei\n"+"set val(ll)           LL                    ;# link layer type \n" +"set val(ant)
Antenna/OmniAntenna       ;# antenna model\n"+"set val(ifqlen)       50                    ;# max
packet in ifq \n"+"set val(nn)           "+str(nnos)+"                 ;# number of
mobilenodes\n"+"set val(rp)           "+protocolo+"                 ;# routing protocol \n"+"set val(x)
"+str(tx)+"               ;# X dimension of topography\n"+"set val(y)           "+str(ty)+"               ;#
Y dimension of topography\n"+"set val(stop)         "+str(temposimu)+"            ;# time of
simulation end\n"+"\n"+"set ns          [new Simulator]\n"+"set tracefd       [open "+ nomesim+".tr w]
\n"+"set windowVsTime2 [open win"+nomesim+".tr w] \n"+"set namtrace      [open
simwrls"+nomesim+".nam w]\n"+"\n$ns trace-all $tracefd\n"+"$ns namtrace-all-wireless
$namtrace $val(x) $val(y)\n"+"\n#define topografia \n"+"\nset topo       [new
Topography]\n"+"\n$topo load_flatgrid $val(x) $val(y) \n" + "\n create-god $val(nn)\n"+ "\n\n#Cria
os nn nos moveis e atrela eles ao canal\n"+"\n     $ns node-config -adhocRouting $val(rp)\\\n"+"
-llType $val(ll) \\\n"+"          -macType $val(mac) \\\n"+"          -ifqType $val(ifq) \\\n"+"
-ifqLen $val(ifqlen) \\\n"+"          -antType $val(ant) \\\n"+"          -propType $val(prop) \\\n"+"
-phyType $val(netif) \\\n"+"          -channelType $val(chan) \\\n"+"          -topoInstance $topo
\\\n"+"          -agentTrace ON \\\n"+"          -routerTrace ON \\\n"+"          -macTrace OFF
```

```
\\\n"+"            -movementTrace ON\n"+"\n for {set i 0} {$i < $val(nn)} { incr i } {\n"+"            set node_($i) [$ns node] \n"+"    }\n" + "\n\n#Posicicoes iniciais dos nos\n"+posicaono+"\n"+"\n"+traffic+"\n"+move+"\n"+"\n\n#Define os nos inicias no nam\n"+"for {set i 0} {$i < $val(nn)} { incr i } {\n"+"# 30 defines the node size for nam\n"+"$ns initial_node_pos $node_($i) 30\n"+"}\n"+"\n\n #Mostra para os nos quando a simualacao termina\n"+"for {set i 0} {$i < $val(nn) } { incr i } {\n"+'    $ns at $val(stop) "$node_($i) reset";\n'+"}\n"+"\n\n #termina nam e a simulacao\n"+'$ns at $val(stop) "$ns nam-end-wireless $val(stop)"\n'+'$ns at $val(stop) "stop"\n'+'$ns at '+str(temposimu)+'.01 "puts \\"end simulation\\" ; $ns halt"\n'+'proc stop {} {\n'+"    global ns tracefd namtrace\n"+"    $ns flush-trace\n"+"    close $tracefd\n"+"    close $namtrace\n"+"}\n"+"\n $ns run"

print(comentscript)
print(options)

with open(str(nomesim)+".tcl","w+") as f:
        f.write(comentscript+options)
```

APÊNDICE B – Script para cenário AODV

```
#Simulacaoo gerada pelo script em python feito por Vinicius Zanin
#Simualcao com 18 nos utilizando protocolo DSR com a grade tendo 500 x 500
#Define as opcoes
set val(chan)          Channel/WirelessChannel    ;# channel type
set val(prop)          Propagation/TwoRayGround   ;# radio-propagation
set val(netif)         Phy/WirelessPhy            ;# network interface type
set val(mac)           Mac/802_11                 ;# MAC type
set val(ifq)           Queue/DropTail/PriQueue    ;# interface queue typei
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)            30                         ;# number of mobilenodes
set val(rp)            AODV                       ;# routing protocol
set val(x)             500                ;# X dimension of topography
set val(y)             500                ;# Y dimension of topography
set val(stop)          150                ;# time of simulation end

set ns          [new Simulator]
set tracefd       [open 30UDP.tr w]
set windowVsTime2 [open win30UDP.tr w]
set namtrace      [open simwrls30UDP.nam w]

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

#define topografia

set topo       [new Topography]

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

 create-god $val(nn)

#Cria os nn nos moveis e atrela eles ao canal

    $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 < $val(nn)} { incr i } {
           set node_($i) [$ns node]
         }

#Posicicoes iniciais dos nos

$node_(29) set X_ 200.0
$node_(29) set Y_ 0.0
$node_(29) set Z_ 0.0

$node_(28) set X_ 400.0
$node_(28) set Y_ 0.0
$node_(28) set Z_ 0.0

$node_(27) set X_ 200.0
$node_(27) set Y_ 100.0
$node_(27) set Z_ 0.0

$node_(26) set X_ 400.0
$node_(26) set Y_ 100.0
$node_(26) set Z_ 0.0

$node_(25) set X_ 200.0
$node_(25) set Y_ 200.0
$node_(25) set Z_ 0.0

$node_(24) set X_ 400.0
$node_(24) set Y_ 200.0
$node_(24) set Z_ 0.0

$node_(23) set X_ 200.0
$node_(23) set Y_ 300.0
$node_(23) set Z_ 0.0

$node_(22) set X_ 400.0
$node_(22) set Y_ 300.0
$node_(22) set Z_ 0.0

$node_(21) set X_ 200.0
$node_(21) set Y_ 400.0
$node_(21) set Z_ 0.0

$node_(20) set X_ 400.0
$node_(20) set Y_ 400.0
$node_(20) set Z_ 0.0

$node_(19) set X_ 200.0
$node_(19) set Y_ 500.0
$node_(19) set Z_ 0.0
```

```
$node_(18) set X_ 400.0
$node_(18) set Y_ 500.0
$node_(18) set Z_ 0.0

$node_(17) set X_ 500.0
$node_(17) set Y_ 500.0
$node_(17) set Z_ 0.0

$node_(16) set X_ 300.0
$node_(16) set Y_ 500.0
$node_(16) set Z_ 0.0

$node_(15) set X_ 100.0
$node_(15) set Y_ 500.0
$node_(15) set Z_ 0.0

$node_(14) set X_ 500.0
$node_(14) set Y_ 400.0
$node_(14) set Z_ 0.0

$node_(13) set X_ 300.0
$node_(13) set Y_ 400.0
$node_(13) set Z_ 0.0

$node_(12) set X_ 100.0
$node_(12) set Y_ 400.0
$node_(12) set Z_ 0.0

$node_(11) set X_ 500.0
$node_(11) set Y_ 300.0
$node_(11) set Z_ 0.0

$node_(10) set X_ 300.0
$node_(10) set Y_ 300.0
$node_(10) set Z_ 0.0

$node_(9) set X_ 100.0
$node_(9) set Y_ 300.0
$node_(9) set Z_ 0.0

$node_(8) set X_ 500.0
$node_(8) set Y_ 200.0
$node_(8) set Z_ 0.0

$node_(7) set X_ 300.0
$node_(7) set Y_ 200.0
$node_(7) set Z_ 0.0

$node_(6) set X_ 100.0
$node_(6) set Y_ 200.0
```

```
$node_(6) set Z_ 0.0

$node_(5) set X_ 500.0
$node_(5) set Y_ 100.0
$node_(5) set Z_ 0.0

$node_(4) set X_ 300.0
$node_(4) set Y_ 100.0
$node_(4) set Z_ 0.0

$node_(3) set X_ 100.0
$node_(3) set Y_ 100.0
$node_(3) set Z_ 0.0

$node_(2) set X_ 500.0
$node_(2) set Y_ 0.0
$node_(2) set Z_ 0.0

$node_(1) set X_ 300.0
$node_(1) set Y_ 0.0
$node_(1) set Z_ 0.0

$node_(0) set X_ 100.0
$node_(0) set Y_ 0.0
$node_(0) set Z_ 0.0

 #Define a conexao UDP entre 0 e16 Comecando em 15.0
set udp_(1) [new Agent/UDP]
$ns attach-agent $node_(0) $udp_(1)
set null_(1) [new Agent/Null]
$ns attach-agent $node_(18) $null_(1)
set cbr_(1) [new Application/Traffic/CBR]
$cbr_(1) set packetSize_ 512
$cbr_(1) set interval_ 0.05
$cbr_(1) set random_ 1
$cbr_(1) set maxpkts_ 10000
$cbr_(1) attach-agent $udp_(1)
$ns connect $udp_(1) $null_(1)
$ns at 15.0 "$cbr_(1) start"

#Define os nos inicias no nam
for {set i 0} {$i < $val(nn)} { incr i } {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}

 #Mostra para os nos quando a simualacao termina
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "$node_($i) reset";
}

#termina nam e a simulacao
```

```
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 150.01 "puts \"end simulation\" ; $ns halt"
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
}

 $ns run
```

APÊNDICE C – CENÁRIO DSR COM MOVIMENTAÇÃO

```
#Simulacaoo gerada pelo script em python feito por Vinicius Zanin
#Simualcao com 18 nos utilizando protocolo DSR com a grade tendo 500 x 500
#Define as opcoes
set val(chan)          Channel/WirelessChannel    ;# channel type
set val(prop)          Propagation/TwoRayGround   ;# radio-propagation
set val(netif)         Phy/WirelessPhy            ;# network interface type

set val(mac)           Mac/802_11                 ;# MAC type
set val(ifq)           CMUPriQueue    ;# interface queue typei
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)            30                         ;# number of mobilenodes
set val(rp)            DSR                        ;# routing protocol
set val(x)             500                ;# X dimension of topography
set val(y)             500                ;# Y dimension of topography
set val(stop)          1000               ;# time of simulation end
set ns         [new Simulator]
set tracefd       [open 30TCP.tr w]
set windowVsTime2 [open win30TCP.tr w]
set namtrace      [open simwrls30TCP.nam w]
$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

#define topografia
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
 create-god $val(nn)

#Cria os nn nos moveis e atrela eles ao canal
    $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 < $val(nn)} { incr i } {
           set node_($i) [$ns node]
         }
#Posicicoes iniciais dos nos
```

```
$node_(29) set X_ 200.0
$node_(29) set Y_ 0.0
$node_(29) set Z_ 0.0

$node_(28) set X_ 400.0
$node_(28) set Y_ 0.0
$node_(28) set Z_ 0.0

$node_(27) set X_ 200.0
$node_(27) set Y_ 100.0
$node_(27) set Z_ 0.0

$node_(26) set X_ 400.0
$node_(26) set Y_ 100.0
$node_(26) set Z_ 0.0

$node_(25) set X_ 200.0
$node_(25) set Y_ 200.0
$node_(25) set Z_ 0.0

$node_(24) set X_ 400.0
$node_(24) set Y_ 200.0
$node_(24) set Z_ 0.0

$node_(23) set X_ 200.0
$node_(23) set Y_ 300.0
$node_(23) set Z_ 0.0

$node_(22) set X_ 400.0
$node_(22) set Y_ 300.0
$node_(22) set Z_ 0.0

$node_(21) set X_ 200.0
$node_(21) set Y_ 400.0
$node_(21) set Z_ 0.0

$node_(20) set X_ 400.0
$node_(20) set Y_ 400.0
$node_(20) set Z_ 0.0

$node_(19) set X_ 200.0
$node_(19) set Y_ 500.0
$node_(19) set Z_ 0.0

$node_(18) set X_ 400.0
$node_(18) set Y_ 500.0
$node_(18) set Z_ 0.0

$node_(17) set X_ 500.0
$node_(17) set Y_ 500.0
$node_(17) set Z_ 0.0
```

```
$node_(16) set X_ 300.0
$node_(16) set Y_ 500.0
$node_(16) set Z_ 0.0

$node_(15) set X_ 100.0
$node_(15) set Y_ 500.0
$node_(15) set Z_ 0.0

$node_(14) set X_ 500.0
$node_(14) set Y_ 400.0
$node_(14) set Z_ 0.0

$node_(13) set X_ 300.0
$node_(13) set Y_ 400.0
$node_(13) set Z_ 0.0

$node_(12) set X_ 100.0
$node_(12) set Y_ 400.0
$node_(12) set Z_ 0.0

$node_(11) set X_ 500.0
$node_(11) set Y_ 300.0
$node_(11) set Z_ 0.0

$node_(10) set X_ 300.0
$node_(10) set Y_ 300.0
$node_(10) set Z_ 0.0

$node_(9) set X_ 100.0
$node_(9) set Y_ 300.0
$node_(9) set Z_ 0.0

$node_(8) set X_ 500.0
$node_(8) set Y_ 200.0
$node_(8) set Z_ 0.0

$node_(7) set X_ 300.0
$node_(7) set Y_ 200.0
$node_(7) set Z_ 0.0

$node_(6) set X_ 100.0
$node_(6) set Y_ 200.0
$node_(6) set Z_ 0.0

$node_(5) set X_ 500.0
$node_(5) set Y_ 100.0
$node_(5) set Z_ 0.0

$node_(4) set X_ 300.0
$node_(4) set Y_ 100.0
$node_(4) set Z_ 0.0
```

```
$node_(3) set X_ 100.0
$node_(3) set Y_ 100.0
$node_(3) set Z_ 0.0

$node_(2) set X_ 500.0
$node_(2) set Y_ 0.0
$node_(2) set Z_ 0.0

$node_(1) set X_ 300.0
$node_(1) set Y_ 0.0
$node_(1) set Z_ 0.0

$node_(0) set X_ 100.0
$node_(0) set Y_ 0.0
$node_(0) set Z_ 0.0

$ns at 469.0 "$node_(0) setdest 275.0 93.0 1.0"
$ns at 744.0 "$node_(1) setdest 145.0 135.0 1.0"
$ns at 865.0 "$node_(2) setdest 249.0 397.0 1.0"
$ns at 816.0 "$node_(3) setdest 453.0 304.0 1.0"
$ns at 17.0 "$node_(4) setdest 119.0 327.0 1.0"
$ns at 667.0 "$node_(5) setdest 252.0 149.0 1.0"
$ns at 548.0 "$node_(6) setdest 44.0 211.0 1.0"
$ns at 16.0 "$node_(7) setdest 127.0 78.0 1.0"
$ns at 34.0 "$node_(8) setdest 35.0 235.0 1.0"
$ns at 584.0 "$node_(9) setdest 115.0 350.0 1.0"
$ns at 312.0 "$node_(10) setdest 54.0 139.0 1.0"
$ns at 369.0 "$node_(11) setdest 472.0 119.0 1.0"
$ns at 754.0 "$node_(12) setdest 499.0 88.0 1.0"
$ns at 386.0 "$node_(13) setdest 467.0 3.0 1.0"
$ns at 902.0 "$node_(14) setdest 295.0 42.0 1.0"
$ns at 122.0 "$node_(15) setdest 466.0 211.0 1.0"
$ns at 567.0 "$node_(16) setdest 439.0 123.0 1.0"
$ns at 114.0 "$node_(17) setdest 100.0 131.0 1.0"
$ns at 664.0 "$node_(18) setdest 186.0 304.0 1.0"
$ns at 670.0 "$node_(19) setdest 173.0 450.0 1.0"
$ns at 562.0 "$node_(20) setdest 282.0 354.0 1.0"
$ns at 572.0 "$node_(21) setdest 16.0 389.0 1.0"
$ns at 134.0 "$node_(22) setdest 56.0 407.0 1.0"
$ns at 182.0 "$node_(23) setdest 159.0 324.0 1.0"
$ns at 131.0 "$node_(24) setdest 423.0 105.0 1.0"
$ns at 714.0 "$node_(25) setdest 385.0 37.0 1.0"
$ns at 332.0 "$node_(26) setdest 409.0 182.0 1.0"
$ns at 844.0 "$node_(27) setdest 32.0 308.0 1.0"
$ns at 884.0 "$node_(28) setdest 71.0 160.0 1.0"
$ns at 245.0 "$node_(29) setdest 447.0 336.0 1.0"
$ns at 272.0 "$node_(0) setdest 226.0 477.0 1.0"
$ns at 104.0 "$node_(1) setdest 298.0 339.0 1.0"
$ns at 916.0 "$node_(2) setdest 165.0 211.0 1.0"
$ns at 148.0 "$node_(3) setdest 407.0 81.0 1.0"
$ns at 163.0 "$node_(4) setdest 397.0 118.0 1.0"
$ns at 899.0 "$node_(5) setdest 326.0 177.0 1.0"
```

```
$ns at 634.0 "$node_(6) setdest 303.0 143.0 1.0"
$ns at 667.0 "$node_(7) setdest 442.0 33.0 1.0"
$ns at 509.0 "$node_(8) setdest 311.0 243.0 1.0"
$ns at 264.0 "$node_(9) setdest 475.0 492.0 1.0"
$ns at 504.0 "$node_(10) setdest 326.0 457.0 1.0"
$ns at 58.0 "$node_(11) setdest 435.0 304.0 1.0"
$ns at 455.0 "$node_(12) setdest 259.0 310.0 1.0"
$ns at 897.0 "$node_(13) setdest 225.0 495.0 1.0"
$ns at 751.0 "$node_(14) setdest 10.0 478.0 1.0"
$ns at 964.0 "$node_(15) setdest 339.0 128.0 1.0"
$ns at 328.0 "$node_(16) setdest 379.0 313.0 1.0"
$ns at 530.0 "$node_(17) setdest 378.0 358.0 1.0"
$ns at 90.0 "$node_(18) setdest 403.0 60.0 1.0"
$ns at 44.0 "$node_(19) setdest 136.0 1.0 1.0"
$ns at 622.0 "$node_(20) setdest 369.0 41.0 1.0"
$ns at 552.0 "$node_(21) setdest 469.0 26.0 1.0"
$ns at 52.0 "$node_(22) setdest 199.0 471.0 1.0"
$ns at 947.0 "$node_(23) setdest 99.0 124.0 1.0"
$ns at 230.0 "$node_(24) setdest 384.0 135.0 1.0"
$ns at 25.0 "$node_(25) setdest 289.0 130.0 1.0"
$ns at 776.0 "$node_(26) setdest 498.0 15.0 1.0"
$ns at 727.0 "$node_(27) setdest 92.0 368.0 1.0"
$ns at 178.0 "$node_(28) setdest 75.0 15.0 1.0"
$ns at 110.0 "$node_(29) setdest 126.0 334.0 1.0"
$ns at 969.0 "$node_(0) setdest 266.0 438.0 1.0"
$ns at 52.0 "$node_(1) setdest 65.0 200.0 1.0"
$ns at 661.0 "$node_(2) setdest 259.0 473.0 1.0"
$ns at 445.0 "$node_(3) setdest 296.0 492.0 1.0"
$ns at 150.0 "$node_(4) setdest 61.0 50.0 1.0"
$ns at 902.0 "$node_(5) setdest 416.0 99.0 1.0"
$ns at 725.0 "$node_(6) setdest 74.0 389.0 1.0"
$ns at 657.0 "$node_(7) setdest 3.0 178.0 1.0"
$ns at 933.0 "$node_(8) setdest 288.0 87.0 1.0"
$ns at 535.0 "$node_(9) setdest 68.0 200.0 1.0"
$ns at 691.0 "$node_(10) setdest 297.0 142.0 1.0"
$ns at 678.0 "$node_(11) setdest 322.0 165.0 1.0"
$ns at 723.0 "$node_(12) setdest 246.0 489.0 1.0"
$ns at 142.0 "$node_(13) setdest 38.0 490.0 1.0"
$ns at 660.0 "$node_(14) setdest 92.0 321.0 1.0"
$ns at 230.0 "$node_(15) setdest 75.0 156.0 1.0"
$ns at 937.0 "$node_(16) setdest 68.0 255.0 1.0"
$ns at 284.0 "$node_(17) setdest 2.0 473.0 1.0"
$ns at 301.0 "$node_(18) setdest 183.0 327.0 1.0"
$ns at 622.0 "$node_(19) setdest 440.0 451.0 1.0"
$ns at 703.0 "$node_(20) setdest 120.0 287.0 1.0"
$ns at 684.0 "$node_(21) setdest 336.0 349.0 1.0"
$ns at 449.0 "$node_(22) setdest 456.0 9.0 1.0"
$ns at 63.0 "$node_(23) setdest 180.0 140.0 1.0"
$ns at 195.0 "$node_(24) setdest 94.0 254.0 1.0"
$ns at 858.0 "$node_(25) setdest 127.0 484.0 1.0"
$ns at 576.0 "$node_(26) setdest 49.0 121.0 1.0"
$ns at 443.0 "$node_(27) setdest 457.0 249.0 1.0"
```

```
$ns at 536.0 "$node_(28) setdest 97.0 22.0 1.0"
$ns at 46.0 "$node_(29) setdest 258.0 227.0 1.0"

#Define a conexao TCP entre 0 e 16 Comecando em 15.0
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
$tcp set packetSize_ 512
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0)  $tcp
$ns attach-agent $node_(18) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 10.0 "$ftp start"

 #Exibe o tamanho da janela
proc plotWindow {tcpSource file} {
global ns
set time 0.01
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
puts $file "$now $cwnd"
$ns at [expr $now+$time] "plotWindow $tcpSource $file" }
$ns at 10.1 "plotWindow $tcp $windowVsTime2"

#Define os nos inicias no nam
for {set i 0} {$i < $val(nn)} { incr i } {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}

 #Mostra para os nos quando a simualacao termina
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "$node_($i) reset";
}

 #termina nam e a simulacao
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 1000.01 "puts \"end simulation\" ; $ns halt"
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
}

 $ns run
```

APÊNDICE D – CÓDIGO EM SHELL PARA AUTOMATIZAR OBTENÇÃO DE RESULTADOS.

```
#!/bin/bash
# set i to 9
i=9
while [ $i -lt 31 ]; do
        echo "No $i"
        gawk -f ~/Documentos/TG/Scripts/Throughput2007.awk
~/Documentos/TG/TCP/1Tr/"$i"TCP/AODV/*.tr >> AODV1TrThr.txt
        gawk -f ~/Documentos/TG/Scripts/e2edelay.awk
~/Documentos/TG/TCP/1Tr/"$i"TCP/AODV/*.tr >> AODV1TrDelay.txt
        gawk -f ~/Documentos/TG/Scripts/e2edelay.awk
~/Documentos/TG/TCP/1Tr/"$i"TCP/AODV/*.tr >> AODV1TrPDR.txt
      gawk -f ~/Documentos/TG/Scripts/Throughput2007.awk
~/Documentos/TG/TCP/1Tr/"$i"TCP/DSR/*.tr >> DSR1TrThr.txt
      gawk -f ~/Documentos/TG/Scripts/e2edelay.awk
~/Documentos/TG/TCP/1Tr/"$i"TCP/DSR/*.tr >> DSR1TrDelay.txt
      gawk -f ~/Documentos/TG/Scripts/e2edelay.awk
~/Documentos/TG/TCP/1Tr/"$i"TCP/DSR/*.tr >> DSR1TrPDR.txt
        let i=i+1;
done
```

ANEXO A – CÓDIGO AWK OBTER DELAY e PDR

```
# http://205.196.121.184/fnufnnc17mwg/zjkxzk4bkrwqhkc/e2edelay.awk
# http://mohittahiliani.blogspot.dk/2010/02/few-more-awk-scripts-for-ns2.html
# ===================================================================

# AWK Script for calculating:

#     => Average End-to-End Delay.

# ===================================================================


BEGIN {
   seqno = -1;

   droppedPackets = 0;

   receivedPackets = 0;

   count = 0;
}

{
   if($4 == "AGT" && $1 == "s" && seqno < $6) {
        seqno = $6;

   }
         else if(($4 == "AGT") && ($1 == "r")) {
         receivedPackets++;

   } else if ($1 == "D" && $7 == "tcp" && $8 > 512){
         droppedPackets++;

  }
   #end-to-end delay
   if($4 == "AGT" && $1 == "s") {
        start_time[$6] = $2;
   } else if(($7 == "tcp") && ($1 == "r")) {
      end_time[$6] = $2;
   } else if($1 == "D" && $7 == "tcp") {
        end_time[$6] = -1;
   }
}

END {
   for(i=0; i<=seqno; i++) {
        if(end_time[i] > 0) {
           delay[i] = end_time[i] - start_time[i];
              count++;
```

```
        }
           else
           {
                delay[i] = -1;
           }
    }
    for(i=0; i<=seqno; i++) {
         if(delay[i] > 0) {
            n_to_n_delay = n_to_n_delay + delay[i];
        }
    }
   n_to_n_delay = n_to_n_delay/count;

    print "\n";
    print "GeneratedPackets            = " seqno+1;
    print "ReceivedPackets             = " receivedPackets;
    print "Packet Delivery Ratio       = " receivedPackets/(seqno+1)*100 "%";
    print receivedPackets/(seqno+1)*100;
    print "Total Dropped Packets = " droppedPackets;
    print  n_to_n_delay * 1000;
    print "\n";
}
```

ANEXO B – CÓDIGO PARA OBTER VAZÃO

```
# http://mailman.isi.edu/pipermail/ns-users/2007-August/060808.html
#
#
#         AWK script for calculate the throughput
#   Throughput.awk


 BEGIN {
     recvdSize = 0
     startTime = 1e6
     stopTime = 0
 }
 {
     # Trace line format: normal
     if ($2 != "-t") {
          event = $1
          time = $2
          if (event == "+" || event == "-") node_id = $3
          if (event == "r" || event == "d") node_id = $4
          flow_id = $8
          pkt_id = $12
          pkt_size = $6
          flow_t = $5
          level = "AGT"
     }
     # Trace line format: new
     if ($2 == "-t") {
          event = $1
          time = $3
          node_id = $5
          flow_id = $39
          pkt_id = $41
          pkt_size = $37
          flow_t = $45
          level = $19
     }

 # Store start time
 if (level == "AGT" && (event == "+" || event == "s") && pkt_size >= 512) {
   if (time < startTime) {
          startTime = time
          }
     }

 # Update total received packets' size and store packets arrival time
 if (level == "AGT" && event == "r" && pkt_size >= 512) {
     if (time > stopTime) {
          stopTime = time
          }
```

```
        # Rip off the header
        hdr_size = pkt_size % 512
        pkt_size -= hdr_size
        # Store received packet's size
        recvdSize += pkt_size
        }
    }
    END {
        printf("%.2f\t\t \n",(recvdSize/(stopTime-startTime))*(8/1000),startTime,stopTime)
    }
BEGIN {
        recvdSize = 0
        startTime = 1e6
        stopTime = 0
    }
    {
        # Trace line format: normal
        if ($2 != "-t") {
            event = $1
            time = $2
            if (event == "+" || event == "-") node_id = $3
            if (event == "r" || event == "d") node_id = $4
            flow_id = $8
            pkt_id = $12
            pkt_size = $6
            flow_t = $5
            level = "AGT"
        }
        # Trace line format: new
        if ($2 == "-t") {
            event = $1
            time = $3
            node_id = $5
            flow_id = $39
            pkt_id = $41
            pkt_size = $37
            flow_t = $45
            level = $19
        }
    # Store start time
    if (level == "AGT" && (event == "+" || event == "s") && pkt_size >= 512) {
      if (time < startTime) {
            startTime = time
            }
        }
    # Update total received packets' size and store packets arrival time
    if (level == "AGT" && event == "r" && pkt_size >= 512) {
        if (time > stopTime) {
            stopTime = time
            }
        # Rip off the header
        hdr_size = pkt_size % 512
```

```
        pkt_size -= hdr_size
        # Store received packet's size
        recvdSize += pkt_size
        }
    }
    END {
        #printf("Average Throughput[kbps] = %.2f\t\t
StartTime=%.2f\tStopTime=%.2f\n",(recvdSize/(stopTime-
startTime))*(8/1000),startTime,stopTime)
    }
```