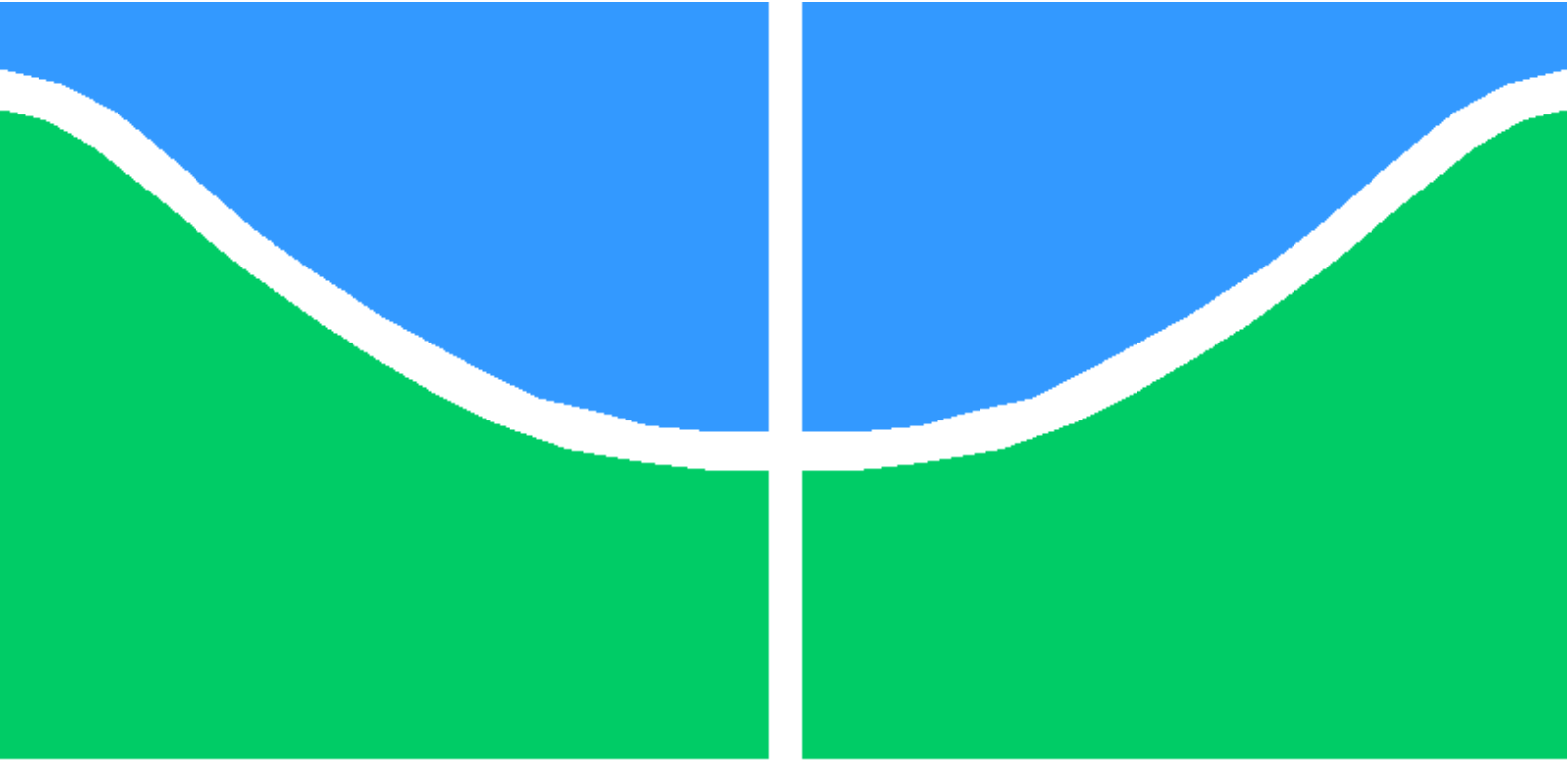


TRABALHO DE GRADUAÇÃO

IMPLEMENTAÇÃO E SIMULAÇÃO DE REDES AD HOC DE COMUNICAÇÕES SEM FIO COM MÚLTIPLAS ANTENAS

Ana Carolina de Oliveira Christófaro

Fadhil Firyaguna

Brasília, julho de 2012

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

IMPLEMENTAÇÃO E SIMULAÇÃO DE REDES AD HOC DE COMUNICAÇÕES SEM FIO COM MÚLTIPLAS ANTENAS

Ana Carolina de Oliveira Christófaro

Fadhil Firyaguna

Relatório submetido ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Engenheiro de Redes de Comunicação

Banca Examinadora

Prof. Marcelo Menezes de Carvalho, ENE/UnB _____
Orientador

Prof. Renato Mariz de Moraes, ENE/UnB _____
Examinador interno

Profª. Katia Obraczka, SOE/UCSC _____
Examinadora externa

Dedicatórias

Ao leitor que pesquisou algum termo deste trabalho, por acaso encontrou este e ficou curioso em lê-lo. Ao leitor que não encontrou o que estava procurando e leu este trabalho por ter sido sua última opção. Ao leitor que estava querendo buscar algo sobre o tema deste trabalho e o encontrou na primeira opção. Ao leitor o qual pedi para procurar este trabalho para ler e o fez. E a todos que desejaram o sucesso deste trabalho.

Fadhil Firyaguna

À nova geração de pesquisadores.

Ana Carolina de Oliveira Christófaro

Agradecimentos

Agradeço aos meus pais, Marília e Ângelo, que compartilharam os meus ideais. Pelo apoio, dedicação e companheirismo. Aos meus avós, pelo exemplo de vida; à minha dupla, Fadhil, por dividir comigo as dificuldades e as alegrias desde o começo; ao Éverton e ao Tiago, pela colaboração e trabalho em equipe; aos colegas que me receberam, pela amizade e aprendizado; à Vera, à Anna Carolina e ao professor Anderson, pelo apoio e incentivo; aos professores que acreditaram em mim e me deram alicerce para chegar até aqui, em especial ao professor Marcelo, pela oportunidade e pelo alcance do trabalho desenvolvido; aos meus amigos que fizeram os meus dias melhores; e ao meu computador, que aguentou até o final.

Ana Carolina de Oliveira Christófaro

Agradeço ao Universo por ter conspirado a favor deste trabalho. Obrigado por ter explodido. Também agradeço à Ana Carolina, ao Éverton e ao Tiago, pelo trabalho em grupo; ao professor Marcelo, por todo o trabalho, sim, todo o trabalho; ao Pedro e à Amanda por terem me ajudado a mexer no simulador no começo do projeto; a todos os colegas de turma pela convivência, amizade e aprendizado em conjunto no decorrer destes valiosos 5 anos; aos professores da universidade que tornaram este aprendizado possível; aos professores das escolas onde passei que sem eles nada disso teria nem sido imaginado; aos meus amigos que apoiaram as minhas escolhas; e à minha família por ter criado a pessoa que sou hoje. E também ao software livre, pelas ferramentas gratis.

Fadhil Firyaguna

RESUMO

Este trabalho estuda o desempenho de redes sem fio *ad hoc* com nós habilitados com múltiplas antenas (MIMO) utilizando tecnologias-chave para permitir altas taxas de transmissão e/ou maior robustez à transmissão. Para tanto, busca-se avaliar as técnicas de transmissão do sistema *Vertical Bell Labs Layered Space-Time* (V-BLAST) (o qual permite a multiplexagem espacial) e do esquema de Alamouti (que explora a diversidade de transmissão). Para realizar este estudo, propomos uma implementação de um sistema MIMO que utiliza estas técnicas no simulador de rede NS-3. Esta implementação é baseada em modelos analíticos e expressões matemáticas resultantes das abstrações da decodificação do sinal na recepção na camada física e consiste em simples modificações na norma IEEE 802.11b que incluem não só os aspectos do cálculo da SINR e da BER, mas também como o mecanismo de detecção de portadora é realizado e como a negociação de 4 vias é configurada. Os resultados de simulação para redes *ad hoc* IEEE 802.11 habilitadas com o esquema de Alamouti, sob canais com desvanecimento Rice, indicam ganhos significativos na vazão média da rede sobre sistemas de única antena, especialmente sob forte desvanecimento de múltiplos percursos. Além disso, os resultados para redes habilitadas com o sistema V-BLAST, sob canais com desvanecimento Rayleigh, mostram que, dependendo da configuração de antenas, ganhos significativos de vazão quase lineares podem ser alcançados, se os ganhos de diversidade forem explorados. Com base nestes resultados, é proposto um sistema MIMO híbrido com protocolo MAC adaptativo que comuta a técnica de transmissão de acordo com as condições do canal, a fim de prover melhores ganhos de desempenho explorando as vantagens da diversidade e da multiplexagem. Este protocolo segue o paradigma do projeto com camadas interrelacionadas ("*cross-layer design*") para realizar a comutação da técnica segundo informações trocadas pelas camadas física e MAC. Estudos são realizados para avaliar os melhores limiares de comutação a fim de otimizar o desempenho. Os resultados de simulação mostram ganhos de vazão de rede maiores que o desempenho de redes utilizando cada técnica individualmente em toda a rede.

Palavras-chave IEEE 802.11, Wifi, Redes Ad Hoc, MIMO, múltiplas antenas, multiplexagem espacial, V-BLAST, diversidade de transmissão, Alamouti, simulação, NS-3, sistema MIMO híbrido, protocolo MAC adaptativo, avaliação de desempenho, vazão média da rede.

ABSTRACT

This work studies the performance of wireless *ad hoc* networks when nodes are equipped with multiple antennas (MIMO) utilizing key technologies that allow higher data rates and/or robustness to transmission. To do so, we seek to evaluate the transmission techniques known as *Vertical Bell Labs Layered Space-Time* (V-BLAST) system (which allows spatial-division multiplexing) and the Alamouti scheme (which exploits transmit diversity). To accomplish this task, we propose an implementation of these MIMO techniques in the network simulator NS-3. This implementation is based on analytical models and expressions obtained from signal decoding abstractions from the physical layer reception, and consists in simple modifications to the IEEE 802.11b standard. These modifications include not only SINR and BER computation aspects, but also how clear channel assessment is performed and how the four-way handshake mechanism is set. Simulation results for IEEE 802.11 *ad hoc* networks enabled with the Alamouti scheme, under Rice fading channel, indicate significant gains in network throughput over single antenna systems, especially under strong multipath fading. Furthermore, simulation results for networks enabled with V-BLAST show that, depending on the antenna configuration, significant quasi-linear network throughput gains can be achieved, especially if the diversity gains are also exploited. Based on these results, we propose a MIMO hybrid system with adaptive MAC protocol that selects the transmission technique depending on channel conditions, in order to provide better performance gains exploiting diversity and multiplexing gains. This protocol follows the cross-layer design paradigm to choose the best technique according to information exchanged by the physical and MAC layers. Studies are carried out to evaluate the best selection thresholds in order to optimize throughput. Simulation results demonstrate higher network throughput gains compared to the performance of networks that utilize each technique individually in all network nodes (or elements).

Keywords IEEE 802.11, Wifi, Ad Hoc networks, MIMO, multiple-input multiple-output, spatial-division multiplexing, V-BLAST, transmit diversity, Alamouti, simulation, NS-3, hybrid MIMO system, adaptative MAC protocol, performance evaluation, average network throughput.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

Acrônimos

ACK	Acknowledgment
AWGN	Additive White Gaussiane Noise
BER	Bit Error Rate
BLAST	Bell Labs Layered Space-Time
BPSK	Binnary Phase Shift Keying
BSS	Basic Service Set
CCA	Clear Channel Assessment
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CTS	Clear-to-Send
DBPSK	Diferential Binary Phase Shift Keying
DCF	Distributed Coordenation Function
DIFS	Distributed Inter Frame Space
DSSS	Direct Sequence Spread Spectrum
DQPSK	Diferential Quadrature Phase Shift Keying
ERP	Extended Rate PHY
FAMA	Floor Acquisition Multiple Access
FHSS	Frequency Hopping Spread Spectrum
HCF	Hybrid Coordination Function
HR	High Rate
HT	High Throughput
IBSS	Independent Basic Service Set
IEEE	Institute of Eletrical and Eletronic Engineers
IR	Infra-Red
ISM	Industrial, Scientific and Medical
JC	Janela de Contenção
LOS	Line-of-Sight
MAC	Medium Access Control
MAI	Multiple Access Interference
MC	Monte Carlo
MCS	Modulation/Coding Scheme
MIMO	Multiple-Input Multiple-Output
MISO	Multiple-Input Single-Output
MMSE	Minimum Mean-Squared Error

Acrônimos

ML	Maximum Likelihood
MPR	Multiple Packet Reception
NAV	Network Allocation Vector
NLOS	Non Line-of-Sight
NS-2	Network Simulator 2
NS-3	Network Simulator 3
OFDM	Orthogonal Frequency Division Multiplexing
PDU	Protocol Data Unit
PHY	Physical Layer (Camada física)
PN	Pseudo-Noise
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
RIMA	Receiver Initiated Multiple Access
RSSI	Received Signal Strenght Indicator
RTS	Request-to-Send
SDM	Spatial-Division Multiplexing
SIFS	Short Inter Frame Space
SISO	Single-Input Single-Output
SINR	Signal-to-Interference-Plus-Noise Ratio
SNR	Signal-to-Noise Ratio
STBC	Space-Time Block Coding
TCP	Transport Control Protocol
TDCA	Time Division Collision Avoidance
TXOP	Transmit Oportunity
V-BLAST	Vertical Bell Labs Layered Space-Time
UDP	User Datagram Protocol
Wi-Fi	Wireless Fidelity
ZF	Zero-Forcing
ZF-SIC	Zero-Forcing Successive Interference Cancellation

Abreviaturas

HYB-A	Modo de operação utilizando apenas o limiar inferior
HYB-B	Modo de operação utilizando apenas o limiar superior
HYB-C	Modo de operação utilizando os limiares inferior e superior
M	Número de antenas transmissoras
N	Número de antenas receptoras
RX	Receptor
rxAnt	Número de antenas receptoras
rxMAC	Camada MAC do receptor
rxPHY	Camada PHY do receptor
snrMin	Limiar inferior de comutação
snrMax	Limiar superior de comutação
TX	Transmissor
txAnt	Número de antenas transmissoras
txMAC	Camada MAC do transmissor
txPHY	Camada PHY do transmissor

Capítulo 1

Introdução

1.1 Contextualização

Com a crescente popularidade dos dispositivos móveis e o drástico aumento da demanda por tráfego multimídia nos últimos anos, a busca por soluções que tornem o rendimento de redes com largura de banda limitada mais eficiente se tornou um objetivo comum na academia e na indústria. Várias soluções já foram propostas na literatura para esta questão e, em particular, o uso de sistemas habilitados com múltiplas antenas transmissoras e receptoras (MIMO, do inglês *Multiple-Input Multiple-Output*) já se tornou uma realidade após quase três décadas desde que foram inicialmente propostas [3], [4]. Padrões recentes de comunicação sem fio já adotaram MIMO como uma das tecnologias-chave para permitir altas taxas de transmissão e/ou maior robustez à transmissão.

Apesar da sua larga aceitação, até então, a maioria dos estudos relacionados ao uso de tecnologias MIMO têm se concentrado na camada física (PHY, do inglês *Physical Layer*), tendo como foco a análise do seu desempenho em um simples enlace. Contudo, no âmbito de redes de comunicação sem fio, a interferência de múltiplos acessos (MAI, do inglês *Multiple Access Interference*) resultante do protocolo de controle de acesso ao meio (MAC, do inglês *Medium Access Control*), e da topologia da rede — definida pela conectividade dos nós via rádio — podem comprometer severamente os ganhos esperados. Como resultado, a avaliação do impacto de soluções MIMO no desempenho global da rede requer mais estudos, uma vez que sua integração no sistema pode impor questões adicionais de projeto que demandarão soluções mais inteligentes para obter os ganhos esperados com o uso de MIMO [5].

Não obstantemente, a carência de implementações apropriadas das funcionalidades MIMO nos simuladores de redes atuais tem impedido, de certa maneira, o desenvolvimento de pesquisas mais consistentes no contexto de redes. Enquanto simulações Monte Carlo são capazes de reproduzir mecanismos MIMO precisamente em nível de enlace (mas são inapropriadas para simulações em nível de rede), simuladores de redes a eventos discretos geralmente se utilizam de abstrações matemáticas da camada física, dada sua incapacidade de lidar com eventos em nível de símbolo.

Com relação às pesquisas realizadas no âmbito de redes *ad hoc* que utilizam tecnologias com múltiplas antenas, um número considerável de trabalhos já foram realizados na aplicação de técnicas

de conformação de feixes para uso das chamadas "antenas direcionáveis" (também conhecida como técnica "*beamforming*"). Contudo, o estudo de redes *ad hoc* habilitadas com sistemas MIMO com o objetivo de explorar a diversidade espacial (de transmissão ou de recepção para aumentar a robustez da comunicação) ou ganhos com a multiplexagem (enviando dados simultaneamente através das múltiplas antenas) ainda merecem mais investigação. Até o momento, a maioria dos trabalhos desenvolvidos nesta área têm considerado cenários em que os dispositivos da rede estão totalmente conectados (dentro do raio de alcance de todos) ou cenários simplificados em que não se permitem colisões [6], [7], [8] ou utilizam uma abstração simples para representar o comportamento MIMO na camada física [9]. Sendo assim, existe ainda a necessidade de se investigar o impacto da interferência de múltiplo acesso, do problema do terminal escondido, e de técnicas de avaliação de ocupação do canal nessas redes.

Motivados por este fato, apresentamos um estudo sobre o desempenho de redes sem fio *ad hoc* habilitadas com múltiplas antenas a partir da implementação no simulador NS-3 de dois sistemas MIMO representativos das técnicas de diversidade a partir do transmissor e de multiplexagem espacial, além da questão do mecanismo de detecção de canal livre ao utilizar múltiplas antenas. Avaliamos as técnicas de transmissão MIMO pelo seu comportamento em nível de rede a partir de abstrações da camada física feitas de abordagens analíticas e expressões matemáticas do desempenho em nível de enlace. Propomos então um sistema híbrido de técnicas de transmissão MIMO com o protocolo MAC adaptativo que comuta a técnica de transmissão do quadro de dados de acordo com a qualidade do canal. Para isso, fazemos uma análise a respeito dos parâmetros de comutação para serem definidos na avaliação do protocolo a fim de otimizar seu desempenho. Finalmente, avaliamos o desempenho do protocolo e demonstramos seus ganhos em relação a outros sistemas.

1.2 Descrição do Trabalho

Este trabalho apresenta uma investigação do desempenho de redes *ad hoc* sem fio com nós habilitados com a tecnologia MIMO utilizando as técnicas de transmissão do sistema V-BLAST (*Vertical Bell Labs Layered Space-Time*) [1], o qual permite a multiplexagem espacial, e do esquema de Alamouti [10], que explora a diversidade de transmissão. Para tanto, será apresentada uma proposta de implementação dessas técnicas no simulador NS-3 [11], baseada no trabalho original realizado por Loyka e Gagnon [2], onde foi desenvolvida uma expressão analítica para a taxa média de erro de bit (BER, do inglês *Bit Error Rate*) de sistemas V-BLAST e baseada na abordagem analítica realizada por Carvalho e Garcia-Luna-Aceves [12] do esquema de Alamouti.

A implementação consiste em uma modificação no padrão IEEE 802.11b, que inclui não apenas aspectos relacionados ao cálculo da relação sinal-interferência mais ruído (SINR, do inglês *Signal-to-Interference-Plus-Noise Ratio*) e do BER, mas também a forma como é realizada a detecção de portadora e como o mecanismo de estabelecimento da conexão, conhecido como negociação de 4 vias (*4-way handshake*), deve ser configurado para transmissões habilitadas com MIMO. Em cima dessa modificação, propomos e implementamos um protocolo em camadas MAC e PHY cujo objetivo é utilizar a melhor técnica de transmissão dependendo do estado do canal a fim de diminuir

a taxa de perda de quadros em canais sujeitos a muita interferência de diversas fontes de sinais (ganho de diversidade) ou aumentar a taxa de transmissão em canais sujeitos a pouca interferência (ganho de multiplexagem).

As investigações foram feitas sobre topologias aleatórias, considerando transmissões concomitantes, para diferentes configurações de antenas. A propagação do sinal transmitido ocorre sob atenuação de percurso em grande escala e sob efeito de desvanecimento em pequena escala. A avaliação das técnicas de transmissão do sistema V-BLAST e do esquema de Alamouti são realizadas em comparação com o desempenho de sistemas com o padrão IEEE 802.11b, que utilizam uma antena para transmissão e recepção (SISO, do inglês *Single-Input Single-Output*). Os resultados de simulação mostram que, em comparação com a configuração SISO, o esquema de Alamouti e o sistema V-BLAST obtiveram ganhos de diversidade ao aumentar o número de antenas receptoras, e que o sistema V-BLAST obteve ganhos de multiplexagem ao aumentar o número de antenas transmissoras.

Os resultados da avaliação de desempenho de redes *ad hoc* habilitadas com o sistema V-BLAST e a descrição da implementação do sistema no simulador, realizados em nosso trabalho, foram publicados em [13]:

- Fadhil Firyaguna, Ana Carolina O. Christófaro, Éverton L. Andrade, Tiago S. Bonfim e Marcelo M. Carvalho, "Throughput Performance of V-BLAST-Enabled Wireless Ad Hoc Networks", the 1st IEEE International Conference on Communications in China (ICCC), Pequim, Agosto, 2012.

Esta publicação estuda o desempenho da vazão média da rede *ad hoc* de comunicação sem fio com nós equipados com o sistema V-BLAST. O estudo é realizado a partir da implementação no simulador de rede apresentado neste trabalho e neste estudo é discutido o compromisso entre o ganho de diversidade de recepção e o ganho de multiplexagem.

A avaliação do protocolo proposto é realizada, inicialmente, fazendo um estudo prévio do comportamento da rede quando os nós podem comutar entre as técnicas que tem melhor desempenho sob condições mais adversas do canal, e em seguida, quando os nós podem comutar entre as técnicas que tem melhor desempenho sob condições mais favoráveis do canal. Este estudo serve como base na escolha dos parâmetros que irão otimizar o desempenho do sistema. A avaliação é feita tendo como referência o melhor desempenho de vazão média da rede habilitada com o sistema V-BLAST e os resultados de simulação mostram ganhos no desempenho da rede com a utilização do protocolo adaptativo proposto.

1.3 Objetivos do Projeto

Este projeto teve como objetivo fazer um estudo do impacto, em nível de rede, da utilização de técnicas de transmissão MIMO, implementando-os em um simulador computacional de redes, e a partir deste estudo, propor um sistema híbrido, no qual estejam habilitadas as duas técnicas (V-BLAST e Alamouti), com um protocolo adaptativo que permite selecionar a melhor técnica em função da condição do canal. A avaliação de desempenho, por meio de simulações de redes sem fio

ad hoc, teve como meta analisar os ganhos relativos dos sistemas em avaliação.

1.4 Contribuições do Projeto

Entre as principais contribuições deste trabalho, destacam-se:

- Implementação do sistema V-BLAST e do esquema de Alamouti no simulador de redes NS-3;
- Avaliação do desempenho de redes sem fio do tipo *ad hoc* habilitadas com MIMO, considerando redes com transmissões concomitantes;
- Avaliação de desempenho do sistema V-BLAST e do esquema de Alamouti para diferentes configurações de antenas;
- Proposta de um protocolo híbrido (utiliza as duas técnicas) e adaptativo (escolhe de acordo com o estado do canal) em camadas MAC e PHY;
- Implementação do protocolo híbrido adaptativo no simulador NS-3;
- Avaliação de desempenho do protocolo híbrido adaptativo.

1.5 Apresentação do Manuscrito

Inicialmente, o Capítulo 2 apresenta a fundamentação teórica. No Capítulo 3 é feita uma revisão bibliográfica sobre o tema de estudo. Em seguida, o Capítulo 4 descreve as modificações realizadas no código do simulador para implementar a tecnologia MIMO, suas técnicas de transmissão e o protocolo adaptativo proposto. Resultados de simulação são discutidos no Capítulo 5, seguido das conclusões no Capítulo 6. Enfim, os anexos contêm o código da implementação no simulador.

Capítulo 2

Fundamentação Teórica

2.1 Introdução

Neste capítulo introduzimos os principais conceitos abordados neste trabalho. Apresentamos na Seção 2.2 o padrão para comunicações sem fio IEEE 802.11, a arquitetura de seu sistema e as especificações para a camada de controle de acesso ao meio (MAC) e para a camada física (PHY). Em seguida, na Seção 2.3, apresentamos o canal de propagação sem fio, os efeitos de propagação e descrevemos os modelos de canal. Nesta seção, também introduzimos o conceito de transmissão por múltiplas antenas (MIMO) e descrevemos o modelo de canal MIMO. Finalmente, apresentamos as técnicas de transmissão MIMO e seus respectivos modelos analíticos na Seção 2.4.

2.2 O Padrão IEEE 802.11

Com a popularidade das estações de trabalho e dos computadores pessoais em meados dos anos 90, surgiu um grande interesse em prover as altas velocidades já disponíveis em conexões cabeadas para redes locais de acesso sem fio. Neste contexto, foram desenvolvidos produtos com padrões proprietários de comunicação, que sofriam com muitos problemas, incluindo desde os elevados custos de fabricação e o baixo volume de produção, até a incompatibilidade entre os fornecedores [14]. Motivado por este fato, o IEEE 802.11 [15] surgiu como um padrão internacional para definir as operações das redes locais sem fio conhecidas como Wi-Fi (do inglês *Wireless Fidelity*).

Atualmente, o IEEE 802.11 engloba um conjunto de normas que descrevem as funções e os serviços requeridos por um dispositivo móvel para operar tanto no modo em infra-estrutura como no modo dito *ad hoc*. No primeiro caso, as estações canalizam todo tráfego através de um ponto de acesso centralizado, enquanto no segundo caso elas podem estabelecer comunicação direta com qualquer outra estação. Esses padrões definem os procedimentos de controle de acesso ao meio (MAC), as técnicas de sinalização da camada física (PHY) e as funções da interface que são controladas pelo IEEE 802.11 MAC, além de descreverem os requerimentos e procedimentos para prover confiabilidade da informação do usuário e da autenticação dos dispositivos, entre outras especificações [15].

O padrão IEEE 802.11 original especifica a operação na faixa de frequência de 2,45 GHz, conhecida como banda ISM (do inglês *Industrial, Scientific, and Medical*). Em sua especificação original, o IEEE 802.11 foi definido para operação em taxas de 1 Mbps e 2 Mbps, usando técnicas de espalhamento espectral por salto em frequência (FHSS, do inglês *Frequency Hopping Spread Spectrum*) e por sequência direta (DSSS, do inglês *Direct Sequence Spread Spectrum*). Visando atingir maiores taxas de dados, foram desenvolvidos os padrões IEEE 802.11a e IEEE 802.11b. O primeiro implementa o esquema de modulação conhecido como OFDM (do inglês *Orthogonal Frequency Division Multiplexing*) e, operando em uma banda de frequência de 5 GHz, é capaz de prover taxas de dados a 54 Mbps. O segundo mantém a abordagem da técnica DSSS e, operando na banda ISM, é capaz de prover 11Mbps. No decorrer dos anos, novas abordagens surgiram, sendo possível alcançar taxas de dados superiores a 600 Mbps com o padrão IEEE 802.11n, e até mesmo melhores níveis de qualidade de serviço (QoS, do inglês *Quality of Service*) com o padrão IEEE 802.11e.

No projeto desses padrões deve-se encarar alguns desafios e restrições que não são impostos no meio cabeado, como a maior susceptibilidade à interferência, aos efeitos de dispersão, difração, reflexão e espalhamento do sinal ao propagar-se no canal (causando o desvanecimento do sinal). Deve-se atentar para os efeitos de redução ou perda da conectividade causados pela mobilidade dos dispositivos e a distância entre eles, a exposição à propagação do sinal por múltiplos percursos, a existência de terminais escondidos e expostos, as falhas de segurança devido ao meio de comunicação aberto e as limitações impostas pelo consumo de energia.

A seguir, detalhamos alguns aspectos relacionados ao padrão IEEE 802.11, e os desafios e restrições que devem ser considerados no projeto desses padrões. Na Seção 2.2.1 discutimos a arquitetura do sistema sem fio do IEEE 802.11, que pode ser definida de dois modos distintos: *ad hoc* ou em infra-estrutura. Nas Seções 2.2.2 e 2.2.3 apresentamos uma breve descrição dos procedimentos da camada de controle de acesso ao meio e da camada física, adotados pelo padrão IEEE 802.11, respectivamente.

2.2.1 Arquitetura do Sistema

A arquitetura do IEEE 802.11 pode ser definida de dois modos distintos: *ad hoc* ou em infra-estrutura. Esses modos se diferem nas definições do método de acesso ao meio e nas funcionalidades da rede.

Uma rede em infra-estrutura é composta por estações que, sob controle de uma função de coordenação, canalizam todo tráfego através de um ponto de acesso centralizado, formando um conjunto de serviço básico (BSS, do inglês *Basic Service Set*) na sua zona de cobertura. Por outro lado, a rede *ad hoc* é um agrupamento de estações em um conjunto de serviço básico independente (IBSS, do inglês *Independent Basic Service Set*), que podem estabelecer comunicação direta com qualquer outra estação da IBSS, sem precisar canalizar todo tráfego através de um ponto de acesso centralizado, com o auxílio de uma função de coordenação distribuída. Uma rede pode ser formada por várias IBSSs, com base na distância entre elas ou usando diferentes frequências de portadora. A Figura 2.1 ilustra um exemplo de rede *ad hoc* formada por duas IBSS.

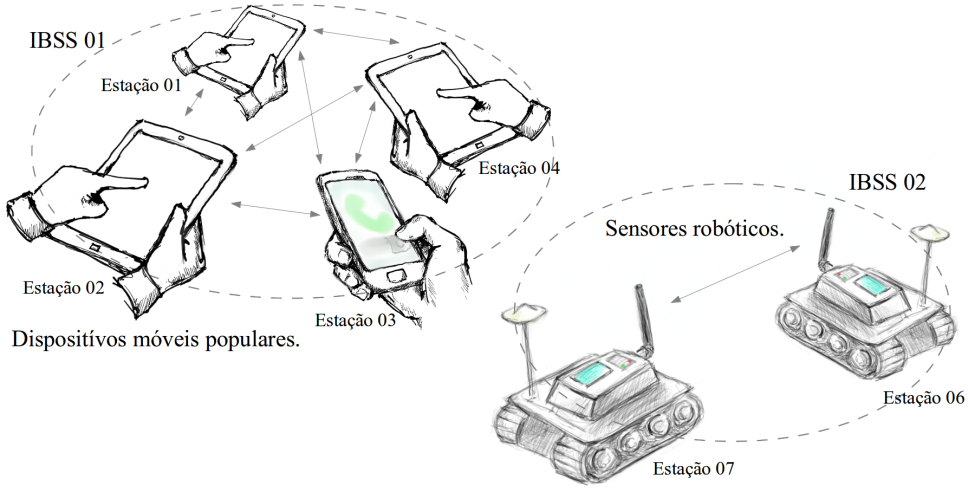


Figura 2.1: Exemplo de redes *ad hoc*.

Com o crescimento da utilização de aplicações de redes de comunicação em cenários que exigem rápida instalação, tolerância a falhas e conectividade entre todos os dispositivos (os quais podem ainda apresentar mobilidade), as redes *ad hoc* se apresentam como uma alternativa onde todos os dispositivos móveis se auto-organizam em redes de acordo com suas localizações e necessidades [14]. As principais vantagens dessa arquitetura são o baixo custo de implantação e a alta flexibilidade, podendo ser totalmente desprovidas de hierarquia ou possuir um controlador, escolhido no momento em que a rede é formada.

Exemplos de aplicações de redes *ad hoc* incluem operações de resgate em áreas remotas, ou quando uma cobertura local deve ser rapidamente implantada em um sítio de construção remoto. Também poderia servir para prover acesso a rede sem fio em áreas públicas com rápida implantação e cobertura estendida. Outras propostas apresentam o uso de redes *ad hoc* como uma rede de sensores, por exemplo, para monitoramento do meio ambiente, de animais ou de objetos [16].

2.2.2 Camada de Controle de Acesso ao Meio

A camada de controle de acesso ao meio (MAC) tem um papel fundamental na forma como a capacidade do enlace é utilizada, principalmente em redes *ad hoc* onde não há uma função de coordenação centralizada. Os protocolos desta camada são responsáveis por gerenciar quando uma estação pode proceder uma tentativa de transmissão. Entretanto, as atividades de acesso ao canal de todas as estações na rede irão contribuir para a interferência agregada em um receptor particular, que, por sua vez, será um fator determinante para o desempenho do protocolo.

Para controlar o acesso ao canal em redes *ad hoc* habilitadas com o padrão IEEE 802.11, a camada MAC faz uso do método baseado em serviço de contenção, oferecido pela função de coordenação distribuída (DCF, do inglês *Distributed Coordination Function*). Por serviço de contenção, ou disputa, entende-se que cada estação com um quadro enfileirado para transmissão deve disputar o acesso ao canal e, uma vez o quadro transmitido, deve disputar novamente pelo acesso ao canal para todos os quadros subsequentes, promovendo a equidade no acesso ao canal para todas as estações. Sendo assim, o método da função DCF permite que todos os usuários com dados para

transmitir tenham uma chance igualmente justa para acessar a rede [17].

Além dos procedimentos de acesso ao canal, a camada MAC também é responsável pelo endereçamento de unidades de dados do protocolo (PDU, do inglês *Protocol Data Unit*), formatação de quadros, verificação de erros, fragmentação e remontagem.

Dando ênfase ao nosso trabalho, na Seção 2.2.2.1 descrevemos o método de acesso adotado pela função DCF e, para melhor entendimento do impacto desse método no comportamento global da rede, apresentamos na Seção 2.2.2.2 os problemas de terminal escondido e exposto. Tais problemas são verificados em redes *ad hoc* sem fio, que fazem uso de protocolos MAC baseados na escuta do canal, e são capazes de causar colisões e acrescentar atraso às transmissões, respectivamente.

2.2.2.1 Função de Coordenação Distribuída

A função DCF oferece o método de acesso fundamental utilizado para transferência assíncrona de dados baseado no melhor esforço e em serviço de contenção, onde todos os usuários com dados para transmitir têm uma chance igualmente justa para acessar a rede [17]. Para isso, todas as estações da rede devem suportar a função DCF.

A IEEE 802.11 DCF é baseada no protocolo de controle de acesso ao meio com detecção da portadora e prevenção de colisão (CSMA/CA, do inglês *Carrier Sense Multiple Access with Collision Avoidance*). Neste protocolo, a prevenção de colisão é feita através do procedimento de retardo aleatório (*backoff*). Se uma estação com um quadro a transmitir percebe o canal ocupado, por meio da escuta do canal em nível físico, a estação espera, por um período aleatório de retardo, até o canal ficar inativo. No final deste período, a estação pode realizar uma nova tentativa de transmissão, até que esta seja bem sucedida.

Além de realizar a escuta do canal, em nível físico, a função IEEE 802.11 DCF faz uso do mecanismo adicional de detecção virtual de portadora (*virtual carrier sense*), o qual utiliza quadros de controle como o RTS (do inglês *Request-To-Send*) e o CTS (do inglês *Clear-To-Send*) para o estabelecimento da conexão (*handshaking*), e o ACK (do inglês *Acknowledgement*) para confirmações positivas. O uso de quadros de controle RTS e CTS, associado ao uso de quadros de confirmação positiva ACK, é definido pelo procedimento conhecido como negociação de 4 vias (*4-way handshake*).

No processo de detecção virtual de portadora, a prevenção de colisões é feita pela reserva do canal, por meio do cabeçalho dos quadros RTS, CTS e quadros de dados, onde as estações enviam a informação da duração do dado a ser transmitido, indicando a quantidade de tempo que o canal será utilizado para completar a transmissão bem sucedida. Com essa informação, todas as estações do IBSS são capazes de ajustar o seu vetor de alocação do canal (NAV, do inglês *Network Allocator Vector*), que indica a quantidade de tempo que deve passar até a atual transmissão se completar e o canal estar inativo novamente.

Como ilustrado na Figura 2.2, o quadro RTS é primeiro transmitido pela estação fonte especificando a estação destino. Todas as estações do IBSS que recebem o RTS, lêem o campo de duração e ajustam seus NAVs de acordo. A estação destino responde o RTS com um CTS após um período de SIFS (do inglês *Short Interframe Space*) inativo. As estações que recebem o CTS, lêem

o campo de duração e novamente atualizam seus NAVs. Com a recepção bem sucedida do CTS, a estação fonte está virtualmente certificada que o canal está estável e reservado para a transmissão do quadro de dados.

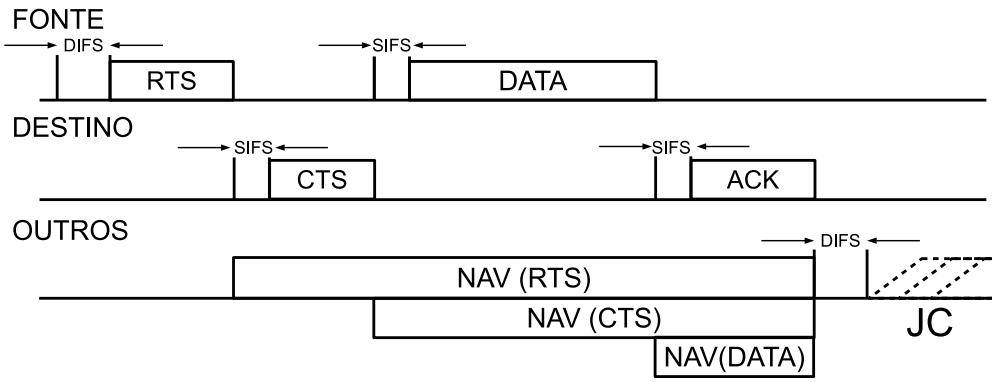


Figura 2.2: Transmissão usando RTS/CTS. Os nós vizinhos (OUTROS) atualizam o NAV a cada vez que recebem um quadro da transmissão. Ao terminar o período do NAV, os nós esperam um período de DIFS (do inglês *Distributed Interframe Space*) e iniciam a disputa pelo canal durante a janela de contenção (JC).

Pacotes de dados de camadas superiores com tamanho excedente ao tamanho do quadro MAC podem ser fragmentados para aumentar a confiança na transmissão. O canal não é liberado até que todo o quadro seja totalmente transmitido ou a estação fonte falhe em receber uma confirmação positiva (ACK) de um fragmento transmitido. A confirmação positiva é enviada da estação destino de volta para a fonte, para cada fragmento recebido com sucesso.

2.2.2.2 Problemas do Terminal Escondido e do Terminal Exposto

Em redes *ad hoc* sem fio que fazem uso de protocolos MAC baseados na escuta do canal, como as redes habilitadas com o padrão IEEE 802.11, as características do meio sem fio geram fenômenos complexos, como os problemas de terminal escondido e terminal exposto.

O problema do terminal escondido é identificado quando uma ou mais estações com dados para transmitir não são capazes de detectar uma transmissão em andamento, uma vez que se encontram fora de seus respectivos raios de detecção, resultando em transmissões simultâneas, susceptíveis à colisão. A Figura 2.3 ilustra um exemplo básico deste problema, quando as estações fazem uso apenas do mecanismo de escuta do canal em nível físico. Suponha que a estação B seja capaz de se comunicar com as estações A e C, e que A e C encontram-se fora da área de escuta um do outro. Suponha ainda que a estação C seja capaz de se comunicar com as estações B e D, e que B e D encontram-se fora da área de escuta um do outro. Se A transmitir para B, C não será capaz de detectar a transmissão em andamento usando o mecanismo de escuta do canal. Neste contexto, uma possível transmissão de C para D irá levar à uma colisão em B.

Com o intuito de amenizar o problema do terminal escondido, foi adicionado ao padrão IEEE 802.11 o mecanismo de detecção virtual de portadora, baseado no uso de quadros de controle RTS e CTS, a fim de realizar a reserva do canal, como já foi mencionado. Apesar de amenizar o problema,

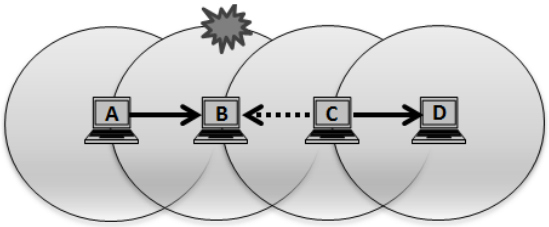


Figura 2.3: Exemplo do problema de terminal escondido

o terminal escondido continuou existindo. Um exemplo é ilustrado na Figura 2.4. Suponha agora que a estação A deseja transmitir para a estação B, e para isso A envia um quadro RTS para B. Todas as estações dentro do raio de transmissão da estação A irão ler o campo de duração deste quadro e ajustar seus NAVs. Como a estação C está fora do alcance de A, e não é capaz de escutar a transmissão do RTS de A, a estação C poderá transmitir seu RTS destinado a D. A estação B está dentro do raio de A e C, portanto o RTS de A sofrerá colisão (interferência) com o RTS de C. Garcia-Luna-Aceves e Fullmer propõem uma solução à colisão por terminais escondidos com a proposta de um protocolo de múltiplo acesso denominado FAMA (do inglês *Floor Acquisition Multiple Access*) [18].

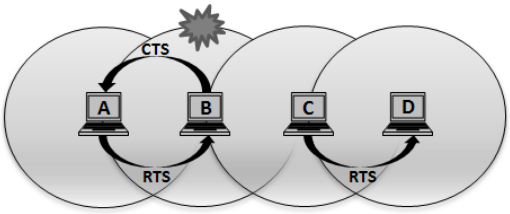


Figura 2.4: Exemplo do problema de terminal escondido, dado o uso do mecanismo de detecção virtual de portadora

Enquanto o terminal escondido é capaz de causar colisões, o problema do terminal exposto é responsável por acrescentar atraso às transmissões e reduzir o tráfego agregado da rede. O problema do terminal exposto resulta de situações em que uma estação, com dados para transmitir, retarda a sua transmissão devido às atividades de transmissão irrelevantes, entre outras estações, dentro do seu raio de transmissão. Um exemplo deste problema é apresentado na Figura 2.5. Fazendo uso apenas do mecanismo de escuta do canal em nível físico, suponha que C deseja transmitir para D. Se B desejar transmitir, como ele está dentro da área de escuta de C, irá escutar que o canal está ocupado e adiar a sua transmissão até que o meio se torne ocioso novamente.

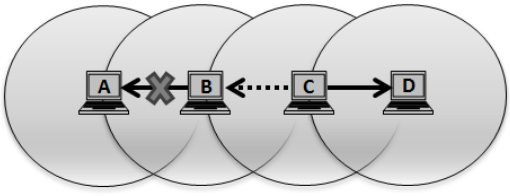


Figura 2.5: Exemplo do problema de terminal exposto

O exemplo mais simples de ocorrência do problema do terminal exposto, dado o uso de quadros

RTS e CTS, é apresentado na Figura 2.6. Suponha um cenário distinto em que as estações A, B e C sejam capazes de se comunicar umas com as outras. Porém, a estação D encontra-se fora das áreas de transmissão de A e B, e dentro da área de transmissão de C. Suponha agora que a estação B deseja transmitir para a estação A, e para isso é feita a reserva do canal usando os quadros de controle RTS e CTS. A estação D não será capaz de escutar essa transmissão. Se D iniciar a transmissão de um quadro RTS para C, não haverá colisão, uma vez que A e B encontram-se fora da área de transmissão de D. Porém, como a estação C é capaz de escutar a transmissão de B para A, ela só será capaz de enviar uma resposta CTS quando a transmissão entre B e A for finalizada, e o canal estiver novamente livre. Neste contexto, a estação C estará exposta a transmissão entre B e A.

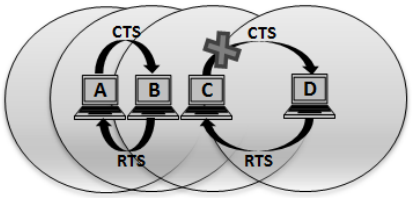


Figura 2.6: Exemplo do problema de terminal exposto, dado o uso do mecanismo de detecção virtual de portadora

No estudo realizado por Jayasuriya *et al.* [19], sobre os problemas do terminal escondido e exposto em redes *ad hoc* habilitadas com o padrão IEEE 802.11, em regime estacionário, foi possível verificar por meio de simulações que o uso de quadros de controle pode causar uma redução significativa na vazão quando são consideradas redes bastante densas, devido principalmente ao aumento do problema do terminal escondido.

2.2.3 Camada Física

O IEEE 802.11 especifica três tipos de implementações na camada física: espalhamento espectral por salto em frequência (FHSS), espalhamento espectral por sequência direta (DSSS) e infra-vermelho (IR, do inglês *Infra-Red*), além das extensões DSSS de alta taxa (HR-DSSS, do inglês *High Rate*-DSSS), multiplexagem por divisão de frequências ortogonais (OFDM, do inglês *Orthogonal Frequency Division Multiplexing*), PHY de taxa extendida (ERP, do inglês *Extended Rate PHY*) e PHY de alta vazão (HT PHY, do inglês *High Throughput PHY*). Os sistemas baseados em DSSS são os mais populares, devido principalmente ao maior aproveitamento da largura de banda disponível. Na especificação mais básica do padrão IEEE 802.11, esse tipo de sistema usa modulação de banda base DBPSK e QDPSK para prover taxas de dados de 1 Mbps e 2 Mbps, respectivamente, e deve operar na faixa de frequências de 2,4 GHz a 2,4835 GHz em um dos 14 canais de 22 MHz de largura de banda. Na especificação do HT PHY, o sistema pode prover taxas de dados até 600 Mbps [15].

Dentre outras atribuições, a camada física é responsável por prover o sinal de avaliação de canal livre (CCA, do inglês *Clear Channel Assessment*), necessário aos mecanismos de controle de acesso ao meio para a realização de detecção de portadora, como é o caso do protocolo MAC CSMA/CA.

Para melhor entendimento da implementação da camada física adotada neste trabalho, na Seção 2.2.3.1 apresentamos uma breve descrição do método DSSS. O método de avaliação de canal livre CCA, bem como a sua importante relação com o desempenho da rede, é discutido na Seção 2.2.3.2.

2.2.3.1 Espalhamento Espectral por Sequência Direta

O espalhamento espectral por sequência direta (DSSS) é um dos métodos de espalhamento espectral mais populares. Para prover uma boa robustez diante de interferência e insensibilidade à propagação múltiplo percurso, o DSSS faz uso de espalhamento do sinal, com a utilização de um código pseudo-aleatório (PN, do inglês *Pseudo-Noise*) multiplicado ao sinal. No IEEE 802.11 DSSS PHY, um mesmo código PN é utilizado por todos os usuários da rede.

No sistema de modulação DSSS, o sinal de dados $s(t)$, com duração T_s, é multiplicado pelo sinal do código pseudo-aleatório $s_c(t)$, que consiste de pulsos menores, com duração T_c e amplitudes iguais a 1 ou -1, resultando no sinal transmitido $s(t) \cdot s_c(t)$, com resposta de frequência igual à convolução desses dois sinais $S(f) * S_c(f)$, sendo $S(f)$ e $S_c(f)$ a Transformada de Fourier de $s(t)$ e $s_c(t)$, respectivamente. Assim, se o sinal original precisa de uma largura de banda B, o sinal resultante vai precisar de uma largura de banda de tamanho $B+B_c$ depois do espalhamento, sendo B_c a largura de banda do código pseudo-aleatório.

2.2.3.2 Método de Avaliação de Canal Livre

Os protocolos da camada MAC são responsáveis por definir quando e como uma tentativa de transmissão pode ocorrer. No caso específico do protocolo CSMA/CA adotado no IEEE 802.11 DCF, a escuta de canal é realizada a cada tentativa de transmissão através do método de avaliação de canal livre (CCA). Isto permite verificar se o canal está livre e, assim, prosseguir com a tentativa de transmissão.

O CCA é feito em nível físico, e parte do fato de que qualquer informação sendo transmitida transporta consigo uma intensidade de sinal grande o suficiente para exceder um determinado limiar. Se um sinal com intensidade acima do limiar for detectado no canal, é assumido que este está ocupado, e a estação com dados para transmitir deverá adiar a sua tentativa de transmissão. A estação só poderá prosseguir com a tentativa de acesso ao canal se a intensidade do sinal de rádio detectada for inferior ao limiar especificado [20].

Para equilibrar as vantagens e desvantagens deste mecanismo, a escolha do limiar de detecção deve ser cautelosa, uma vez que esse parâmetro, além de influenciar diretamente na distância de escuta (a qual também irá depender das características do canal, sendo variante no tempo), também é responsável por permitir a existência de transmissões concorrentes, ou seja, o reuso espacial. Um grande raio de detecção de portadora, com estratégias apropriadas de avaliação de canal livre, pode tratar eficientemente os problemas de terminal escondido e bloqueio no receptor. Entretanto, um grande raio de detecção de portadora pode agravar o problema do terminal exposto e reduzir o reuso espacial [21].

2.3 Canal de Propagação Sem Fio

A propagação de um sinal eletromagnético em um canal de rádio sem fio não é susceptível apenas ao ruído e à interferência, mas também à perda de caminho e à presença de obstáculos que podem causar variações no sinal recebido, fenômeno conhecido como sombreamento (*shadowing*) ou desvanecimento em larga escala. A perda de caminho é causada pela atenuação do sinal transmitido ao longo do caminho entre o transmissor e o receptor devido à distância entre eles e às propriedades eletromagnéticas do meio de transmissão. O sombreamento é causado pela presença de obstáculos entre o transmissor e o receptor que podem atenuar a potência do sinal por absorção, reflexão, espalhamento e difração [14].

Para prever a intensidade média do sinal recebido a uma determinada distância do transmissor, são utilizados modelos de propagação em larga escala. O modelo mais simples é o de propagação no espaço livre, conhecido como modelo Friis. Neste, o caminho de propagação do sinal entre duas estações não apresenta qualquer tipo de obstáculo, tendo associado a ele um caminho de linha-de-visada (LOS, do inglês *Line-of-Sight*). Entretanto, em ambientes reais, raramente um sinal será transmitido por um único caminho. Neste contexto, o modelo de dois raios (também conhecido como modelo *Two-Ray*) se apresenta mais usual, considerando um caminho LOS e um caminho refletido no solo (NLOS, do inglês *Non Line-of-Sight*), entre o transmissor e o receptor.

Uma vez que o sinal recebido será a soma dos sinais dos diferentes caminhos, a intensidade instantânea do sinal recebido sofrerá flutuações rápidas para deslocamentos curtos de distâncias, na ordem do comprimento de onda, devido à combinação das múltiplas versões do sinal transmitido que ocorre de maneira tanto construtiva quanto destrutiva, caracterizando o desvanecimento em pequena escala. Este efeito é crucial para o desempenho do receptor (em comparação à propagação em larga escala) já que varia dramaticamente em curtas distâncias e curtos períodos de tempo. Para prever essas flutuações, são usados modelos de propagação em pequena escala, como os modelos estatísticos de Rice e Rayleigh. As condições de propagação, determinadas pelos modelos de propagação, terão grande influência sobre a capacidade do canal MIMO. Entretanto, este canal também pode ser modelado a partir de sua descrição estatística.

Para melhor entendimento do canal de propagação sem fio, na Seção 2.3.1 apresentamos uma breve definição dos efeitos de propagação inerentes a ele. As descrições dos modelos Friss e de dois raios, de propagação em larga-escala, e dos modelos de desvanecimento Rice e Rayleigh, são apresentadas nas Seções 2.3.2, 2.3.3 e 2.3.4, respectivamente. Por fim, descrevemos o modelo de canal MIMO, utilizado para o desenvolvimento deste trabalho, na Seção 2.3.5.

2.3.1 Efeitos de Propagação

Os efeitos básicos de propagação são: atenuação, reflexão, espalhamento e difração.

A atenuação é responsável pela perda gradativa da potência do sinal transmitido, durante o seu caminho até o receptor. Considerando a propagação no espaço livre, o sinal transmitido apresenta uma atenuação contínua. Entretanto, em ambientes reais, a perda por atenuação pode ser acentuada pela chuva, gases atmosféricos e até mesmo pela absorção de parte da potência

do sinal por obstáculos no caminho da transmissão. Essa perda será definida pelas propriedades eletromagnéticas do meio ou do material incidente.

A reflexão ocorre quando a onda eletromagnética propagada incide sobre um obstáculo, causando a mudança completa na direção da onda incidente, ou parcial (refração). O obstáculo pode ser uma parede, um objeto ou até mesmo o solo. O coeficiente de reflexão, ou refração, depende das propriedades eletromagnéticas do obstáculo, da polarização da onda, do ângulo de incidência e da frequência de propagação da onda.

O espalhamento funciona como uma refleção não direcional. Quando a onda eletromagnética propagada incide sobre um obstáculo, são geradas cópias do sinal transmitido, conhecidas como componentes de múltiplos percursos, que podem ser atenuadas em potência, atrasadas no tempo ou apresentar deslocamento em fase e/ou frequência, em relação ao componente LOS do sinal no receptor [14].

A difração ocorre quando o caminho entre o transmissor e o receptor é obstruído por obstáculos pontudos. A incidência da onda eletromagnética propagada sobre o obstáculo dará origem à novas ondas, fazendo surgir uma curvatura em torno deste, mesmo quando não há um caminho LOS entre o transmissor e o receptor. Em altas frequências, assim como a reflexão, a difração depende das propriedades eletromagnéticas do meio, da polarização da onda, do ângulo de incidência e da frequência de propagação da onda.

2.3.2 Modelo Friis de Propagação em Larga-Escala

O modelo de propagação no espaço livre é usado para prever a intensidade do sinal recebido quando transmissor e receptor possuem um caminho de linha-de-visada limpo, desobstruído, entre eles [22]. Este modelo, assim como a maioria dos modelos de propagação em larga escala, prevê que a potência do sinal recebido P_r é inversamente proporcional ao quadrado da distância d entre o transmissor e o receptor. Essa potência é dada pela equação de Friis [22]:

$$P_r(d) = \left(\frac{G_t G_r \lambda^2}{(4\pi)^2 d^2 L}\right) P_t, \tag{2.1}$$

sendo P_t a potência de transmissão, G_t e G_r os ganhos das antenas transmissora e receptora, respectivamente, L o fator de perda do sistema não relacionado à propagação ($L \geq 1$), e λ o comprimento de onda em metros. O fator L é definido de acordo com as perdas no *hardware* do sistema, quando não há perdas, tem-se que $L = 1$. O ganho de uma antena é relacionado à sua abertura (ou área) efetiva A_e [22] de forma que

$$G = \left(\frac{4\pi A_e}{\lambda^2}\right). \tag{2.2}$$

A perda de caminho PL causada pela atenuação do sinal transmitido ao longo do caminho entre o transmissor e o receptor, devido à distância entre eles e às propriedades eletromagnéticas do meio de transmissão, é definida pela diferença entre a potência efetiva transmitida e a potência recebida medida em dB. No modelo de propagação no espaço livre, ela pode ser descrita [22] por

$$PL(dB) = 10log\left(\frac{P_t}{P_r}\right) = -10log\left(\frac{G_t G_r \lambda^2}{(4\pi)^2 d^2}\right). \tag{2.3}$$

2.3.3 Modelo de Dois Raios para Propagação em Larga Escala

No modelo de dois raios (também conhecido como modelo Two-Ray) apenas o efeito de múltiplo percurso causado pela reflexão no solo é considerado para prever a intensidade do sinal recebido a uma determinada distância do transmissor. Assim sendo, o sinal recebido irá consistir do componente LOS, propagado no espaço livre, e de um componente de múltiplo percurso, obtido da reflexão do sinal no solo, como ilustra a Figura 2.7.

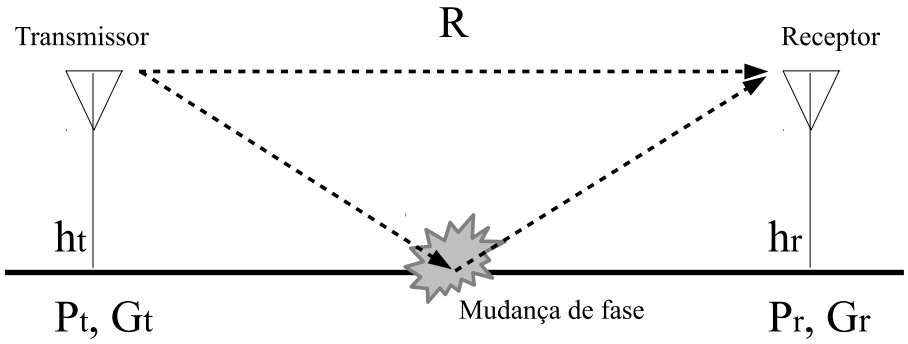


Figura 2.7: Modelo de propagação em larga escala de Dois Raios.

A potência recebida, a uma distância d do transmissor pode ser aproximada [22] por

$$P_r(d) = \left(\frac{{h_t}^2 {h_r}^2}{d^4}\right) P_t G_t G_r, \tag{2.4}$$

sendo P_t a potência transmitida, G_t e G_r os ganhos das antenas transmissora e receptora, respectivamente, e h_t e h_r a altura das antenas transmissora e receptora, respectivamente. Neste modelo, o atraso de espalhamento equivale ao atraso entre o componente LOS e o componente refletido. A perda de caminho PL, para o modelo Two-Ray, pode ser expressa em dB [22] por

$$PL(dB) = 40log(R) - [10log(G_t) + 10log(G_r) + 20log(h_t) + 20log(h_r)]. \tag{2.5}$$

Esse modelo é considerado razoavelmente preciso para prever a intensidade do sinal em larga escala para distâncias de vários quilômetros em sistemas de rádio móvel que utilizam torres altas (alturas que ultrapassam os 50 m), além de canais de microcélula de linha de visada em ambientes urbanos [22].

2.3.4 Modelo de Desvanecimento Rice e Rayleigh

Para prever a intensidade do sinal em um canal sob o efeito da propagação em pequena escala, são usados os modelos estatísticos de Rice e Rayleigh. O canal de desvanecimento de Rice pode ser descrito por dois parâmetros: K, a razão entre as potências LOS e NLOS, e Ω, a potência do sinal no caminho de visada. O modelo de desvanecimento Rayleigh é um caso particular do modelo Rice,

quando $K = 0$ e não há contribuição da potência LOS. A intensidade do sinal recebido pelo canal desvanecido é uma variável aleatória com distribuição Rice, com a seguinte função de densidade de probabilidade [23]:

$$f(x) = \frac{2(K+1)x}{\Omega} \exp\left(-K - \frac{(K+1)x^2}{\Omega}\right) \times I_0\left(2\sqrt{\frac{K(K+1)}{\Omega}}x\right), \tag{2.6}$$

sendo $x \in X \sim Rice(K, \Omega)$, I_0 a função de Bessel modificada do primeiro tipo e de ordem zero.

2.3.5 Modelo de Canal MIMO

Nesta seção, consideramos sistemas com múltiplas antenas no transmissor e no receptor, o qual é mais conhecido como sistema de múltiplas entradas e saídas (MIMO). Em sistemas MIMO, as antenas transmissoras e receptoras podem ser usadas para obter ganhos de diversidade. As múltiplas antenas podem ser usadas para aumentar a taxa de dados através de multiplexagem ou para melhorar o desempenho através de diversidade. A multiplexagem é obtida explorando a estrutura da matriz de ganhos de canal para obter caminhos independentes que podem ser usados para enviar dados independentes. O uso de múltiplas antenas em sistemas sem fio melhora sua eficiência espectral, e tal ganho requer conhecimentos precisos sobre o canal no receptor, e às vezes, no transmissor também [14]. A seguir, apresentamos como é modelado o canal MIMO para M antenas transmissoras e N antenas receptoras.

As características do ambiente sem fio apresentadas nas Seções anteriores são importantes para o entendimento do modelo de canal MIMO, assim como o entendimento das técnicas de MIMO. Os efeitos de desvanecimento em pequena escala com MIMO são dados pelas seguintes observações, considerando a Figura 2.8:

- A resposta do canal para qualquer antena transmissora i e para qualquer antena receptora j é a soma de vários sinais de caminhos refletidos. Tais caminhos podem ter diferentes atrasos de propagação, podendo a resposta ter um grande espalhamento no tempo.
- O sinal recebido em cada antena receptora é a soma dos sinais de todas as antenas transmissoras (com a aplicação do canal).

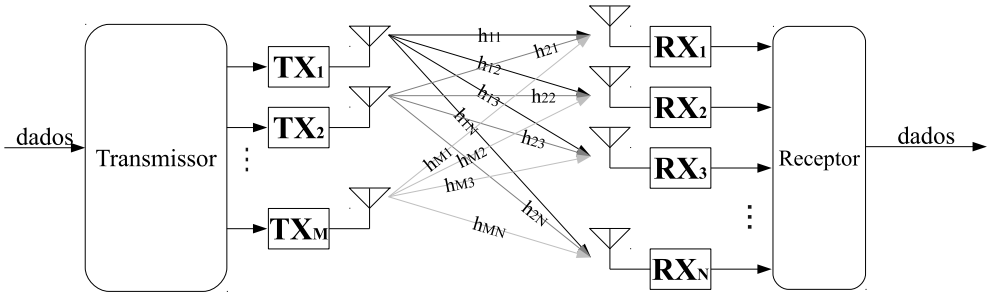


Figura 2.8: Canal de múltiplas antenas.

Sendo assim, pode-se descrever o sinal recebido após atravessar o canal MIMO como

$$r(t) = \sum_{k=-\infty}^{\infty} \mathbf{H}(t - kT)\mathbf{a}(k) + \mathbf{n}(t), \tag{2.7}$$

sendo T, o intervalo de tempo entre a transmissão de cada símbolo $a_i(k)$ do vetor $\mathbf{a}(k)$, sendo $\mathbf{a}(k)$, a sequência transmitida (cada elemento através de uma antena transmissora), $\mathbf{n}(t)$, um vetor de amostras de ruído Gaussiano aditivo branco (AWGN, do inglês *Additive White Gaussian Noise*), e $\mathbf{H}(\tau)$, matriz de canal MIMO no domínio do tempo, assumindo M antenas transmissoras e N antenas receptoras, tal que

$$\mathbf{a}(k) = \begin{bmatrix} a_1(k) \\ \vdots \\ a_M(k) \end{bmatrix}, \tag{2.8}$$

$$\mathbf{n}(t) = \begin{bmatrix} n_1(t) \\ \vdots \\ n_N(t) \end{bmatrix}, \tag{2.9}$$

$$\mathbf{H}(\tau) = \begin{bmatrix} h_{11}(\tau) & h_{12}(\tau) & \dots & h_{1N}(\tau) \\ h_{21}(\tau) & h_{22} & \dots & h_{2L}(\tau) \\ \vdots & \vdots & \ddots & \vdots \\ h_{M1}(\tau) & h_{M2(\tau)} & \dots & h_{MN}(\tau) \end{bmatrix}. \tag{2.10}$$

Na matriz $\mathbf{H}(\tau)$, cada elemento $h_{ij}(\tau)$ é a resposta impulsional do canal da antena transmissora i para a antena receptora j que pode variar no tempo devido à movimentação dos objetos ao redor e das próprias estações. Como a recepção de quadros requer um canal invariante no tempo, os quadros são projetados para ter durações curtas, onde o canal é aproximadamente constante no tempo [5].

2.4 Técnicas de Transmissão MIMO

O uso de múltiplas antenas para transmissão e recepção tem se apresentado como uma tecnologia chave para alcançar melhor desempenho e maiores taxas de transmissão de dados em enlaces de comunicação sem fio. Para tanto, diversas técnicas têm sido propostas. Destas técnicas, duas grandes classes de técnicas de transmissão MIMO se destacam: a codificação em bloco espaço-temporal (STBC, do inglês *Space-Time Block Coding*) e a multiplexagem por divisão espacial (SDM, do inglês *Spatial-Division Multiplexing*).

A codificação STBC permite ativar *diversidade* espacial usando múltiplas antenas, visando melhorar a qualidade do sinal no receptor em detrimento aos efeitos do canal com desvanecimento. Na transmissão por diversidade espacial, são usadas múltiplas antenas transmissoras, entre as quais a potência transmitida é dividida, para o envio de réplicas do sinal. O esquema mais comum desse tipo de codificação foi proposto por Alamouti [10], e se caracteriza por não requerer que o canal seja conhecido no transmissor. O esquema original de Alamouti é baseado no uso de duas antenas

transmissoras e uma antena receptora (sistema MISO, do inglês *Multiple-Input, Single-Output*), que suporta detecção máxima de verossimilhança baseada apenas em um processamento linear no receptor. Esse esquema pode facilmente ser generalizado para o caso de duas antenas transmissoras e N antenas receptoras para prover diversidade em ordem de $2N$ [12].

Enquanto o uso da codificação STBC permite uma transmissão mais robusta, técnicas SDM são usadas para ativar maiores taxas de transmissão de dados em enlaces de comunicação sem fio. Nessas técnicas, múltiplas antenas transmissoras, apropriadamente espaçadas, são usadas para transmitir segmentos de um quadro de dados de maneira independente, os quais são individualmente recuperados no receptor. A *multiplexagem* é obtida explorando a estrutura da matriz de ganho do canal para obter percursos de sinais independentes que podem ser usados para enviar dados independentes [14]. Uma forma eficiente de se realizar tal exploração é proposta pela arquitetura de processamento de sinais *Vertical Bell Labs Layered Space-Time* (V-BLAST) [4].

A seguir, na Seção 2.4.1 apresentamos os fundamentos básicos do esquema de Alamouti. E, na Seção 2.4.2 apresentamos os fundamentos básicos do sistema V-BLAST. Para análise de desempenho do algoritmo V-BLAST, nosso trabalho se apoiou na abordagem analítica geométrica proposta por Loyka e Gagnon [2], que é descrita na Seção 2.4.3.

2.4.1 Esquema de Alamouti

O esquema de Alamouti [10] visa melhorar a qualidade do sinal no receptor de um lado do enlace através de um simples processamento entre duas antenas de transmissão do lado oposto, sem requerer que o canal seja conhecido no transmissor e com pouca complexidade computacional [10]. O esquema opera em dois períodos de símbolo e, para simplificar, apresentamos o caso para duas antenas no transmissor e duas no receptor (antenas 1 e 2). No primeiro período de símbolo, dois símbolos diferentes s_1 e s_2 são transmitidos simultaneamente pelas antenas 1 e 2, respectivamente. Durante o segundo período de símbolo, o símbolo $-s_2^*$ é transmitido pela antena 1, e o símbolo s_1^* é transmitido pela antena 2 (onde o sobrescrito * indica a operação conjugado complexo). Em outras palavras, em dois períodos de símbolo consecutivos, o transmissor envia os vetores de símbolo $\mathbf{x}_1$ e $\mathbf{x}_2$

$$\mathbf{x}_1 = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}, \qquad \mathbf{x}_2 = \begin{bmatrix} -s_2^* \\ s_1^* \end{bmatrix}, \tag{2.11}$$

A cada período de símbolo, a energia E_s disponível no tranmissor é igualmente dividida entre as duas antenas, ou seja, cada símbolo tem energia $E_s/2$ (não é necessária energia extra). Assume-se que o canal se mantém constante em dois períodos de símbolo consecutivos e sofre desvanecimento plano, com $h_{kl} = |h_{kl}|e^{j\theta_{kl}},\ k,l = 1,2$, representando os ganhos complexos do canal entre a l-ésima antena transmissora e a k-ésima antena receptora. O receptor usa os símbolos sequencialmente recebidos (possivelmente contaminados por interferência de múltiplo acesso (MAI, do inglês *Multiple Access Interference*) e por ruído de fundo) para formar um novo vetor de sinal que eficientemente desacopla as duas transmissões de símbolo independentemente do estado do canal [12]. Por exemplo, considerando um esquema 2×2, a matriz de canal pode ser definida por

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix}, \tag{2.12}$$

e os sinais $\mathbf{y}_1$ e $\mathbf{y}_2$ recebidos em ambas as antenas receptoras nos dois períodos consecutivos, respectivamente, são dados por

$$\mathbf{y}_1 = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \end{bmatrix}, \qquad \mathbf{y}_2 = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} -s_2^* \\ s_1^* \end{bmatrix} + \begin{bmatrix} n_3 \\ n_4 \end{bmatrix}, \tag{2.13}$$

sendo n_1, n_2, n_3 e n_4 amostras de ruído AWGN, descorrelacionadas, de média zero e potência (variância) N_0. Usando os sinais recebidos $\mathbf{y}_1$ e $\mathbf{y}_2$, o receptor forma o vetor $\mathbf{y} = [\mathbf{y}_1\, \mathbf{y}_2^*]^T$, dado por

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2^* \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \\ h_{12}^* & -h_{11}^* \\ h_{22}^* & -h_{21}^* \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \\ n_3^* \\ n_4^* \end{bmatrix} = \mathbf{H}_A\, \mathbf{s} + \mathbf{n}, \tag{2.14}$$

sendo $\mathbf{s} = [s_1\ s_2]^T$ e $\mathbf{n} = [n_1\ n_2\ n_3^*\ n_4^*]^T$. Da forma como foi definida, a matriz $\mathbf{H}_A$ é uma matriz ortogonal independentemente do estado do canal, ou seja, $\mathbf{H}_A^H \mathbf{H}_A = \|\mathbf{H}\|_F^2 \mathbf{I}_2$, sendo $\mathbf{I}_2$ a matriz identidade 2×2 e $\|\mathbf{H}\|_F^2$ a *norma de Frobenius* de $\mathbf{H}$, definida como

$$\|\mathbf{H}\|_F^2 = \sum_k \sum_l |h_{kl}|^2, \tag{2.15}$$

sendo, como já mencionado, h_{kl} o ganho complexo do canal entre a l-ésima antena transmissora e a k-ésima antena receptora. Definindo o novo vetor $\mathbf{z} = \mathbf{H}_A^H \mathbf{y}$, obtemos

$$\mathbf{z} = \|\mathbf{H}\|_F^2 \mathbf{I}_2\, \mathbf{s} + \tilde{\mathbf{n}}, \tag{2.16}$$

sendo $\tilde{\mathbf{n}} = \mathbf{H}_A^H \mathbf{n}$ um vetor de ruído Gaussiano complexo com $E\{\tilde{\mathbf{n}}\} = \mathbf{0}$ e matriz de covariância $E\{\tilde{\mathbf{n}}\tilde{\mathbf{n}}^H\} = \|\mathbf{H}\|_F^2 N_0 \mathbf{I}_2$. A natureza diagonal de $\mathbf{z}$ desacopla efetivamente os dois símbolos transmitidos, e cada componente de $\mathbf{z}$ corresponde a um dos símbolos transmitidos, tal que

$$z_i = \|\mathbf{H}\|_F^2\, s_i + \tilde{n}_i, \quad i = 1, 2. \tag{2.17}$$

Assim, a razão sinal-ruído (SNR, do inglês *Signal-to-Noise Ratio*) recebida corresponde a SNR de z_i, que é dada por [14]

$$\text{SNR} = \frac{\|\mathbf{H}\|_F^2 E_s}{2N_0}, \tag{2.18}$$

sendo, como já mencionado, E_s a energia disponível no tranmissor e $\|\mathbf{H}\|_F^2$ a *norma de Frobenius* da matriz de canal $\mathbf{H}$ entre as 2 antenas transmissoras e as 2 antenas receptoras..

Utilizando o esquema de Alamouti e generalizando para um número arbitrário de antenas receptoras, a Eq. (2.18) pode ser modificada para considerar os efeitos de interferência de múltiplo

acesso (MAI) dos nós que podem transmitir simultaneamente no canal. Utilizando os resultados obtidos anteriormente [12], pode-se mostrar que a relação sinal-ruído e interferência (SINR) é dada por

$$\text{SINR} = \frac{\|\mathbf{H}_i\|_F^2 E_s/2}{N_0 + \sum_j \|\mathbf{H}_j\|_F^2 E_s/2}, \tag{2.19}$$

sendo o índice i a identificação do nó remetente do sinal desejado, e o índice j a identificação de todo nó que transmite simultaneamente e interfere com a transmissão individual do nó i. Presume-se que todos os nós da rede estão equipados com a mesma quantidade de antenas transmissoras e receptoras.

2.4.2 Arquitetura V-BLAST

O sistema *Vertical Bell Labs Layered Space-Time* (V-BLAST) [4] foi proposto para alcançar alta eficiência espectral pelo uso de múltiplas antenas. No sistema original, fluxos paralelos de dados são transmitidos simultaneamente através do uso de múltiplas antenas na mesma banda de frequência. Esses segmentos são decodificados no receptor utilizando um detector com cancelamento de interferência por forçação a zero (ZF-SIC, do inglês *Zero-Forcing Successive Interference Cancellation*), que permite alcançar alta eficiência espectral com razoável complexidade de decodificação.

A Figura 2.9 retrata uma generalização do sistema V-BLAST, considerando um canal de comunicação ponto-a-ponto sob desvanecimento Rayleigh, com M antenas transmissoras e N antenas receptoras. Assume-se um canal quase estático sob desvanecimento plano, isto é, o desvanecimento no canal é aproximadamente constante ao longo de um símbolo, e varia apenas de um símbolo para o outro. Denotamos por $\boldsymbol{H}^{N\times M}$ a matriz do canal e assumimos que ela é estimada corretamente no receptor após detecção coerente do sinal recebido. O coeficiente de desvanecimento h_{ij} é o ganho de caminho complexo da antena transmissora j para a antena receptora i. Também assumimos que $N \geq M$. A potência total transmitida é assumida como sendo P_t, independente de M, e igualmente dividida entre cada antena transmissora.

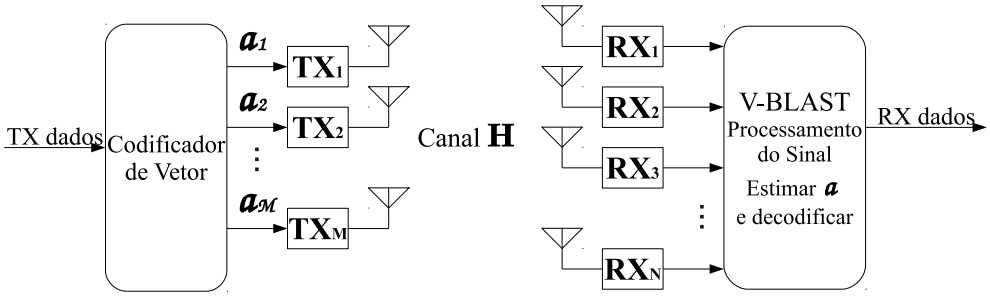


Figura 2.9: Arquitetura V-BLAST. Diagrama do sistema em alto nível com M antenas transmissoras e N antenas receptoras [1].

O sinal recebido pode ser representado por um vetor $\boldsymbol{r}$ de dimensões $N \times 1$ e pode ser modelado em banda base como

$$\boldsymbol{r} = \sqrt{\frac{P_t}{M}}\boldsymbol{H}\cdot\boldsymbol{a} + \boldsymbol{n}, \tag{2.20}$$

sendo $\boldsymbol{a} = \{a_1, a_2, ..., a_M\}^T$ um vetor $M \times 1$, no qual o j-ésimo componente representa o sinal transmitido da j-ésima antena transmissora, e $\boldsymbol{n}$ o vetor $N \times 1$ do ruído AWGN com média nula e variância σ^2. Assumimos sincronismo perfeito no receptor.

A seguinte notação é usada: letras maíusculas em negrito denotam matrizes, letras minúsculas em negritos denotam vetores de colunas, e letras minúsculas normais denotam escalares. Os sobrescritos $(\cdot)^T$ e $(\cdot)^\dagger$ denotam matrizes transpostas e pseudoinversas de Moore-Penrose, respectivamente, e $||\cdot||$ denota a norma de Frobenius.

Para realizar a detecção, o receptor explora o conceito de anulação combinatória linear: cada segmento de dados a ser decodificado é considerado o sinal desejado, e os demais segmentos são considerados interferência. A anulação é efetuada pela ponderação linear dos sinais recebidos de forma a satisfazer algum critério de desempenho[1]. De acordo com a abordagem de forçação a zero, o vetor de pesos $\boldsymbol{w_i}$, $i = 1, 2, ..., M$, deve ser escolhido de forma que

$$\boldsymbol{w_i^T (H)_j} = \delta_{ij}, \tag{2.21}$$

sendo $\boldsymbol{(H)_j}$ a j-ésima coluna de $\boldsymbol{H}$, e δ a função delta de Kronecker. O vetor de pesos $\boldsymbol{w_i}$ deve ser a i-ésima coluna de $\boldsymbol{H^\dagger}$ de modo a satisfazer a Eq. (2.21). Assim, a decisão estatística do i-ésimo segmento é dada por

$$y_i = \boldsymbol{w_i^T r_i}. \tag{2.22}$$

Os valores estatísticos resultantes dados pela Eq. (2.22) podem ser usados para se obter uma estimativa $\hat{a}_i$ do símbolo desejado a_i por

$$\hat{a}_i = Q(y_i), \tag{2.23}$$

sendo $Q(\cdot)$ a operação de quantização apropriada para a constelação em uso. Antes de realizar a detecção do $(i+1)$-ésimo símbolo, é importante observar que um desempenho superior pode ser obtido pela combinação de técnicas não lineares com o algoritmo ZF. Uma técnica não linear particularmente atraente envolve a exploração do tempo de sincronismo inerente ao modelo do sistema, sendo acompanhada pelo uso de cancelamento de símbolo. Com o uso de cancelamento de símbolo, a interferência dos componentes de $\boldsymbol{a}$ já detectados é subtraída do vetor do sinal recebido, resultando em um vetor recebido modificado que apresenta menor interferência. Quando cancelamento de símbolo é usado, a ordem em que os componentes de $\boldsymbol{a}$ são detectados se torna importante para o desempenho total do sistema. Considere o conjunto ordenado $\boldsymbol{S} = \{k_1, k_2, ..., k_M\}$ como sendo uma permutação dos inteiros $\{i : 1, 2, ..., M\}$, especificando a ordem na qual os componentes do vetor de símbolo transmitido $\boldsymbol{a}$ é extraído. Para este caso particular, é possível mostrar que o desempenho ótimo pode ser obtido se esta ordem seguir a expressão dada por [1]

$$k_{i+1} = \arg \min_{j \notin \{k_1, ..., k_i\}} ||(\boldsymbol{H^\dagger_{k_i}})_{\boldsymbol{j}}||^2, \tag{2.24}$$

sendo k_{i+1} o próximo segmento a ser detectado, $\boldsymbol{H^\dagger_{k_i}}$ a matriz pseudo-inversa do canal obtida no i-ésimo passo do processo de detecção, e j cada uma das linhas contidas em $\boldsymbol{H^\dagger_{k_i}}$. Agora, assumindo

[1] Apesar de usarmos o algoritmo *zero-forcing* (ZF) em nosso trabalho, outros critérios como erro quadrático médio mínimo (MMSE, do inglês *Minimum Mean-Squared Error*) ou máxima verossimilhança (ML, do inglês *Maximum Likelihood*) também podem ser empregados.

que $\hat{a}_i = a_i$, é possível cancelar o vetor recebido $\boldsymbol{r_i}$ do sinal original recebido fazendo

$$\boldsymbol{r_{i+1}} = \boldsymbol{r_i} - (\boldsymbol{H})_{k_i}\hat{a}_{k_i}, \tag{2.25}$$

sendo $(\boldsymbol{H})_{k_i}$ a k_i-ésima coluna de $\boldsymbol{H}$. Após proceder o cancelamento de símbolo, a coluna $\boldsymbol{(H)}_{k_i}$ deverá ser zerada. Este processo de anulação (ZF) e o cancelamento sucessivo de interferência (SIC, do inglês *Successive Interference Cancellation*) deve ser realizado até que $\boldsymbol{a}$ tenha sido completamente detectado. O valor da SNR do k_i-ésimo componente detectado de $\boldsymbol{a}$ pode ser obtido substituindo as Eqs. (2.20) e (2.21) em (2.22). Logo,

$$(SNR)_{k_i} = \frac{|a_{k_i}|^2}{\sigma^2||\boldsymbol{w_{k_i}}||^{\boldsymbol{2}}}. \tag{2.26}$$

Apesar de parecer simples, o uso da Eq. (2.26) não é uma solução adequada para implementação do sistema V-BLAST em um simulador de rede a eventos discretos. A razão para isto deve-se à natureza iterativa do algoritmo envolvido para detecção de todos os segmentos de dados. Além disso, apenas com a Eq. (2.26), ainda se desconhece a relação exata entre a taxa de erro de bit (BER) e a razão sinal ruído (SNR) do sistema. Entretanto, simuladores a eventos discretos tipicamente utilizam expressões analíticas fechadas ou tabelas de consulta de modo que simplificações numéricas tornam-se possíveis. Claramente, uma alternativa para a Eq. (2.26) é necessária. Tal alternativa é apresentada a seguir, na Seção 2.4.3. Trata-se de uma abordagem analítica geométrica para a análise de desempenho do algoritmo V-BLAST proposta por Loyka e Gagnon [2].

2.4.3 Abordagem Analítica do Desempenho do Sistema V-BLAST

Loyka e Gagnon [2] propuseram uma abordagem analítica geométrica para a análise de desempenho do algoritmo V-BLAST, sendo baseada no processo de ortogonalização de Gram-Schmidt. Esta abordagem introduziu uma nova visão geométrica do algoritmo V-BLAST, que explica algumas das suas propriedades de forma completa e rigorosa, incluindo uma análise estatística do pós-processamento do SNR para um sistema $M \times N$. Além disso, expressões analíticas fechadas são derivadas para a BER média de um sistema $M \times N$ BPSK. A análise desenvolvida por eles também levou em consideração os efeitos da ordenação ótima no processo iterativo de detecção, sendo M e N o número de antenas transmissoras e receptoras, respectivamente.

De acordo com os resultados obtidos por Loyka [2], a BER média total $\overline{P_{et}}$ (calculada sobre todas as antenas) pode ser expressa por

$$\overline{P_{et}} = a_t\overline{P_{e1}}, \tag{2.27}$$

sendo

$$a_t = \frac{1}{M}\sum_{i=1}^{M} a_i, \qquad \overline{P_{e1}} \approx \frac{C_{2(N-M)+1}^{N-M+1}}{(4\gamma_0)^{N-M+1}}, \tag{2.28}$$

Vale a pena mencionar que $\overline{P_{et}}$ é uma média calculada com relação à ordem de detecção de fluxos de dados também. A fim de verificar a exatidão da Eq. (2.27), Bonfim *et. al* [13] realizaram um conjunto de simulações Monte Carlo para três diferentes configurações de antenas. Os resultados estão representados na Figura 2.10. Como pode ser observado, o modelo analítico proposto por

Loyka e Gagnon capta muito bem o desempenho V-BLAST. Por esta razão, nós adotamos o seu modelo para implementação do sistema V-BLAST no simulador NS-3.

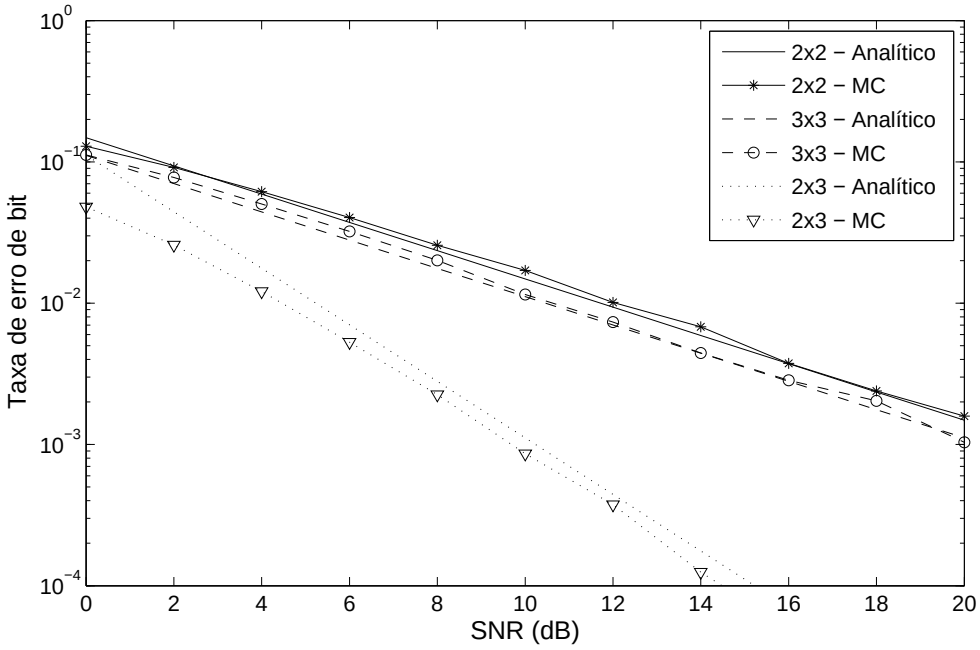


Figura 2.10: Comparação das curvas de taxa de erro de bit do sistema V-BLAST providas por simulações Monte Carlo ("MC") e pela expressão fechada derivada por Loyka e Gagnon [2] ("Analítico").

Capítulo 3

Revisão Bibliográfica

Uma quantidade significativa de trabalhos já foi desenvolvida para o projeto de redes *ad hoc* sem fio habilitadas com sistemas de múltiplas antenas que utilizam conformação de feixe — as chamadas "antenas direcionáveis". O projeto de protocolos MAC que utilizam antenas direcionáveis tem como objetivo principal o aumento da capacidade de transferência de dados na rede a partir de um maior reuso espacial da rede, ou seja, um aumento do número possível de transmissões simultâneas dentro de uma rede [24], [25]. Apesar de todo esforço realizado, o uso de antenas direcionáveis em redes *ad hoc* está sujeito ao problema de "surdez" inerente à direcionalidade da comunicação e à incapacidade de percepção de possíveis tentativas de transmissão de nós vizinhos.

No caso do uso de MIMO com antenas omnidirecionais, o problema de surdez dos nós não ocorre, porém o reuso espacial da rede deve ser tratado de outras maneiras. Para explorar ganhos de diversidade e/ou multiplexagem, Stamoulis e Al-Dhahir [26] investigaram o impacto de códigos de bloco espaço-temporais (STBC, do inglês *Space-Time Block Codes*) em redes sem fio locais operando no modo *ad hoc* e habilitadas com o padrão IEEE 802.11a. Para isto, os autores realizaram simulações no MATLAB® para gerar sequências de transmissão de pacotes para o simulador NS-2, a fim de avaliar os benefícios da codificação STBC no desempenho de protocolos de camadas superiores, como o TCP. Para avaliação de desempenho, eles assumiram cenários de redes onde os nós encontram-se totalmente conectados, ou seja, todos os nós dentro do raio de transmissão de todos os outros nós, e habilitados apenas com o esquema mais simples de Alamouti, com 2 antenas transmissoras e 1 receptora.

Hu e Zhang [6] tentaram modelar MIMO em redes *ad hoc* focando no padrão IEEE 802.11 com o esquema STBC. Em seu trabalho foi desconsiderado o impacto da topologia da rede, ao assumir que os "eventos experimentados por um usuário eram estatisticamente os mesmos dos outros usuários", afirmando que cada nó tem a *mesma* vazão média. Tomando esta premissa, cada nó é rodeado pelo mesmo número médio de nós, e a rede de múltiplos saltos é simplificada, na prática, em várias redes de único salto, onde as interações acontecem apenas com os vizinhos diretos. Por sua vez, Rosseto e Zorzi [8] estudaram o uso de STBC para transmitir pacotes de controle como um meio de prover extensão do raio de transmissão omnidirecional sem precisar de utilizar redes direcionáveis. Neste trabalho, os autores consideraram apenas redes em que todos os nós/dispositivos estão totalmente conectados.

Levorato *et al.* [7] também consideraram redes com todos os dispositivos totalmente conectados (todos os nós estão dentro do alcance de todos) para avaliar o desempenho de redes *ad hoc* habilitadas com o esquema MIMO. No caso deste trabalho, os autores fazem o uso da arquitetura de processamento BLAST. Eles apresentaram um modelo analítico que prediz a propagação do erro de detecção de símbolo durante o processo iterativo de detecção do sistema BLAST. Eles utilizaram este modelo como uma abstração da camada física em uma simulação de rede.

Gelal *et al.* [9] examinaram o ganho de diversidade oriundo do esquema STBC para explorar o compromisso entre o aumento do raio de alcance e o aumento da taxa de transmissão. O modelo da camada física desenvolvido por eles assume a operação da rede em regime de alta SINR. Com isto, eles utilizam um valor *fixo* para representar o ganho de diversidade proporcionado pelo STBC, ou seja, eles não utilizam qualquer modelagem para abstrair o comportamento do uso de STBC nos nós. O valor de ganho de diversidade utilizado é de aproximadamente 15 dB para todas as comunicações, assumindo uma configuração fixa de 4×1 antenas. Portanto, em vez de propor um modelo analítico para a operação do STBC, eles implementaram uma extensão do raio de alcance fornecida pelo ganho de diversidade como uma diminuição temporária do limiar de detecção de SNR em um dado receptor (pelo valor previamente mencionado e fixado em 15 dB).

Xi *et al.* [27] propuseram um algoritmo adaptativo para enlace de malha aberta (*open-loop*) para redes MIMO com o padrão IEEE 802.11n. Este algoritmo adapta o modo de transmissão do enlace variando o esquema de modulação/codificação (MCS, do inglês *Modulation/Coding Scheme*) e o modo MIMO de acordo com informações sobre o canal do último pacote recebido. O enlace pode se adaptar desde o modo mais confiável (modulação BPSK, codificação 1/2, STBC) ao modo de maior taxa de dados (modulação 64-QAM, codificação 5/6, multiplexagem), em um total de 16 modos. O algoritmo é distribuído no transmissor e no receptor e coordenado pelas camadas MAC e física em um processo colaborativo entre as duas camadas. Basicamente, o receptor, a partir da SNR do pacote recebido do transmissor e do modelo de canal, define qual MCS será mais apropriado para prover a maior taxa de dados possível. Devido ao foco no impacto dos modos de transmissão na vazão, eles desconsideraram o impacto da colisão de quadros na avaliação da vazão. Para simular este algoritmo, eles implementaram uma abstração da camada física MIMO com tabelas de consulta para valores de BER versus SNR para as diferentes técnicas usando o simulador OPNET. Para obter seus resultados de simulação, eles usaram cenários *indoor* e *outdoor* para uma topologia aleatória de apenas seis nós, ou seja, a técnica não foi avaliada para redes maiores.

Xia *et al.* [28] propuseram um algoritmo de taxa automatizada de malha aberta (*autorate fallback open-loop*) para rede MIMO com o padrão IEEE 802.11n. O algoritmo consiste em estimar a qualidade do enlace e, dinamicamente, escolher o esquema de transmissão mais apropriado compreendendo o modo MIMO e o MCS. O feedback consiste apenas no ACK e um valor do indicador de intensidado do sinal recebido (RSSI, do inglês *Received Signal Strenght Indicator*) medido em cada antena receptora com o objetivo de obter uma relação que permite o transmissor estimar a dinâmica do canal. As estatísticas acumuladas de transmissões anteriores são atualizadas dinamicamente. Com estes parâmetros, o enlace pode transmitir usando um dos 32 modos de MCS definidos, escolhendo o número de segmentos de dados e o modo MIMO. Eles usaram o MATLAB® em conjunto ao NS-2 para desenvolver detalhes da camada física do IEEE 802.11n e, além disso,

o trabalho teve como foco apenas o desempenho em *nível de enlace*, ou seja, não examinaram o impacto do uso de MIMO em nível de rede.

Zhu *et al.* [29] apresentaram um esquema de sistema MIMO que é uma combinação de STBC e V-BLAST e o aplica em redes *ad hoc*. Este esquema [30] propõe que as antenas transmissoras sejam divididas em grupos, enquanto os códigos STBC são aplicados aos diferentes grupos e o algoritmo V-BLAST é usado no receptor. Para aplicar este esquema em uma rede *ad hoc*, eles propuseram um protocolo de controle de acesso ao meio, o CSMA/TDCA (do inglês, *Carrier Sense Multiple Access with Time Division Collision Avoidance*), que aproveita o fato do STBC operar em períodos de tempo incluindo o conceito de múltiplo acesso por divisão de tempo (TDMA, do inglês *Time Division Multiple Access*) para rejeitar interferências, de modo que, em cada período de tempo definido, apenas um nó pode transmitir. Da forma que o protocolo foi proposto, este esquema de combinação das técnicas MIMO é utilizado para todos os enlaces da rede independentemente do estado do canal. O trabalho apresentou o desempenho em nível de enlace apenas e mostrou que o esquema tem taxas de erro de bit menores que o sistema V-BLAST, porém maiores que o esquema STBC. Entretanto, não apresentou resultados númericos acerca do desempenho em nível de rede para comparação com o nosso trabalho.

O atual desenvolvimento do padrão IEEE 802.11n no NS-3 foi iniciado pela Universidade de Florença (LART lab) [11]. A implementação visa as seguintes características: *frame aggregation*, *block* ACK, HCF (do inglês *Hybrid Coordination Function*), TXOP (do inglês *Transmit Opportunity*), HT (do inglês *High Throughput*) *terminal*, e MIMO. Até a última versão do NS-3, algumas características já foram implementadas, como *frame aggregation*, HCF, TXOP, e *block* ACK. Infelizmente, até então, a possibilidade de uso de MIMO não foi implementado. A proposta de implementação era tratar cada segmento de dados como apenas fluxos paralelos, e esta abordagem iria requerer bastante esforço computacional nas camadas de nível mais baixo.

Considerando o trabalho desenvolvido até então, nosso objetivo é apresentar uma implementação viável do uso dos sistemas MIMO V-BLAST e do esquema de Alamouti para o simulador NS-3, e que permita a avaliação de desempenho de redes *ad hoc* genéricas, não totalmente conectadas, sob topologias diversas. Mais ainda, a partir de nossas investigações preliminares, propomos um protocolo adaptativo que utiliza ambas as técnicas (V-BLAST e Alamouti) que, de acordo com nossa avaliação, permite ganhos maiores de vazão com relação ao uso individual de V-BLAST ou de Alamouti apenas.

Capítulo 4

Implementação de Sistemas MIMO em NS-3

4.1 Introdução

A escolha do modo de transmissão tem um impacto direto na característica fundamental de uma rede sem fio, a vazão [28]. Para tanto, o uso de múltiplas antenas para transmissão e recepção tem se apresentado como chave para alcançar melhores desempenhos e maiores taxas de dados, por meio de técnicas como o esquema de Alamouti e o sistema V-BLAST, respectivamente.

Para estudar estas técnicas, é de grande interesse a implementação de sistemas MIMO em simuladores de rede e, até então, há uma carência nos simuladores computacionais populares. Em particular, é de interesse a implementação de técnicas de diversidade, especialmente de diversidade a partir do transmissor na forma do Esquema de Alamouti [10], e técnicas que implementem multiplexagem espacial, no caso, o sistema V-BLAST [1]. A implementação destas técnicas se dá a partir de abstrações da camada física realizadas por meio de modelos analíticos ou expressões matemáticas. Para o esquema de Alamouti, temos o modelo analítico proposto por Carvalho e Garcia-Luna-Aceves [12], e para o sistema V-BLAST, temos a expressão da taxa média de erro de bit proposta por Loyka e Gagnon [2]. Contudo, devemos inicialmente implementar as características básicas de sistemas MIMO no simulador, como por exemplo, as múltiplas antenas da estação e o modelo de canal.

A fim de investigar como essas duas técnicas podem ser utilizadas simultaneamente em uma mesma rede, dada as suas diferentes vantagens, propomos um protocolo híbrido/adaptativo capaz de trocar informações entre as camadas MAC e PHY (*cross-layer*) para realizar a escolha do melhor método a ser utilizado em determinadas condições da rede. Esta proposta busca uma solução para aproveitar o melhor de cada técnica e obter desempenhos melhores do que os obtidos com o uso individual de cada técnica em toda a rede. Um modo de transmissão dinâmico pode maximizar a utilização da rede sem fio MIMO tirando vantagem da multiplexagem espacial e da diversidade de transmissão.

Neste capítulo apresentamos a abordagem utilizada para a implementação do esquema de

Alamouti e do sistema V-BLAST no simulador, assim como o protocolo adaptativo proposto. Inicialmente, na Seção 4.2, apresentamos a implementação das características básicas de sistemas de rede sem fio e dos mecanimos de propagação do sinal. Para implementação da camada física, apresentamos na Seção 4.3 a forma como abstraímos o processamento do sinal pelas múltiplas antenas. Na Seção 4.4, descrevemos a operação de avaliação de canal livre, importante para protocolos MAC do tipo CSMA/CA, e descrevemos na Seção 4.5, como o NS-3 calcula a razão sinal-interferência-e-ruído (SINR) no receptor e a taxa de erro de bit (BER), de um dado esquema de modulação. A implementação do esquema de Alamouti e do sistema V-BLAST é descrita nas Seções 4.6 e 4.7, respectivamente. Na Seção 4.8 apresentamos as modificações feitas nas camadas MAC e PHY para sistemas MIMO. Finalmente, descrevemos os princípios e a implementação do protocolo híbrido adaptaivo proposto na Seção 4.9.

4.2 Modelo de Canal de Desvanecimento

Em redes sem fio, onde a informação se propaga via sinal eletromagnético e sofre perdas no ambiente, o desempenho depende fortemente dos efeitos de propagação em larga escala e de pequena escala. O sinal transmitido sofre atenuação à medida que a distância até o receptor aumenta e também sofre o efeito de desvanecimento causado pela propagação do sinal via múltiplos percursos. Em nossa implementação, utilizamos o modelo Two-Ray, para propagação em larga escala, e o modelo Rice, para desvanecimento em pequena escala, para avaliar o desempenho das redes sem fio.

O fator K do modelo Rice, razão entre a potência LOS e a potência NLOS, foi implementado alterando-se o valor do parâmetro $\wp$, um atributo da classe *ns3::NakagamiPropagationLossModel*. Esta classe está relacionada ao desvanecimento de pequena escala, em que a potência do sinal transmitido é degradada por reflexões do mesmo sinal. Esta degradação é modelada estatisticamente como uma variável aleatória, de forma que, se a amplitude sofre desvanecimento de distribuição Nakagami-m, a potência segue a distribuição Gama. Esta relação é possível pois, em particular, dada uma variável aleatória $Y \sim Gamma(k, \theta)$ é possível obter uma variável aleatória $X \sim Nakagami(\mu, \omega)$ a partir das substituições $k = \mu$ e $\theta = \omega/\mu$, tal que

$$X = \sqrt{Y}. \tag{4.1}$$

As distribuições Nakagami e Rice têm formas bastante semelhantes, e uma pode ser usada para aproximar a outra. Para $\wp > 1$, pode se mostrar que o fator $\wp$ pode ser calculado a partir de K de acordo com a seguinte equação [31]:

$$\wp = \frac{(K+1)^2}{2K+1}, \tag{4.2}$$

sendo $\wp$ o fator de propagação Nakagami, e K o fator de propagação Rice.

Devido à necessidade de se avaliar o desempenho do protocolo sujeito ao desvanecimento Rice, verificamos se o simulador NS-3 gerava amostras de amplitudes de sinais de acordo com os modelos de desvanecimento Rice e Rayleigh (caso especfífico em que $K = 0$). Para isto, foi realizado um teste prévio para saber se esta implementação era válida. Valores de potência com média unitária

foram amostrados de uma simulação genérica para diferentes valores de K. Esses valores foram devidamente tratados com a Eq. (4.1), e um histograma gerado para comparação com a curva da função de densidade de probabilidade de Rice descrita na Eq. (2.6).

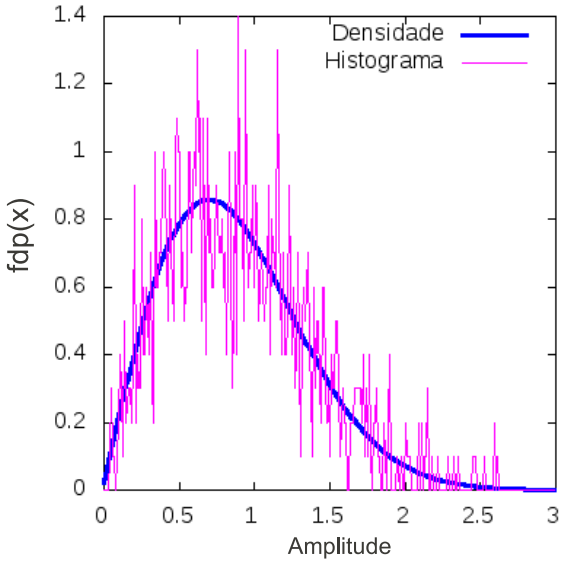


Figura 4.1: $K = 0$.

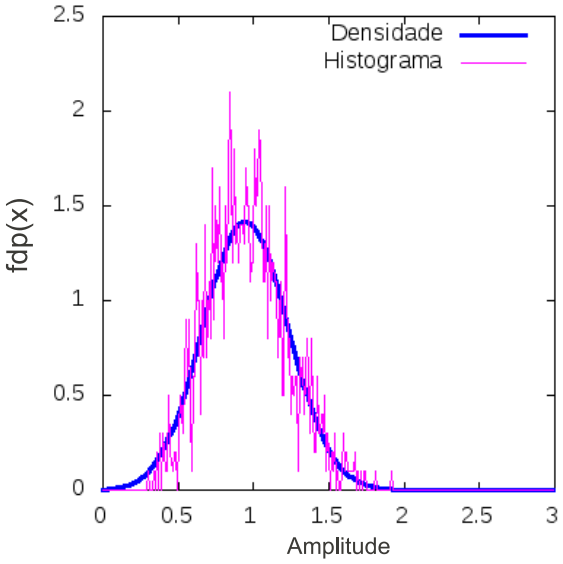


Figura 4.2: $K = 5$.

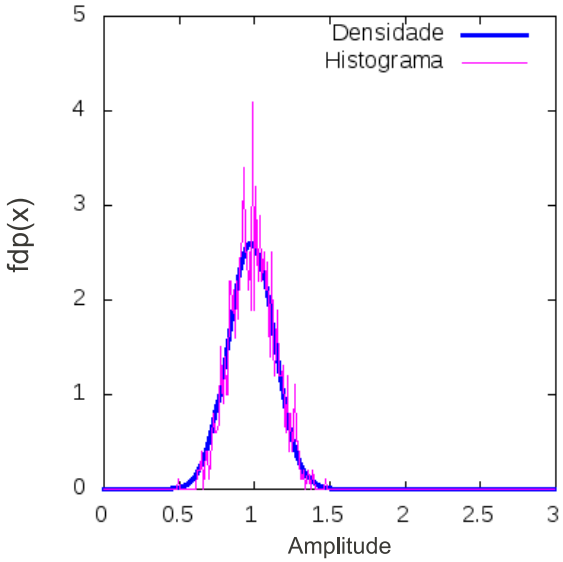


Figura 4.3: $K = 20$.

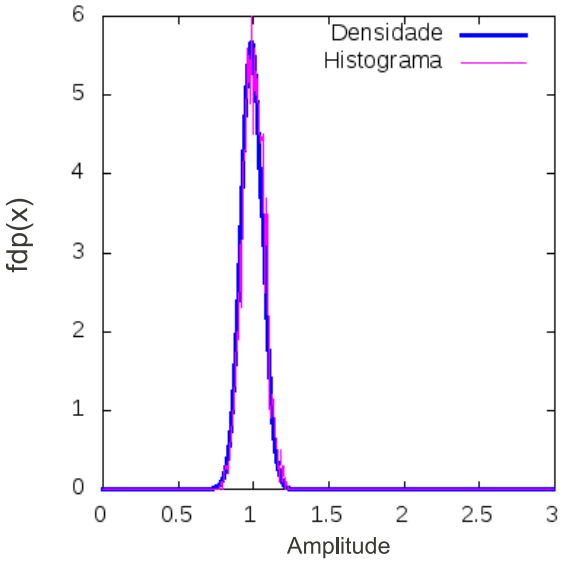


Figura 4.4: $K = 100$.

Figura 4.5: Histogramas das amostras de amplitude para diferentes valores de K em comparação com a função de densidade de probabilidade de Rice.

De acordo com os resultados obtidos, apresentados nas Figuras 4.1, 4.2, 4.3 e 4.4, as amostras de amplitude geradas pelo simulador com diferentes sementes atendem aos modelos teóricos para diferentes valores de K. As figuras mostram histogramas de um conjunto de 1000 amostras da variável aleatória gerada, para cada valor de K, e compara com a função densidade de probabilidade

da distribuição Rice correspondente.

4.3 Efeitos do Canal MIMO

Para definição do número de antenas transmissoras e receptoras, foram adicionados dois parâmetros básicos na classe do modelo de camada física, *ns3::YansWifiPhy*. Além disso, o cálculo da potência recebida de cada caminho, entre as múltiplas antenas, foi adicionado na classe do modelo de canal *ns3::YansWifiChannel*. O método original implementado na classe do modelo de canal *ns3::YansWifiChannel::Send* atribuía o resultado do cálculo da potência recebida (dados a potência de transmissão, posição dos nós e modelo de perda) a uma única variável. A fim de adaptar esse método às nossas necessidades, essa classe foi alterada de forma que o valor calculado da potência de cada caminho passou a ser armazenado em uma *matriz* que é passada para classe do modelo de camada física. Na camada física, a matriz (na qual cada elemento armazena a potência recebida de um caminho) é processada de acordo com a técnica MIMO utilizada. O tamanho da matriz é função dos parâmetros do número de antenas do modelo de camada física.

Esta matriz simula a potência recebida de cada uma das M antenas transmissoras para cada uma das N antenas receptoras, como a matriz de canal $\boldsymbol{H}$ representada na Eq. (2.10). Assumimos que a potência de transmissão é igualmente atenuada pelo modelo de perda em larga escala em todos os caminhos percorridos das antenas transmissoras às antenas receptoras em um único enlace de par de nós. Ao mesmo tempo, para estudo do desempenho dos esquemas de Alamouti ou V-BLAST, a potência média recebida (devido às perdas de larga escala) é degradada de acordo com o modelo de desvanecimento apropriado (Rice ou Rayleigh) independentemente para cada elemento da matriz. Uma vez que o NS-3 assume que a potência recebida de um dado transmissor é a mesma por toda a duração da transmissão do quadro, a matriz de canal é mantida constante durante a transmissão do quadro. No entanto, todo quadro recebido (de transmissores distintos) tem associado sua própria matriz de canal. Na recepção, cada antena recebe a potência de todas as antenas transmissoras (fonte e interferentes).

4.4 Mecanismo de Detecção de Canal Livre

A operação de avaliação de canal livre (CCA) é um importante aspecto a ser considerado no uso de sistemas MIMO, uma vez que o seu uso pode ter implicações no reuso espacial da rede, como apresentado na Seção 2.2.3.2. Portanto, uma decisão cuidadosa deve ser tomada na execução da detecção de energia.

Em modo de recepção, o receptor captura a energia das M antenas transmissoras em suas N antenas receptoras. Dado que cada segmento de dados enviado por cada uma das M antenas em um nó é transmitido com uma energia correspondente a $1/M$ da energia total permitida, o receptor detectará a energia total transmitida (sob atenuação e desvanecimento) em *cada* antena receptora. Consequentemente, a energia total recebida por todas as N antenas receptoras será aproximadamente N vezes a energia recebida em uma única antena.

Para permitir um maior reuso espacial, nós decidimos executar a detecção de portadora baseada na *média da energia recebida* em N antenas receptoras. Este valor médio é comparado a um dado limiar para decidir se o canal está ocupado ou ocioso.

Esta decisão é feita no modelo de camada física, no método *ns3::YansWifiPhy ::StartReceivePacket*, onde o valor da potência recebida passado pelo modelo de canal é comparado com o valor do limiar de CCA. Nesta implementação, o modelo de canal repassa uma matriz contendo o valor da potência de cada caminho e então calcula a média do valor dos elementos dessa matriz.

4.5 Cálculo do BER no NS-3

O NS-3 é um simulador de rede a eventos discretos. Como tal, ele não foi projetado para tratar eventos em nível de símbolos enviados por um transmissor, mas sim, um quadro composto por muitos bits. Consequentemente, o comportamento de um dado esquema de modulação/codificação deve ser abstraído através do uso de expressões matemáticas ou de tabelas de consulta que descrevam o desempenho geral de um dado esquema em função da SINR calculada em um dado receptor. Um quadro, durante sua recepção, pode sofrer interferência de outros quadros em diferentes momentos. Cada um destes momentos terão SINR distintas entre si, e portanto, BER distintas dado o mesmo esquema de modulação/codificação. Desta forma, o quadro pode ser dividido em fragmentos (do inglês *chunk*) onde há um conjunto de bits em que a BER e a taxa de transmissão são assumidas constantes [32], como ilustrado na Figura 4.6.

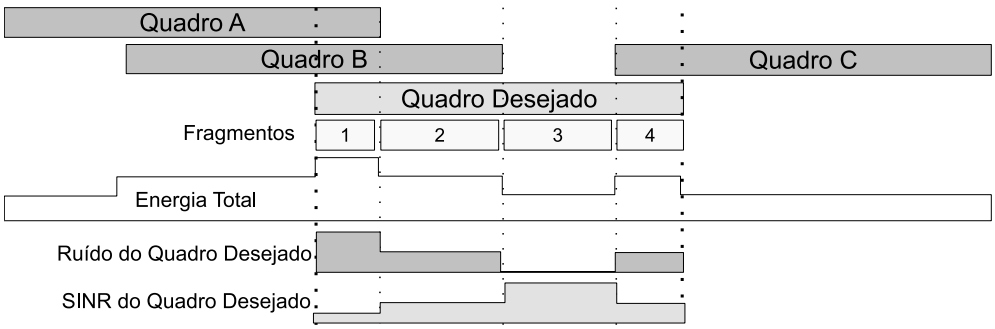


Figura 4.6: Exemplo de fragmentação de um evento de recepção de quadro no NS-3. O quadro recebido desejado é dividido em quatro partes desiguais que têm diferentes SINRs, em relação aos quadros recebidos A, B e C.

Para cada fragmento, a SINR é calculada no NS-3 como

$$\text{SINR} = \frac{P_s^r}{P_n + \sum_k P_k^r}, \tag{4.3}$$

sendo P_s^r a potência recebida no nó r a partir de um sinal transmitido pelo nó s, P_n a potência média do ruído de fundo gerada por fontes quaisquer e P_k^r a potência de interferência recebida de outras fontes k que agendaram transmissões no mesmo intervalo/fragmento. Uma vez que a SINR é calculada, seu valor é usado para calcular a BER em função da SINR de um dado esquema de modulação (ex. DBPSK, 16-QAM, etc). Tal cálculo é baseado em uma expressão matemática fechada para a dada modulação (e codificação). Dada a taxa de transmissão do sistema e a duração

no tempo do fragmento, é possível saber a quantidade de bits que o fragmento contém. Então, sabendo-se a BER da modulação/codificação específica, e o número de bits no fragmento dado, calcula-se a taxa de sucesso do fragmento. Um importante aspecto a salientar a respeito da camada física do NS-3 é o pressuposto da independência entre erros de bits em uma dado quadro. De fato, se *nbits* é o número de bits em um dado fragmento, sujeito a uma dada BER, o NS-3 calcula a probabilidade P_{sc}^{i} de sucesso da transmissão do i-ésimo fragmento como

$$P_{sc}^{i} = (1 - BER)^{nbits}. \tag{4.4}$$

Finalmente, a taxa de erro do quadro (PER) é calculada considerando-se todos os fragmentos no quadro, de acordo com

$$\text{PER} = 1 - \prod_{i=1}^{nchunks} P_{sc}^{i}, \tag{4.5}$$

sendo *nchunks* o número total de fragmentos em um dado quadro. Este valor de PER é passado para a classe responsável pela recepção do pacote, onde o NS-3 seleciona aleatoriamente um número no intervalo [0,1], como forma de simular a recepção de quadros com sucesso ou com erros. Se este número for maior que o valor de PER, o pacote é recebido com sucesso. Caso contrário, o pacote é descartado.

4.6 Implementação do Esquema de Alamouti

Como mostrado no Capítulo 2, Seção 2.4.1, é possível implementar o comportamento da operação do esquema de Alamouti a partir da observação do efeito final na SINR no receptor após as operações de decodificação próprias do esquema de Alamouti. Assim, podemos abstrair no simulador o uso de antenas MIMO a partir do cálculo apropriado da SINR na recepção de um quadro de dados. A SINR para o esquema de Alamouti é calculada como na Eq. (2.19). Para implementar o cálculo do SINR, modificamos a cálculo da potência recebida para configurações MIMO para transformar a Eq. (4.3) na Eq. (4.6) a seguir, a qual nada mais é do que a Eq. (2.19) em função da potência recebida. Assim, a SINR resultante é dada por

$$\text{SINR} = \frac{\sum_i \sum_j P_{ij}}{P_N + \sum_k \sum_i \sum_j I_{ij_k}}, \tag{4.6}$$

sendo P_{ij} a potência recebida entre a j-ésima antena transmissora e a i-ésima antena receptora, P_N a potência média do ruído, e I_{ij}^{k} a potência recebida na i-ésima antena receptora da j-ésima antena transmissora do k-ésimo interferente. A potência recebida P_{ij} é a potência transmitida por uma antena, P_T/M, degradada pelos efeitos de propagação em larga escala e pequena escala. O valor deste SINR é passado para o cálculo do BER que, no caso do esquema Alamouti, é a função de BER da modulação utilizada DBPSK dada por

$$\text{BER}_{DBPSK} = 0.5 \times \exp(-\text{SINR}). \tag{4.7}$$

A modulação DBPSK foi particular para o nosso caso. Entretanto, o esquema de Alamouti pode ser implementado com várias modulações.

4.7 Implementação do Sistema V-BLAST

Assim como na implementação do esquema de Alamouti, nossa implementação do Sistema V-BLAST foca no efeito final, combinado, do uso de múltiplas antenas e operações específicas do V-BLAST na decodificação do sinal recebido. Logo, para implementação do modelo analítico do cálculo do BER do V-BLAST proposto por Loyka e Gagnon [11], foi necessária a criação de um novo método contendo a função de BER na classe de modelo de erro *ns3::DsssErrorRateModel*, que contém outros métodos com funções de BER de outras modulações/codificações. Novamente, é importante enfatizar que não simulamos a divisão do quadro em segmentos independentes para transmissão. Mas o valor da potência de transmissão P_T é divido pelo número de antenas transmissoras M, e a duração original da transmissão do quadro é divida de acordo com o número de antenas transmissoras. Assim, em vez de tratar múltiplas transmissões de quadros, na verdade nós tratamos um quadro mais curto recebido com "múltiplas potências" para representar o canal MIMO $N \times M$. Portanto, a característica mais importante que nós adicionamos é o pós-processamento na recepção de um dado quadro.

O modelo proposto por Loyka considera a BER *média* total calculada para todas as N antenas. Então, a potência recebida média entre as N antenas é utilizada para o cálculo do SINR. Logo, a SINR resultante é dada por

$$\text{SINR} = \frac{\sum_i \sum_j P_{ij}/N}{P_N + \sum_k \sum_i \sum_j I_{ij}^k/N}, \tag{4.8}$$

sendo P_{ij} a potência recebida entre a j-ésima antena transmissora e a i-ésima antena receptora, P_N é a potência média de ruído, e I_{ij}^k é a potência recebida na i-ésima antena receptora da j-ésima antena transmissora do k-ésimo interferente. A potência recebida P_{ij} é a potência transmitida por uma antena, P_T/M, degradada pelos efeitos de propagação em larga escala e pequena escala. O valor deste SINR é passado para o cálculo do BER com o novo método criado usando a função descrita na Eq. (2.27).

O processamento da potência recebida, tanto do esquema Alamouti quanto do sistema V-BLAST, é realizado no modelo de camada física utilizando a matriz de canal passada pelo modelo de canal. Com isto, calcula-se a média dos elementos da matriz (no caso do cálculo do SINR do sistema V-BLAST e SISO, e para a operação de CCA) e a soma dos elementos (no caso do cálculo do SINR do esquema Alamouti). Esses dois valores são passados para o modelo de interferência, onde de fato é calculada a SINR e a taxa de erro do quadro. No modelo de interferência, são calculados todos os eventos de todos os nós da rede. Para calcular a SINR, o modelo verifica a potência dos eventos que estão ocorrendo simultaneamente ao evento desejado e os define como potência interferente, além de considerar a potência de ruído.

4.8 Implementação do Protocolo MAC com MIMO

O sistema V-BLAST proporciona a diminuição no tempo de transmissão total de um quadro ao utilizar mais antenas para transmitir partições do quadro simultaneamente. Os nós que participam da transmissão, tanto os comunicantes quanto os vizinhos, devem estar conscientes desta diminuição da duração de transmissão total do quadro. Neste trabalho, adaptamos a função de

coordenação distribuída do padrão IEEE 802.11b para uso dos sistemas MIMO implementados em redes *ad hoc* de comunicações sem fio. De acordo com as especificações do padrão IEEE 802.11 original, a informação sobre o tempo de duração do quadro que é introduzida no cabeçalho depende apenas da quantidade de bits e da taxa de transmissão adotada. Com a multiplexagem espacial do quadro dividido em segmentos, a duração do quadro corresponderá agora à duração de um segmento, que seria a duração do quadro inteiro dividida pelo *número de antenas transmissoras*. Portanto, é necessária a atribuição de um novo parâmetro no protocolo MAC, o número de antenas transmissoras. Tal parâmetro é repassado à camada MAC pela camada física.

Diferentemente do sistema V-BLAST, o esquema de Alamouti não diminui a duração do quadro, justamente por enviar a mesma quantidade de símbolos em um mesmo intervalo de tempo em relação à configuração SISO. O tempo de duração do quadro não é afetado quando a transmissão é realizada usando o esquema de Alamouti. Por isso, é necessária a atribuição de um outro novo parâmetro, a *técnica MIMO utilizada*. Assim, com o protocolo modificado, a informação sobre o tempo de duração de transmissão de um quadro agora depende da quantidade de bits, da taxa de transmissão, do número de antenas transmissoras e da técnica MIMO utilizada.

A transmissão de quadros de controle, como RTS, CTS e ACK, é feita usando a configuração SISO, em vez de utilizar configuração MIMO. Isto permite que (possivelmente) dispositivos sem MIMO próximos detectem o canal e evitem colisões. Este procedimento é adotado pelo padrão IEEE 802.11n para compatibilidade com dispositivos 802.11a/b/g. Apesar de não estarmos implementando o padrão IEEE 802.11n, usamos o mesmo argumento para fazer este procedimento.

Com o canal reservado para transmissão de dados, os quadros de dados são enviados utilizando a configuração MIMO. Devido ao sistema V-BLAST permitir transmissões de dados mais rápidas comparado à configuração SISO, precisamos adaptar o vetor de alocação de rede (NAV, network allocation vector) nos nós adequadamente. A Figura 4.7 demonstra a redução no tempo da transmissão do quadro de dados e o correspondente tempo economizado na negociação de 4 vias (*4-way handshake*) descrita na Seção 2.2.2.1. Desta maneira, o período de contenção começa mais cedo e mais transmissões podem ocorrer no mesmo período de tempo, em comparação ao caso sem multiplexagem. O campo de duração de transmissão dos cabeçalhos dos quadros RTS, CTS e DATA variam com o número de antenas transmissoras. Quanto mais antenas, menor é o tamanho dos segmentos do mesmo quadro. Quanto menor o tamanho dos segmentos, menor o tempo de transmissão. Sabendo o número de antenas transmissoras da camada física, a camada MAC pode calcular o tempo de transmissão e atualizar o NAV adequadamente.

Estas modificações foram feitas na classe *ns3::MacLow*, onde é tratado o envio e recepção de quadros da negociação de 4 vias (*4-way handshake*).

Além disso, foi adicionada uma sinalização no método *ns3::MacLow:: SendDataAfterCts* onde é feito o envio do quadro de dados. Esta sinalização avisa à camada física durante a recepção de um quadro se este deve ser processado com configuração MIMO e qual técnica está sendo indicada.

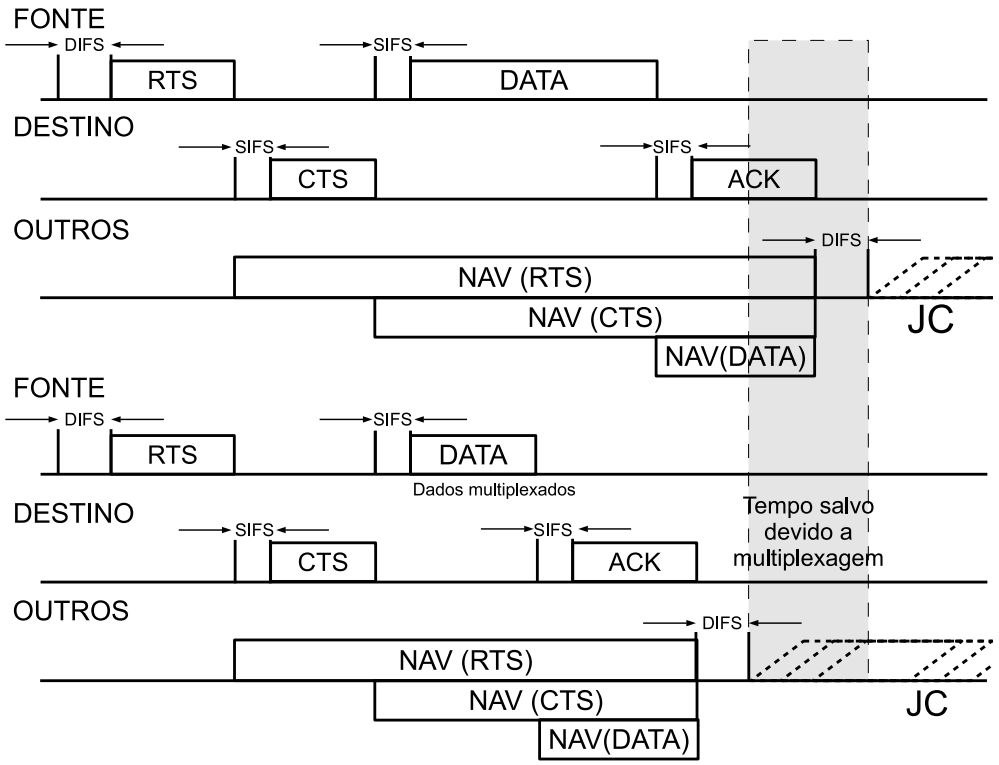


Figura 4.7: Comparação das negociações de quatro vias que utilizam a multiplexagem espacial com relação à transmissão sem uso da multiplexagem espacial (abaixo e acima, respectivamente). A área cinza destacada mostra que a negociação de 4 vias termina mais cedo com o uso da multiplexagem, o que contribui para redução do tempo até ocorrência de uma nova disputa de canal.

4.9 Implementação do Protocolo MAC Híbrido Adaptativo

Em uma rede *ad hoc*, cuja topologia apresenta pares fonte-destino com variadas distâncias entre os nós, é de se esperar que a SINR das transmissões seja diferente para cada par. Em uma topologia aleatória, é esperado haver pares com nós muito próximos e outros muito distantes entre si. Portanto, em uma mesma rede é possível observar uma diferença muito grande no valor das SINRs percebidas nos receptores. Assim, utilizando o mesmo esquema de transmissão para todos os nós da rede, vários nós poderiam estar descartando seus pacotes devido à alta taxa de erro, ou, por outro lado, estar com uma taxa de erro muito baixa e não utilizar a capacidade total do sistema MIMO. É nesse contexto que surge a proposta de um protocolo capaz de maximizar a utilização da rede por meio do uso de diferentes esquemas de transmissão. A proposta e a descrição deste protocolo serão apresentadas na Seção 4.9.1. A sua implementação no simulador NS-3, para fins de validação, será descrita na Seção 4.9.2.

4.9.1 Proposta do Protocolo MAC Híbrido Adaptativo

A escolha do modo de transmissão tem um impacto direto na característica fundamental de uma rede sem fio, a vazão [28]. Um modo de transmissão estático não maximiza a vazão, pois impossibilita o uso das múltiplas antenas da melhor forma possível, dependendo das condições do

canal, cujas características variam no tempo e espaço, e não se adapta às rápidas alterações nas condições de tráfego. Por outro lado, a possibilidade de uso simultâneo e adaptativo dos dois modos de transmissão (Alamouti e V-BLAST) — a depender do estado do canal em cada enlace individual — pode maximizar a capacidade da rede a partir da exploração dos benefícios da multiplexagem espacial e da diversidade de transmissão.

Nesta seção apresentamos a descrição de um novo protocolo para a camada de controle de acesso ao meio (MAC), o qual foi projetado segundo o paradigma de projeto de protocolos com camadas interrelacionadas (o chamado "*cross-layer design*"), ou seja, cujo projeto não assume independência de operação entre as camadas, como nos protocolos tradicionais. No caso, o protocolo MAC proposto utiliza informações disponibilizadas pela camada física (PHY) para determinar os mecanismos apropriados para comunicação com um determinado destinatário. Por sua vez, o protocolo MAC determina à camada PHY o melhor mecanismo de transmissão com múltiplas antenas para o enlace em questão. O protocolo MAC proposto, assim como o uso das tecnologias MIMO V-BLAST e Alamouti, substituem a arquitetura do protocolo tradicional do padrão IEEE 802.11 nas camadas MAC e física, para uso em redes sem fio no modo *ad hoc*. A idéia principal do protocolo proposto consiste no aumento da vazão desse tipo de rede, a partir do uso de diferentes técnicas de múltiplas antenas em cada par de comunicação, e considerando modificações apropriadas na negociação de 4 vias (*4-way handshake*) definido na norma IEEE 802.11. Com as modificações propostas, é possível comutar, adaptativamente, a técnica MIMO e a configuração de antenas mais apropriada para transmissão no enlace em questão, considerando a relação sinal-interferência-e-ruído (SINR) calculada no destinatário intencionado da transmissão.

O protocolo tem como chave de comutação apenas a SINR calculada *na recepção de um RTS no destinatário intencionado*. Isto porque o fator determinante para transmissão de um quadro com sucesso para um dado destinatário é a SINR percebida por este. Em uma rede *ad hoc* sem fio, um par fonte-destino pode selecionar modos de transmissão diferentes, dependendo da distância entre os nós, da interferência devido à natureza de múltiplo acesso e da intensidade de ruído de fundo, causado por outras fontes. De modo geral, os nós que estiverem sujeitos a valores de SINR muito baixos devem preferir utilizar o Esquema de Alamouti para obter uma transmissão mais robusta (*diversidade de transmissão*), enquanto que os nós que observarem valores altos de SINR tenderão a utilizar o sistema V-BLAST para obter maiores taxas de dados (*multiplexagem espacial*). No caso do V-BLAST, a transmissão é concluída mais rapidamente e o canal é liberado mais cedo.

Para realizar a comutação, é necessário estabelecer os limiares que determinam a comutação entre uma técnica e outra. A partir dos resultados de simulações e do desempenho de modelos analíticos, identificamos a necessidade de definirmos dois limiares para obtermos três níveis de compromisso. O sistema V-BLAST apresenta taxas de erro mais elevadas em baixos valores de SINR e sua utilização nesta faixa de valores pode ser desvantajosa. Neste caso, o esquema de Alamouti é a melhor opção por garantir uma transmissão mais robusta. O sistema V-BLAST, quando é utilizado com um grau a mais de diversidade de recepção, ou seja, a transmissão é realizada usando uma antena transmissora a menos que o número de antenas receptoras ($M = N-1$), apresentou melhores desempenhos comparado quando é utilizado com a mesma quantidade de antenas transmissoras e receptoras ($M = N$), como é demonstrado no Capítulo 5. A razão deste fato se dá porque a utilização do sistema com diversidade proporciona maior robustez às

transmissões e mais quadros são recebidos com sucesso. Também por isso, a taxa de erro do sistema com diversidade é menor, como pode ser observado no gráfico da Figura 2.10, discutido na Seção 2.4.3. Em uma faixa de valores de SINR grandes o suficiente, a taxa de erro de bit do sistema sem diversidade se torna tão pequena quanto a taxa do sistema com diversidade. Assim, pode ser mais vantajoso utilizar mais antenas para multiplexar os dados transmitidos. Logo, neste caso, o sistema V-BLAST sem diversidade ($M = N$) permitirá obter maiores vazões, com a garantia de que nesta faixa de SINR a quantidade de perdas de quadros é significadamente menor, ao mesmo tempo que reduz o tempo necessário para transmitir o dado.

Basicamente, o nó pode escolher um modo de transmissão entre três possíveis dependendo da SINR calculada no receptor: transmissão utilizando *diversidade de transmissão*, *multiplexagem com diversidade de recepção* ou *multiplexagem sem diversidade de recepção* (*full-multiplexing*).

4.9.1.1 Descrição do Protocolo MAC Híbrido Adaptativo

No protocolo proposto, a comutação dinâmica da técnica MIMO e da configuração de antenas mais apropriada para transmissão é feita de acordo com o valor de SINR calculado no destinatário do quadro de dados ao receber um quadro RTS. Para tanto, com os limiares de decisão previamente configurados para cada faixa de valores de SINR, a qual define o esquema de transmissão a ser utilizado, o destinatário dos dados é o responsável por definir em qual dessas faixas o valor determinado de SINR se encontra.

De acordo com resultados de simulação apresentados na Seção 5.8, verificamos que para diferentes valores de SINR, a utilização conjunta do esquema de Alamouti e do sistema V-BLAST se torna mais apropriada para se atingir melhores desempenhos da rede. Para valores grandes de SINR, os esquemas de modulação apresentam baixas taxas de erro de bit. Logo, nestes casos, o receptor do RTS decidirá pela transmissão por multiplexagem espacial em que todas as antenas transmissoras e receptoras são utilizadas, ou seja, uma configuração do tipo $M = N$, com o número máximo de segmentos de dados em paralelo. Para valores de SINR situados na faixa intermediária entre os dois limiares, nos quais as taxas de erro de bit começam a tornar-se significativas, o receptor do RTS decidirá pela transmissão por multiplexagem espacial com um grau de diversidade, usando uma antena transmissora a menos que o número de antenas receptoras (configuração $M = N - 1$), visto que com este grau de diversidade a taxa de erro se apresenta inferior, proporcionando uma vazão mais elevada. Por fim, para valores baixos de SINR, sujeitos a taxas de erro de bit mais elevadas, a decisão será pela transmissão por diversidade de transmissão (esquema de Alamouti com configuração $2 \times N$), dado que esta técnica proporciona maior robustez sob condições de canal mais adversas.

Após decidir o esquema MIMO, o destinatário intencionado adiciona essa informação ao cabeçalho do quadro CTS (um byte extra), o que permite avisar ao transmissor qual esquema a ser utilizado. O transmissor, ao receber o CTS, verifica esse byte e repassa a informação à camada física, que irá selecionar o esquema MIMO definido para a transmissão do quadro de dados. A transmissão de dados é realizada desde que o canal esteja devidamente reservado, com ambos os nós (transmissor e receptor) cientes do esquema MIMO e da configuração de antenas a serem utilizados. Ao final da recepção do quadro de dados, o receptor envia uma confirmação positiva

pelo quadro ACK.

As Figuras 4.8 e 4.9 apresentam os pseudocódigos dos algoritmos utilizados no protocolo. A variável s representa a técnica de transmissão MIMO a ser utilizada e depende do valor de SINR calculado na recepção de um RTS. No caso, a variável s pode indicar os valores: *alamouti*, $VBLAST(M = N - 1)$, $VBLAST(M = N)$, sendo M o número de antenas transmissoras, N o número de antenas receptoras, $snrMin$ o limiar de decisão inferior, $snrMax$ o limiar de decisão superior e $tempoDADOS$ o tempo da duração da transmissão do quadro de dados.

Algoritmo 1: Selecionar Técnica MIMO

Entrada *SNR* (SNR recebida no RTS)
Saída *s* (técnica selecionada)

Se (*SNR* < *snrMin*) **então**
    s ← *alamouti;*
Se não, Se (*snrMin* < *SNR* < *snrMax*) **então**
    s ← *VBLAST(M=N-1);*
Se não, Se (*SNR* > *snrMax*) **então**
    s ← *VBLAST(M=N);*
Fim Se

Retornar *s;*

Figura 4.8: Pseudocódigo referente a operação de seleção da técnica MIMO.

Algoritmo 2: Atualizar duração do quadro de dados

Entrada *s* (técnica selecionada)
Saída *tempoDADOS* (duração do quadro)

Se (*s = VBLAST(M=N-1)*
        ou *VBLAST(M=N)*) **então**
    tempoDADOS = tempoDADOS / M;
Fim Se

Retornar tempoDADOS*;*

Figura 4.9: Pseudocódigo referente a operação de atualização da duração do quadro de dados.

A Figura 4.10 ilustra o procedimento da negociação de 4 vias com o protocolo proposto em operação e as funções nas camadas MAC e PHY.

4.9.2 Implementação no Simulador

Podemos dividir a implementação no simulador em duas partes, implementação em camada MAC e em camada PHY, que se interrelacionam. A negociação de 4 vias do protocolo é apresentada a seguir:

A camada MAC do transmissor (txMAC), ao receber um pacote de sua fila de transmissão e após adquirir o canal por meio do CCA, inicia a requisição do canal enviando um quadro RTS. A

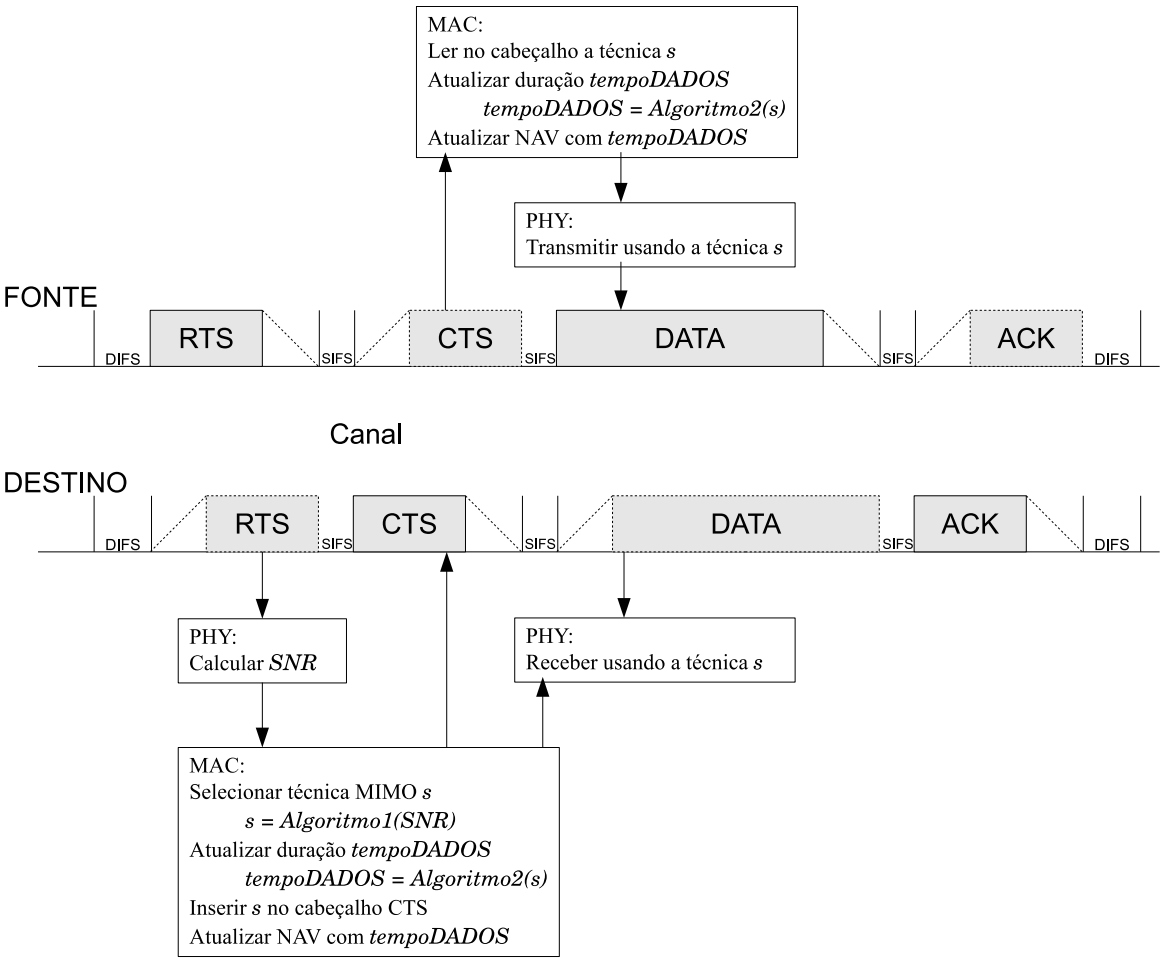


Figura 4.10: Protocolo proposto operando sobre a negociação de 4 vias.

sua camada PHY (txPHY) envia o quadro utilizando a configuração SISO após verificar que é um quadro de controle.

A camada PHY do receptor (rxPHY) recebe o quadro e, ao verificar que se trata de um quadro de controle, calcula a SNR e a taxa de erro do quadro para saber se a recepção foi bem sucedida, utilizando o modelo de erro padrão da configuração SISO. A camada MAC do receptor (rxMAC) é responsável por, ao receber o RTS com sucesso, ler o valor da SNR calculada e repassada pela camada PHY. Tendo este valor, compara-o aos limiares de decisão pré-estabelecidos, para poder decidir o esquema de transmissão. Feita a decisão, adiciona uma sinalização, avisando de sua decisão, no quadro CTS, para ser enviado ao transmissor. A rxMAC atualiza o campo de duração da transmissão no cabeçalho do quadro, caso o esquema decido seja a multiplexagem, para poder atualizar o NAV dos vizinhos. Neste caso, o valor da duração da transmissão do quadro de dados é dividido pelo número de antenas transmissoras utilizado. A rxPHY envia o quadro utilizando a configuração SISO após verificar que é um quadro de controle.

A txPHY recebe o quadro e verifica se a recepção foi bem sucedida da mesma forma que o rxPHY fez no caso anterior. A txMAC, ao receber o CTS com sucesso, lê a sinalização adicionada pela rxMAC e a copia para adicioná-la no quadro de dados (DATA). A txPHY lê a sinalização do quadro e o envia utilizando a configuração sinalizada para transmissão.

A rxPHY recebe o quadro e, ao verificar que se trata de um quadro de dados, lê a sinalização e calcula a SNR e a taxa de erro do quadro, usando o modelo de erro do esquema sinalizado. A rxMAC, ao receber o quadro com sucesso, repassa o pacote de dados para a camada superior e envia o ACK. A rxPHY envia o quadro utilizando a configuração SISO após verificar que é um quadro de controle.

A txPHY recebe o quadro e, ao verificar que se trata de um quadro de controle, usa o modelo de erro padrão. A txMAC, ao receber o ACK com sucesso, encerra a negociação de 4 vias.

Capítulo 5

Avaliação de Desempenho

5.1 Introdução

Neste capítulo são apresentados os resultados das simulações para as redes sem fio *ad hoc* habilitadas com o esquema proposto por Alamouti, o sistema V-BLAST puro e o protocolo híbrido adaptativo proposto implementado sobre o padrão IEEE 802.11b, em seus diferentes modos de operação.

A avaliação de desempenho da rede é feita em termos da vazão média, do atraso médio de pacote e do índice de justiça da rede para diferentes configurações de antenas. Por vazão média efetiva, entende-se o tráfego total transmitido com sucesso divido pela quantidade de usuários na rede, e por atraso médio, entende-se o atraso total dos pacotes transmitidos com sucesso dividido pela quantidade desses pacotes. O atraso é medido a partir do momento que o pacote da camada IP entra na fila de transmissão da camada MAC até o momento quando é recebido na camada IP do receptor, considerando assim o atraso de fila e de retransmissão de quadros até seu sucesso, sem se considerar atraso devido a roteamento. O índice de justiça da rede é um importante fator a ser considerado em estudos de desempenho, uma vez que indica a justiça com a qual o canal de transmissão é compartilhado, e a equidade de condições dos nós para acessá-lo. Jain *et al.* [33] define que o índice de justiça $\Im$ de um sistema cujo meio é disputado por n usuários, pode ser calculado por

$$\Im(x_1, x_2, ...x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n.\sum_{i=1}^{n} {x_i}^2}, \tag{5.1}$$

sendo x_i a vazão efetiva da i-ésima conexão. Esse índice varia no intervalo [0,1], sendo seu valor ótimo igual à 1, o qual indica que todos os n usuários possuem a mesma condição de disputa ao meio.

Para realizar tal avaliação de desempenho, fez-se necessária a definição dos cenários de simulação. As simulações, por sua vez, se deram com base em níveis de confiança desejados, a fim de garantir a coerência dos resultados apresentados.

Na Seção 5.2 descrevemos os cenários de simulação considerados, e apresentamos na Seção 5.4 o método pelo qual o nível de confiança desejado é atingido. Apresentamos a avaliação de

desempenho do esquema de Alamouti na Seção 5.6 e do sistema V-BLAST na Seção 5.7. Em seguida, discutimos a possibilidade de ganhos ao introduzir um sistema híbrido que utiliza as duas técnicas com um MAC adaptativo e as questões que implicam no impacto de seu desempenho na Seção 5.8. Apresentamos o estudo da variação dos limiares de comutação nas Seções 5.8.1 e 5.8.2. Por fim, apresentamos a avaliação de desempenho do protocolo considerando o estudo anterior na Seção 5.8.3.

5.2 Cenários de Simulação

Dado o interesse em investigar o impacto da contenção e dos mecanismos de controle MAC na vazão global da rede, focamos nossas simulações em redes *ad hoc* sem fio estáticas e saturadas. Para tanto, foram consideradas topologias de 100 nós, aleatoriamente posicionados em um terreno plano de 1600 × 1600 m, de forma que cada nó atua tanto como transmissor quanto como receptor, permitindo a existência de 100 pares fonte-destino simultâneos.

Utilizamos o modelo de propagação *Two-Ray* com desvanecimento Rayleigh para calcular a potência recebida em um dado nó. Entretanto, o desvanecimento Rice foi utilizado para demonstrar o comportamento do esquema de Alamouti, sob diferentes níveis de desvanecimento. O limiar CCA foi configurado de forma que a sensibilidade de detecção de sinal no canal correspondesse a uma distância 1,5 vezes maior que o raio de transmissão estabelecido, ou seja, capaz de detectar um sinal de um nó a uma distância de 225 m. Estes parâmetros foram considerados para obtermos uma quantidade considerável de grupos de nós não-diretamente conectados para a limitação da área escolhida. Assim poderemos considerar a interferência de transmissões concomitantes no desempenho do sistema. A Tabela 5.1 resume o restante do parâmetros usados para as camadas PHY e MAC.

Tabela 5.1: Parâmetros de simulação

Limiar de detecção de energia: -73.8764 dBm	Limiar de avaliação CCA: -80.9201 dBm
Figura de ruído: 7 dB	Potência de transmissão: 10 dBm
Altura da antena: 1.2 m	Tamanho do pacote de dados: 1412 bytes
Taxa de transmissão DSSS PHY: 1 Mbps	Modulação: DBPSK
Taxa de tráfego CBR: 1.001 Mbps	Aplicação: UDP

5.3 Geração de Topologias

As topologias foram geradas considerando-se 100 nós posicionados em um terreno plano de 1600 × 1600 m, por meio de distribuição aleatória uniforme, de forma que cada nó atua tanto como transmissor quanto como receptor, permitindo a existência de 100 pares fonte-destino simultâneos. Para tanto, a geração dessas topologias foi dividida em dois processos: criação e verificação. O processo de criação envolve o posicionamento dos 100 nós no terreno plano de 1600 × 1600 m. Este foi feito por meio de três métodos distintos para atingir diferentes níveis de densidade e,

consequentemente, contenção na rede. O processo de verificação visa validar as topologias criadas capazes de formar 100 pares fonte-destino simultâneos, com a restrição de que cada par deveria possuir um raio de transmissão máximo de 150 m.

No primeiro método do processo de criação, os 100 nós são posicionados aleatoriamente no terreno plano de 1600 × 1600 m. Realizando o processo de verificação, foi possível validar topologias como a apresentada na Figura 5.1 (a). Observamos que, devido à restrição do processo de verificação (100 pares fonte-destino simultâneos com um raio de transmissão máximo de 150 m), as topologias validadas com este método se apresentavam bastante densas, resultando em redes com alta contenção.

Com a finalidade de forçar a geração de cenários menos densos, modificamos a forma como era realizado o posicionamento aleatório dos nós no terreno plano de 1600 × 1600 m, no processo de criação. Neste segundo método, dividimos o terreno original em 25 regiões de 320 × 320 m, nas quais 4 nós são aleatoriamente posicionados. Com isso, ao realizar o processo de verificação, conseguimos validar topologias menos densas e, consequentemente, redes com menor contenção, como o exemplo apresentado na Figura 5.1 (b).

Ainda, com o intuito de dar maior aleatoriedade à distribuição dos nós no terreno, em um terceiro método do processo de criação, definimos que a quantidade de nós em cada região de 320 × 320 m deveria ser aleatoriamente escolhida entre 1 e 4. Após o preenchimento sequencial das 25 regiões, os nós ainda não posicionados são aleatoriamente distribuídos no terreno original de 1600 × 1600 m. Um exemplo de topologia obtida por este método, após o processo de verificação, é apresentado na Figura 5.1 (c). Neste caso, observamos que as topologias validadas apresentavam densidade e contenção medianas em relação às geradas anteriormente.

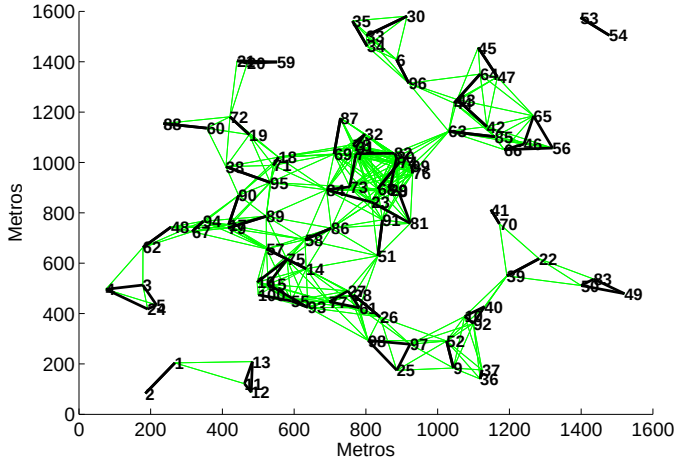
A fim de considerar os mais variados tipos de redes, foram utilizadas nas nossas simulações topologias obtidas pelos três diferentes métodos apresentados. É importante ressaltar que existe a possibilidade de se encontrar topologias menos densas já com o primeiro método do processo de criação apresentado. Entretanto, devido à restrição do processo de verificação de que as topologias deveriam ser compostas por 100 pares fonte-destino simultâneos com um raio de transmissão máximo de 150 m, o que levava à topologias mais densas, houve a necessidade de se forçar a criação de topologias menos densas capazes de ser validadas pelo processo de verificação, dada a sua restrição, em menor tempo.

5.4 Nível de Confiança dos Resultados Apresentados

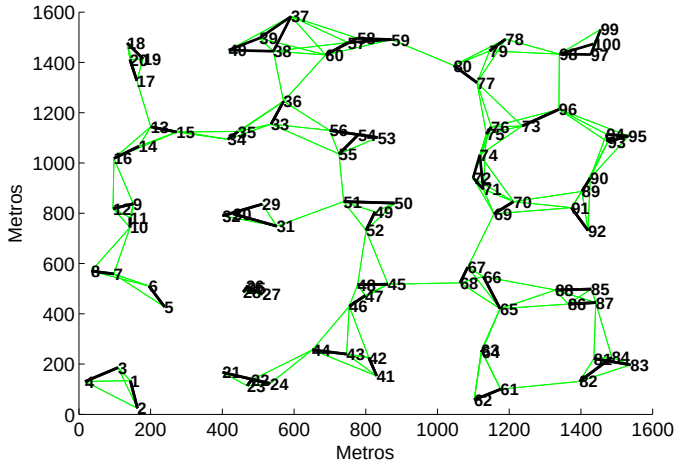
A fim de garantir a confiabilidade dos resultados apresentados, utilizamos o conceito de intervalo de confiança. Para tanto, através da distribuição de probabilidade "t" de Student, é possível calcular o intervalo de confiança para uma média de amostras a partir de um nível de confiança desejado. Para um pequeno número de amostras, o intervalo de confiança pode ser calculado por

$$I = \bar{x} \pm t_{[(1-\alpha)/2, n-1]} \frac{s}{\sqrt{n}}, \tag{5.2}$$

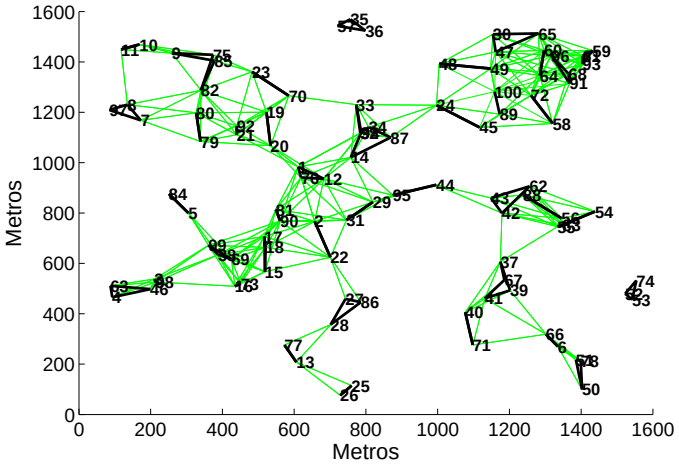
sendo $\bar{x}$ a média das amostras, α o nível de significância (tal que, para um nível de confiança desejado igual a 95%, por exemplo, 1 - $\alpha = 0.95$), s o desvio padrão das amostras, n a quantidade



(a) Maior contenção.



(b) Menor contenção.



(c) Contenção mediana.

Figura 5.1: Topologias com diferentes níveis de contenção. Os números indicados representam a identificação de cada um dos 100 nós da topologia. As linhas escuras indicam os pares fonte-destino, enquanto que as linhas claras indicam os nós dentro do alcance de sensibilidade.

de amostras e t o valor extraído da tabela da distribuição "t" de Student para os parâmetros definidos entre colchetes. Para um número grande de amostras, o intervalo de confiança para uma média de amostras pode ser calculada por

$$I = \bar{x} \pm z_{[(1-\alpha)/2]} \frac{s}{\sqrt{n}}, \tag{5.3}$$

sendo, também, z o valor extraído da tabela da distribuição "t" para os parâmetros definidos entre colchetes. A tabela da distribuição de probabilidade "t" de Student, que inclui os possíveis valores de t e z, é apresentada no Anexo I.

Ao fazer com que as simulações sejam baseadas em níveis e intervalos de confiança mínimos pré-estabelecidos, conseguimos não só garantir a coerência dos dados apresentados, como também automatizar as simulações, de forma a gerarem a quantidade de amostras necessárias para atingir tais parâmetros. Essas amostras são obtidas por meio de simulações, com diferentes sementes, realizadas sobre topologias definidas.

5.5 Metodologia das Simulações

Para realizar as simulações e garantir a confiabilidade dos resultados apresentados, nos baseamos no conceito de intervalo de confiança. Para tanto, desenvolvemos um algoritmo capaz de gerar um arquivo *bash* responsável por proceder com as simulações, dados os parâmetros (configuração de antenas, esquema MIMO, nível de desvanecimento, etc.) definidos pelo usuário.

Neste arquivo *bash* é definido um laço no qual, após simuladas todas as topologias para determinados parâmetros definidos pelo usuário, é realizada a avaliação do intervalo de confiança obtido para as amostras de vazão geradas pelas simulações, dado o nível de confiança desejado. Caso o intervalo de confiança obtido seja inferior ao intervalo de confiança desejado, todas as topologias são simuladas novamente usando-se uma nova semente de simulação, a fim de se obter um número maior de amostras. Esse procedimento é repetido, até que o intervalo de confiança desejado para o nível de confiança desejado seja atingido. A escolha da vazão como parâmetro de parada foi feita dado o foco do trabalho no impacto dos modos de transmissão na vazão. Outros parâmetros, como atraso médio, poderiam ser usados.

Em cada rodada de simulação são colhidas as amostras da vazão, do atraso médio e do índice de justiça. Quando o intervalo de confiança desejado para o nível de confiança desejado é atingido, a média desses três parâmetros é calculada e armazenada. Em um mesmo arquivo *bash* é possível listar vários laços contendo diferentes parâmetros a serem simulados. Assim, ao final de um laço, um outro já é automaticamente acionado. Essa listagem é também gerada pelo algoritmo gerador do arquivo *bash*.

5.6 Redes Ad Hoc MIMO via Alamouti

Nesta Seção, apresentamos os resultados de simulação computacional de redes *ad hoc* MIMO quando habilitadas para utilizar apenas o esquema de Alamouti na camada física, e sob as modificações correspondentes no protocolo MAC e no procedimento de avaliação de ocupação de canal.

A Figura 5.2 ilustra o desempenho da rede calculado sobre sete topologias do tipo que apresenta maiores níveis de contenção, como a apresentada na Figura 5.1 (a), rodadas para diferentes sementes de simulação, com diferentes configurações de antena ($M \times N$) usando o esquema de Alamouti. Para comparação, mostramos a vazão média da rede para a configuração SISO IEEE 802.11b. Os resultados apresentados possuem um nível de confiança de 99.8%.

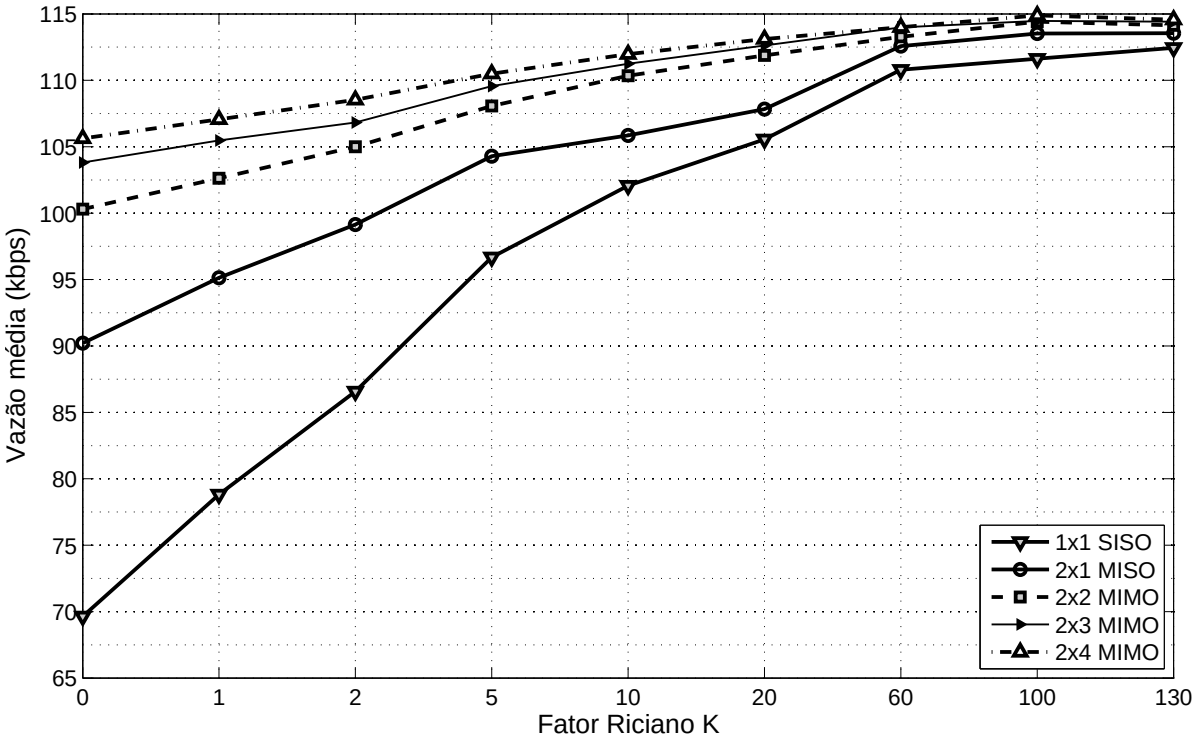


Figura 5.2: Vazão média da rede sob diferentes níveis de desvanecimento para diferentes configurações de antenas.

Pode-se observar que a vazão diminui, ao diminuir valor de K. Pode-se observar que, para valores menores de K (forte desvanecimento), a vazão aumenta à medida que mais antenas são adicionadas ao receptor. Este comportamento é esperado devido ao ganho de diversidade proporcionado pelo esquema de Alamouti (maior robustez à transmissão). Além disso, nota-se que este aumento é menos acentuado quando o valor de K aumenta. Tais resultados são esperados, pois ao aumentar o número de antenas, os erros no canal são mais inibidos e a vazão fica limitada basicamente pelo grau de contenção da rede. Esta inibição de erros do canal começa a se tornar desnecessária para valores maiores de K devido à presença de baixo desvanecimento. Neste caso, a vazão tende a ficar constante para diferentes configurações de antenas.

O desempenho da vazão média da rede usando o esquema de Alamouti sob desvanecimento Rayleigh ($K = 0$) é apresentado na Figura 5.3. Os resultados de simulação são calculados para nove topologias (divididas igualmente entre os três diferentes níveis de contenção, como as apresentadas na Figura 5.1), rodadas para diferentes sementes de simulação, com diferentes configurações de antena ($M \times N$) usando o esquema de Alamouti. Para comparação, mostramos a vazão média da rede para a configuração SISO IEEE 802.11b. As barras de erro representam a margem de erro com nível de confiança de 95%.

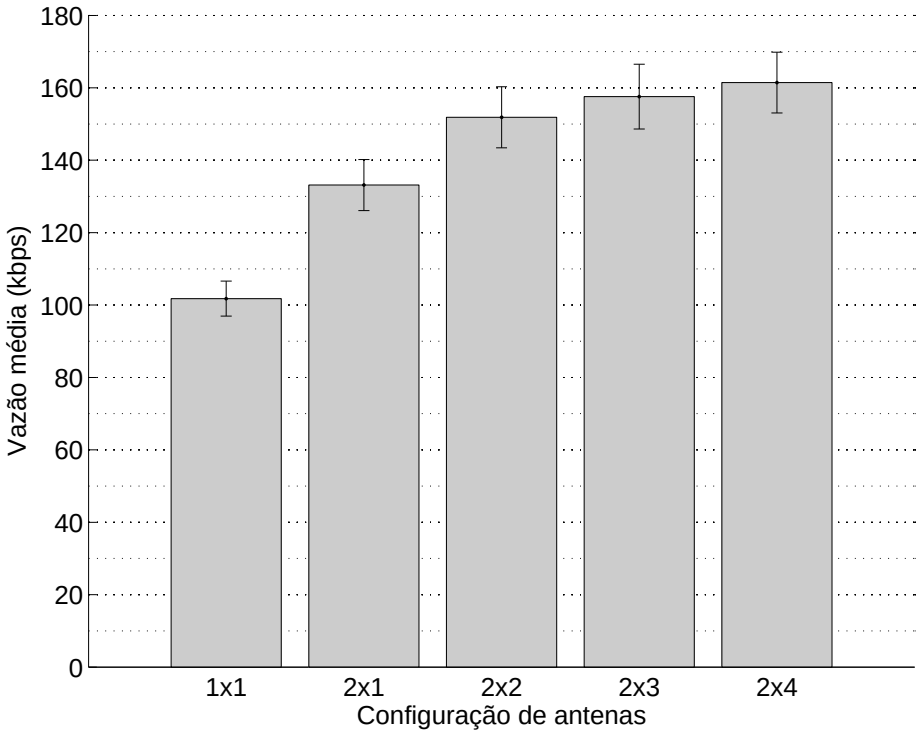


Figura 5.3: Vazão média da rede habilitada apenas com o esquema de Alamouti sob desvanecimento Rayleigh ($K = 0$) para diferentes configurações de antenas.

Pode-se observar que sua utilização melhora o desempenho da rede em relação ao caso SISO. Entretanto, à medida que se aumenta o número de antenas receptoras, o ganho em relação à configuração anterior diminui. Observam-se ganhos de 1,3082, 1,4920, 1,5481 e 1,5863 para as configurações 2×1, 2×2, 2×3 e 2×4, respectivamente, em relação ao caso 1×1. Ao aumentar o número de antenas, os erros no canal são mais inibidos e a vazão fica limitada basicamente pelo grau de contenção da rede. Ou seja, a camada MAC limita a vazão atingível uma vez que a camada PHY alcança seu melhor desempenho. Estes resultados são esperados pois o esquema de Alamouti, por se tratar de uma técnica de *diversidade de transmissão*, não é projetado para obter maiores taxas de transmissão como a técnica de multiplexagem espacial, e sim, projetado para prover uma transmissão mais robusta, menos susceptível a erros [12].

Nas Figuras 5.4 e 5.5 são apresentados os desempenhos do atraso médio de pacote e do índice de justiça da rede usando o esquema de Alamouti. A sua utilização diminui o atraso médio dos pacotes em aproximadamente 20% para todas as configurações de antenas. O comportamento do índice de justiça se mostrou semelhante ao comportamento da vazão. Uma das causas do atraso de pacote, além do tempo de propagação e do processamento das camadas mais baixas, é o tempo perdido devido à recepção sem sucesso, isto é, a retransmissão. Com uma transmissão mais robusta, os quadros tem mais chances de serem recebidos com sucesso evitando o pacote perder tempo com sua retransmissão e o canal pode ser melhor compartilhado tendo menos ocupações mal utilizadas.

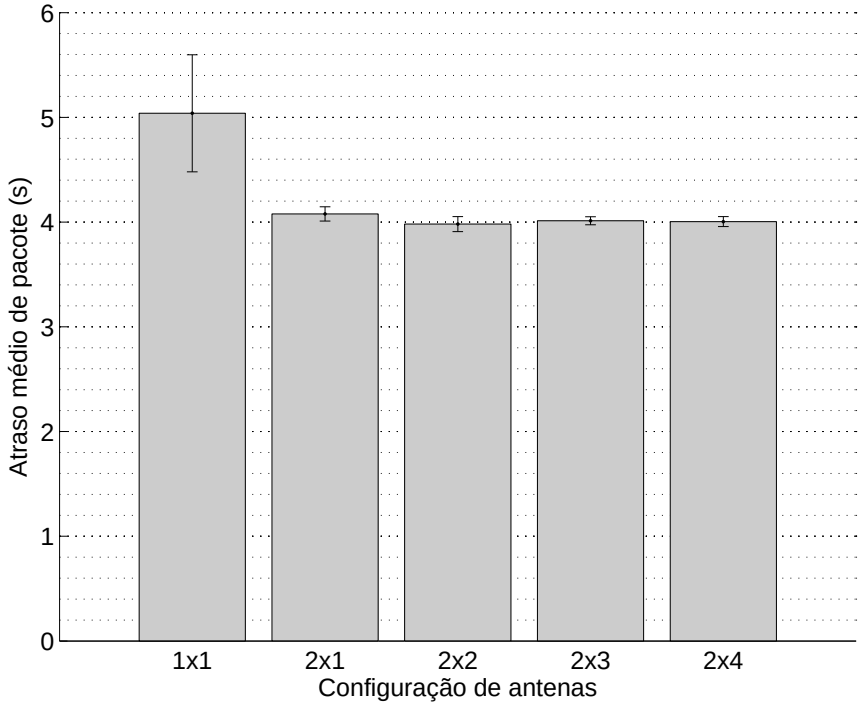


Figura 5.4: Atraso médio de pacote da rede habilitada apenas com o esquema de Alamouti sob desvanecimento Rayleigh ($K = 0$) para diferentes configurações de antenas.

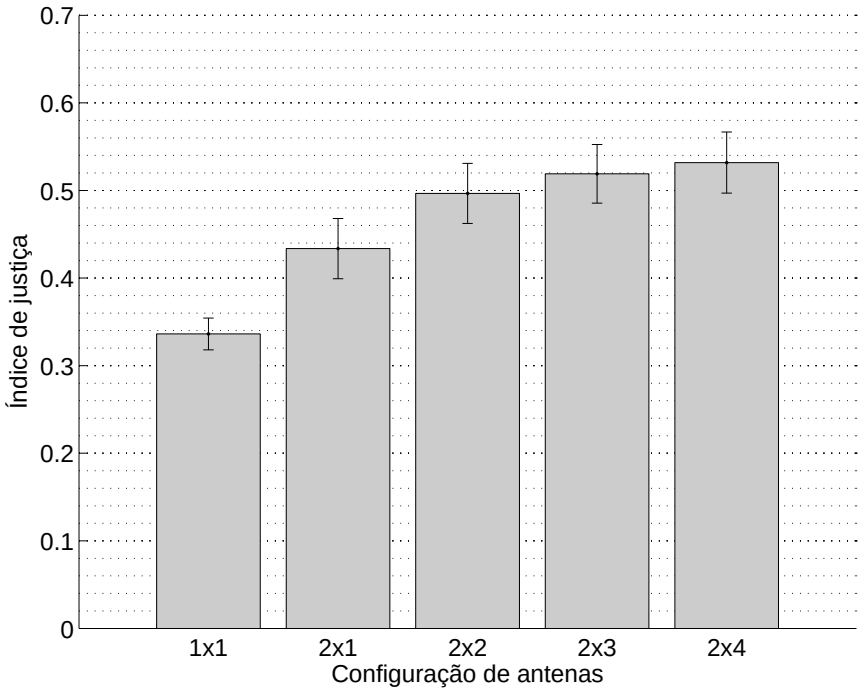


Figura 5.5: Índice de justiça da rede habilitada apenas com o esquema de Alamouti sob desvanecimento Rayleigh ($K = 0$) para diferentes configurações de antenas.

5.7 Redes Ad Hoc MIMO via V-BLAST

Nesta Seção, apresentamos os resultados de simulação computacional de redes *ad hoc* MIMO quando habilitadas para utilizar apenas o sistema V-BLAST na camada física, e sob as modificações

correspondentes no protocolo MAC e no procedimento de avaliação de ocupação de canal.

Os resultados de simulação são calculados para nove topologias (divididas igualmente entre os três diferentes níveis de contenção, como as apresentadas na Figura 5.1), rodadas para diferentes sementes de simulação, com diferentes configurações de antena ($M \times N$) usando o sistema V-BLAST. Para comparação, mostramos a vazão média da rede para a configuração SISO IEEE 802.11b. As barras de erro representam a margem de erro com nível de confiança de 95%.

Na Figura 5.6, é apresentado o desempenho da vazão média da rede usando o sistema V-BLAST. De acordo com os resultados da capacidade de sistemas MIMO [14], é esperado um aumento linear na taxa de dados quando mais antenas são adicionadas para transmissão e/ou recepção. Entretanto, pelos nossos resultados, este aumento linear não é observado em todos os casos. Isto é melhor explicado através do conceito de *diversidade de recepção*. Loyka e Gagnon [2] mostraram que o grau de diversidade (*diversity order*) do V-BLAST no i-ésimo passo de processamento é $N - M + i$ (sob desvanecimento Rayleigh e ordenamento ótimo). Tal resultado pode ser verificado na Figura 2.10, que reflete o ganho de diversidade médio sobre todos os passos de processamento. Assim, para um número fixo M de antenas transmissoras, o efeito de aumentar o número N de antenas receptoras pode ser observado no gráfico da Figura 2.10 como uma mudança na inclinação das curvas, isto é, a BER diminui para uma dada SNR ao aumentar o número de antenas receptoras (observe os casos 2×2 e 2×3).

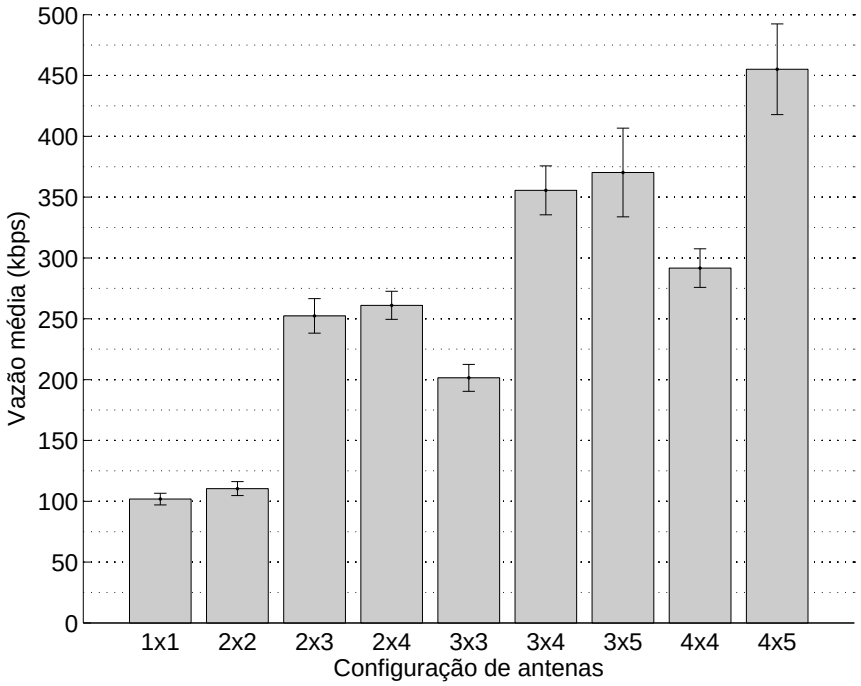


Figura 5.6: Vazão média da rede habilitada apenas com o sistema V-BLAST sob desvanecimento Rayleigh ($K = 0$) para diferentes configurações de antenas.

Como é bem conhecido, o efeito da diversidade é melhorar a robustez da transmissão, de modo que é esperado uma diminuição no tempo gasto com tentativas mal sucedidas de transmissão pelo protocolo IEEE 802.11. Por isso, considerando apenas os casos em que $M < N$ (ex.: 2×3, 3×4 e 4×5), de fato observa-se um crescimento quase linear na vazão média em relação ao caso SISO.

Por exemplo, a vazão média para o caso 2×3 (266,725 kbps) é mais que o dobro (2,6207 vezes) do caso SISO (101.776 kbps), enquanto a vazão média do caso 3×4 (355,576 kbps) é 3,4937 vezes o caso SISO, e para o caso 4×5 (455,112 kbps), a vazão média é 4,4717 vezes a vazão média do caso SISO.

Apesar dos ganhos de vazão obtidos serem significativos quando $M < N$, não há melhorias significativas se mais antenas receptoras são adicionadas enquanto mantém-se constante o número de antenas transmissoras (note os casos 2×3 e 2×4, e os casos 3×4 e 3×5 na Figura 5.6). Isto é devido ao fato de que não importa quão pequena a BER se torne aumentando o número N de antenas receptoras (para dada SNR), não há mais dados multiplexados no transmissor (desde que M seja mantido constante). Consequentemente, como apenas é aumentado o número de antenas receptoras, espera-se uma saturação na vazão.

Por outro lado, é interessante observar o compromisso entre diversidade e multiplexagem quando o número de antenas receptoras é mantida fixa enquanto o número de antenas transmissoras varia. Entretanto, a partir dos resultados obtidos até então, pode-se concluir que, para os cenários investigados, em alguns momentos pode ser mais vantajoso garantir transmissões robustas mas com menos dados multiplexados (caso 3×4) do que adicionar mais uma antena para acelerar as transmissões e liberar o canal mais cedo (caso 4×4).

A Figura 5.7 apresenta o desempenho do atraso médio de pacote da rede usando o sistema V-BLAST. Como é esperado, o comportamento do atraso médio de pacote é inverso ao comportamento da vazão. A diminuição do atraso em função do número de antenas é explicada principalmente pelo fato da diminuição do tempo de transmissão causado pela multiplexagem do quadro.

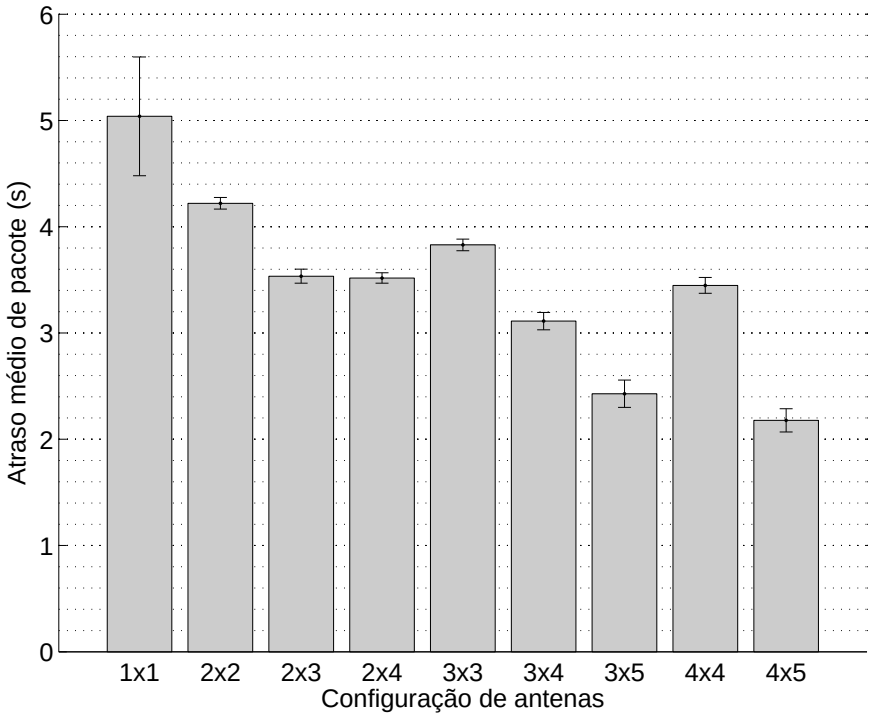


Figura 5.7: Atraso médio de pacote da rede habilitada apenas com o sistema V-BLAST sob desvanecimento Rayleigh ($K = 0$) para diferentes configurações de antenas.

A Figura 5.8 apresenta o desempenho do índice de justiça da rede usando o sistema V-BLAST. Podemos notar que o comportamento deste índice é semelhante ao comportamento da vazão. Uma rede tem uma partilha do canal mais justa quando os nós desocupam o canal mais rapidamente, liberando-o mais cedo para a transmissão de nós vizinhos. O índice de justiça, quando não é utilizada a multiplexagem, é limitado pelo controle de acesso ao meio, como podemos observar no índice do esquema de Alamouti na Figura 5.5. A Figura 5.9 permite fazer um comparativo entre o comportamento dos índices de justiça do esquema de Alamouti e do sistema V-BLAST. Podemos notar um ganho devido à multiplexagem, além da limitação do controle de acesso ao meio a partir da configuração 2×3. A Figura 5.10, por sua vez, permite fazer um comparativo entre o comportamento da vazão do esquema de Alamouti e do sistema V-BLAST. Podemos notar uma certa semelhança entre o comportamento de vazão e do índice de justiça para essas duas técnicas.

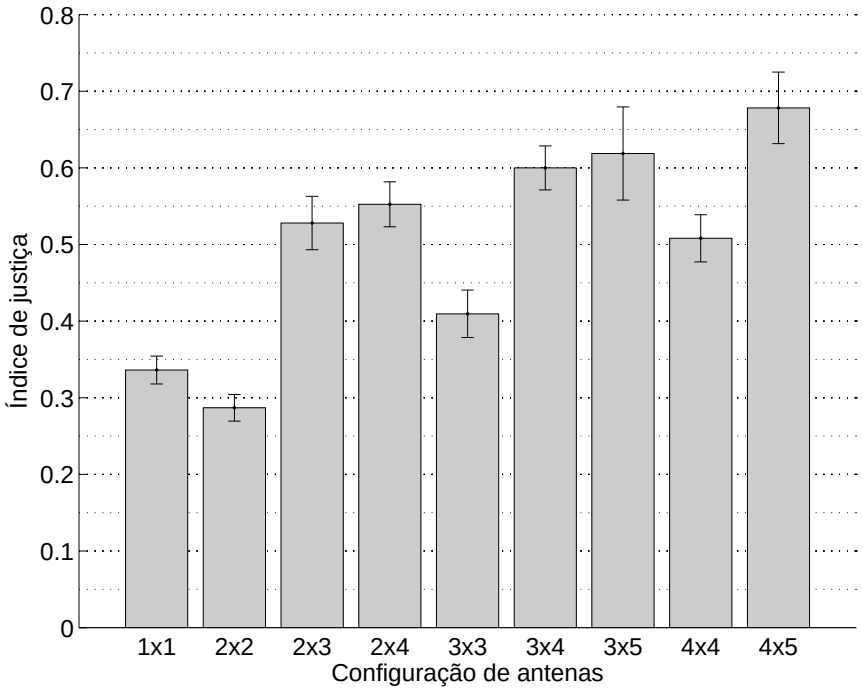


Figura 5.8: Índice de justiça da rede habilitada apenas com o sistema V-BLAST sob desvanecimento Rayleigh ($K = 0$) para diferentes configurações de antenas.

5.8 Redes Ad Hoc MIMO Híbridas com MAC Adaptativo

A avaliação de desempenho de redes *ad hoc* habilitadas com as técnicas MIMO isoladas permitiu observar ganhos nas variadas situações em que o canal se encontra. Na Seção 5.6, mostramos que as redes habilitadas para utilizar apenas o esquema de Alamouti apresentam melhores resultados de desempenho (maiores ganhos relativos) quando o canal está com alto nível de desvanecimento. Este melhor desempenho é justificado pelo aumento da SINR do quadro recebido e, consequentemente, pela diminuição da BER providos pelo uso da proposta de Alamouti, como demonstrado na Seção 2.4.1. Por isto, esta é uma ótima técnica para ser utilizada quando a SINR é pequena, como nas situações de transmissões a distâncias próximas à fronteira de raio de alcance de transmissão ou em regiões com alta densidade de nós e elevada susceptibilidade a interferências. Na Seção 5.7,

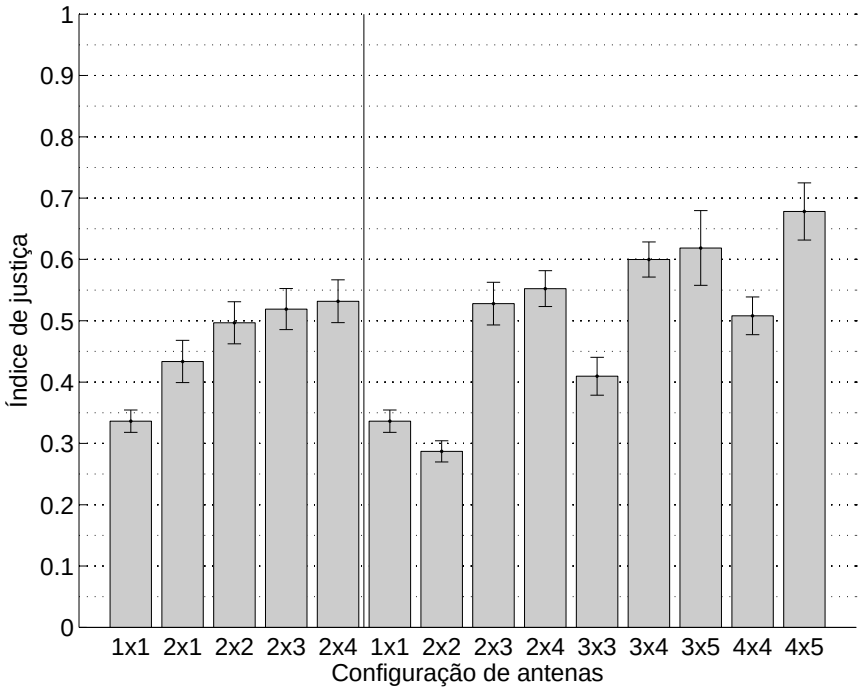


Figura 5.9: Comparação do índice de justiça entre as técnicas de transmissão. À esquerda da linha vertical escura, encontra-se o desempenho do esquema de Alamouti e à direita, o sistema V-BLAST.

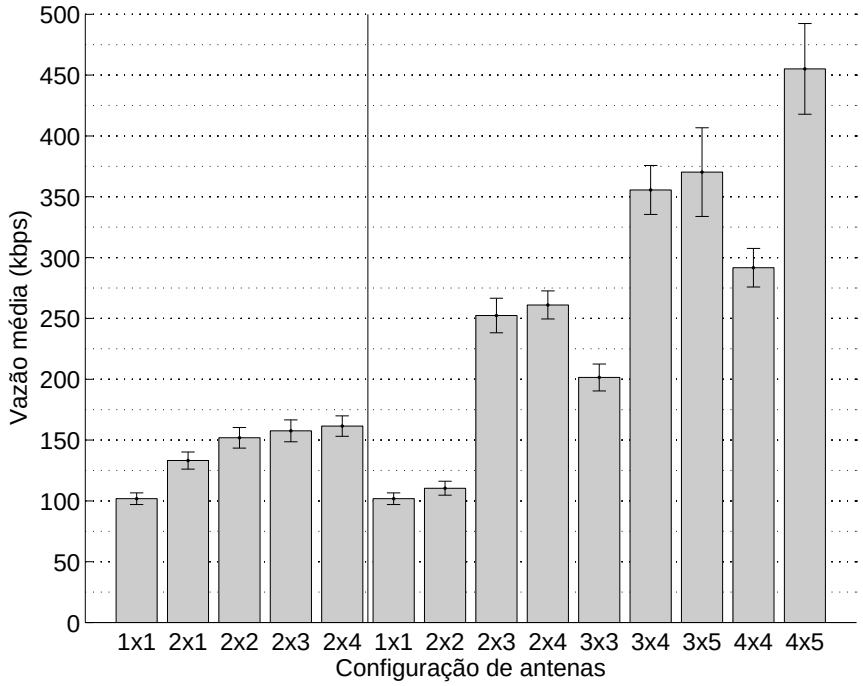


Figura 5.10: Comparação da vazão média entre as técnicas de transmissão. À esquerda da linha vertical escura, encontra-se o desempenho do esquema de Alamouti e à direita, o sistema V-BLAST.

mostramos que as redes habilitadas para utilizar apenas o sistema V-BLAST apresentam melhores resultados de desempenho ao utilizar uma antena transmissora a menos do que o número de antenas receptoras. Este melhor desempenho é devido a menor BER apresentada, fixada a SINR,

em detrimento a menor quantidade de dados multiplexados. Entretanto, se considerarmos o caso em que a SINR é suficientemente grande a ponto de apresentar uma taxa de descarte do quadro praticamente nula, é possível supor que os resultados de desempenho ao utilizar o mesmo número de antenas transmissoras e receptoras serão melhores, devido ao aumento da quantidade de dados multiplexados e ao comportamento descrescente da curva da BER, apresentada na Figura 2.10, em relação a SINR.

Desta forma, o compromisso observado entre diversidade (de transmissão e de recepção) e multiplexagem pode ser medido a partir da relação SINR versus BER de cada técnica de transmissão. Assim, podemos separar cada técnica em uma região de valor de SINR que propicia melhor ganho. É neste contexto que se insere a proposta de protocolo apresentada neste trabalho. Ao adotar o uso deste protocolo, os nós poderão escolher qual compromisso assumir (transmissão robusta ou aumento de vazão) em função da SINR recebida. Todavia, a SINR é uma medida particular de cada recepção de símbolo ou quadro. Por isso, cada par de nós deve, individualmente, avaliar qual a melhor técnica a ser utilizada durante o estabelecimento da conexão, tendo como referência a SINR recebida no quadro RTS. É importante salientar que os quadros de controle RTS e CTS, bem como o ACK, são transmitidos com a configuração SISO.

Para que o nó seja capaz de alternar entre as técnicas de transmissão, de acordo com a SINR percebida na recepção, fez-se necessário definir os limiares de comutação. Para tanto, definimos dois limiares: um inferior, o $snrMin$, e um superior, o $snrMax$. O limiar inferior $snrMin$ servirá de base para fazer a comutação entre as técnicas que atingem melhores taxas de erro de bit sob valores baixos de SINR. No caso, propõe-se a comutação entre o esquema de Alamouti e o sistema V-BLAST com diversidade de recepção. O limiar superior $snrMax$ controlará a escolha entre as técnicas que operam com alta taxa de transmissão para valores altos de SINR. No caso, a comutação acontece entre os sistemas V-BLAST com e sem diversidade. A partir do estudo do impacto na variação destes dois limiares foi possível definir valores de limiares próximos ao adequado de comutação (caso se comprove vantajoso, como iremos demonstrar).

Nosso estudo tem como referência o desempenho do sistema V-BLAST com um grau de diversidade de recepção (ou seja, quando o número de antenas transmissoras M possui uma antena a menos que o número de antenas receptoras N, ou seja, $M = N - 1$). Isto porque, como discutido na Seção 5.5, esta configuração é utilizada na região intermediária entre os limiares de comutação $snrMin$ e $snrMax$ e, portanto, demonstraremos como é possível obter maiores ganhos utilizando outras configurações MIMO quando o valor de SINR está fora deste intervalo.

Neste experimento, analisamos o desempenho do protocolo híbrido funcionando com três modos de operação diferentes. No primeiro modo, nomeado HYB-A, verificamos a vazão média final da rede para diferentes valores do limiar $snrMin$, de forma a ter a comutação apenas entre as técnicas de Alamouti e V-BLAST ($M = N-1$). Assim, mantendo o valor de $snrMax$ fixo e suficientemente grande, garantimos que a comutação se dá apenas em função do valor de SINR comparado ao limiar $snrMin$. No segundo modo, nomeado HYB-B, verificamos a vazão média da rede para diferentes valores do limiar $snrMax$, mantendo o limiar $snrMin$ fixo e com valor suficientemente baixo de forma a ter a comutação apenas entre as técnicas V-BLAST ($M = N - 1$) e V-BLAST ($M = N$). No terceiro modo, nomeado HYB-C, fixamos ambos os limiares ($snrMin, snrMax$), em valores

dedicidos com base em estudos prévios, para avaliação da comutação entre essas três técnicas.

Apresentamos o estudo do desempenho dos modos HYB-A, HYB-B e HYB-C nas Seções 5.8.1, 5.8.2 e 5.8.3, respectivamente.

5.8.1 Estudo da Variação do Limiar Inferior de Comutação

Nesta seção, apresentamos o estudo do desempenho do primeiro modo de operação (HYB-A) do protocolo híbrido proposto. Este estudo foi realizado variando-se o valor do limiar inferior $snrMin$, responsável pela comutação entre as técnicas de Alamouti e V-BLAST($M = N - 1$). A Figura 5.11 ilustra a vazão média da rede calculada sobre nove topologias (divididas igualmente entre os três diferentes níveis de contenção, como as apresentadas na Figura 5.1), rodadas para diferentes sementes de simulação, com diferentes valores do $snrMin$ para dispositivos de rede habilitados com 3 e 4 antenas disponíveis para transmissão. Além disso, é feita uma comparação com a vazão média do sistema V-BLAST ($M = N - 1$) puro, indicada por linhas horizontais nos gráficos. As barras de erro representam a margem de erro com nível de confiança de 95%.

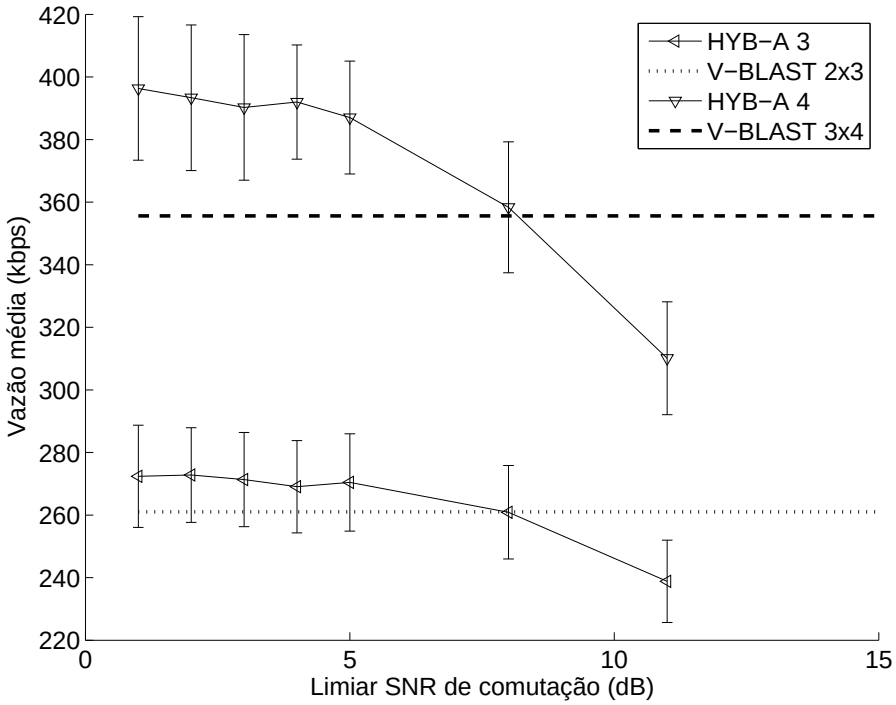


Figura 5.11: Vazão média da rede operando no modo HYB-A em função do limiar inferior de comutação com 3 e 4 antenas disponíveis para transmissão em comparação com a vazão média do sistema V-BLAST($M = N - 1$) puro, indicada pelas linhas horizontais.

Observamos que a vazão do modo de operação HYB-A diminui à medida que aumentamos o valor do $snrMin$, tanto quando se tem 3 antenas disponíveis para transmissão, como quando se tem 4 antenas disponíveis. Isto é devido ao fato de, ao aumentarmos o valor do $snrMin$, mais transmissões utilizando o esquema de Alamouti acontecem na rede. Como pode ser observado na Figura 5.10, a vazão de transmissões utilizando o esquema de Alamouti é inferior à de transmissões que utilizam o sistema V-BLAST com diversidade de recepção.

Na Figura 5.11, para valores inferiores ao limiar $snrMin = 8$ dB, observa-se um ganho significativo do modo HYB-A em relação ao sistema V-BLAST($M = N - 1$) puro. Podemos observar também que, para valores do limiar $snrMin < 5$ dB, os ganhos em vazão parecem tender à uma saturação. Isto se deve ao fato de as transmissões com SNRs menores que 8 dB estarem com uma alta taxa de descarte de quadros quando utilizam apenas o V-BLAST. Entretanto, ao utilizar o esquema de Alamouti nestas mesmas transmissões, o descarte de quadros diminui, aumentando a vazão média da rede. Em seu melhor desempenho, este modo de operação alcançou um ganho máximo em relação ao sistema V-BLAST($M = N-1$) de 8,11% considerando 3 antenas disponíveis para transmissão, e de 10,24% considerando 4 antenas disponíveis.

As Figuras 5.12 e 5.13 ilustram os desempenhos do atraso médio de pacote e do índice de justiça da rede. É fácil notar que o desempenho do atraso médio de pacote do modo HYB-A se mostrou melhor que o desempenho do sistema V-BLAST($M = N - 1$) ao observarmos que ambas as curvas estão abaixo das linhas horizontais. Nos melhores casos, o modo de operação HYB-A reduziu o atraso em 56,69% considerando 3 antenas disponíveis para transmissão em relação à configuração V-BLAST (2×3) puro, e a 49,43% considerando 4 antenas disponíveis em relação à configuração V-BLAST (3×4) puro. O comportamento do índice de justiça se assemelha ao comportamento da vazão. Entretanto, considerando 3 antenas disponíveis para transmissão, o HYB-A não apresentou ganhos em nenhum caso quando comparada ao sistema V-BLAST(2×3), apenas igualando-se, no melhor caso. Isto acontece devido à limitação do controle de acesso ao meio sobre o canal compartilhado, como já discutido nas seções anteriores.

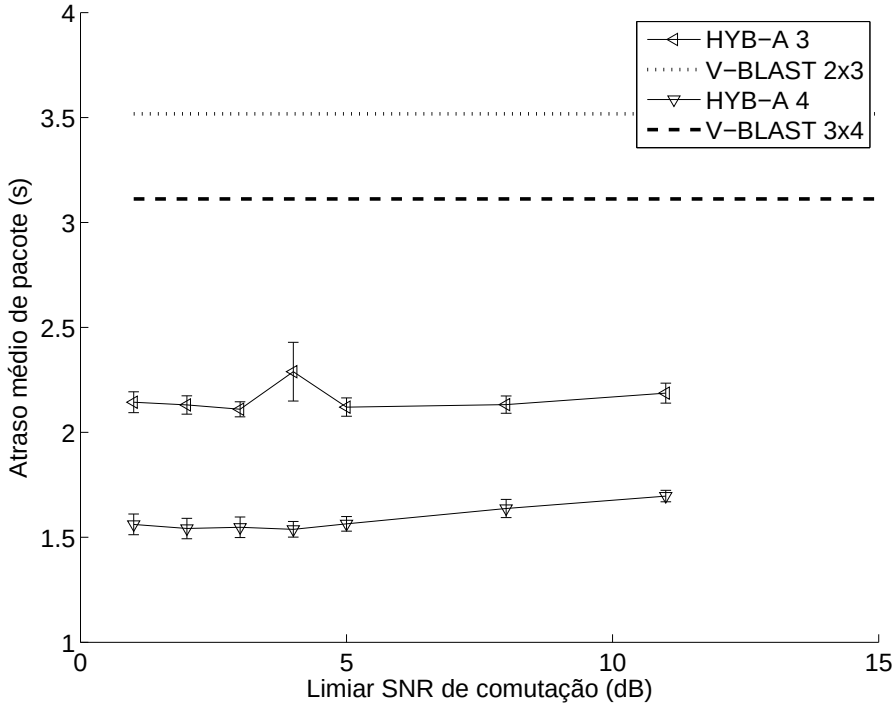


Figura 5.12: Atraso médio de pacote da rede operando em modo HYB-A em função do limiar inferior de comutação com 3 e 4 antenas disponíveis para transmissão em comparação com a vazão média do sistema V-BLAST($M = N - 1$).

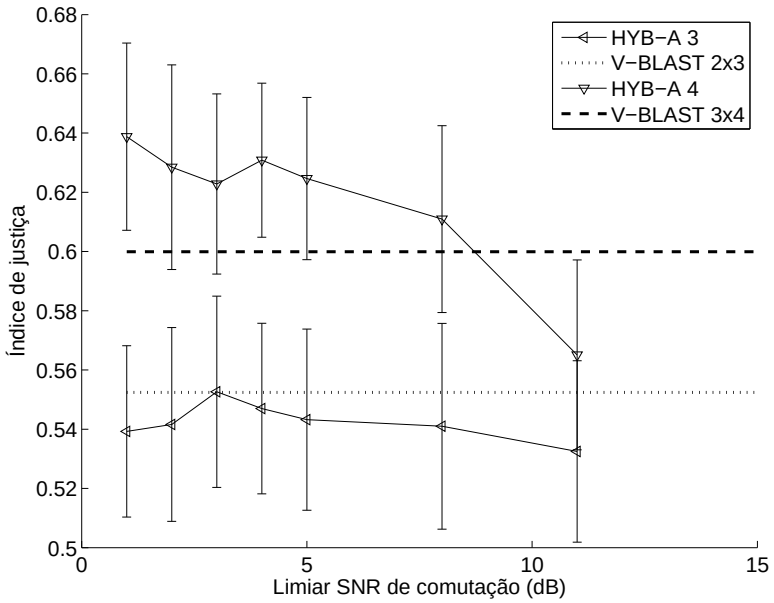


Figura 5.13: Índice de justiça da rede operando em modo HYB-A em função do limiar inferior de comutação com 3 e 4 antenas disponíveis para transmissão em comparação com a vazão média do sistema V-BLAST($M = N - 1$).

5.8.2 Estudo da Variação do Limiar Superior de Comutação

Nesta seção, apresentamos o estudo do desempenho do segundo modo de operação (HYB-B) do protocolo híbrido proposto. Este estudo foi realizado variando-se o valor do limiar superior $snrMax$, responsável pela comutação entre as técnicas V-BLAST($M = N - 1$) e V-BLAST($M = N$). A Figura 5.14 ilustra o vazão média da rede calculada sobre nove topologias (divididas igualmente entre os três diferentes níveis de contenção, como as apresentadas na Figura 5.1), rodadas para diferentes sementes de simulação, com diferentes valores do $snrMax$, para dispositivos de rede habilitados com 3 e 4 antenas disponíveis para transmissão. Além disso, é feita uma comparação com a vazão média do sistema V-BLAST ($M = N-1$) puro, indicada por linhas horizontais nos gráficos. As barras de erro representam a margem de erro com nível de confiança de 95%.

Observamos que a vazão diminui à medida que diminuímos o valor do $snrMax$, tanto quando se tem 3 antenas disponíveis para transmissão, como quando se tem 4 antenas disponíveis. Esse comportamento é explicado pela maior ocorrência de transmissões utilizando o modo de multiplexagem com todas as antenas disponíveis, que resulta em mais quadros descartados devido à maior taxa de erro de bit desta técnica de transmissão. O ganho em relação à vazão média da rede utilizando apenas o sistema V-BLAST com diversidade puro, indicada pelas linhas horizontais, é observado a partir do limiar $snrMax = 14$ dB e atinge o máximo quando $snrMax$ está aproximadamente entre 20 e 23 dB. Com valores de SNR acima deste limiar, a taxa de erro de bit do sistema V-BLAST sem diversidade se torna tão pequena quanto à taxa de erro do sistema com diversidade, e os quadros não são descartados. Portanto, ao utilizar mais antenas para transmissão nesta região de SNR, os quadros são recebidos com sucesso e aumenta a vazão média da rede. Em seu melhor desempenho, este modo de operação alcançou um ganho máximo em relação ao

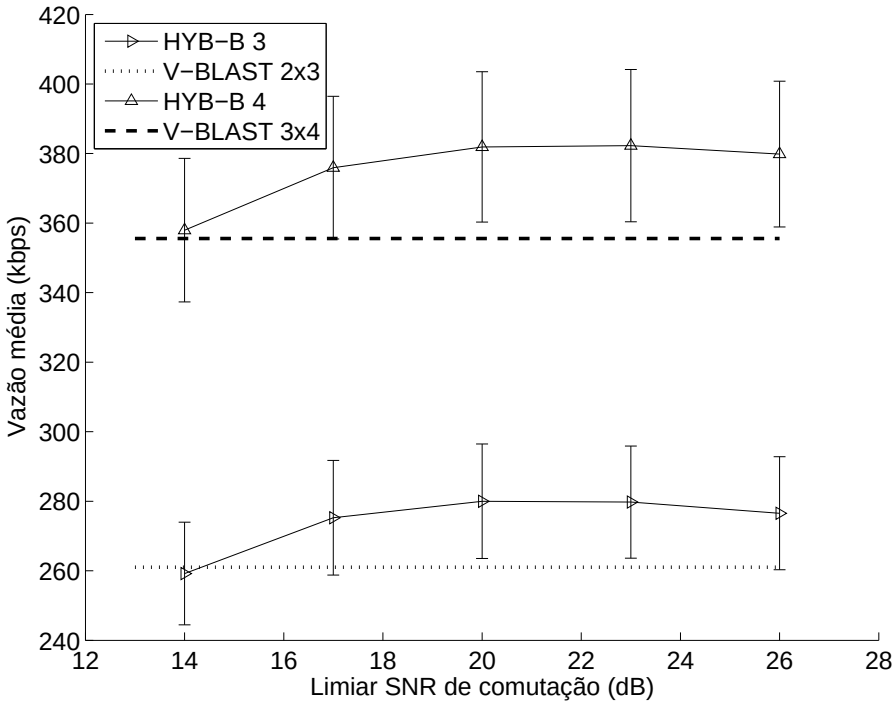


Figura 5.14: Vazão média da rede operando no modo HYB-B em função do limiar superior de comutação com 3 e 4 antenas disponíveis para transmissão em comparação com a vazão média do sistema V-BLAST($M = N - 1$) puro, indicada pelas linhas horizontais.

sistema V-BLAST($M = N - 1$) de 10,96% considerando 3 antenas disponíveis para transmissão, e de 7,51% considerando 4 antenas disponíveis.

As Figuras 5.15 e 5.16 apresentam os desempenhos do atraso médio de pacote e do índice de justiça da rede. Observamos que o desempenho do atraso médio da rede operando no modo HYB-B também se mostrou melhor que o desempenho do sistema V-BLAST($M = N-1$). Em seu melhor desempenho, o atraso médio obteve redução de 59,61% considerando 3 antenas disponíveis para transmissão, e de 77,91% considerando 4 antenas disponíveis. Quanto ao índice de justiça, este mostrou uma suave diminuição ao aumentar o limiar de comutação, mas nada muito significativo. Considerando 3 antenas disponíveis para transmissão, o modo HYB-B não apresentou muita diferença em relação ao sistema V-BLAST (2×3) puro. Entretanto, considerando 4 antenas disponíveis, este apresentou um certo ganho em relação ao sistema V-BLAST (3×4) puro. Tais ganhos podem ser devido ao ganho de multiplexagem se sobressaindo à limitação do controle de acesso ao meio.

5.8.3 Avaliação do Protocolo MAC Híbrido Adaptativo

Nesta Seção, avaliamos o desempenho do terceiro modo de operação (HYB-C) do protocolo híbrido proposto. Baseado nos resultados com os modos HYB-A e HYB-B discutidos nas Seções 5.8.1 e 5.8.2, definimos os limiares $snrMin$ e $snrMax$ a serem usados nas simulações do modo HYB-C de modo a otimizar o desempenho da vazão. Escolhemos $snrMin = 5$ dB e $snrMax = 23$ dB por serem os limiares a partir dos quais as *curvas de vazão média* começaram a ficar constantes nos modos HYB-A e HYB-B em ambos os casos de 3 e 4 antenas nos dispositivos. Os resultados do

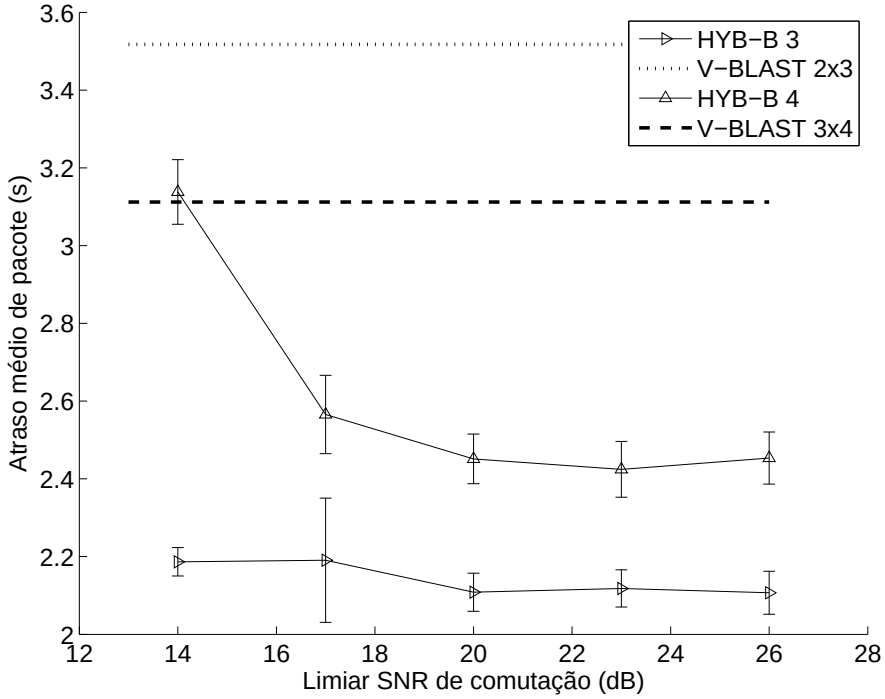


Figura 5.15: Atraso médio de pacote da rede operando em modo HYB-AB em função do limiar superior de comutação com 3 e 4 antenas disponíveis para transmissão em comparação com a vazão média do sistema V-BLAST($M = N - 1$).

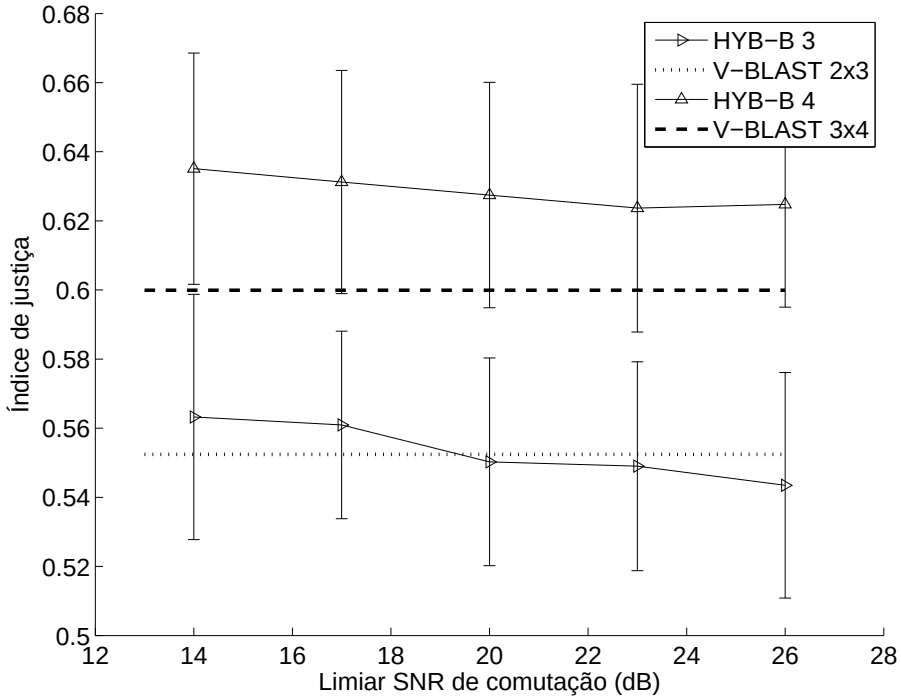


Figura 5.16: Índice de justiça da rede operando em modo HYB-B em função do limiar superior de comutação com 3 e 4 antenas disponíveis para transmissão em comparação com a vazão média do sistema V-BLAST($M = N - 1$).

desempenho da vazão média da rede do modo HYB-C são apresentados na Figura 5.17 juntamente com as curvas de desempenho dos modos HYB-A e HYB-B. É esperado que o desempenho do modo

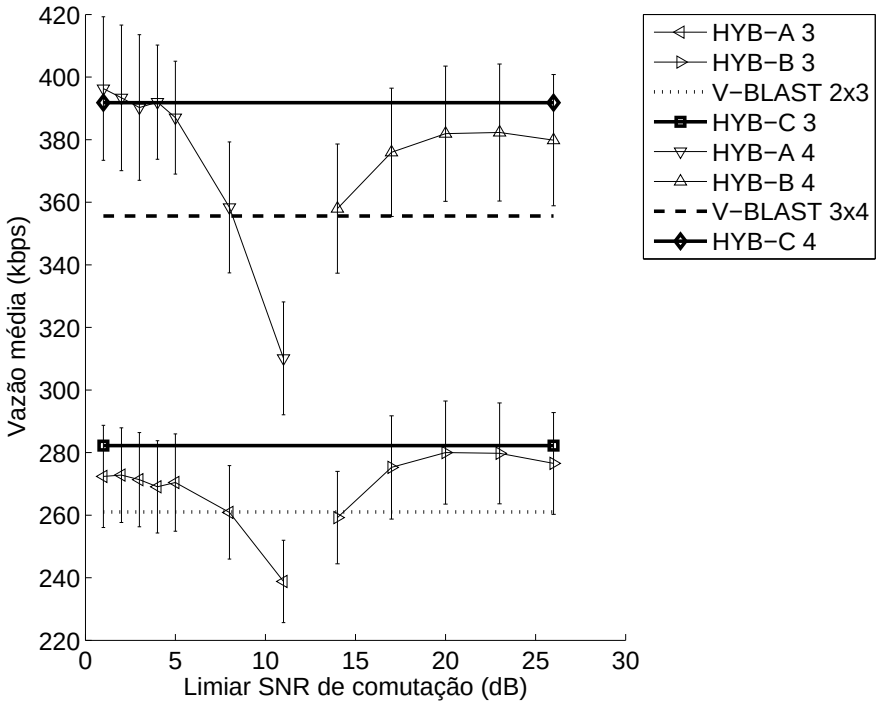


Figura 5.17: Vazão média da rede operando no modo HYB-C (linha em negrito) em comparação com os modos HYB-A e HYB-B e com o sistema V-BLAST($M = N - 1$) para as configurações de 3 e 4 antenas.

HYB-C seja melhor ou igual ao desempenho dos outros modos por estar utilizando as configurações com melhor desempenho dentre as três técnicas (Alamouti, VBLAST($M = N-1$) e VBLAST($M = N$)). De fato, o gráfico mostra que a vazão média da rede no modo HYB-C (linhas em negrito) estão acima das curvas dos modos HYB-A e HYB-B em ambos os casos de 3 e 4 antenas nos dispositivos. Obteve-se um ganho de vazão de 11,84% ao utilizar o modo HYB-C com três antenas (282,217 kbps) em relação ao sistema V-BLAST(2×3) (252,330 kbps). Da mesma forma, obteve-se um ganho de vazão de 10,19% ao utilizar o modo HYB-C com 4 antenas (391,792 kbps) em relação ao sistema V-BLAST(3×4) (355,576 kbps).

As Figuras 5.18, 5.19 e 5.20 exibem o desempenho da rede operando no modo HYB-C em comparação ao desempenho da rede utilizando a configuração SISO e às outras técnicas de transmissão MIMO. A utilização do protocolo MIMO híbrido e MAC adaptativo tornou os ganhos do sistema MIMO quase lineares, em relação ao caso SISO, ao aumentar o número de antenas, como é esperado em teoria [14] (mesmo sob condições de contenção e limitações da camada MAC). Com os dispositivos da rede dispondo de 3 antenas, a vazão é 2,77 vezes a vazão da configuração SISO (101,776 kbps) contra 2,48 vezes do sistema V-BLAST. Com os dispositivos dispondo de 4 antenas, a vazão é 3,85 vezes contra 3,49 do sistema V-BLAST. Isto evidencia que uma rede utilizando este protocolo usufrui mais da capacidade total do sistema MIMO do que uma rede utilizando apenas uma técnica de transmissão. Quanto ao atraso médio de pacote, este foi reduzido em 34,93%, em relação à configuração SISO, no caso de 3 antenas e em 30,79%, no caso de 4 antenas, apresentando uma redução mais drástica em relação às outras configurações. Observando agora o índice de justiça da rede, este se apresentou maior que o índice da rede utilizando as outras técnicas.

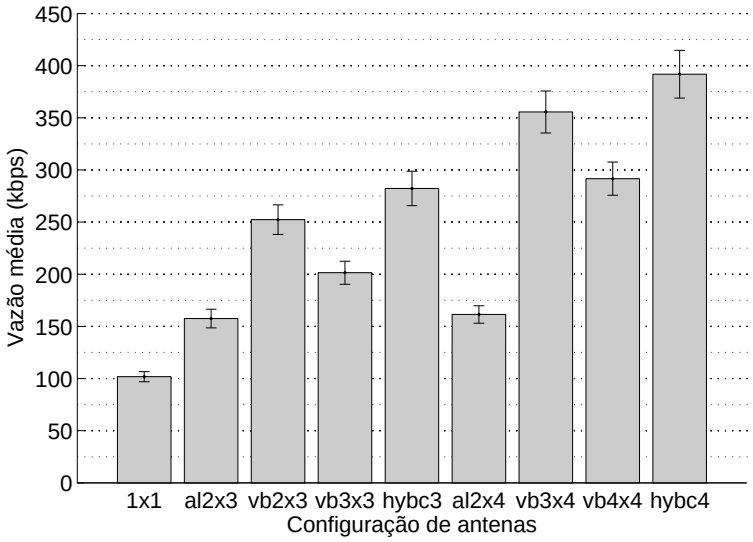


Figura 5.18: Vazão média da rede comparando os diferentes modos de operação. O sufixo "al" indica o esquema de Alamouti, "vb", o sistema V-BLAST e "hybc", o modo de operação HYB-C.

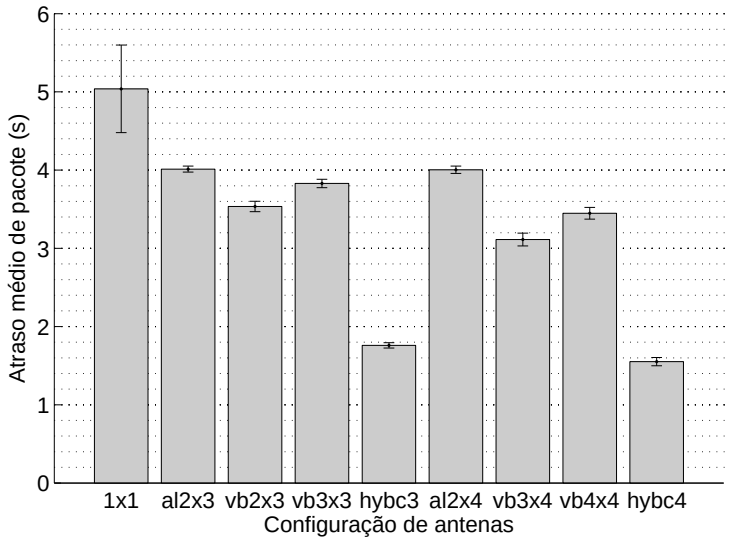


Figura 5.19: Atraso médio da rede comparando os diferentes modos de operação. O sufixo "al" indica o esquema de Alamouti, "vb", o sistema V-BLAST e "hybc", o modo de operação HYB-C.

Em todas as avaliações, o desempenho da rede utilizando o protocolo híbrido adaptativo proposto foi melhor que o desempenho das outras redes utilizando a mesma técnica MIMO em todos os nós. Os ganhos foram obtidos sem aumentar a carga dos mecanismos de controle da camada MAC da rede, já que o protocolo utiliza os mecanismos e informações já existentes para operar. Vale lembrar que estes resultados são uma média do desempenho de redes com três tipos de níveis de contenção e densidade espacial. E mesmo assim, foram possíveis ganhos positivos no atraso médio de pacote, no índice de justiça e, principalmente, na vazão média da rede. Em topologias com menores níveis de contenção, é esperado um ganho maior devido a menor limitação de conteção da camada MAC, e com níveis maiores, ganhos menores.

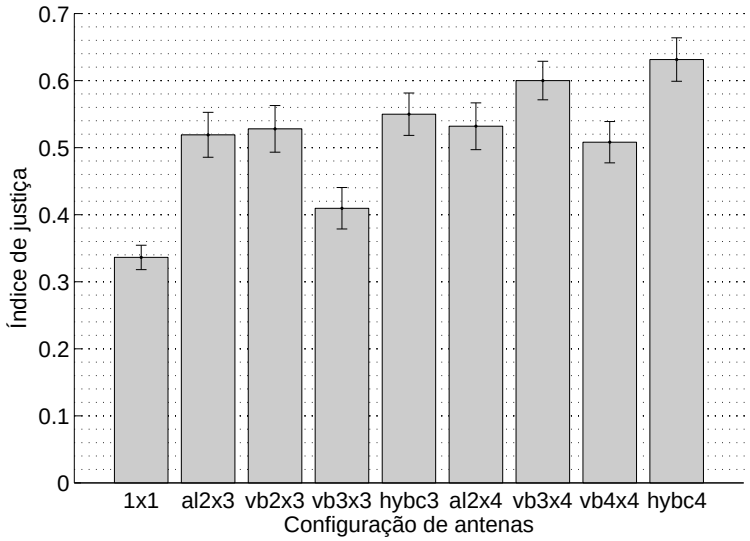


Figura 5.20: Índice de justiça da rede comparando os diferentes modos de operação. O sufixo "al" indica o esquema de Alamouti, "vb", o sistema V-BLAST e "hybc", o modo de operação HYB-C.

Neste Capítulo foi avaliado o desempenho das redes utilizando sistemas MIMO implementados com nossas modificações no simulador NS-3. Avaliamos as técnicas de Alamouti e V-BLAST e observamos seus ganhos em relação à configuração SISO. Discutimos a proposta do protocolo híbrido adaptativo e a necessidade de definir seus parâmetros. Em seguida, estudamos os limiares de comutação que otimizavam o desempenho do protocolo proposto. Então, definimos os limiares e avaliamos o desempenho da rede utilizando o protocolo híbrido adaptativo e seus resultados demonstraram desempenhos superiores.

Capítulo 6

Conclusões

Neste trabalho, foi apresentado um estudo sobre o desempenho de redes sem fio *ad hoc* habilitadas com múltiplas antenas. Descrevemos a implementação realizada no simulador NS-3 do sistema MIMO e de suas técnicas de transmissão, além da questão do mecanismo de detecção de canal livre ao utilizar múltiplas antenas. Então, avaliamos as técnicas de transmissão MIMO pelo seu comportamento em nível de rede a partir de abstrações da camada física feitas a partir de abordagens analíticas e expressões matemáticas do desempenho em nível de enlace. Com base nesta avaliação, propusemos um sistema híbrido de técnicas de transmissão MIMO com o protocolo MAC adaptativo que comuta a técnica para a transmissão do quadro de dados de acordo com o nível de SINR percebido no receptor ao receber o quadro RTS. Esta proposta buscou ser uma solução para aproveitar o melhor de cada técnica para obter desempenhos melhores do que os obtidos com o uso individual de cada técnica em toda a rede. Para demonstrar tal fato, fizemos uma análise a respeito dos limiares ótimos de comutação para serem definidos na avaliação do protocolo a fim de otimizar seu desempenho. Finalmente, avaliamos o desempenho do protocolo e demonstramos seus ganhos em relação ao melhor desempenho do sistema V-BLAST.

Nesta proposta, foi discutido o compromisso que é assumido ao se utilizar múltiplas antenas para comunicação sem fio. As múltiplas antenas podem ser utilizadas para prover ganhos de diversidade de transmissão e/ou recepção ou para prover ganhos de multiplexagem ao enviar fluxos paralelos de dados simultaneamente. O sistema híbrido, cujas técnicas habilitadas já provêem estes ganhos ao serem utilizadas individualmente na rede por todos os nós, pode agregar estes ganhos e melhorar ainda mais o desempenho da rede, desde que haja um mecanismo que se adapte à qualidade do canal. Este mecanismo de adaptação é um fator muito importante pois cada técnica de transmissão mostrou obter maiores ganhos quando operava em determinado nível de qualidade do canal.

Nos resultados apresentados da avaliação do sistema híbrido/adaptativo, os ganhos obtidos conseguiram se aproximar mais do ganho linear esperado ao aumentar o número de antenas dos dispositivos [14]. Sob as limitações do controle de acesso ao meio da camada MAC, o sistema V-BLAST conseguiu apresentar um ganho quase linear no desempenho da vazão média da rede, quando os dispositivos utilizavam configurações de antenas cujas comunicações apresentavam menores taxas de erro, porém esta análise considerava o ganho apenas em relação ao número de antenas transmissoras, e não o número total de antenas (transmissoras e receptoras) do dispositivo, portanto, sendo

apenas o ganho de multiplexagem de transmissão. Contudo, o sistema proposto obteve ganhos quase lineares considerando o número de antenas disponíveis nos dispositivos da rede (no caso, três e quatro antenas por nó).

Futuramente, deve-se analisar se o nível de complexidade do sistema proposto justifica os ganhos apresentados. Para redes *ad hoc* com dispositivos móveis com fontes limitadas de energia (uma rede de pequenos sensores em uma área remota, por exemplo), uma carga extra de processamento computacional pode consumir mais energia e diminuir a vida útil do dispositivo, podendo prejudicar a rede de certa maneira. Portanto, um sistema híbrido/adaptativo completo deve assumir, além dos compromissos de diminuição da taxa de erro e de aumento da taxa de dados, um compromisso de complexidade e de eficiência energética do sistema.

Com a implementação do sistema MIMO em um simulador de rede, surge a oportunidade de se estudar outros protocolos que utilizam as múltiplas antenas para melhorar o desempenho. Um exemplo, é o protocolo de múltipla recepção de pacote (MPR, do inglês *Multiple Packet Reception*). Este protocolo tem com fundamentação o protocolo de controle de acesso ao meio onde o acesso é iniciado pelo receptor dos dados (RIMA, do inglês *Receiver Initiated Medium Access*). Este tipo de protocolo demonstra algumas vantagens em relação ao processo tradicional onde o acesso é iniciado pelo transmissor, podendo ser mais eficiente [34]. A partir deste tipo controle de acesso ao meio, usando as múltiplas antenas e a sincronização dos vários transmissores com o receptor (possibilitado pelo protocolo RIMA, no recebimento do quadro de requisição de dados - *Ready-To-Receive*), o protocolo permite a recepção de múltiplos pacotes de transmissores diferentes simultaneamente, com a utilização do receptor V-BLAST. Desta forma, a interferência de múltiplo acesso, que é um problema nos protocolos atuais, passa a ser um aliado na busca de maiores capacidades da rede [35]. A técnica V-BLAST permite a detecção de fluxos de dados independentes simultaneamente por meio de cancelamento sucessivo de interferência. A proposta inicial desta técnica é a detecção de fluxos oriundos de um mesmo transmissor que multiplexa o seu pacote em vários segmentos e transmite cada segmento através de uma de suas múltiplas antenas. Na múltipla recepção de pacotes, a ideia é que os fluxos de dados independentes sejam oriundos de transmissores diferentes e que seja possível identificá-los. Assim, pode-se melhorar o reuso espacial da rede e carregar mais informação em um único enlace *ad hoc*.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-BLAST: an architecture for realizing very high data rates over the rich-scattering wireless channel," *Proc. ISSSE*, pp. 295–300, 1998.

[2] Sergey Loyka and Francois Gagnon, "Performance analysis of the V-BLAST algorithm: an analytical approach," *IEEE Trans. on Wireless Communications*, vol. 3, pp. 1326–1337, July 2004.

[3] Jack Winters, "On the capacity of radio communication systems with diversity in a Rayleigh fading environment," *IEEE Journal on Selected Areas in Communications*, vol. SAC-5, pp. 871–878, June 1987.

[4] G. Foschini, "Layered space-time architecture for wireless communication in fading environments when using multi-element antennas," *Bell Labs Technical Journal*, pp. 41–59, 1996.

[5] Thomas Paul and Tokunbo Ogunfunmi, "Wireless LAN comes of age: Understanding the IEEE 802.11n amendment," *IEEE Circuits and Systems Magazine*, vol. 8, 2008.

[6] Ming Hu and Junshan Zhang, "MIMO ad hoc networks: Medium access control, satuation throughput, and optimal hop distance," *Journal of Communications and Networks*, pp. 317–330, Dec. 2004.

[7] Marco Levorato, Stefano Tomasin, Paolo Casari, and Michele Zorzi, "Physical layer approximations for cross-layer performance analysis in MIMO-BLAST ad hoc networks," *IEEE Trans. on Wireless Communications*, vol. 6, no. 11, Nov. 2007.

[8] Francesco Rossetto and Michele Zorzi, "A low-delay MAC solution for MIMO ad hoc networks," *IEEE Trans. on Wireless Communications*, vol. 8, no. 1, pp. 130–135, Jan. 2009.

[9] Ece Gelal, Gentian Jakllari, and Srikanth V. Krishnamurthy, "Exploiting diversity gain in MIMO equipped ad hoc networks," 2006, 14th Asilomar Conference on Signals, Systems, and Computers.

[10] Siavash Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 8, pp. 1451–1458, Oct. 1998.

[11] "ns-3, the network simulator," www.nsnam.org/, acesso em 25 de julho de 2012 .

[12] Marcelo M. Carvalho and J. J. Garcia-Luna-Aceves, "Analytical modeling of ad hoc networks that utilize space-time coding," in *In Proc. IEEE 4th Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2006.

[13] Fadhil Firyaguna, Ana Carolina O. Christófaro, Everton L. Andrade, Tiago S. Bonfim, and Marcelo M. Carvalho, "Throughput performance of V-BLAST-enabled wireless ad hoc networks," *IEEE ICCC '12 Procedings*, Aug. 2012.

[14] Andrea Goldsmith, "Wireless communications," Cambridge University Press, 2005.

[15] "IEEE standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements part 11: Wireless lan medium access control (MAC) and physical layer (PHY) specifications," IEEE Std 802.11-2012, 2012.

[16] Magnus Frodigh, Per Johansson, and Peter Larsson, "Wireless ad hoc networking - the art of networking without a network," *Ericsson Review*, , no. 04, 2000.

[17] B. P. Crow, I. K. Widjaja, G. Jeong, and P. T. SakaL, "IEEE-802.11 wireless local area networks," *IEEE Communications Magazine*, vol. 35, no. 9, pp. 116–126, Sept. 1997.

[18] Chane L. Fullmer and J. J. Garcia-Luna-Aceves, "Floor acquisition multiple access (FAMA) for packet-radio networks," *ACM SIGCOMM Computer Communication Review*, vol. 25, pp. 262 – 273, Oct. 1995.

[19] Aruna Jayasuriya, Sylvie Perreau, Arek Dadej, and Steven Gordon, "Hidden vs. exposed terminal problem in ad hoc networks," *Communications, 2005 Asia-Pacific Conference*, pp. 759–763, Oct. 2005.

[20] Xue Yang and Nitin Vaidya, "On physical carrier sensing in wireless ad hoc networks," *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proc. IEEE*, pp. 2525–2535, Mar. 2005.

[21] Hongqiang Zhai and Yuguang Fang, "Physical carrier sensing and spatial reuse in multirate and multihop wireless ad hoc networks," *Proc. IEEE Infocom*, 2006.

[22] Theodore S. Rappaport, *Wireless Communications: Principles and Practice*, Prentice Hall, New Jersey, NJ, EUA, 2 edition, 2002.

[23] Gregory D. Durgin, *Space-Time Wireless Channels*, Prentice Hall, 1 edition, 2002.

[24] Romit Roy Choudhury and Nitin H. Vaidya, "Deafness: A MAC problem in ad hoc networks when using directional antennas," *Proc. ICNP*, pp. 283–292, Oct. 2005.

[25] Ram Ramanathan, Jason Redi, Cesar Santivanez, David Wiggins, and Stephen Polit, "Ad hoc networking with directional antennas: A complete system solution," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 3, pp. 496–506, Mar. 2005.

[26] Anastasios Stamoulis and Naofal Al-Dhahir, "Impact of space-time block codes on 802.11 network throughput," *IEEE Trans. on Wireless Communications*, vol. 2, no. 5, pp. 1029–1039, 2003.

[27] Weihua Helen Xi, Alistair Munro, and Michael Barton, "Link adaptation algorithm for the IEEE 802.11n MIMO system," *NETWORKING'08 Proc. IFIP-TC6 Networking Conference*, 2008.

[28] Qiuyan Xia, Mounir Hamdi, and Khaled Ben Letaief, "Open-loop link adaptation for next-generation IEEE 802.11n wireless networks," *IEEE Trans. on Vehicular Technology*, vol. 58, no. 7, pp. 3713–3725, Sept. 2009.

[29] Songsheng Zhu, Zhenyu Hu, and Kuixi Yin, "A STBC and V-BLAST combining MIMO system applies in ad hoc networks," *WiCOM '08*, pp. 1–5, Oct. 2008.

[30] Wang Zhong-peng, Qiu Zhong-yuan, and Wu Wei-ling, "MIMO system combining space-time block codes and VBLAST," *Journal of Eletronics and Information Technology*, vol. 27, pp. 1098–1100, July 2005.

[31] Andreas F. Molisch, *Wireless Communications*, John Wiley & Sons Ltd, 2 edition, 2011.

[32] M. Lacage and T.R. Henderson, "Yet another network simulator," in *Proc. WNS2 '06*, NY, USA, 2006, WNS2 '06.

[33] Rajendra K. Jain, Dah-Ming W. Chiu, and William R. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer system," *Technical Report 301, Digital Equipment Corporation*, Sept. 1984.

[34] J. J. Garcia-Luna-Aceves, "Reversing the collision-avoidance handshake in wireless networks," in *ACM/IEEE Mobicom '99*, 1999, pp. 120–131.

[35] P. Casari, "MAC/PHY cross-layer design of MIMO ad hoc networks with layered multiuser detection," *IEEE Trans. on Wireless Communications*, vol. 7, Nov. 2008.

ANEXOS

I. TABELA T-STUDENT

Tabela *t*

cum. prob	$t_{.50}$	$t_{.75}$	$t_{.80}$	$t_{.85}$	$t_{.90}$	$t_{.95}$	$t_{.975}$	$t_{.99}$	$t_{.995}$	$t_{.999}$	$t_{.9995}$
one-tail	0.50	0.25	0.20	0.15	0.10	0.05	0.025	0.01	0.005	0.001	0.0005
two-tails	1.00	0.50	0.40	0.30	0.20	0.10	0.05	0.02	0.01	0.002	0.001
df											
1	0.000	1.000	1.376	1.963	3.078	6.314	12.71	31.82	63.66	318.31	636.62
2	0.000	0.816	1.061	1.386	1.886	2.920	4.303	6.965	9.925	22.327	31.599
3	0.000	0.765	0.978	1.250	1.638	2.353	3.182	4.541	5.841	10.215	12.924
4	0.000	0.741	0.941	1.190	1.533	2.132	2.776	3.747	4.604	7.173	8.610
5	0.000	0.727	0.920	1.156	1.476	2.015	2.571	3.365	4.032	5.893	6.869
6	0.000	0.718	0.906	1.134	1.440	1.943	2.447	3.143	3.707	5.208	5.959
7	0.000	0.711	0.896	1.119	1.415	1.895	2.365	2.998	3.499	4.785	5.408
8	0.000	0.706	0.889	1.108	1.397	1.860	2.306	2.896	3.355	4.501	5.041
9	0.000	0.703	0.883	1.100	1.383	1.833	2.262	2.821	3.250	4.297	4.781
10	0.000	0.700	0.879	1.093	1.372	1.812	2.228	2.764	3.169	4.144	4.587
11	0.000	0.697	0.876	1.088	1.363	1.796	2.201	2.718	3.106	4.025	4.437
12	0.000	0.695	0.873	1.083	1.356	1.782	2.179	2.681	3.055	3.930	4.318
13	0.000	0.694	0.870	1.079	1.350	1.771	2.160	2.650	3.012	3.852	4.221
14	0.000	0.692	0.868	1.076	1.345	1.761	2.145	2.624	2.977	3.787	4.140
15	0.000	0.691	0.866	1.074	1.341	1.753	2.131	2.602	2.947	3.733	4.073
16	0.000	0.690	0.865	1.071	1.337	1.746	2.120	2.583	2.921	3.686	4.015
17	0.000	0.689	0.863	1.069	1.333	1.740	2.110	2.567	2.898	3.646	3.965
18	0.000	0.688	0.862	1.067	1.330	1.734	2.101	2.552	2.878	3.610	3.922
19	0.000	0.688	0.861	1.066	1.328	1.729	2.093	2.539	2.861	3.579	3.883
20	0.000	0.687	0.860	1.064	1.325	1.725	2.086	2.528	2.845	3.552	3.850
21	0.000	0.686	0.859	1.063	1.323	1.721	2.080	2.518	2.831	3.527	3.819
22	0.000	0.686	0.858	1.061	1.321	1.717	2.074	2.508	2.819	3.505	3.792
23	0.000	0.685	0.858	1.060	1.319	1.714	2.069	2.500	2.807	3.485	3.768
24	0.000	0.685	0.857	1.059	1.318	1.711	2.064	2.492	2.797	3.467	3.745
25	0.000	0.684	0.856	1.058	1.316	1.708	2.060	2.485	2.787	3.450	3.725
26	0.000	0.684	0.856	1.058	1.315	1.706	2.056	2.479	2.779	3.435	3.707
27	0.000	0.684	0.855	1.057	1.314	1.703	2.052	2.473	2.771	3.421	3.690
28	0.000	0.683	0.855	1.056	1.313	1.701	2.048	2.467	2.763	3.408	3.674
29	0.000	0.683	0.854	1.055	1.311	1.699	2.045	2.462	2.756	3.396	3.659
30	0.000	0.683	0.854	1.055	1.310	1.697	2.042	2.457	2.750	3.385	3.646
40	0.000	0.681	0.851	1.050	1.303	1.684	2.021	2.423	2.704	3.307	3.551
60	0.000	0.679	0.848	1.045	1.296	1.671	2.000	2.390	2.660	3.232	3.460
80	0.000	0.678	0.846	1.043	1.292	1.664	1.990	2.374	2.639	3.195	3.416
100	0.000	0.677	0.845	1.042	1.290	1.660	1.984	2.364	2.626	3.174	3.390
1000	0.000	0.675	0.842	1.037	1.282	1.646	1.962	2.330	2.581	3.098	3.300
z	0.000	0.674	0.842	1.036	1.282	1.645	1.960	2.326	2.576	3.090	3.291
	0%	50%	60%	70%	80%	90%	95%	98%	99%	99.8%	99.9%
	Nível de Confiança										

Figura I.1: Distribuição de probabilidade "t" de Student.

II. SCRIPTS PRINCIPAL E AUXILIARES

II.1 hybrid.cc

Script principal onde é realizada todas as configurações da simulação e a execução da simulação.

```
/* -*-  Mode: C++; c-file-style: "gnu"; indent-tabs-mode:nil; -*- */
/*
 * Copyright (c) 2011-2012 NERds GPDS UnB
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *
 * Author: Fadhil Firyaguna <firyaguna@ieee.org>
 */

/*
 * This script simulates a wifi adhoc scenario for testing spatial multiplexing
 * technique and Alamouti scheme for multiple-input multiple-output antenna
 * configuration (MIMO) under a rayleigh/rician fading channel and friis or two-ray
 * propagation loss. It is based over the IEEE 802.11b standard with modifications
 * on following classes:
 *  - WifiPhy
 *  - YansWifiPhy
 *  - InterferenceHelper
 *  - ErrorRateModel
 *  - YansErrorRateModel
 *  - DsssErrorRateModel
 *
 * It reads a topology file ("topo.dat") and a link file ("multihop.routes-static"),
 * varying from 1 to 10, that must be in this directory: "topology/tpX/",
 * where X is the number of the topology.
 *
 * The files must be in this standard:
 *
 * "topo.dat"
 * 526.0783 743.5801
 * 397.6912 435.3065
 * 183.8507 674.9844
 * ...
 * First column for x-axis, second column for y-axis.
 *
 * "multihop.routes-static"
 * 1 61 61
 * 2 70 70
 * 3  1  1
 * ...
 * First column for destination node, second column for source node, third column for nothing.
 *
 * The script output is a NS_LOG_INFO from Ipv4L3Protocol class that shows the number of
 * the node that receives successfully the packet. This is used for calculate individual
 * and global throughput.
 */

#include "ns3/core-module.h"
#include "ns3/common-module.h"
#include "ns3/node-module.h"
#include "ns3/helper-module.h"
#include "ns3/mobility-module.h"
#include "ns3/contrib-module.h"
#include "ns3/wifi-module.h"
#include "ns3/simulator-module.h"
#include "ns3/random-variable.h"
#include "ns3/flow-monitor-module.h"

#include <iostream>
```

```
#include <fstream>
#include <vector>
#include <string>
#include <cstdio>
#include <cstdlib>
#include <stdexcept>

NS_LOG_COMPONENT_DEFINE ("SMUX-CEM");

using namespace ns3;
using namespace std;

Vector GetPosition (Ptr<Node> node);

void LerTopologia (char *arquivo);
double rxPowerDbm (double distance, double height, double txPowerDbm, bool useTwoRay);
char* NameFile (uint16_t txAnt, uint16_t rxAnt, int top, double k, double snrMax, double snrMin, int run, int mimoMode);
void ComputeResults (uint64_t *rxBytesByNode, Time *delayMeanByNode, double stop, int routevetor[][2], char *filename);
void SnrResults (int routevetor[][2], char *filename, uint16_t txAnt, uint16_t rxAnt);

//coordenadas dos nós
float *x;
float *y;
//objeto do arquivo
ifstream in;
//quantidade de nós
int nos = 0;
char ch;
const double PI = 3.14159265358979323846;
const double lambda = (3.0e8 / 2.407e9);

int
main (int argc, char *argv[])
{
  //   LogComponentEnable ("MacLow", LOG_LEVEL_ERROR);
  //   LogComponentEnable ("DcfManager", LOG_LEVEL_DEBUG);
  //   LogComponentEnable ("InterferenceHelper", LOG_LEVEL_ERROR);
  //   LogComponentEnable ("YansWifiPhy", LOG_LEVEL_ERROR);
  //   LogComponentEnable ("PropagationLossModel", LOG_LEVEL_INFO);
  //   LogComponentEnable ("PropagationLossModel", LOG_LEVEL_DEBUG);
  //   LogComponentEnable ("YansErrorRateModel", LOG_LEVEL_INFO);
  //   LogComponentEnable ("YansErrorRateModel", LOG_LEVEL_DEBUG);
  //   LogComponentEnable ("YansWifiChannel", LOG_LEVEL_DEBUG);
  //   LogComponentEnable ("DsssErrorRateModel", LOG_LEVEL_INFO);
  //   LogComponentEnable ("DsssErrorRateModel", LOG_LEVEL_DEBUG);
  //   LogComponentEnable ("Ipv4EndPoint", LOG_LEVEL_DEBUG);
  //   LogComponentEnable ("Ipv4L3Protocol", LOG_LEVEL_INFO); // uncomment to generate throughput data
  //   LogComponentEnable ("MacLow", LOG_LEVEL_DEBUG);

    int top = 4;
    char topofile[100]/* = "test-topology/top3"*/;//nome do arquivo de topologia a ser lido "topo.dat"
    char routesfile[100]/* = "test-topology/link3"*/;//nome do arquivo de rotas a ser lido "multihop.routes-static";
    string phyMode ("DsssRate1Mbps");
    uint32_t packetSize = 1412; // bytes
    uint32_t numPackets = 1;
    double interval = 1.0; // seconds
    double stop = 10; //time of simulation
    bool verbose = false;
    bool tracing = false; // enable pcap tracing
    uint16_t txAnt = 1;
    uint16_t rxAnt = 1;
    double k = 0; // rician factor
    string out ("pcap/teste");
    double factorCca = 1.5;
    float height = 1.2;
    double txPowerDbm = 10;
    double distance = 150;
    bool useTwoRay = true;
    double noiseFig = 7;
    bool alamouti = false;
    bool smux = false;
    double snrMin = 10;
    double snrMax = 15;
    bool antennaHybrid = false;
    int run = 1;
    int mimoMode = 0;

  //----------------------------COMMAND LINE-----------------------------------------//
    CommandLine cmd;
    cmd.AddValue ("phyMode", "Wifi Phy mode", phyMode);
    cmd.AddValue ("packetSize", "size of application packet sent", packetSize);
    cmd.AddValue ("numPackets", "number of packets generated", numPackets);
    cmd.AddValue ("interval", "interval (seconds) between packets", interval);
```

```
    cmd.AddValue ("verbose", "turn on all WifiNetDevice log components", verbose);
    cmd.AddValue ("tracing", "enable pcap tracing", tracing);
    cmd.AddValue ("txAnt", "number of transmiter antennas on device", txAnt);
    cmd.AddValue ("rxAnt", "number of receiver antennas on device", rxAnt);
    cmd.AddValue ("ricianK", "LOS and NLOS ratio", k);
    cmd.AddValue ("stop", "stop simulation at this time in seconds", stop);
    cmd.AddValue ("out", "output file name", out);
    cmd.AddValue ("top", "input topology file name", top);
    cmd.AddValue ("cca", "multiplier factor for CCA distance", factorCca);
    cmd.AddValue ("noise", "noise figure loss in dB", noiseFig);
    cmd.AddValue ("mimoMode", "enable alamouti tech", mimoMode);
    cmd.AddValue ("snrMin", "min snr threshold for antenna switching (dB)", snrMin);
    cmd.AddValue ("snrMax", "max snr threshold for antenna switching (dB)", snrMax);
    cmd.AddValue ("run", "seed", run);
    cmd.Parse (argc, argv);
    //----------------------------fim-COMMAND LINE ------------------------------------//

    sprintf(topofile,"%s%d%s","topology/tp",top,"/topo.dat");
    sprintf(routesfile,"%s%d%s","topology/tp",top,"/multihop-routes.static");

    LerTopologia(topofile);

    //------------ Arquivo Snr Temporário -------------//
    char tempSnrFile[20];
    sprintf(tempSnrFile,"dados/temp%dx%d",txAnt,rxAnt);
    remove(tempSnrFile);
    //---------- fim-Arquivo Snr Temporário -----------//

    // disable fragmentation for frames below 2200 bytes
    Config::SetDefault ("ns3::WifiRemoteStationManager::FragmentationThreshold", StringValue ("2200"));
    // turn off RTS/CTS for frames below 2200 bytes
    Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold", StringValue ("1000"));
    // Fix non-unicast data rate to be the same as that of unicast
    Config::SetDefault ("ns3::WifiRemoteStationManager::NonUnicastMode", StringValue (phyMode));

    NodeContainer c;
    c.Create (nos);

    // The below set of helpers will help us to put together the wifi NICs we want
    WifiHelper wifi;
    wifi.SetStandard (WIFI_PHY_STANDARD_80211b);

    YansWifiPhyHelper wifiPhy =  YansWifiPhyHelper::Default ();
    // ns-3 supports RadioTap and Prism tracing extensions for 802.11b
    wifiPhy.SetPcapDataLinkType (YansWifiPhyHelper::DLT_IEEE802_11_RADIO);
    wifiPhy.Set ("RxNoiseFigure", DoubleValue (noiseFig));
    wifiPhy.SetErrorRateModel ("ns3::YansErrorRateModel");

    //----------------------------MIMO CONFIG-----------------------------------------//
    if(txAnt == 1 && rxAnt == 1) mimoMode = 0;
    switch (mimoMode){
      case 0:
      default: //SISO
txAnt = 1;
rxAnt = 1;
alamouti = false;
smux = false;
antennaHybrid = false;
break;
      case 1: // Just alamouti (alam)
alamouti = true;
smux = false;
antennaHybrid = false;
break;
      case 2: // Just smux (smux)
alamouti = false;
smux = true;
antennaHybrid = false;
break;
      case 3: // hybrid min threshold alamouti/smux(tx=rx-1) (hyba)
alamouti = false;
smux = true;
antennaHybrid = true;
snrMax = 1500;
break;
      case 4: // hybrid max threshold smux(tx=rx-1)/smux(tx=rx) (hybb)
alamouti = false;
smux = true;
antennaHybrid = true;
snrMin = -10;
break;
      case 5: // hybrid full alamouti/smux(tx=rx-1)/smux(tx=rx) (hybc)
alamouti = false;
```

```
smux = true;
antennaHybrid = true;
break;
    }
    wifiPhy.Set ("TxAntenna", UintegerValue (txAnt));
    wifiPhy.Set ("RxAntenna", UintegerValue (rxAnt));

    wifiPhy.Set ("AlamoutiTech", BooleanValue (alamouti));
    wifiPhy.Set ("SmuxTech", BooleanValue (smux));
    wifiPhy.Set ("useAntennaHybrid", BooleanValue(antennaHybrid));
    //----------------------------MIMO CONFIG-----------------------------------------//


    wifiPhy.Set ("rxSnrMinThreshold", DoubleValue (snrMin));
    wifiPhy.Set ("rxSnrMaxThreshold", DoubleValue (snrMax));
    wifiPhy.Set ("useAntennaHybrid", BooleanValue(antennaHybrid));

    wifiPhy.Set ("TxGain", DoubleValue(0));
    wifiPhy.Set ("RxGain", DoubleValue(0));
    wifiPhy.Set ("TxPowerStart", DoubleValue(txPowerDbm));
    wifiPhy.Set ("TxPowerEnd", DoubleValue(txPowerDbm));
    wifiPhy.Set ("EnergyDetectionThreshold", DoubleValue(rxPowerDbm (distance, height, txPowerDbm, useTwoRay)));
    wifiPhy.Set ("CcaMode1Threshold", DoubleValue(rxPowerDbm (distance*factorCca, height, txPowerDbm, useTwoRay)));

    YansWifiChannelHelper wifiChannel ;
    wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");

    if(!useTwoRay){
    wifiChannel.AddPropagationLoss ("ns3::FriisPropagationLossModel");
    }else{
    wifiChannel.AddPropagationLoss ("ns3::TwoRayGroundPropagationLossModel");
    }

    //------------------------------Rician Fading-----------------------------//
    double m;
    m = (pow((k+1),2)) / (2*k+1); // fórmula apresentada no livro "Wireless Communications" (Molisch)
    wifiChannel.AddPropagationLoss ("ns3::NakagamiPropagationLossModel",
"m0", DoubleValue (m),"m1", DoubleValue (m),"m2", DoubleValue (m));
    //------------------------------Rician Fading --------------------------/*/

    wifiPhy.SetChannel (wifiChannel.Create ());

    // Add a non-QoS upper mac, and disable rate control
    NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
    wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager", "DataMode",StringValue(phyMode),
      "ControlMode",StringValue(phyMode));

    // Set it to adhoc mode
    wifiMac.SetType ("ns3::AdhocWifiMac");
    NetDeviceContainer devices = wifi.Install (wifiPhy, wifiMac, c);

    MobilityHelper mobility;
    Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator> ();

    for(int j=0;j<nos;j++) positionAlloc->Add (Vector (x[j],y[j],height)); //alocação das posições dos nós

    mobility.SetPositionAllocator (positionAlloc);
    mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
    mobility.Install (c);

    InternetStackHelper internet;
    internet.Install (c);

    Ipv4AddressHelper ipv4;
    NS_LOG_INFO ("Assign IP Addresses.");
    ipv4.SetBase ("10.1.1.0", "255.255.255.0");
    Ipv4InterfaceContainer interface = ipv4.Assign (devices);

    //----------------------Aplicação Tempos Fixos-----------------------------------//
    UniformVariable uv;
    double start;

    int from=0,to=0,aux=0, l, i=0;
    FILE *fromto;
    fromto = fopen(routesfile,"r");
    int routevetor[100][2];

    do{
fscanf(fromto, "%d %d %d",&to,&from,&aux);
routevetor[i][1]=to; routevetor[i][2]=from; i++;
PacketSinkHelper sink("ns3::UdpSocketFactory",
InetSocketAddress(interface.GetAddress(to-1), 80));
ApplicationContainer sinkApp = sink.Install(c.Get(to-1));
```

```
start = uv.GetValue(0.0,0.5);
sinkApp.Start(Seconds(start));
sinkApp.Stop(Seconds(stop));

OnOffHelper onOff ("ns3::UdpSocketFactory",
InetSocketAddress(interface.GetAddress(from-1), 80));
onOff.SetAttribute ("OnTime", RandomVariableValue (ConstantVariable (1.0)));
onOff.SetAttribute ("OffTime", RandomVariableValue (ConstantVariable (0.0)));
onOff.SetAttribute ("PacketSize", UintegerValue(packetSize));
onOff.SetAttribute ("DataRate", DataRateValue(DataRate("1001000")));
onOff.SetAttribute ("Remote",
AddressValue(InetSocketAddress(interface.GetAddress(to-1),80)));
ApplicationContainer udpApp = onOff.Install(c.Get(from-1));
udpApp.Start(Seconds(start+0.001));
udpApp.Stop(Seconds(stop));
    }while((l=fgetc(fromto))!=EOF);

    fclose(fromto);
    //----------------------fim Aplicação-------------------------------------------/*/

    //--------------- Tracing----------------//
    if (tracing) wifiPhy.EnablePcap (out, devices);
    //---------------fim Tracing------------/*/

    //------------------------------Flow Monitor----------------------/*/
    FlowMonitorHelper flowmon;
    Ptr<FlowMonitor> monitor = flowmon.InstallAll ();

    uint64_t rxBytesByNode[nos];
    Time delayMeanByNode[nos];
    for(int i=0; i<nos; i++){
rxBytesByNode[i] = 0;
delayMeanByNode[i] = NanoSeconds(0);
    }

    Simulator::Stop (Seconds (stop+1));
    Simulator::Run ();

    monitor->CheckForLostPackets ();
    Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier> (flowmon.GetClassifier ());
    std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats ();

    for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator i = stats.begin (); i != stats.end (); ++i)
    {
Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (i->first);
uint8_t addr[4];
t.destinationAddress.Serialize (addr);
int nodeid = addr[3]+0;
rxBytesByNode[nodeid] += i->second.rxBytes;
if (i->second.rxPackets != 0)
delayMeanByNode[nodeid] += (i->second.delaySum / NanoSeconds (i->second.rxPackets));
    }
    //---------------------------fim-Flow Monitor----------------------/*/

    Simulator::Destroy ();

    char *filename;
    filename = NameFile (txAnt, rxAnt, top, k, snrMax, snrMin, run, mimoMode);

    ComputeResults (rxBytesByNode, delayMeanByNode, stop, routevetor, filename);
    SnrResults (routevetor, filename, txAnt, rxAnt);

    //------------ Arquivo Snr Temporário -------------//
    remove(tempSnrFile);
      //---------- fim-Arquivo Snr Temporário ----------//

    return 0;
}

double
rxPowerDbm (double distance, double height, double txPowerDbm, bool useTwoRay)
{
    double lossPowerDbm;

    if (useTwoRay){
    double dCross = (4 * PI * height * height) / lambda;
    if (distance <= dCross){
      lossPowerDbm = 10 * log10( lambda*lambda / (16.0 * PI*PI * distance*distance));
    } else {
      lossPowerDbm = 10 * log10( (height*height*height*height) / (distance*distance*distance*distance) );
    }
    }
    else {
```

```
      lossPowerDbm = 10 * log10( lambda*lambda / (16.0 * PI*PI * distance*distance));
    }

return txPowerDbm + lossPowerDbm;
}

void
LerTopologia (char *arquivo)
{
    in.open(arquivo);//essa primeira parte abre o arquivo para contar o número de linhas

    if(!in){
      cerr << "ARQUIVO 1 MAU!" << endl;
    }else{
      while(in.get(ch)){
if(ch=='\n'){
  nos++;
}
      }
    }

    if(nos != 0){
      nos++;//o número de linhas será o parâmetro para o número de nós criados
    }
    in.close();

    x = (float *)malloc(nos * sizeof(float));
    y = (float *)malloc(nos * sizeof(float));

    in.open(arquivo);//diretôrio do arquivo de topologia

    if(!in){ cerr << "ERRO DE ABERTURA 1!" << endl; }
    else{
      while(in){
for(int i=0; i<nos; i++){ in >> x[i] >> y[i]; }
in.close();
      }
    }

    in.close();

}

Vector
GetPosition (Ptr<Node> node)
{
    //função que retorna as coordenadas de um nó que achei aqui no ns-3-users group
    Ptr<MobilityModel> mobility = node->GetObject<MobilityModel> ();

return mobility->GetPosition ();
}

char*
NameFile (uint16_t txAnt, uint16_t rxAnt, int top, double k, double snrMax, double snrMin, int run, int mimoMode)
{
    char* filename = new char[15];

    if(mimoMode == 0) sprintf(filename,"siso");
    else if(mimoMode == 1) sprintf(filename,"alam");
    else if(mimoMode == 2) sprintf(filename,"smux");
    else if(mimoMode == 3) sprintf(filename,"hyba");
    else if(mimoMode == 4) sprintf(filename,"hybb");
    else if(mimoMode == 5) sprintf(filename,"hybc");

    char filesurname[50];
    sprintf(filesurname,"%dx%dk%.0ft%dr%d",txAnt,rxAnt,k,top,run);
    if(mimoMode == 3) sprintf(filesurname,"%ssnrMin%.0f",filesurname,snrMin);
    if(mimoMode == 4) sprintf(filesurname,"%ssnrMax%.0f",filesurname,snrMax);
    if(mimoMode == 5) sprintf(filesurname,"%ssnrMin%.0fsnrMax%.0f",filesurname,snrMin,snrMax);
    sprintf(filename,"%s%s",filename,filesurname);

return(filename);
}

void
ComputeResults (uint64_t *rxBytesByNode, Time *delayMeanByNode, double stop, int routevetor[][2], char *filename)
{
    double thrptMedia = 0;
    Time delayMedia = NanoSeconds(0);
    int zero = 0;

    char fileuser[60] = "dados/singleuser/";
    char filethrpt[60] = "dados/throughput/";
```

```
    char filedly[60] = "dados/delay/";

    sprintf(fileuser,"%s%s",fileuser,filename);
    sprintf(filethrpt,"%s%s",filethrpt,filename);
    sprintf(filedly,"%s%s",filedly,filename);

    ofstream singleuser;
    singleuser.open(fileuser);
    ofstream thrpt;
    thrpt.open(filethrpt);
    ofstream dly;
    dly.open(filedly);

    for(int i=1; i<nos; i++) {
      if(rxBytesByNode[i] == 0) zero++;
double vazao = (double) rxBytesByNode[i] * 8 / 1024 / stop;
singleuser << i << " " << vazao << " " << delayMeanByNode[i] << endl;
thrptMedia += vazao;
delayMedia += delayMeanByNode[i];
    }

    singleuser.close();

    thrptMedia = (double) thrptMedia / nos;
    delayMedia = delayMedia / NanoSeconds(nos-zero);

    thrpt << thrptMedia << endl;
    thrpt.close();

    dly << delayMedia << endl;
    dly.close();

    cout << "throughput: " << thrptMedia << " kbps" << endl;
    cout << "delay: " << delayMedia << endl;
    cout << "rx eff: " << nos - zero << endl;

}

void
SnrResults (int routevetor[][2], char *filename, uint16_t txAnt, uint16_t rxAnt)
{
    int to, from;
    double snr, m_snr[100][2];

    char tempname[50] = "dados/temp";
    char filesnr[50] = "dados/snr/";
    sprintf(tempname,"%s%dx%d",tempname,txAnt,rxAnt);
    sprintf(filesnr,"%s%s",filesnr,filename);

    FILE *snrDados = fopen(tempname,"r");
    ofstream snrFile;
    snrFile.open(filesnr);

    for (int s=0; s<100; s++){
      m_snr[s][0] = 0;
      m_snr[s][1] = 0;
    }

    while(fscanf(snrDados, "[mac=00:00:00:00:00:%x] [mac=00:00:00:00:00:%x] %lf\n", &to, &from, &snr) != EOF){
      if(from == routevetor[to-1][2]){
m_snr[to-1][0] += snr;
m_snr[to-1][1]++;
      }
    }

    fclose(snrDados);

    for (int s=0; s<100; s++){
      if (m_snr[s][1] != 0){
m_snr[s][0] = m_snr[s][0] / m_snr[s][1];
snrFile << s+1 << " " << m_snr[s][0] << endl;
      }
    }

    snrFile.close();
    remove(tempname);

}
```

II.2 Criador de Topologia

Este programa cria amostras de topologias capazes de formar 100 pares fonte-destino dentro de um raio de transmissão de 150 m, em um terreno plano 1600 x 1600 m com 100 nós distribuídos aleatoriamente.

```
#include "ns3/simulator-module.h"
#include "ns3/core-module.h"

#include <iostream>
#include <iomanip>
#include <fstream>
#include <vector>
#include <string>
#include <ctime>
#include <cstdio>
#include <cstdlib>
#include <stdexcept>

NS_LOG_COMPONENT_DEFINE ("NEW-TOP");

using namespace ns3;
using namespace std;

int main (int argc, char *argv[])
{

//Definições gerais
srand(time(NULL));
UniformVariable uv;

int aux=0;
int maximum=150;
int viz[101];
int top = 1;
int prob[101];
int n=1;
int limit = 1600;
int par=0;
float dist[101][101], select[101][101];
float x[101] , y[101];
int option = 0; //pode assumir os valores 0 (muito densa), 1(meno densa) e 2(densidade mediana).


CommandLine cmd;

cmd.AddValue ("top", "input topology file name", top);
cmd.AddValue ("maximum", "set the maximum value", maximum);
cmd.AddValue ("n", "set the minimum number of neighbors", n);
cmd.AddValue ("option", "set the topology density option", option);

cmd.Parse (argc, argv);


char topology[100], distance[100], routes[100];
sprintf(topology,"%s%d%s","scratch/new-topology/tp",top,"/topo.dat");
sprintf(distance,"%s%d%s","scratch/new-topology/tp",top,"/distance.dat");
sprintf(routes,"%s%d%s","scratch/new-topology/tp",top,"/multihop-routes.static");


//-----------------------------------
//Posiciona os nós de acordo com o padrão de densidade desejado para a topologia
//-----------------------------------
if(option == 0){
  for(int j = 1; j <= 100; j++){   x[j]= uv.GetValue(0,limit); y[j]= uv.GetValue(0,limit); }
}

if(option == 1){
  int j = 1;
  for(int px = 320; px <= 1600; px = px + 320)
    for(int py = 320; py <= 1600; py = py + 320)
      for(int k = 0; k < 4; k++){ x[j]= uv.GetValue((px - 320),px); y[j]= uv.GetValue((py - 320),py); j++; }
}

if(option == 2){
  int j = 1;
  for(int px = 320; px <= 1600; px = px + 320)
    for(int py = 320; py <= 1600; py = py + 320){
      int aleatorio = 1 + rand() % 4;
  for(int k = 0; k < aleatorio; k++){ x[j]= uv.GetValue((px - 320),px); y[j]= uv.GetValue((py - 320),py); j++; }
    }
  if(j != 100)  for(int k = j; k <= 100; k++){   x[k]= uv.GetValue(0,limit); y[k]= uv.GetValue(0,limit); }
}
```

```
//----------------------------------
//Processa enquanto a quantidade de rotas selecionadas for diferente de 100,
//ou seja, enquanto nem todos os nós possuem n vizinhos.
//----------------------------------

while(par < 100){

par=0;

for(int j=1; j <= 100; j++){
aux = 0;

for(int k=1; k <= 100; k++)
if(j != k){
        dist[j][k] = sqrt(pow((x[j]-x[k]),2) + pow((y[j]-y[k]),2));
if(dist[j][k] <= maximum) aux++;
}

if(aux < n) prob[j] = 1;
}

//Reposiciona os nós que não apresentaram no mínimo n vizinhos
for(int j=1; j <= 100; j++){
aux=0;

if(prob[j] == 1){
while(aux < n){

//=============================================================
//Reposicionamento dos nós
if(option == 0 || option == 2){  x[j]= uv.GetValue(0,limit); y[j]= uv.GetValue(0,limit);  }

if(option == 1){
  int m = 0;
  for(int px = 320; px <= 1600; px = px + 320)
    for(int py = 320; py <= 1600; py = py + 320)
      for(int k = 0; k < 4; k++){
m++;
      if(j == m){ x[j]= uv.GetValue((px - 320),px); y[j]= uv.GetValue((py - 320),py);}
    }
}
//=============================================================

                for(int k=1; k <= 100; k++) if(j != k){
         dist[j][k] = sqrt(pow((x[j]-x[k]),2) + pow((y[j]-y[k]),2));
if(dist[j][k] <= maximum) aux++;
}

         }
         }
}

//Gera o arquivo com a distância entre os nós da nova topologia
ofstream novadist;
novadist.open(distance);

for(int j=1; j <= 100; j++)
for (int k = 1; k<=100; k++){
dist[j][k] = sqrt(pow((x[j]-x[k]),2) + pow((y[j]-y[k]),2));
        novadist << j << " " << k << " " << dist[j][k] << endl;
        }

novadist.close();

//Gera arquivo com as posições dos nós na nova topologia
ofstream topo;
topo.open(topology);

for(int j=1; j <= 100; j++)
        topo << x[j] << " " << y[j] << endl;

topo.close();

//Calcula número de vizinhos
for(int j=1; j <= 100; j++){
viz[j]=0;
for (int k = 1; k<=100; k++){
dist[j][k] = sqrt(pow((x[j]-x[k]),2) + pow((y[j]-y[k]),2));
        if(dist[j][k] <= maximum && j!=k){ viz[j]++; }
        }
}
```

```
//Seleciona os nós que formarão os 100 pares e gera arquivo contendo-os
ofstream route;
route.open(routes);

for(int j=1; j <= 100; j++) {
aux = 0;

for (int k = 1; k <= 100; k++){
if(dist[j][k] <= maximum && j != k && aux < viz[j]){
aux++;
        }

if(dist[j][k] <= maximum && j != k && aux == viz[j]){
        select[j][k] = dist[j][k];
route << j << " " << k << " " << k << endl;
par++;
        }
}
}

topo.close();

//Imprime o resultado de cada rodada na tentativa de gerar os 100 pares
cout << "Qtdade pares selecionados: " << par << endl;
}

  return 0;
}
```

II.3 Verificador de pares fonte-destino simultâneos

Este programa verifica, dentre as topologias geradas, quais são capazes de formar 100 pares fonte-destino, dentro de um raio de transmissão de 150 m, simultâneos nas simulações.

```
#include "ns3/simulator-module.h"
#include "ns3/core-module.h"

#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <cstdio>
#include <cstdlib>
#include <stdexcept>

NS_LOG_COMPONENT_DEFINE ("ROUTES-FULL");

using namespace ns3;
using namespace std;

int main (int argc, char *argv[])
{

int top = 1;
int ll = 0;
char distancefile[101], routes[101];

int aux1, aux2;
float aux3;
int rx[101];
int used[101];
int viz[101];
int menor[101][15];
int qtdviz[101];
float dist[101][101];
int qtd = 1;

CommandLine cmd;

cmd.AddValue ("top", "input topology file name", top);

cmd.Parse (argc, argv);


for(int i = 1; i <= 100; i++){ viz[i] = 0; used[i] = 0; rx[i]= 0; }


sprintf(distancefile,"%s%d%s","scratch/topology/tp",top,"/distance.dat");
sprintf(routes,"%s%d%s","scratch/topology/tp",top,"/multihop-routes.static");
```

```
//---------------------------------------------------------------------
// Carrega o arquivo que contem as distâncias entre os nós
//---------------------------------------------------------------------

FILE *arq_geral;
arq_geral = fopen(distancefile,"r");

do{
fscanf(arq_geral, "%d %d %f",&aux1,&aux2,&aux3);
dist[aux1][aux2] = aux3;
if(dist[aux1][aux2] <= 150)
viz[aux1]++;


}while((ll=fgetc(arq_geral))!=EOF);
  fclose(arq_geral);

//---------------------------------------------------------------------
//Processamento
//---------------------------------------------------------------------

//Inicializa vetores

int ordviz[101][101];

for(int i = 1; i <= 100; i++)
for(int j = 1; j <= 100; j++)
ordviz[i][j] = 0;

for(int i = 1; i <= 100; i++)
for(int j = 1; j <= 15; j++)
menor[i][j] = 0;

for(int i = 1; i <= 100; i++)      // se a distância entre um par for menor que 150 m
for(int j = 1; j <= 100; j++)      // o par do nó "i" é colocado em um vetor de vizinhos
if(dist[i][j] <= 150){     // que o nó "i" possui, e que apresentam distância inferior
ordviz[i][j] = viz[j];     // a 150 m.
}

//
int aux = 0;

for(int i = 1; i <= 100; i++) // o vetor composto pelos vizinhos de "i" é ordenado, de foma
for(int k = 1; k <= 100; k++) // a colocar em ordem decrescente os vizinhos de "i" que possuem
for(int j = k+1; j <= 100; j++) // o menor número de vizinhos.
if(ordviz[i][k] > ordviz[i][j]){
aux = ordviz[i][k];
ordviz[i][k] = ordviz[i][j];
ordviz[i][j] = aux;
}


for(int i = 1; i <= 100; i++){ // é criado um vetor menor[][] para colocar em ordem decrescente
int k= 1; // as quantidades de vizinhos que os vizinhos de "i" possuem, para
qtdviz[i] = 0; // futura manipulação.
for(int j = 1; j <= 100; j++){
if(ordviz[i][j] != ordviz[i][j-1]){
menor[i][k] = ordviz[i][j];
qtdviz[i]++;
k++;
}
}
}

while(qtd < 99){
for(int i = 100; i >= 1; i--){       // são criadas as rotas, analisando todos os pares possíveis de cada vez.
aux1 = 0;
aux2 = 1;
while(aux2 <= qtdviz[i]){ // a criação de rotas é feita escolhendo-se inicialmente os vizinhos de "i" que possuem a menor quantidades
for(int j = 1; j <= 100; j++){ // de vizinhos, afim de garantir que esses tenham seus pares anteiormente definidos, já que a chance de
if(dist[i][j] <= 150 && i != j && viz[i] == qtd && used[j] == 0 && aux1 == 0 && viz[j] == menor[i][aux2]){ // quem tem mais vizinhos conseguir
rx[i] = j;    // algum outro vizinho qualquer é maior.
used[j] = 1;
aux1 = 1;
}

}
aux2++;
}
}
qtd++;
}
```

```
//-----------------------------------------------------------------------
//Verificação
//-----------------------------------------------------------------------
int soma = 0;

ofstream route;
route.open(routes);

for(int i = 1; i <= 100; i++){ // verifica a quantidade de nós que conseguiram formar par
cout << i << " " << rx[i] << " " << dist[i][rx[i]] << endl;
if(rx[i] != 0)
soma++;
}

if(soma==100)  // se a quantidade de nós que conseguirem formar para for igual à 100,
for(int i = 1; i <= 100; i++){ // gera um novo arquivo multihop-routes.static.
route << i << " " << rx[i] << " " << rx[i] << endl;
}

// Se a topologia analisada atingir os requisitos descritos nesse script, a fim de registro para o processamento
// de varias topologias, ela vai ser registrada em um arquivo topologias.txt, bem como as outras topologias que
//passaram por esse processamento.
char temp[101];
sprintf(temp,"%s","topologias");

ofstream tempo;
tempo.open(temp,ios::app);

if(soma==100){
cout << "TOPOLOGIA " << top << ": " << soma << " receptores" << endl;
tempo << "TOPOLOGIA " << top << ": " << soma << " receptores" << endl;
}


//---------------------------------

  return 0;
}
```

II.4 Confiability

Este programa calcula a média dos dados extraídos, o desvio padrão e a margem de erro para índice de confiança dado.

```
#include "ns3/simulator-module.h"
#include "ns3/core-module.h"

#include <iostream>
#include <iomanip>
#include <fstream>
#include <vector>
#include <string>
#include <ctime>
#include <cstdio>
#include <cstdlib>
#include <stdexcept>

NS_LOG_COMPONENT_DEFINE ("CONFIABILITY");

using namespace ns3;
using namespace std;


double Table (double alfa, int n);
long double ReadThrpt (char *filename);
long int ReadDelay (char *filename);
long double ReadFairness(char *filename);
void ComputeThrpt(long double *thrptSimul, uint16_t n, double t, bool final, int mimoMode, uint16_t txAnt,
  uint16_t rxAnt, uint16_t k, uint16_t snrMin, uint16_t snrMax);
void ComputeDelay(long int *dlySimul, uint16_t n, double t);
void ComputeFairness(long double *fairnessSimul, uint16_t n, double t);
void PrintOutput(int mimoMode, uint16_t txAnt, uint16_t rxAnt, uint16_t k, uint16_t snrMin, uint16_t snrMax);
void PrintResults(long double average, long double variance, double margin, bool delay, bool throughput, bool jainFairness);
char* NameFile (uint16_t txAnt, uint16_t rxAnt, int top, double k, double snrMin, double snrMax, int run, int mimoMode);


int main (int argc, char *argv[])
{
```

```
uint16_t txAnt = 2;
uint16_t rxAnt = 2;
uint16_t top = 5;
uint16_t k = 0;
uint16_t rngRun = 3;
bool delay = false;
double t = 0;
double confidence = 99.8;  //%
uint16_t n = 0;
bool final = false;
uint16_t snrMax = 100;
uint16_t snrMin = 100;
bool jainFairness = false;
int mimoMode = 0;


CommandLine cmd;

  cmd.AddValue ("txAnt", "number of transmiter antennas on device", txAnt);
  cmd.AddValue ("rxAnt", "number of receiver antennas on device", rxAnt);
  cmd.AddValue ("top", "input topology file name", top);
  cmd.AddValue ("mimoMode", "enable mimo tech", mimoMode);
  cmd.AddValue ("final", "enable the storage of the results after simulations", final);
  cmd.AddValue ("k", "LOS and NLOS ratio", k);
  cmd.AddValue ("rngRun", "Seed", rngRun);
  cmd.AddValue ("confidence", "Confidence Level", confidence);
  cmd.AddValue ("snrMin", "minimum value of snr", snrMin);
  cmd.AddValue ("snrMax", "maximum value of snr", snrMax);
  cmd.AddValue ("delay", "enable delay", delay);
  cmd.AddValue ("fairness", "enable JainFairness", jainFairness);

cmd.Parse (argc, argv);

      double alfa = (100 - confidence)/100;
      n = rngRun*top - 1;

      t = Table(alfa, n);

      //cout << filename << endl;

      long double thrptSimul[100];
      long int delaySimul[100];
      long double fairnessSimul[100];
      int j = 0;

      if(txAnt == 1 && rxAnt == 1) mimoMode = 0;

      for(int run = 1; run <= rngRun; run++){
      for(int tp= 1; tp <= top; tp++){
      char *filename;
      filename = NameFile (txAnt, rxAnt, tp, k, snrMin, snrMax, run, mimoMode);
      thrptSimul[j] = ReadThrpt(filename);
      if(delay && final) delaySimul[j] = ReadDelay(filename);
      if(jainFairness && final) fairnessSimul[j] = ReadFairness(filename);
      j++;
       }
       }
       ComputeThrpt(thrptSimul, n, t, final, mimoMode, txAnt, rxAnt, k, snrMin, snrMax);
       if(delay && final) ComputeDelay(delaySimul, n, t);
       if(jainFairness && final) ComputeFairness(fairnessSimul, n, t);
  return 0;
}

char*
NameFile (uint16_t txAnt, uint16_t rxAnt, int top, double k, double snrMin, double snrMax, int run, int mimoMode)
{
    char* filename = new char[50];

    if(mimoMode == 0) sprintf(filename,"siso");
    else if(mimoMode == 1) sprintf(filename,"alam");
    else if(mimoMode == 2) sprintf(filename,"smux");
    else if(mimoMode == 3) sprintf(filename,"hyba");
    else if(mimoMode == 4) sprintf(filename,"hybb");
    else if(mimoMode == 5) sprintf(filename,"hybc");

    char filesurname[50];
    sprintf(filesurname,"%dx%dk%.0ft%dr%d",txAnt,rxAnt,k,top,run);
    if(mimoMode == 3) sprintf(filesurname,"%ssnrMin%.0f",filesurname,snrMin);
    if(mimoMode == 4) sprintf(filesurname,"%ssnrMax%.0f",filesurname,snrMax);
    if(mimoMode == 5) sprintf(filesurname,"%ssnrMin%.0fsnrMax%.0f",filesurname,snrMin,snrMax);
    sprintf(filename,"%s%s",filename,filesurname);

return(filename);
```

```
}

void PrintOutput(int mimoMode, uint16_t txAnt, uint16_t rxAnt, uint16_t k, uint16_t snrMin, uint16_t snrMax){

  cout << "echo \"========================================\"" << endl;
cout << "\"End of the ";
if(mimoMode == 0) cout << "siso ";
if(mimoMode == 1) cout << "alamouti ";
if(mimoMode == 2) cout << "smux ";
if(mimoMode == 3) cout << "hybrid A ";
if(mimoMode == 4) cout << "hybrid B ";
if(mimoMode == 5) cout << "hybrid C ";
cout << txAnt << "x" << rxAnt << " k=" << k;
if(mimoMode == 3) cout << " snrMin= " << snrMin;
if(mimoMode == 4) cout << " snrMax= " << snrMax;
if(mimoMode == 5) cout << " snrMin= " << snrMin << " snrMax= " << snrMax;
cout << " simulation" << endl;
cout << "\"========================================\"" << endl;

char file[50] = "dados/results";
ofstream results(file, ios_base::app);

if(mimoMode == 0) results << endl << endl << "Siso ";
  if(mimoMode == 1) results << endl << endl << "Alamouti ";
if(mimoMode == 2) results << endl << endl << "Smux ";
if(mimoMode == 3) results << endl << endl << "Hybrid A ";
if(mimoMode == 4) results << endl << endl << "Hybrid B ";
if(mimoMode == 5) results << endl << endl << "Hybrid C ";
results << txAnt << "x" << rxAnt << " k=" << k;
if(mimoMode == 3) results << " snrMin=" << snrMin;
if(mimoMode == 4) results << " snrMax=" << snrMax;
if(mimoMode == 5) results << " snrMin=" << snrMin << " snrMax=" << snrMax;

}

void PrintResults(long double average, long double variance, double margin, bool delay, bool throughput, bool jainFairness){

if(throughput) cout << "Throughput => ";
if(delay) cout << "Atraso => ";
if(jainFairness) cout << "JainFairness => ";
cout << "Average: " << average;
cout << " Variance: " << variance;
cout << " Margin: " << margin << "%" << endl;

char file[50] = "dados/results";
ofstream results(file, ios_base::app);

if(throughput) results << endl << " Throughput => ";
if(delay) results << endl << " Delay => ";
if(jainFairness) results << endl << " JainFairness => ";
results << "Average: " << average;
results << " Variance: " << variance;
results << " Margin: " << margin << "%";
results.close();

}

void ComputeThrpt(long double *thrptSimul, uint16_t n, double t, bool final, int mimoMode, uint16_t txAnt,
  uint16_t rxAnt, uint16_t k, uint16_t snrMin, uint16_t snrMax){

long double sumThrpt = 0;
long double average = 0;
long double variance = 0;
double margin = 0; //%

for(int i = 0; i < (n+1); i++)
sumThrpt += thrptSimul[i];

average = sumThrpt/(n+1);

for(int i = 0; i < (n+1); i++)
variance += pow((thrptSimul[i] - average),2);

variance = variance / (n+1);
variance = pow(variance,0.5);

if(n <= 1000){
margin = (t*variance)/(pow((n+1),0.5));
margin = (((average + margin)/average) - 1)*100;
} if(n > 1000){
margin = (t*100*variance)/(pow((n+1),0.5)*average);
}
```

```
if(final){
  PrintOutput(mimoMode, txAnt, rxAnt, k, snrMin, snrMax);
  PrintResults(average, variance, margin, 0, 1, 0);
}
else{ cout << int(margin) << endl; }
}

void ComputeDelay(long int *dlySimul, uint16_t n, double t){

long int sumDelay = 0;
long double average = 0;
long double variance = 0;
double margin = 0; //%

for(int i = 0; i < (n+1); i++)
sumDelay += dlySimul[i];

average = sumDelay/(n+1);

for(int i = 0; i < (n+1); i++)
variance += pow((dlySimul[i] - average),2);

variance = variance / (n+1);
variance = pow(variance,0.5);

average = average/pow(10,9);
variance = variance/pow(10,9);

if(n <= 1000){
margin = (t*variance)/(pow((n+1),0.5));
margin = (((average + margin)/average) - 1)*100;
}if(n > 1000){
margin = (t*100*variance)/(pow((n+1),0.5)*average);
}

PrintResults(average, variance, margin, 1, 0, 0);
}

void ComputeFairness(long double *fairnessSimul, uint16_t n, double t){

long double sumFairness = 0;
long double average = 0;
long double variance = 0;
double margin = 0; //%

for(int i = 0; i < (n+1); i++)
sumFairness += fairnessSimul[i];

average = sumFairness/(n+1);

for(int i = 0; i < (n+1); i++)
variance += pow((fairnessSimul[i] - average),2);

variance = variance / (n+1);
variance = pow(variance,0.5);

if(n <= 1000){
margin = (t*variance)/(pow((n+1),0.5));
margin = (((average + margin)/average) - 1)*100;
}if(n > 1000){
margin = (t*100*variance)/(pow((n+1),0.5)*average);
}

PrintResults(average, variance, margin, 0, 0, 1);
}

long double ReadThrpt(char *filename){

long double aux = 0;
char name[80] = "dados/throughput/";

        sprintf(name,"%s%s",name,filename);

//cout << name << endl;

ifstream file(name);

file >> aux;
return aux;
}

long int ReadDelay(char *filename){
```

```
long int aux = 0;
long int time = 0;
int nodes = 0;
float vaz;
int id = 0;
char ns[50];

char name[50] = "dados/singleuser/";
sprintf(name,"%s%s",name,filename);

FILE *file = fopen(name, "r");

for(int i = 0; i < 100; i++){
fscanf(file, "%d %f %ld%s", &id, &vaz, &time, ns);
aux += time;
if(vaz > 0) nodes++;
}
aux = (aux/nodes);
return aux;
}

long double ReadFairness(char *filename){

long int time = 0;
int nodes = 100;
float vaz[100];
int id = 0;
char ns[50];
long double sum = 0;
long double sumQd = 0;
long double fairness = 0;

char name[50] = "dados/singleuser/";
sprintf(name,"%s%s",name,filename);

FILE *file = fopen(name, "r");

for(int i = 0; i < 100; i++){
fscanf(file, "%d %f %ld%s", &id, &vaz[i], &time, ns);
sum += vaz[i];
sumQd += vaz[i]*vaz[i];
}
fairness = (sum*sum)/(nodes*sumQd);
return fairness;
}

double Table (double alfa, int n){
  double t = 0;
  //Tabela z
//----------------------------------------
if(n == 1){

 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 1;}
 if(alfa > 0.39 && alfa < 0.41){ t = 1.376;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.963;}
 if(alfa > 0.19 && alfa < 0.21){ t = 3.078;}
 if(alfa > 0.09 && alfa < 0.11){ t = 6.314;}
 if(alfa > 0.049 && alfa < 0.051){ t = 12.71;}
 if(alfa > 0.019 && alfa < 0.021){ t = 31.82;}
 if(alfa > 0.009 && alfa < 0.011){ t = 63.66;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 318.31; }
 if(alfa > 0.0009 && alfa <0.0011){ t = 636.62;}
}
if(n == 2){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.816;}
 if(alfa > 0.39 && alfa < 0.41){ t = 1.061;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.386;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.886;}
 if(alfa > 0.09 && alfa < 0.11){ t = 2.920;}
 if(alfa > 0.049 && alfa < 0.051){ t = 4.303;}
 if(alfa > 0.019 && alfa < 0.021){ t = 6.965;}
 if(alfa > 0.009 && alfa < 0.011){ t = 9.925;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 22.327;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 31.599;}
}
if(n == 3){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.765;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.978;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.250;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.638;}
```

```
 if(alfa > 0.09 && alfa < 0.11){ t = 2.353;}
 if(alfa > 0.049 && alfa < 0.051){ t = 3.182;}
 if(alfa > 0.019 && alfa < 0.021){ t = 4.541;}
 if(alfa > 0.009 && alfa < 0.011){ t = 5.841;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 10.215;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 12.924;}
}
if(n == 4){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.741;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.941;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.190;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.533;}
 if(alfa > 0.09 && alfa < 0.11){ t = 2.132;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.776;}
 if(alfa > 0.019 && alfa < 0.021){ t = 3.747;}
 if(alfa > 0.009 && alfa < 0.011){ t = 4.604;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 7.173;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 8.610;}
}
if(n == 5){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.727;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.920;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.156;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.476;}
 if(alfa > 0.09 && alfa < 0.11){ t = 2.015;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.571;}
 if(alfa > 0.019 && alfa < 0.021){ t = 3.365;}
 if(alfa > 0.009 && alfa < 0.011){ t = 4.032;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 5.893;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 6.869;}
}
if(n == 6){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.718;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.906;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.134;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.44;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.943;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.447;}
 if(alfa > 0.019 && alfa < 0.021){ t = 3.143;}
 if(alfa > 0.009 && alfa < 0.011){ t = 3.707;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 5.208;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 5.959;}
}
if(n == 7){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.711;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.896;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.119;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.415;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.895;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.365;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.998;}
 if(alfa > 0.009 && alfa < 0.011){ t = 3.499;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 4.785;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 5.408;}
}
if(n == 8){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.706;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.889;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.108;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.397;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.860;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.306;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.896;}
 if(alfa > 0.009 && alfa < 0.011){ t = 3.355;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 4.501;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 5.041;}
}
if(n == 9){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.703;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.883;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.1;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.383;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.833;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.262;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.821;}
 if(alfa > 0.009 && alfa < 0.011){ t = 3.25;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 4.297;}
```

```
 if(alfa > 0.0009 && alfa < 0.0011){ t = 4.781;}
}
if(n == 10){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.7;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.879;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.093;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.372;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.812;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.228;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.764;}
 if(alfa > 0.009 && alfa < 0.011){ t = 3.169;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 4.144;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 4.587;}
}
if(n == 11){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.697;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.876;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.088;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.363;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.796;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.201;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.718;}
 if(alfa > 0.009 && alfa < 0.011){ t = 3.106;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 4.025;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 4.437;}
}
if(n == 12){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.695;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.873;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.083;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.356;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.782;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.179;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.681;}
 if(alfa > 0.009 && alfa < 0.011){ t = 3.055;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.930;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 4.318;}
}
if(n == 13){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.694;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.870;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.079;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.350;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.771;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.160;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.650;}
 if(alfa > 0.009 && alfa < 0.011){ t = 3.012;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.852;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 4.221;}
}
if(n == 14){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.692;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.868;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.076;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.345;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.761;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.145;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.624;}
 if(alfa > 0.009 && alfa < 0.011){ t = 2.977;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.787;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 4.140;}
}
if(n == 15){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.691;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.866;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.074;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.341;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.753;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.131;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.602;}
 if(alfa > 0.009 && alfa < 0.011){ t = 2.947;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.733;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 4.073;}
}
if(n == 16){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.690;}
```

```
 if(alfa > 0.39 && alfa < 0.41){ t = 0.865;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.071;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.337;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.746;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.12;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.583;}
 if(alfa > 0.009 && alfa < 0.011){ t = 2.921;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.686;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 4.015;}
}
if(n == 17){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.689;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.863;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.069;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.333;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.74;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.11;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.567;}
 if(alfa > 0.009 && alfa < 0.011){ t = 2.898;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.646;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 3.965;}
}
if(n == 18){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.688;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.862;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.067;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.33;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.734;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.101;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.552;}
 if(alfa > 0.009 && alfa < 0.011){ t = 2.878;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.610;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 3.922;}
}
if(n == 19){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.688;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.861;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.066;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.328;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.729;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.093;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.539;}
 if(alfa > 0.009 && alfa < 0.011){ t = 2.861;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.579;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 3.883;}
}
if(n == 20){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.687;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.86;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.064;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.325;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.725;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.086;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.528;}
 if(alfa > 0.009 && alfa < 0.011){ t = 2.845;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.552;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 3.850;}
}
if(n == 21){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.686;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.859;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.063;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.323;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.721;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.080;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.518;}
 if(alfa > 0.009 && alfa < 0.011){ t = 2.831;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.527;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 3.819;}
}
if(n == 22){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.686;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.858;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.061;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.321;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.717;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.074;}
```

```
 if(alfa > 0.019 && alfa < 0.021){ t = 2.508;}
 if(alfa > 0.009 && alfa < 0.011){ t = 2.819;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.505;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 3.792;}
}
if(n == 23){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.685;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.858;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.06;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.319;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.714;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.069;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.5;}
 if(alfa > 0.009 && alfa < 0.011){ t = 2.807;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.485;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 3.768;}
}
if(n == 24){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.685;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.857;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.059;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.318;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.711;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.064;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.492;}
 if(alfa > 0.009 && alfa < 0.011){ t = 2.797;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.467;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 3.745;}
}
if(n == 25){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.684;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.856;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.058;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.316;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.708;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.060;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.485;}
 if(alfa > 0.009 && alfa < 0.011){ t = 2.787;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.450;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 3.725;}
}
if(n == 26){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.684;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.856;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.058;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.315;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.706;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.056;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.479;}
 if(alfa > 0.009 && alfa < 0.011){ t = 2.779;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.435;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 3.707;}
}
if(n == 27){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.684;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.855;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.057;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.314;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.703;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.052;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.473;}
 if(alfa > 0.009 && alfa < 0.011){ t = 2.771;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.421;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 3.690;}
}
if(n == 28){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.683;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.855;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.056;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.313;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.701;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.048;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.467;}
 if(alfa > 0.009 && alfa < 0.011){ t = 2.763;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.408;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 3.674;}
}
```

```
if(n == 29){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.683;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.854;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.055;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.311;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.699;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.045;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.462;}
 if(alfa > 0.009 && alfa < 0.011){ t = 2.756;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.396;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 3.659;}
}
if(n >= 30 && n <= 39){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.683;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.854;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.055;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.310;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.697;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.042;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.457;}
 if(alfa > 0.009 && alfa < 0.011){ t = 2.750;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.385;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 3.646;}
}
if(n >= 40 && n <= 59 ){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.681;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.851;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.050;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.303;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.684;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2.021;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.423;}
 if(alfa > 0.009 && alfa < 0.011){ t = 2.704;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.307;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 3.551;}
}
if(n >= 60 && n <= 79 ){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.679;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.848;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.045;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.296;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.671;}
 if(alfa > 0.049 && alfa < 0.051){ t = 2;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.390;}
 if(alfa > 0.009 && alfa < 0.011){ t = 2.660;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.232;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 3.460;}
}
if(n >= 80 && n <= 99){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.678;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.846;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.043;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.292;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.664;}
 if(alfa > 0.049 && alfa < 0.051){ t = 1.990;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.374;}
 if(alfa > 0.009 && alfa < 0.011){ t = 2.639;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.195;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 3.416;}
}
if(n >= 100 && n <= 999){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.677;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.845;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.042;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.290;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.660;}
 if(alfa > 0.049 && alfa < 0.051){ t = 1.984;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.364;}
 if(alfa > 0.009 && alfa < 0.011){ t = 2.626;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.174;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 3.390;}
}
if(n == 1000){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.675;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.842;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.037;}
```

```
 if(alfa > 0.19 && alfa < 0.21){ t = 1.282;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.646;}
 if(alfa > 0.049 && alfa < 0.051){ t = 1.962;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.330;}
 if(alfa > 0.009 && alfa < 0.011){ t = 2.581;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.098;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 3.300;}
}
if(n > 1000){
 if(alfa > 0.9 && alfa < 1.1){ t = 0;}
 if(alfa > 0.49 && alfa < 0.51){ t = 0.674;}
 if(alfa > 0.39 && alfa < 0.41){ t = 0.842;}
 if(alfa > 0.29 && alfa < 0.31){ t = 1.036;}
 if(alfa > 0.19 && alfa < 0.21){ t = 1.282;}
 if(alfa > 0.09 && alfa < 0.11){ t = 1.645;}
 if(alfa > 0.049 && alfa < 0.051){ t = 1.960;}
 if(alfa > 0.019 && alfa < 0.021){ t = 2.326;}
 if(alfa > 0.009 && alfa < 0.011){ t = 2.576;}
 if(alfa > 0.0019 && alfa < 0.0021){ t = 3.090;}
 if(alfa > 0.0009 && alfa < 0.0011){ t = 3.291;}
}
//-----------------------------------------

  return t;
}
```

II.5 Gerador de script automatizador de simulações

Este programa gera um script em linguagem Shell. Este script automatiza a execução das simulações para que seja extraído os dados até alcançar a margem de erro desejada.

```
/* -*-  Mode: C++; c-file-style: "gnu"; indent-tabs-mode:nil; -*- */
/*
 * Copyright (c) 2011-2012 NERds GPDS UnB
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *
 * Author: Ana Carolina de Oliveira Christófaro <ana.christofaro@gmail.com>
 */

#include <iostream>
#include <iomanip>
#include <fstream>
#include <vector>
#include <string>
#include <ctime>
#include <cstdio>
#include <cstdlib>
#include <stdexcept>

#include "ns3/core-module.h"

using namespace std;
using namespace ns3;

int main (int argc, char *argv[])
{

  //-------
  int vetor_k[9] = {0,1,2,5,10,20,60,100,130};
  int minK = 0;
  int maxK = 0;
  //-------
//   int vetor_ant[7][2] = {{1,1},{2,2},{2,3},{2,4},{3,3},{3,4},{4,4}};
//   int vetor_ant[2][2] = {{3,3},{4,4}};
  int vetor_ant[3][2] = {{1,1},{2,3},{2,4}};
```

```
  int minAnt = 0;
  int maxAnt = 2;
  //-------
  int stop = 10;
  int nTop = 9;
  //-------
  double confidence = 99.8; //%
  int desiredMargin = 6;
  //-------
  int mimoMode = 1;
  bool delay = true;
  bool fairness = true;
  char configuracao[20];
  //-------
  int snrMax[8] = {15,11,14,17,20,23,26};
  int snrMin[8] = {8,1,2,3,4,5,8,11};
  int nSnrMax = 7;
  int nSnrMin = 7;


  CommandLine cmd;
  cmd.AddValue("maxAnt", "number of antennas", maxAnt);
  cmd.AddValue("minAnt", "number of antennas", minAnt);
  cmd.AddValue("stop", "time of simulation", stop);
  cmd.AddValue("nTop", "number of topologies", nTop);
  cmd.AddValue ("confidence", "confidence lebel %", confidence);
  cmd.AddValue ("desiredMargin", "desired margin %", desiredMargin);
  cmd.AddValue ("minK", "K value", minK);
  cmd.AddValue ("maxK", "K value", maxK);
  cmd.AddValue ("mimoMode", "enable mimo tech", mimoMode);
  cmd.AddValue ("nSnrMax", "number of snr max", nSnrMax);
  cmd.AddValue ("nSnrMin", "number of snr min", nSnrMin);

  cmd.Parse (argc, argv);

    if(mimoMode == 0) sprintf(configuracao,"siso");
    else if(mimoMode == 1) sprintf(configuracao,"alam");
    else if(mimoMode == 2) sprintf(configuracao,"smux");
    else if(mimoMode == 3) sprintf(configuracao,"hyba");
    else if(mimoMode == 4) sprintf(configuracao,"hybb");
    else if(mimoMode == 5) sprintf(configuracao,"hybc");

    if(mimoMode == 0) { nSnrMax = 0; nSnrMin = 0; minAnt=0; maxAnt=0;}
    else if(mimoMode == 1) {    nSnrMax = 0; nSnrMin = 0; }
    else if(mimoMode == 2) { nSnrMax = 0; nSnrMin = 0; }
    else if(mimoMode == 3) { nSnrMax = 0; }
    else if(mimoMode == 4) { nSnrMin = 0; }
    else if(mimoMode == 5) { nSnrMax = 1; nSnrMin = 1; }


  for (int ant = minAnt; ant <= maxAnt; ant++){
      ofstream bash;
      char file[50];
      sprintf(file,"%s%d%s%d%s",configuracao,vetor_ant[ant][0],"x",vetor_ant[ant][1],".sh");
      bash.open(file);
      bash << "#!/bin/bash" << endl << endl;

  for (int n = 0; n <=nSnrMin; n++){
  for (int s = 0; s <=nSnrMax; s++){
  for (int k=minK; k<=maxK; k++){


      bash << "echo \"=======================================================\"" << endl;
      bash << "top=$((" << nTop << "))" << endl;
      bash << "k=$((" << vetor_k[k] << "))" << endl;
      bash << "tx=$((" << vetor_ant[ant][0] << "))" << endl;
      bash << "rx=$((" << vetor_ant[ant][1] << "))" << endl;
      bash << "stop=$((" << stop << "))" << endl;
      bash << "desiredmargin=$((" << desiredMargin << "))" << endl;
      if(mimoMode == 3) bash << "snrMin=$((" << snrMin[n] << "))" << endl;
      if(mimoMode == 4) bash << "snrMax=$((" << snrMax[s] << "))" << endl;
      if(mimoMode == 5){
bash << "snrMin=$((" << snrMin[n] << "))" << endl;
        bash << "snrMax=$((" << snrMax[s] << "))" << endl;
}

      bash << "reached=$((0))" << endl;
      bash << "run=$((0))" << endl;

      bash << "while [ \"$reached\" -ne \"1\" ]" << endl;
      bash << "do" << endl;

      bash << "run=$(( $run + 1 ))" << endl;
```

```
      bash << "t=$((1))" << endl;

      bash << "echo \"========================================\"" << endl;
      bash << "echo \"Run: \" $run \" \" $tx\"x\"$rx \" k = \"$k";
      if(mimoMode == 3) bash << " \"--snrMin=\"$snrMin";
      if(mimoMode == 4) bash << " \"--snrMax=\"$snrMax";
      if(mimoMode == 5){
bash << " \"--snrMin=\"$snrMin";
        bash << " \"--snrMax=\"$snrMax";
      }

      bash << endl << "echo \"========================================\"" << endl;

      bash << "\twhile [ $t -le $top ]" << endl;
      bash << "\tdo" << endl;
      bash << "\techo topology $(( $top - $t + 1 )) \"...\"" << endl;
      bash << "\tdate" << endl;
      bash << "\terro=$(time NS_GLOBAL_VALUE=\"RngRun\"=$run ./waf --run \"hybrid
--mimoMode=" << mimoMode << " --top=$t --stop=$stop --run=$run ";
      bash << "--ricianK=$k --txAnt=$tx --rxAnt=$rx";
      if(mimoMode == 3) bash << " --snrMin=$snrMin";
      if(mimoMode == 4) bash << " --snrMax=$snrMax";
      if(mimoMode == 5){
bash << " --snrMin=$snrMin";
bash << " --snrMax=$snrMax";
      }
      bash << "\")" << endl << "\techo $erro" << endl;
      bash << "\t\twhile [ \"$erro\" == \"\" ]" << endl;
      bash << "\t\tdo" << endl;
      bash << "\t\tseed=$(($RANDOM % 10 + 5))" << endl;
      bash << "\t\techo \"Error! Run again which seed \"$seed\"...\"" << endl;
      bash << "\t\tdate" << endl;
      bash << "\t\terro=$(time NS_GLOBAL_VALUE=\"RngRun\"=$seed ./waf --run \"hybrid
--mimoMode=" << mimoMode << " --top=$t --stop=$stop --run=$run ";
      bash << "--ricianK=$k --txAnt=$tx --rxAnt=$rx";
      if(mimoMode == 3) bash << " --snrMin=$snrMin";
      if(mimoMode == 4) bash << " --snrMax=$snrMax";
      if(mimoMode == 5){
bash << " --snrMin=$snrMin";
bash << " --snrMax=$snrMax";
      }
      bash << "\")" << endl << "\techo $erro" << endl;
      bash << "\t\tdone" << endl;
      bash << "\tt=$(( $t + 1 ))" << endl;
      bash << "\tdone" << endl;

      bash << "\t\tif [ \"$run\" -gt \"1\" ]" << endl;
      bash << "\t\tthen" << endl;
      bash << "\t\tmargin=$(./waf --run \"confiability --mimoMode=" << mimoMode
<< " --top=$top --RngRun=$run --k=$k --txAnt=$tx --rxAnt=$rx --confidence=" << confidence
<< " --delay=" << delay << " --fairness=" << fairness;
      if(mimoMode == 3) bash << " --snrMin=$snrMin";
      if(mimoMode == 4) bash << " --snrMax=$snrMax";
      if(mimoMode == 5){
bash << " --snrMin=$snrMin";
bash << " --snrMax=$snrMax";
      }
      bash << "\")" << endl;
      bash << "\t\techo \"Margin: \"" << endl;
      bash << "\t\techo $margin" << endl;
      bash << "\t\tif [ \"$margin\" -lt \"$desiredmargin\" ]" << endl;
bash << "\t\t\tthen" << endl;
bash << "\t\t\techo \"Margin reached!\"" << endl;
bash << "\t\t\treached=$((1))" << endl;
        bash << "\t\t\telse" << endl;
bash << "\t\t\treached=$((0))" << endl;
bash << "\t\t\techo \"Margin not reached... keep running...\"" << endl;
      bash << "\t\tfi" << endl;
      bash << "\t\tfi" << endl;
      bash << "\tdone" << endl;
      bash << "echo \"========================================================\"" << endl;
      bash << "./waf --run \"confiability --mimoMode=" << mimoMode << " --top=$top --RngRun=$run
--k=$k --txAnt=$tx --rxAnt=$rx --confidence=" << confidence << "
--final=1 --delay=" << delay << " --fairness=" << fairness;
      if(mimoMode == 3) bash << " --snrMin=$snrMin";
      if(mimoMode == 4) bash << " --snrMax=$snrMax";
      if(mimoMode == 5){
bash << " --snrMin=$snrMin";
bash << " --snrMax=$snrMax";
      }
      bash << "\"" << endl;
      bash << "echo \"========================================================\"" << endl << endl << endl;
}
```

```
}
}
}

}
```

III. MAC LOW

III.1 mac-low.h

```
@@   class MacLow : public Object {
     public:
     ...
     private:
     ...
+ void SendDataAfterCts (Mac48Address source, Time duration, WifiMode txMode);
-void SendDataAfterCts (Mac48Address source, Time duration, WifiMode txMode,
uint8_t mux);
     ...
     };

+    class SnrTag : public Tag
+    {
+       /*
+       * An extra attribute is added to notify that SNR is
+       * above the threshold to use multiplexing. This
+       * attribute is switched if rxSnr from RTS is above
+       * threshold, thus it is sent with the CTS packet
+       * to the data transmitter station.
+       */
+    public:
+
+     static TypeId GetTypeId (void);
+     virtual TypeId GetInstanceTypeId (void) const;
+
+     virtual uint32_t GetSerializedSize (void) const;
+     virtual void Serialize (TagBuffer i) const;
+     virtual void Deserialize (TagBuffer i);
+     virtual void Print (std::ostream &os) const;
+
+     void Set (double snr);
+     double Get (void) const;
+     void SetMux (uint8_t mux);
+     uint8_t GetMux (void) const;
+    private:
+     double m_snr;
+     uint8_t multiplex;
+    };
```

III.2 mac-low.cc

```
-    class SnrTag : public Tag
-    {
-    public:
-
-     static TypeId GetTypeId (void);
-     virtual TypeId GetInstanceTypeId (void) const;
-
-     virtual uint32_t GetSerializedSize (void) const;
-     virtual void Serialize (TagBuffer i) const;
-     virtual void Deserialize (TagBuffer i);
-     virtual void Print (std::ostream &os) const;
-
-     void Set (double snr);
-     double Get (void) const;
-    private:
-     double m_snr;
-    };

    TypeId
    SnrTag::GetTypeId (void)
    {
      static TypeId tid = TypeId ("ns3::SnrTag")
.SetParent<Tag> ()
.AddConstructor<SnrTag> ()
.AddAttribute ("Snr", "The snr of the last packet received",
      DoubleValue (0.0),
      MakeDoubleAccessor (&SnrTag::Get),
      MakeDoubleChecker<double> ())
+ .AddAttribute ("Multiplex", "Switch for multiplexing packet",
```

```
+       UintegerValue (0),
+       MakeUintegerAccessor (&SnrTag::GetMux),
+       MakeUintegerChecker<uint8_t> ())
      ;
      return tid;
    }
...
    uint32_t
    SnrTag::GetSerializedSize (void) const
    {
-     return sizeof (double));
+     return sizeof (double) + sizeof (uint8_t);
    }
    void
    SnrTag::Serialize (TagBuffer i) const
    {
      i.WriteDouble (m_snr);
+     i.WriteU8 (multiplex);
    }
    void
    SnrTag::Deserialize (TagBuffer i)
    {
      m_snr = i.ReadDouble ();
+     multiplex = i.ReadU8 ();
    }
    void
    SnrTag::Print (std::ostream &os) const
    {
      os << "Snr=" << m_snr;
+     os << " mux=" << multiplex;
    }
...
+    void
+    SnrTag::SetMux (uint8_t mux)
+    {
+      multiplex = mux;
+    }
+    uint8_t
+    SnrTag::GetMux (void) const
+    {
+      return multiplex;
+    }
...
@@  void
    MacLow::ReceiveOk (Ptr<Packet> packet, double rxSnr, WifiMode txMode,
    WifiPreamble preamble)
    {
    ...
    else if (hdr.IsCts () &&
    hdr.GetAddr1 () == m_self &&
    m_ctsTimeoutEvent.IsRunning () &&
    m_currentPacket != 0)
      {
NS_LOG_DEBUG ("receive cts from="<<m_currentHdr.GetAddr1 ());

SnrTag tag;
packet->RemovePacketTag (tag);

+ /*
+  * Multiplexing tag was set when receiving RTS depending on rxSnr.
+  */
+ uint8_t mux = tag.GetMux ();

m_stationManager->ReportRxOk (m_currentHdr.GetAddr1 (), &m_currentHdr,
      rxSnr, txMode);
m_stationManager->ReportRtsOk (m_currentHdr.GetAddr1 (), &m_currentHdr,
      rxSnr, txMode, tag.Get ());

m_ctsTimeoutEvent.Cancel ();
NotifyCtsTimeoutResetNow ();
m_listener->GotCts (rxSnr, txMode);
NS_ASSERT (m_sendDataEvent.IsExpired ());
m_sendDataEvent = Simulator::Schedule (GetSifs (),
      &MacLow::SendDataAfterCts, this,
      hdr.GetAddr1 (),
      hdr.GetDuration (),
-      txMode);
+       txMode, mux);
      }
    ...
    }
...
@@  void
```

```
    MacLow::SendDataPacket (void)
    {
    ...
    m_currentHdr.SetDuration (duration);

+    SnrTag tag;
+    if (m_phy->GetJustAlamouti ()){
+      tag.SetMux (0);
+    }else if (m_phy->GetJustSmux ()){
+      tag.SetMux (3);
+    }
+    m_currentPacket->AddPacketTag (tag);

    m_currentPacket->AddHeader (m_currentHdr);
    WifiMacTrailer fcs;
    m_currentPacket->AddTrailer (fcs);

    ForwardDown (m_currentPacket, &m_currentHdr, dataTxMode);
    m_currentPacket = 0;
    }
...
@@  void
    MacLow::SendCtsAfterRts (Mac48Address source, Time duration, WifiMode rtsTxMode,
double rtsSnr)
    {
      NS_LOG_FUNCTION (this << source << duration << rtsTxMode << rtsSnr);
      /* send a CTS when you receive a RTS
       * right after SIFS.
       */

+ /*
+ * If channel is very good or very poor, use full spatial multiplexing.
+ * Else, use one degree of receive diversity.
+ * It needs to recalculate tx duration because it will be shortened.
+ * Enable a tag to notify transmissor to use multiplexing.
+ */

+      Time muxTime = MicroSeconds (0);
+      uint8_t mux = 0;
+      double rxSnrMinThreshold = pow(10, m_phy->GetRxSnrMinThreshold () / 10.0);
+      double rxSnrMaxThreshold = pow(10, m_phy->GetRxSnrMaxThreshold () / 10.0);

+      if (m_phy->GetJustAlamouti ()
+   or (m_phy->GetUseAntennaHybrid () and rtsSnr < rxSnrMinThreshold)){
+ mux = 0;
+      }else if (m_phy->GetJustSmux ()){
+      if ((m_phy->GetUseAntennaHybrid () and rtsSnr > rxSnrMaxThreshold)
+   or !m_phy->GetUseAntennaHybrid ()
+   or m_phy->GetTxAnt () < 3)
+      {
+ uint64_t m_tx = m_phy->GetTxAnt ();
+ muxTime = duration - GetSifs () - GetSifs () - GetSifs ()
  - GetCtsDuration (source, rtsTxMode) - GetAckDuration (source, rtsTxMode);
+ muxTime = muxTime * Scalar (m_tx-1) / Scalar (m_tx);
+ mux = 1;
+ NS_LOG_ERROR ("SNR OUT");
+      }else{
+ uint64_t m_tx = m_phy->GetTxAnt () - 1;
+ muxTime = duration - GetSifs () - GetSifs () - GetSifs ()
  - GetCtsDuration (source, rtsTxMode) - GetAckDuration (source, rtsTxMode);
+ muxTime = muxTime * Scalar (m_tx-1) / Scalar (m_tx);
+ mux = 2;
+ NS_LOG_ERROR ("SNR IN");
+      }
+      }
+      //---- May 4th 2012 - antenna hybrid added ----//
+
+      //---- May 13th 2012 - print snr ----//
+      char file[50] = "dados/temp";
+      sprintf(file,"%s%dx%d",file,m_phy->GetTxAnt (),m_phy->GetRxAnt ());
+      ofstream snrFile(file, ios_base::app);
+      snrFile <<  "[mac=" << m_self << "] " << "[mac=" << source << "] " << rtsSnr << endl;
+      //NS_LOG_ERROR (/*"rtsSnr = " << */"[mac=" << source << "] " << rtsSnr);
+      snrFile.close();
+      //---- May 13th 2012 - print snr ----//

       WifiMode ctsTxMode = GetCtsTxModeForRts (source, rtsTxMode);
       WifiMacHeader cts;
       cts.SetType (WIFI_MAC_CTL_CTS);
       cts.SetDsNotFrom ();
       cts.SetDsNotTo ();
       cts.SetNoMoreFragments ();
       cts.SetNoRetry ();
```

```
      cts.SetAddr1 (source);
      duration -= GetCtsDuration (source, rtsTxMode);
      duration -= GetSifs ();
+      duration -= muxTime; // time compensation for multiplexed transmission
      NS_ASSERT (duration >= MicroSeconds (0));
      cts.SetDuration (duration);

      Ptr<Packet> packet = Create<Packet> ();
      packet->AddHeader (cts);
      WifiMacTrailer fcs;
      packet->AddTrailer (fcs);

      SnrTag tag;
      tag.Set (rtsSnr);
+      tag.SetMux (mux);
      packet->AddPacketTag (tag);

      ForwardDown (packet, &cts, ctsTxMode);
    }

    void
    MacLow::SendDataAfterCts (Mac48Address source, Time duration, WifiMode txMode,
uint8_t mux)
    {
      NS_LOG_FUNCTION (this);
      /* send the third step in a
      * RTS/CTS/DATA/ACK hanshake
      */
      NS_ASSERT (m_currentPacket != 0);
      StartDataTxTimers ();
    //   NS_LOG_ERROR ("SendDataAfterCts mux = " << mux+0);

+      uint64_t m_tx = m_phy->GetTxAnt ();

      WifiMode dataTxMode = GetDataTxMode (m_currentPacket, &m_currentHdr);
      Time newDuration = Seconds (0);
      newDuration += GetSifs ();
      newDuration += GetAckDuration (m_currentHdr.GetAddr1 (), dataTxMode);
      Time txDuration = m_phy->CalculateTxDuration (GetSize (m_currentPacket,
&m_currentHdr), dataTxMode, WIFI_PREAMBLE_LONG);
+      if (mux == 1)
+      {
+ /*
+ * When multiplexing tag is enabled, tx duration is reduced
+ * and the multiplex bit in wifi mac header is enabled.
+ */
+ txDuration = txDuration / Scalar (m_tx);
+      } else if (mux == 2)
+      {
+ txDuration = txDuration / Scalar (m_tx - 1);
+      }

+      SnrTag tag;
+      tag.SetMux (mux);
+      m_currentPacket->AddPacketTag (tag);

      duration -= txDuration;
      duration -= GetSifs ();

      duration = std::max (duration, newDuration);
      NS_ASSERT (duration >= MicroSeconds (0));
      m_currentHdr.SetDuration (duration);

      m_currentPacket->AddHeader (m_currentHdr);
      WifiMacTrailer fcs;
      m_currentPacket->AddTrailer (fcs);

      ForwardDown (m_currentPacket, &m_currentHdr, dataTxMode);
      m_currentPacket = 0;
    }
```

IV. WIFI CHANNEL

IV.1 yans-wifi-channel.h

```
@@  class YansWifiChannel : public WifiChannel
    {
    ...
    private:
      typedef std::vector<Ptr<YansWifiPhy> > PhyList;
+     //-----------------February 3rd 2012 - rxPowerDbm matrix----------------------/*/
+     void Receive (uint32_t i, Ptr<Packet> packet, double **rxPowerDbm,
+     WifiMode txMode, WifiPreamble preamble) const;
+     //-----------------February 3rd 2012 - rxPowerDbm matrix----------------------/*/
    ...
    };
```

IV.2 yans-wifi-channel.cc

```
@@  void
    YansWifiChannel::Send (Ptr<YansWifiPhy> sender, Ptr<const Packet> packet,
    double txPowerDbm, WifiMode wifiMode, WifiPreamble preamble) const
    {
+     //------------------------February 3rd 2012 - added MIMO-------------//
+     uint16_t txAnt = sender->GetTxAnt ();
+     //------------------------February 3rd 2012 - added MIMO-------------//
      Ptr<MobilityModel> senderMobility =
sender->GetMobility ()->GetObject<MobilityModel> ();
      NS_ASSERT (senderMobility != 0);
      uint32_t j = 0;
      for (PhyList::const_iterator i = m_phyList.begin ();
    i != m_phyList.end (); i++, j++)
{
  if (sender != (*i))
    {
      // For now don't account for inter channel interference
      if ((*i)->GetChannelNumber() != sender->GetChannelNumber())
  continue;
      Ptr<MobilityModel> receiverMobility =
      (*i)->GetMobility ()->GetObject<MobilityModel> ();
      Time delay = m_delay->GetDelay (senderMobility, receiverMobility);

-      rxPowerDbm = m_loss->CalcRxPower (txPowerDbm, senderMobility, receiverMobility);
+       //-----------------February 3rd 2012 - added MIMO-------------------//
+       uint16_t rxAnt = (*i)->GetRxAnt ();
+       double **rxPowerDbm = (double**)malloc(txAnt * sizeof(double*));
+       for(int k=0; k < txAnt; k++){
+ rxPowerDbm[k] = (double*)malloc(rxAnt * sizeof(double));
+       }
+       /* rxPowerDbm [txAnt][rxAnt]
+       * This matrix keeps the received power in each MIMO path.
+       */
+       for(int k=0; k < txAnt; k++){
+ for(int l=0; l < rxAnt; l++){
+ /*
+   * The power of each path is the ratio between total power and tx antennas number.
+   */
+   rxPowerDbm [k][l] = m_loss->CalcRxPower (txPowerDbm, senderMobility, receiverMobility)
      - 10.0 * log10(txAnt);
+   NS_LOG_DEBUG (k << l << ": " << rxPowerDbm[k][l]);
+ }
+       }
+       //-----------------February 3rd 2012 - added MIMO-------------------//

      Ptr<Packet> copy = packet->Copy ();
      Ptr<Object> dstNetDevice = m_phyList[j]->GetDevice ();
      uint32_t dstNode;
      if (dstNetDevice == 0)
{
  dstNode = 0xffffffff;
}
      else
{
  dstNode = dstNetDevice->GetObject<NetDevice> ()->GetNode ()->GetId ();
}
```

```
      Ptr<Object> srcNetDevice = sender->GetDevice ();
      Simulator::ScheduleWithContext (dstNode,
      delay, &YansWifiChannel::Receive, this,
      j, copy, rxPowerDbm, wifiMode, preamble);
    }
}
    }
    //-----------------------February 3rd 2012 - added MIMO-------------------------//
@@  void
-   YansWifiChannel::Receive (uint32_t i, Ptr<Packet> packet, double rxPowerDbm,
+   YansWifiChannel::Receive (uint32_t i, Ptr<Packet> packet, double **rxPowerDbm,
      WifiMode txMode, WifiPreamble preamble) const
    {
      m_phyList[i]->StartReceivePacket (packet, rxPowerDbm, txMode, preamble);
    }
    //-----------------------February 3rd 2012 - added MIMO-------------------------//
```

V. WIFI PHY

V.1 wifi-phy.h

```
@@  class WifiPhy : public Object
    {
    public:
    ...
+   //---------------December 23th 2011----------------------//
+     virtual void SetTxAnt (uint16_t tx) = 0;
+     virtual uint16_t GetTxAnt (void) const = 0;
+     virtual void SetRxAnt (uint16_t rx) = 0;
+     virtual uint16_t GetRxAnt (void) const = 0;
+   //---------------December 23th 2011----------------------//
+
+   //---- March 1st 2012 - mimo tech switch added ----//
+     virtual void SetJustAlamouti (bool alamouti) = 0;
+     virtual void SetJustSmux (bool smux) = 0;
+     virtual bool GetJustAlamouti (void) const = 0;
+     virtual bool GetJustSmux (void) const = 0;
+   //---- March 1st 2012 - mimo tech switch added ----//
+
+   //---- May 4th 2012 - antenna hybrid added ----//
+     virtual void SetRxSnrMinThreshold (double snr) = 0;
+     virtual void SetRxSnrMaxThreshold (double snr) = 0;
+     virtual void SetUseAntennaHybrid (bool use) = 0;
+     virtual double GetRxSnrMinThreshold (void) const = 0;
+     virtual double GetRxSnrMaxThreshold (void) const = 0;
+     virtual bool GetUseAntennaHybrid (void) const = 0;
+   //---- May 4th 2012 - antenna hybrid added ----//
    ...
    };
```

V.2 yans-wifi-phy.h

```
@@  class YansWifiPhy : public WifiPhy
    {
    public:
    ...
+    //-----------------February 3rd 2012 - added MIMO----------------------//
+     void StartReceivePacket (Ptr<Packet> packet,
+       double **rxPowerDbm,
+       WifiMode mode,
+       WifiPreamble preamble);
+     //-----------------February 3rd 2012 - added MIMO----------------------//
    ...
+     //------------------December 23th 2011-------------------------//
+     void SetTxAnt (uint16_t tx);
+     uint16_t GetTxAnt (void) const;
+     void SetRxAnt (uint16_t rx);
+     uint16_t GetRxAnt (void) const;
+     uint16_t txAnt;
+     uint16_t rxAnt;
+     //---------------------December 23th 2011----------------------/*/

+     //---- March 1st 2012 - mimo tech switch added ----//
+     void SetJustAlamouti (bool alamouti);
+     void SetJustSmux (bool smux);
+     bool GetJustAlamouti (void) const;
+     bool GetJustSmux (void) const;
+     //---- March 1st 2012 - mimo tech switch added ----//

+     //---- May 4th 2012 - antenna hybrid added ----//
+     void SetRxSnrMinThreshold (double snr);
+     void SetRxSnrMaxThreshold (double snr);
+     void SetUseAntennaHybrid (bool use);
+     double GetRxSnrMinThreshold (void) const;
+     double GetRxSnrMaxThreshold (void) const;
+     bool GetUseAntennaHybrid (void) const;
+     //---- May 4th 2012 - antenna hybrid added ----//
    ...
    private:
    ...
+     //---- March 1st 2012 - mimo tech switch added ----//
```

```
+     bool m_alamouti;
+     bool m_smux;
+     //---- March 1st 2012 - mimo tech switch added ----//

+     //---- May 4th 2012 - antenna hybrid added ----//
+     double m_rxSnrMinThreshold;
+     double m_rxSnrMaxThreshold;
+     bool m_useAntennaHybrid;
+     //---- May 4th 2012 - antenna hybrid added ----//
    };
```

V.3 yans-wifi-phy.cc

```
+   //----- February 26th 2012 - multiplexing tag added -----//
+   #include "ns3/boolean.h"
+   #include "mac-low.h"
+   #include "wifi-mac-header.h"
+   //----- February 26th 2012 - multiplexing tag added -----//
    ...
    TypeId
    YansWifiPhy::GetTypeId (void)
    {
      static TypeId tid = TypeId ("ns3::YansWifiPhy")
.SetParent<WifiPhy> ()
.AddConstructor<YansWifiPhy> ()
+ //-----------December 23th 2011------------//
+ .AddAttribute ("TxAntenna",
+       "Number of antennas at the transmiter. Default is 1",
+       UintegerValue (1),
+       MakeUintegerAccessor (&YansWifiPhy::SetTxAnt,
+     &YansWifiPhy::GetTxAnt),
+       MakeUintegerChecker<uint16_t> ())
+ .AddAttribute ("RxAntenna",
+       "Number of antennas at the receiver. Default is 1",
+       UintegerValue (1),
+       MakeUintegerAccessor (&YansWifiPhy::SetRxAnt,
+     &YansWifiPhy::GetRxAnt),
+       MakeUintegerChecker<uint16_t> ())
+     //------------December 23th 2011-----------/*/
+     //---- March 1st 2012 - mimo tech switch added ----//
+     /*
+ * If both variables are false, hybrid system will be used.
+ * If both are true, spatial multiplexing will be used.
+ */
+ .AddAttribute ("AlamoutiTech",
+       "Boolean variable to switch to MIMO Alamouti technique",
+       BooleanValue (false),
+       MakeBooleanAccessor (&YansWifiPhy::SetJustAlamouti,
+     &YansWifiPhy::GetJustAlamouti),
+       MakeBooleanChecker ())
+ .AddAttribute ("SmuxTech",
+       "Boolean variable to switch to MIMO Spatial Multiplexing technique",
+       BooleanValue (false),
+       MakeBooleanAccessor (&YansWifiPhy::SetJustSmux,
+     &YansWifiPhy::GetJustSmux),
+       MakeBooleanChecker ())
+     //---- March 1st 2012 - mimo tech switch added ----//
+     //---- May 4th 2012 - antenna hybrid added ----//
+ .AddAttribute ("rxSnrMinThreshold",
+       "Min Snr threshold for antenna switching. Default is 1.5",
+       DoubleValue (1.5),
+       MakeDoubleAccessor (&YansWifiPhy::SetRxSnrMinThreshold,
+     &YansWifiPhy::GetRxSnrMinThreshold),
+       MakeDoubleChecker<double> ())
+ .AddAttribute ("rxSnrMaxThreshold",
+       "Max Snr threshold for antenna switching. Default is 25",
+       DoubleValue (25.0),
+       MakeDoubleAccessor (&YansWifiPhy::SetRxSnrMaxThreshold,
+     &YansWifiPhy::GetRxSnrMaxThreshold),
+       MakeDoubleChecker<double> ())
+ .AddAttribute ("useAntennaHybrid",
+       "Boolean variable to switch to Antenna Hybrid Multiplexing",
+       BooleanValue (false),
+       MakeBooleanAccessor (&YansWifiPhy::SetUseAntennaHybrid,
+     &YansWifiPhy::GetUseAntennaHybrid),
+       MakeBooleanChecker ())
+     //---- May 4th 2012 - antenna hybrid added ----//
    ...
    }
    ...
```

```
+   //-----------------December 23th 2011--------------------//
+   void
+   YansWifiPhy::SetTxAnt (uint16_t tx)
+   {
+     NS_LOG_FUNCTION (this << tx);
+     txAnt = tx;
+     m_interference.SetTxAnt (tx);
+   }
+   void
+   YansWifiPhy::SetRxAnt (uint16_t rx)
+   {
+     NS_LOG_FUNCTION (this << rx);
+     rxAnt = rx;
+     m_interference.SetRxAnt (rx);
+   }
+   uint16_t
+   YansWifiPhy::GetTxAnt (void) const
+   {
+     return txAnt;
+   }
+   uint16_t
+   YansWifiPhy::GetRxAnt (void) const
+   {
+     return rxAnt;
+   }
+   //-----------------December 23th 2011--------------------/*/
+
+   //---- March 1st 2012 - mimo tech switch added ----//
+   void
+   YansWifiPhy::SetJustAlamouti (bool alamouti)
+   {
+     NS_LOG_FUNCTION (this << alamouti);
+     m_alamouti = alamouti;
+   }
+   void
+   YansWifiPhy::SetJustSmux (bool smux)
+   {
+     NS_LOG_FUNCTION (this << smux);
+     m_smux = smux;
+   }
+   bool
+   YansWifiPhy::GetJustAlamouti (void) const
+   {
+     return m_alamouti;
+   }
+   bool
+   YansWifiPhy::GetJustSmux (void) const
+   {
+     return m_smux;
+  }
+   //---- March 1st 2012 - mimo tech switch added ----//
+
+   //---- May 4th 2012 - antenna hybrid added ----//
+   void
+   YansWifiPhy::SetRxSnrMinThreshold (double snr)
+   {
+     NS_LOG_FUNCTION (this << snr);
+     m_rxSnrMinThreshold = snr;
+   }
+   void
+   YansWifiPhy::SetRxSnrMaxThreshold (double snr)
+   {
+     NS_LOG_FUNCTION (this << snr);
+     m_rxSnrMaxThreshold = snr;
+   }
+   double
+   YansWifiPhy::GetRxSnrMinThreshold (void) const
+   {
+     return m_rxSnrMinThreshold;
+   }
+   double
+   YansWifiPhy::GetRxSnrMaxThreshold (void) const
+   {
+     return m_rxSnrMaxThreshold;
+   }
+   void
+   YansWifiPhy::SetUseAntennaHybrid (bool use)
+   {
+     NS_LOG_FUNCTION (this << use);
+     m_useAntennaHybrid = use;
+   }
+   bool
+   YansWifiPhy::GetUseAntennaHybrid (void) const
```

```
+   {
+     return m_useAntennaHybrid;
+   }
+   //---- May 4th 2012 - antenna hybrid added ----//
    ...
@@  void
    YansWifiPhy::StartReceivePacket (Ptr<Packet> packet,
-    double rxPowerDbm,
+     double **rxPowerDbm,
    WifiMode txMode,
    enum WifiPreamble preamble)
    {
    ...
-     double rxPowerW = DbmToW (rxPowerDbm += m_rxGainDb);
+     WifiMacHeader hdr;
+     packet->PeekHeader (hdr);
+     SnrTag tag;
+     packet->PeekPacketTag (tag);
+     uint8_t mux = tag.GetMux ();
+     bool isData = hdr.IsData ();
+     if(!isData and m_smux) mux = 3;
+     //---
+     uint16_t txAnt_t = txAnt;
+     if(m_alamouti) {
+ mux = 0;
+ txAnt_t = 2;
+     }
+     //----- February 26th 2012 - multiplexing tag added -----//
+
+     //-----------------February 3rd 2012 - added MIMO----------------------//
+     double **rxPowerW = (double**)malloc(txAnt_t * sizeof(double*));
+     for(int i=0; i<txAnt ; i++){
+ rxPowerW[i] = (double*)malloc(rxAnt * sizeof(double));
+     }
+
+     for(int i=0; i<txAnt_t ; i++) { for(int j=0; j<rxAnt ; j++)
+ {
+   rxPowerW [i][j] = DbmToW (rxPowerDbm [i][j] += m_rxGainDb);
+   //       NS_LOG_DEBUG (i << j << ": " << rxPowerW[i][j]);
+ }
+     }
+
+     /* norma frobenius adaptada na forma de potência
+     * frobenius norm adapted to power
+     */
+     double mimo_rxPowerW = 0;
+     for(int i=0; i<txAnt_t ; i++) { for(int j=0; j<rxAnt ; j++)
+ {
+   mimo_rxPowerW += rxPowerW [i][j];
+ }
+     }
+   //   NS_LOG_DEBUG ("\nMIMO = " << mimo_rxPowerW);
+
+     /* potência recebida no canal mimo emulando um canal siso
+     * tira-se a média das potências dos caminhos a partir de cada antena transmissora
+     * soma-se as médias
+     *
+     * received power on mimo channel emulating a siso channel
+     * it gets the mean of each path from tx antenna power
+     * then sums the means
+     */
+     double siso_rxPowerW = mimo_rxPowerW / rxAnt;
+
+   //   NS_LOG_DEBUG ("SISO = " << siso_rxPowerW);
+     //-----------------February 3rd 2012 - added MIMO----------------------//
+
      Time rxDuration = CalculateTxDuration (packet->GetSize (), txMode, preamble);

+     //---- May 4th 2012 - antenna hybrid added ----//
+     NS_LOG_ERROR ("mux: " << mux+0);
+     if (mux == 2){
+ rxDuration = NanoSeconds (rxDuration.GetNanoSeconds () / (txAnt-1));
+     }else{
+ if (mux == 1) {
+ rxDuration = NanoSeconds (rxDuration.GetNanoSeconds () / (txAnt));
+ }
+     }
+     //---- May 4th 2012 - antenna hybrid added ----//
+
      Time endRx = Simulator::Now () + rxDuration;

      Ptr<InterferenceHelper::Event> event;
```

```
      event = m_interference.Add (packet->GetSize (),
  txMode,
  preamble,
  rxDuration,
-  rxPowerW);
+   siso_rxPowerW,
+   mimo_rxPowerW,
+   mux); // multiplexing tag added


      switch (m_state->GetState ()) {
      case YansWifiPhy::SWITCHING:
      ...
      case YansWifiPhy::IDLE:
-if (rxPowerW > m_edThresholdW)
+ if (siso_rxPowerW > m_edThresholdW)
...
break;
      }
    ...
    }

@@  void
    YansWifiPhy::SendPacket (Ptr<const Packet> packet, WifiMode txMode,
  WifiPreamble preamble, uint8_t txPower)
    {
    ...
+     //----- February 26th 2012 - multiplexing tag added -----//
      /*
      * Read wifi mac header to know if multiplexing technique has to be used.
      */
+     WifiMacHeader hdr;
+     packet->PeekHeader (hdr);
+     SnrTag tag;
+     packet->PeekPacketTag (tag);
+     bool mux = tag.GetMux ();

      Time txDuration = CalculateTxDuration (packet->GetSize (), txMode, preamble);
+     //---- May 4th 2012 - antenna hybrid added ----//

+     if (mux == 2){
+ txDuration = NanoSeconds (txDuration.GetNanoSeconds () / (txAnt-1));
+     }else{
+ if (mux == 1) {
+ txDuration = NanoSeconds (txDuration.GetNanoSeconds () / (txAnt));
+ }
      }

+     //---- May 4th 2012 - antenna hybrid added ----//
+     //----- February 26th 2012 - multiplexing tag added -----//
    ...
    }
```

VI. INTERFERENCE HELPER

VI.1 interference-helper.h

```
@@  class InterferenceHelper
    {
    public:
      class Event : public SimpleRefCount<InterferenceHelper::Event>
      {
      public:
-Event (uint32_t size, WifiMode payloadMode,
-enum WifiPreamble preamble,
-Time duration, double rxPower);
+ //---------------February 3rd 2012 - multipath received power added-----//
+ //---------------February 26th 2012 - multiplexing tag added -----------//
+ Event (uint32_t size, WifiMode payloadMode,
+ enum WifiPreamble preamble,
+ Time duration, double rxPower, double mimo_rxPower, uint8_t useMux);
+ //---------------February 3rd 2012 - multipath received power added-----//
...
+   //---------------February 3rd 2012 - add multipath received power -----------//
+ double GetMimoRxPowerW (void) const;
+ //---------------February 3rd 2012 - add multipath received power -----------//
...
+ //---------------February 26th 2012 - multiplexing tag added -----------//
+ uint8_t GetUseMux (void) const;
+ //---------------February 26th 2012 - multiplexing tag added -----------//

      private:
...
+ //---------------February 3rd 2012 - add multipath received power -----------//
+ double m_mimo_rxPowerW;
+ //---------------February 3rd 2012 - add multipath received power -----------//

+ //---------------February 26th 2012 - multiplexing tag added -----------//
+ uint8_t m_useMux;
+ //---------------February 26th 2012 - multiplexing tag added -----------//
      };
      ...
+     //-----------------December 23th 2011-------------------------/*/
+     static uint16_t m_tx;
+     static uint16_t m_rx;
+     void SetTxAnt (uint16_t tx);
+     uint16_t GetTxAnt (void) const;
+     void SetRxAnt (uint16_t rx);
+     uint16_t GetRxAnt (void) const;
+     //-----------------December 23th 2011-------------------------/*/

      ...
-     Ptr<InterferenceHelper::Event> Add (uint32_t size, WifiMode payloadMode,
-enum WifiPreamble preamble,
-Time duration, double rxPower);
+     //---------------February 3rd 2012 - add multipath received power -----------//
+ //---------------February 26th 2012 - multiplexing tag added -----------//
+     Ptr<InterferenceHelper::Event> Add (uint32_t size, WifiMode payloadMode,
+ enum WifiPreamble preamble,
+ Time duration, double rxPower,
+ double mimo_rxPower, uint8_t useMux);
+      //---------------February 3rd 2012 - add multipath received power -----------//

      ...
    private:
      //---------------February 3rd 2012 - add multipath received power -----------//
      class NiChange {
      public:
NiChange (Time time, double delta, double mimo_delta);
Time GetTime (void) const;
double GetDelta (void) const;
+ double GetMimoDelta (void) const;
bool operator < (const NiChange& o) const;
      private:
Time m_time;
double m_delta;
+ double m_mimo_delta;
      };
      //---------------February 3rd 2012 - add multipath received power -----------//
```

```
      ...
-     double CalculateChunkSuccessRate (double snir, Time delay, WifiMode mode,
   uint8_t useMux) const;
+ //---------------February 26th 2012 - multiplexing tag added -----------//
+     double CalculateChunkSuccessRate (double snir, Time delay, WifiMode mode,
  uint8_t useMux) const;
+ //---------------February 26th 2012 - multiplexing tag added -----------//

      ...
+     //---------------February 3rd 2012 - add multipath received power -----------//
+     double m_mimo_firstPower;
+     //---------------February 3rd 2012 - add multipath received power -----------//
      ...
    };
```

VI.2 interference-helper.cc

```
@@  InterferenceHelper::Event::Event (uint32_t size, WifiMode payloadMode,
      enum WifiPreamble preamble,
-      Time duration, double rxPower)
+       Time duration, double rxPower,
+       double mimo_rxPower, uint8_t useMux)
      : m_size (size),
m_payloadMode (payloadMode),
m_preamble (preamble),
m_startTime (Simulator::Now ()),
m_endTime (m_startTime + duration),
m_rxPowerW (rxPower),
+ m_mimo_rxPowerW (mimo_rxPower),
+ m_useMux (useMux)
    {}
    ...
+   //---------------February 3rd 2012 - add multipath received power -----------//
+   double
+   InterferenceHelper::Event::GetMimoRxPowerW (void) const
+   {
+     return m_mimo_rxPowerW;
+   }
+   //---------------February 3rd 2012 - add multipath received power -----------//

+   //---------------February 26th 2012 - add multiplexing tag ------------------//
+   uint8_t
+   InterferenceHelper::Event::GetUseMux (void) const
+   {
+     return m_useMux;
+   }
+   //---------------February 26th 2012 - add multiplexing tag ------------------//
    ...
-   InterferenceHelper::NiChange::NiChange (Time time, double delta)
-     : m_time (time), m_delta (delta)
-   {}
+   //---------------February 3rd 2012 - add multipath received power -----------//
+   InterferenceHelper::NiChange::NiChange (Time time, double delta, double mimo_delta)
+     : m_time (time), m_delta (delta), m_mimo_delta (mimo_delta)
+   {}
+   //---------------February 3rd 2012 - add multipath received power -----------//
    ...
    //---------------February 3rd 2012 - add multipath received power -----------//
+   double
+   InterferenceHelper::NiChange::GetMimoDelta (void) const
+   {
+     return m_mimo_delta;
+   }
+   //---------------February 3rd 2012 - add multipath received power -----------//
    ...
+   //---------------February 3rd 2012 - add multipath received power -----------//
@@  InterferenceHelper::InterferenceHelper ()
      : m_errorRateModel (0),
m_firstPower (0.0),
+ m_mimo_firstPower (0.0),
m_rxing (false)
    {}
+   //----------February 3rd 2012 - add multipath received power -----------//
    ...
+   //--------February 3rd 2012 - add multipath received power -----------//
+   //---------------February 26th 2012 - multiplexing tag added -----------//
@@  Ptr<InterferenceHelper::Event>
    InterferenceHelper::Add (uint32_t size, WifiMode payloadMode,
    enum WifiPreamble preamble,
-    Time duration, double rxPowerW)
```

```
+     Time duration, double rxPowerW,
+     double mimo_rxPower, uint8_t useMux)
    {
      Ptr<InterferenceHelper::Event> event;

      event = Create<InterferenceHelper::Event>
(size,
payloadMode,
preamble,
duration,
rxPowerW,
+ mimo_rxPower,
+ useMux);
      AppendEvent (event);
      return event;
    }
+   //--------February 3rd 2012 - add multipath received power -----------//

+   //--------------December 23th 2011--------------------//
+   void
+   InterferenceHelper::SetTxAnt (uint16_t tx)
+   {
+     m_tx = tx;
+   }
+   uint16_t
+   InterferenceHelper::GetTxAnt (void) const
+   {
+     return m_tx;
+   }
+   void
+   InterferenceHelper::SetRxAnt (uint16_t rx)
+   {
+     m_rx = rx;
+   }
+   uint16_t
+   InterferenceHelper::GetRxAnt (void) const
+   {
+     return m_rx;
+   }
+   uint16_t
+   InterferenceHelper::m_tx = 0;
+   uint16_t
+   InterferenceHelper::m_rx = 0;
+   //-------------------December 23th 2011-------------------/*/
    ...
+   //-----------February 3rd 2012 - add multipath received power -----------//
@@  void
    InterferenceHelper::AppendEvent (Ptr<InterferenceHelper::Event> event)
    {
      Time now = Simulator::Now ();
      if (!m_rxing)
{
  NiChanges::iterator nowIterator = GetPosition (now);
  for (NiChanges::iterator i = m_niChanges.begin (); i != nowIterator; i++)
    {
      m_firstPower += i->GetDelta ();
+       m_mimo_firstPower += i->GetMimoDelta ();
    }
  m_niChanges.erase (m_niChanges.begin (), nowIterator);
-  m_niChanges.insert (m_niChanges.begin (), NiChange (event->GetStartTime (),
event->GetRxPowerW ()));
+   m_niChanges.insert (m_niChanges.begin (), NiChange (event->GetStartTime (),
    event->GetRxPowerW (), event->GetMimoRxPowerW ()));
}
      else
{
-  AddNiChangeEvent (NiChange (event->GetStartTime (), event->GetRxPowerW ()));
+   AddNiChangeEvent (NiChange (event->GetStartTime (), event->GetRxPowerW (),
   event->GetMimoRxPowerW ()));
}
-     AddNiChangeEvent(NiChange (event->GetEndTime (), -event->GetRxPowerW ()));
+     AddNiChangeEvent(NiChange (event->GetEndTime (), -event->GetRxPowerW (),
    -event->GetMimoRxPowerW ()));

    }
+   //---------------February 3rd 2012 - add multipath received power -----------//
    ...
+   //---------------February 3rd 2012 - add multipath received power -----------//
@@  double
    InterferenceHelper::CalculateNoiseInterferenceW (Ptr<InterferenceHelper::Event> event,
    NiChanges *ni) const
    {
+     double mimo_noiseInterference = m_mimo_firstPower;
```

```
      double noiseInterference = m_firstPower;
      NS_ASSERT (m_rxing);
      for (NiChanges::const_iterator i = m_niChanges.begin () + 1;
      i != m_niChanges.end (); i++)
{
-  if ((event->GetEndTime () == i->GetTime ())
      && event->GetRxPowerW () == -i->GetDelta ())
+   if ((event->GetEndTime () == i->GetTime ())
      && event->GetMimoRxPowerW () == -i->GetMimoDelta ())
    {
      break;
    }
  ni->push_back (*i);
}
-     ni->insert (ni->begin (), NiChange (event->GetStartTime (), noiseInterference));
+     ni->insert (ni->begin (), NiChange (event->GetStartTime (), noiseInterference,
      mimo_noiseInterference));
-     ni->push_back (NiChange (event->GetEndTime (), 0));
+     ni->push_back (NiChange (event->GetEndTime (), 0, 0));
      return noiseInterference;
    }
+   //---------------February 3rd 2012 - add multipath received power -----------//

+   //------------------February 26th 2012 - add multiplexing tag------------------//
@@  double
-   InterferenceHelper::CalculateChunkSuccessRate (double snir, Time duration,
  WifiMode mode) const
+   InterferenceHelper::CalculateChunkSuccessRate (double snir, Time duration,
  WifiMode mode, uint8_t useMux) const
    {
      if (duration == NanoSeconds (0)) {
return 1.0;
      }
      uint32_t rate = mode.GetPhyRate ();
      uint64_t nbits = (uint64_t)(rate * duration.GetSeconds ());

+     /*
+     * For Spatial Multiplexing MIMO technique, bit error rate
+     * is calculated with Sergey & Loyka V-Blast equation that
+     * depends on antenna configuration and on SNIR based on
+     * SISO received power. The technique is used only for
+     * DATA packets, ACK, RTS and CTS packets use Alamouti, and
+     * its ber is calculated commonly with DBPSK BER function.
+     */

+     if (useMux == 1){
+ m_errorRateModel->SetTxAnt (m_tx);
+ m_errorRateModel->SetRxAnt (m_rx);
+     } else if (useMux == 2){
+ m_errorRateModel->SetTxAnt (m_tx-1);
+ m_errorRateModel->SetRxAnt (m_rx);
+     }else if (useMux == 0 or useMux == 3){
+ m_errorRateModel->SetTxAnt (1);
+ m_errorRateModel->SetRxAnt (1);
+     }

      double csr = m_errorRateModel->GetChunkSuccessRate (mode, snir, (uint32_t)nbits);
      return csr;
    }
+   //------------------February 26th 2012 - add multiplexing tag----------------//


+   //------------------February 26th 2012 - add multiplexing tag----------------//
@@  double
    InterferenceHelper::CalculatePer (Ptr<const InterferenceHelper::Event> event,
      NiChanges *ni) const
    {
      double psr = 1.0; /* Packet Success Rate */
      NiChanges::iterator j = ni->begin ();
      Time previous = (*j).GetTime ();
      WifiMode payloadMode = event->GetPayloadMode ();
      WifiPreamble preamble = event->GetPreambleType ();
      WifiMode headerMode = GetPlcpHeaderMode (payloadMode, preamble);
      Time plcpHeaderStart = (*j).GetTime ()
    + MicroSeconds (GetPlcpPreambleDurationMicroSeconds (payloadMode, preamble));
      Time plcpPayloadStart = plcpHeaderStart
    + MicroSeconds (GetPlcpHeaderDurationMicroSeconds (payloadMode, preamble));
      double noiseInterferenceW = (*j).GetDelta ();
      double powerW = event->GetRxPowerW ();
+   //---------------February 3rd 2012 - add multipath received power -----------//
+     double mimo_noiseInterferenceW = (*j).GetMimoDelta ();
+     double mimo_powerW = event->GetMimoRxPowerW ();
+   //   NS_LOG_DEBUG ("[error ] mimo: " << powerW);
```

```
+   //---------------February 3rd 2012 - add multipath received power ----------//

+     uint8_t useMux = event->GetUseMux ();
+     double use_powerW = 0, use_noiseInterferenceW = 0;

+     if (useMux != 0){
+ use_powerW = powerW;
+ use_noiseInterferenceW = noiseInterferenceW;
+     }else{
+ use_powerW = mimo_powerW;
+ use_noiseInterferenceW = mimo_noiseInterferenceW;
+     }

      j++;
      while (ni->end () != j)
{
  Time current = (*j).GetTime ();
  NS_ASSERT (current >= previous);

  if (previous >= plcpPayloadStart)
    {
-      psr *= CalculateChunkSuccessRate (CalculateSnr (powerW,
-      noiseInterferenceW,
-      payloadMode),
-current - previous,
-payloadMode);
+       psr *= CalculateChunkSuccessRate (CalculateSnr (use_powerW,
+       use_noiseInterferenceW,
+       payloadMode),
+ current - previous,
+ payloadMode, useMux);
    }
  else if (previous >= plcpHeaderStart)
    {
      if (current >= plcpPayloadStart)
{
-  psr *= CalculateChunkSuccessRate (CalculateSnr (powerW,
-  noiseInterferenceW,
-  headerMode),
-    plcpPayloadStart - previous,
-    headerMode);
-  psr *= CalculateChunkSuccessRate (CalculateSnr (powerW,
-  noiseInterferenceW,
-  payloadMode),
-    current - plcpPayloadStart,
-    payloadMode);
+   psr *= CalculateChunkSuccessRate (CalculateSnr (use_powerW,
+   use_noiseInterferenceW,
+   headerMode),
+     plcpPayloadStart - previous,
+     headerMode, useMux);
+   psr *= CalculateChunkSuccessRate (CalculateSnr (use_powerW,
+   use_noiseInterferenceW,
+   payloadMode),
+     current - plcpPayloadStart,
+     payloadMode, useMux);
}
      else
{
  NS_ASSERT (current >= plcpHeaderStart);
-  psr *= CalculateChunkSuccessRate (CalculateSnr (powerW,
-  noiseInterferenceW,
-  headerMode),
-    current - previous,
-    headerMode);
+   psr *= CalculateChunkSuccessRate (CalculateSnr (use_powerW,
+   use_noiseInterferenceW,
+   headerMode),
+     current - previous,
+     headerMode, useMux);
}
    }
  else
    {
      if (current >= plcpPayloadStart)
{
-  psr *= CalculateChunkSuccessRate (CalculateSnr (powerW,
-  noiseInterferenceW,
-  headerMode),
-    plcpPayloadStart - plcpHeaderStart,
-    headerMode);
-  psr *= CalculateChunkSuccessRate (CalculateSnr (powerW,
-  noiseInterferenceW,
```

```
-  payloadMode),
-    current - plcpPayloadStart,
-    payloadMode);
+   psr *= CalculateChunkSuccessRate (CalculateSnr (use_powerW,
+   use_noiseInterferenceW,
+   headerMode),
+     plcpPayloadStart - plcpHeaderStart,
+     headerMode, useMux);
+   psr *= CalculateChunkSuccessRate (CalculateSnr (use_powerW,
+   use_noiseInterferenceW,
+   payloadMode),
+     current - plcpPayloadStart,
+     payloadMode, useMux);
}
      else if (current >= plcpHeaderStart)
{
-  psr *= CalculateChunkSuccessRate (CalculateSnr (powerW,
-  noiseInterferenceW,
-  headerMode),
-    current - plcpHeaderStart,
-    headerMode);
+   psr *= CalculateChunkSuccessRate (CalculateSnr (use_powerW,
+   use_noiseInterferenceW,
+   headerMode),
+     current - plcpHeaderStart,
+     headerMode, useMux);
}
    }

+   //---------------February 3rd 2012 - add multipath received power -----------//
  noiseInterferenceW += (*j).GetDelta ();
+   mimo_noiseInterferenceW += (*j).GetMimoDelta ();
+   //---------------February 3rd 2012 - add multipath received power -----------//
  previous = (*j).GetTime ();
  j++;
}

      double per = 1 - psr;
      return per;
    }
+   //------------------February 26th 2012 - add multiplexing tag----------------//
    ...
+   //---------------February 3rd 2012 - add multipath received power -----------//
    void
    InterferenceHelper::EraseEvents (void)
    {
      m_niChanges.clear ();
      m_rxing = false;
      m_firstPower = 0.0;
+     m_mimo_firstPower = 0.0;
    }


    InterferenceHelper::NiChanges::iterator
    InterferenceHelper::GetPosition (Time moment)
    {
-     return std::upper_bound (m_niChanges.begin (), m_niChanges.end (),
NiChange (moment, 0, 0));
+     return std::upper_bound (m_niChanges.begin (), m_niChanges.end (),
 NiChange (moment, 0, 0));

    }
+   //---------------February 3rd 2012 - add multipath received power -----------//
```

VII. ERROR RATE MODEL

VII.1 error-rate-model.h

```
@@  class ErrorRateModel : public Object
    {
    public:
    ...
+     //-----------------December 23th 2011-------------------//
+     uint16_t m_tx;
+     uint16_t m_rx;
+     virtual void SetTxAnt (uint16_t tx);
+     virtual uint16_t GetTxAnt (void) const;
+     virtual void SetRxAnt (uint16_t rx);
+     virtual uint16_t GetRxAnt (void) const;
+     //-------------------December 23th 2011----------------/*/

    };
```

VII.2 error-rate-model.cc

```
+   //-----------------December 23th 2011--------------------//
+   void
+   ErrorRateModel::SetTxAnt (uint16_t tx)
+   {
+     m_tx = tx;
+   }
+   uint16_t
+   ErrorRateModel::GetTxAnt (void) const
+   {
+     return m_tx;
+   }
+   uint16_t
+   ErrorRateModel::GetRxAnt (void) const
+   {
+     return m_rx;
+   }
+   void
+   ErrorRateModel::SetRxAnt (uint16_t rx)
+   {
+     m_rx = rx;
+   }
+   //---------------December 23th 2011-----------------------/*/
```

VII.3 yans-error-rate-model.h

```
    class YansErrorRateModel : public ErrorRateModel
    {
    public:
    ...
+     //-------------December 23th 2011----------------//
+     uint16_t m_tx;
+     uint16_t m_rx;
+
+     virtual void SetTxAnt (uint16_t tx);
+     virtual uint16_t GetTxAnt (void) const;
+     virtual void SetRxAnt (uint16_t rx);
+     virtual uint16_t GetRxAnt (void) const;
+     //--------------December 23th 2011--------------/*/
    ...
    };
```

VII.4 yans-error-rate-model.cc

```
+   //-----------------December 23th 2011--------------------//
+   void
+   YansErrorRateModel::SetTxAnt (uint16_t tx)
```

```
+   {
+     m_tx = tx;
+   }
+   uint16_t
+   YansErrorRateModel::GetTxAnt (void) const
+   {
+     return m_tx;
+   }
+   uint16_t
+   YansErrorRateModel::GetRxAnt (void) const
+   {
+     return m_rx;
+   }
+   void
+   YansErrorRateModel::SetRxAnt (uint16_t rx)
+   {
+     m_rx = rx;
+   }
+   //------------------December 23th 2011--------------------/*/
    ...
@@  double
    YansErrorRateModel::GetChunkSuccessRate (WifiMode mode, double snr,
uint32_t nbits) const
    {
    ...
      else if (mode.GetModulationClass () == WIFI_MOD_CLASS_DSSS)
{
  switch (mode.GetDataRate ())
    {
    case 1000000:
-      return DsssErrorRateModel::GetDsssDbpskSuccessRate (snr, nbits);
+   //------------------December 23th 2011--------------------//
+       if ((m_tx == 1) and (m_rx == 1))
+       {
+ NS_LOG_DEBUG("SISO: snr = " << snr);
+ return DsssErrorRateModel::GetDsssDbpskSuccessRate (snr, nbits);
+       }else{
+ NS_LOG_DEBUG("MIMO: snr = " << snr);
+ // Use this method to calculate success rate throw closed function
+ // obtained by Sergey and Loyka
+ return DsssErrorRateModel::SergeyLoykaVblastSuccessRate (snr, nbits, m_tx, m_rx);
+       }
+   //------------------December 23th 2011--------------------/*/
    case 2000000:
      return DsssErrorRateModel::GetDsssDqpskSuccessRate (snr, nbits);
    case 5500000:
      return DsssErrorRateModel::GetDsssDqpskCck5_5SuccessRate (snr, nbits);
    case 11000000:
      return DsssErrorRateModel::GetDsssDqpskCck11SuccessRate (snr, nbits);
    }
}
      return 0;
    }
```

VII.5 dsss-error-rate-model.h

```
@@  class DsssErrorRateModel
    {
    public:
+     //----------------------MIMO-------December 23th 2011--------------------//
+     static double SergeyLoykaVblastSuccessRate(double sinr, uint32_t nbits,
  uint16_t m_tx, uint16_t m_rx);
+     //-----------------------------MIMO------December 23th 2011--------------/*/
    ...
    };
```

VII.6 dsss-error-rate-model.cc

```
+   //--------------December 23th 2011---------------//
+   #include "sergey_loyka_vblast_analytical/math_new.cc"
+   //--------------December 23th 2011---------------/*/
    ...
+   //-----------------------December 23th 2011-----------------------------------//
+   double
+   DsssErrorRateModel::SergeyLoykaVblastSuccessRate(double sinr, uint32_t nbits,
      uint16_t m_tx, uint16_t m_rx)
+   {
```

```
+     double EbN0 = sinr * 22000000.0 / 1000000.0;
+     NS_LOG_DEBUG (evalBER(m_rx, m_tx, EbN0) << " " << sinr);
+     return pow((1 - evalBER(m_rx, m_tx, EbN0)), nbits);
+   }
+   //-----------------------December 23th 2011-----------------------------------//
```

VIII. V-BLAST BIT ERROR RATE MODEL

Expressão matemática para a taxa de erro de bit do sistema V-BLAST proposto por Loyka *et al.* [2].

VIII.1 math_new.h

```
#define ERROR_CODE -1

namespace ns3 {
double fatorial(double input);
double comb(double n,double k);
double evalBER(uint16_t nRX, uint16_t nTX, double sinr);
}
```

VIII.2 math_new.cc

```
/*
 * All rights reserved by NERDs,
 * The file may not be re-distributed without explicit authorization
 * from UnB NERDs .
 * Checked for proper operation with Visual Studio
 * Author        : Tiago da Silva Bonfim
 * Email         : tiago.bonfim@gmail.com
 * Version       : 1.0
 * Date          : 30 November 2011
 *

 * Library for computing the BER for BPSK modulation in a
 * Rayleigh fading channel with nTX, nRX MIMO channel
 * Zero forcing Error Equalization with Successive Interference
 * Cancellation (ZF-SIC) without optimal ordering
 */
//-------------------Includes & Variables--------------------//
#include <math.h>
#include "math_new.h"
#include <stdio.h>

namespace ns3 {
float at[10] = {1,0.594,0.445,0.364,0.311,0.275,0.247,0.224,0.206,0.191};
// Vector with nTX impact


//----------------------Functions --------------------------//

/*-------------------------------------------------------------
 *  Brief Calcula a fatorial de um número
 *  Input:  (double input) valor de entrada
 *  Output: (double result) fatorial calculada
 -------------------------------------------------------------*/

double fatorial(double input)
{
   static double result;
   if(input)
   {
      result = input;
   }
   else
   {
      result = 1;
   }
   while(input > 1)
   {
      input--;
      result = result*input;
   }
   return result;
}

/*-------------------------------------------------------------
 *  Brief Calcula a combinação n k a k
 *  Input:  (double n) valor de entrada 1
 *          (double k) valor de entrada 2
```

```
 *  Output: (double result) combinação calculada
 ------------------------------------------------------------*/

double comb(double n,double k)
{
   double result = (fatorial(n))/(fatorial(k)*fatorial(n-k));

   if(n>=k)
   {
      return result;
   }
   else
   {
      printf ("Erro no cálculo da relação sinal ruído");
   }
   return ERROR_CODE;
}

/*------------------------------------------------------------
 *  Brief Calcula a BER do V-blast com SIC e ZF para os valores
 *  de entrada fornecidos
 *  Input:  (char nRX) número de antenas receptoras
 *          (char nTX) número de antennas transmissoras
 *  Output: (float result) BER calculada
 ------------------------------------------------------------*/

double evalBER(uint16_t nRX, uint16_t nTX, double sinr)
{
   float result;
//   float SNR = pow(10,(SNRdb/10));
//   float debug1,debug2,debug3;

   result = ((at[nTX-1])*(comb((2*(nRX-nTX)+1),(nRX-nTX+1))))
    /(pow(4*sinr,(nRX-nTX+1)));

   if(nRX>=nTX)
   {
      return result;
   }
   else
   {
      printf ("ERRO: NTX > NRX");
   }
   return ERROR_CODE;
}
}
```